Modélisation 3D de bâtiments : reconstruction automatique de superstructures de toits et recalage cinétique de toits polyédriques prenant en compte la topologie

Mathieu Brédif

École Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

Thèse

présentée pour obtenir le grade de docteur

de Télécom ParisTech

Spécialité : Signal et Images

Mathieu BREDIF

Modélisation 3D de bâtiments

Recalage cinétique à topologie variable de toits polyédriques
et
Reconstruction automatique de superstructures de toits

Soutenue le 27 mai 2010 devant le jury composé de

Isabelle Bloch	**Présidente**
Dominique Bechmann	**Rapporteurs**
Renaud Keriven	
Pierre Alliez	**Examinateurs**
Caroline Baillard	
Didier Boldo	
Henri Maître	**Directeurs de thèse**
Marc Pierrot-Deseilligny	

Avant-propos / Foreword

Ce manuscrit s'articule autour de deux parties relativement indépendantes (les parties II et III), intitulées respectivement *Reconstruction automatique de superstructures de toits* et *Recalage cinétique à topologie variable de toits polyédriques*.

Si votre principal centre d'intérêt est la reconstruction de bâtiments, vous pouvez en première lecture passer les détails techniques de la partie III, et en particulier les chapitres 5 et 6. Les résultats obtenus avec l'approche proposée sont présentés dans le chapitre 7, et devraient vous convaincre que les problèmes topologiques mentionnés dans le chapitre 4 ont été traités avec succès.

Au contraire, la partie III s'adresse aux lecteurs passionnés par la géométrie algorithmique. Le chapitre 4 pose le contexte de l'approche proposée de recalage de bâtiment, avant que les chapitres 5 et 6 ne plongent dans nos contributions au domaine de la géométrie algorithmique.

D'autre part, pour donner un sens aux temps de calcul relevés dans ce manuscrit, nous précisons que tous ces temps ont été mesurés sur un unique coeur CPU Intel® Xeon® 5110 de fréquence 1.60GHz, muni de 2.0Go de mémoire vive. Enfin, si la thèse elle-même est rédigée en anglais, elle est précédée d'un résumé étendu en français qui présente les idées directrices du manuscrit et leur enchaînement.

Happy reading...

The core of this thesis falls into the two independent parts II and III, entitled respectively *Automatic Roof Superstructure Reconstruction* and *Topology-Aware Kinetic Fitting of Polyhedral Roofs*.

If your primary interest is building reconstruction, you may skip the inner details of part III, namely chapters 5 and 6. The results of the proposed approach, presented in chapter 7, will convince you that the topological issues mentionned in chapter 4 have been addressed.

By contrast, readers primarily interested in computational geometry may find relevant materials in part III. Chapter 4 provides some context about our building fitting approach, before chapters 5 and 6 dive into the contributions to the field of computational geometry.

As a sidenote, to make the computing times mentionned in this work meaningful, the reader should note that all timings are measured on a single Intel® Xeon® 5110 CPU core running at 1.60GHz, with 2.0GB of main memory. Finally, although this thesis is written in English, it is prefixed with an extended summary in French.

Bonne lecture...

Remerciements / Acknowledgments

Cette thèse s'est déroulée au laboratoire MATIS de l'Institut Géographique National, en collaboration avec le département TSI de l'école Télécom ParisTech. Je souhaite tout d'abord exprimer toute ma gratitude à Henri Maître, qui a accepté de diriger mes recherches et dont j'ai beaucoup apprécié la précision des remarques et les critiques attentives tout au long de la thèse. Je remercie également très sincèrement Marc Pierrot-Deseilligny, qui a encadré et codirigé cette thèse, pour de nombreuses discussions scientifiques et toute la confiance qu'il m'a témoignée dans le cadre de ce travail.

Je remercie par ailleurs tous les membres du jury pour l'intérêt dont ils ont fait preuve à l'égard de ce travail. Isabelle Bloch m'a fait l'honneur de présider ce jury et m'a montré une grande bienveillance. Je remercie très sincèrement Dominique Bechmann et Renaud Keriven, tous deux rapporteurs de cette thèse, pour le temps qu'ils ont accepté de consacrer à la lecture de ce manuscrit, Pierre Alliez, Caroline Baillard et Didier Boldo pour la pertinence de leurs questions et leurs remarques positives sur mes travaux.

Prior to this thesis, I feel tremenduously lucky to have worked as a research assistant with James Arvo and Mike Stark at UC Irvine, and later with Pat Hanrahan, Marc Levoy and Ren Ng at Stanford University. I owe them my strong inclination for research and my fascination for computer graphics. If I had a single teacher to thank, it would definitely be Leonidas Guibas, who instilled me with its passion of computational geometry. My thesis would not have lead to a new kinetic data structure without me taking his class.

Je remercie par ailleurs mes chefs de laboratoire successifs, Didier et Nicolas, pour leur confiance indéfectible. De plus, je tiens à remercier tout particulièrement tous ceux qui ont tour à tour partagé mon bureau : Mélanie, Arnaud, Bruno, Fadi, Hassan et enfin l'inoubliable Florent. Ils m'ont vaillamment et gaiement supporté tout au long de ma thèse (aux sens français et anglais du terme). Merci à Marie-Claude pour son indulgence et son extrême efficacité administrative. Merci aussi à François de nous faire partager les bénéfices de sa connaissance des rouages de l'IGN et pour sa capacité à résoudre immédiatement tout problème logistique ou technique.

Je tiens à remercier Grégoire et David pour la mise à disposition du formidable outil qu'est la maquette. Les tests d'intégration de mes travaux sur la plateforme de production Bati3D n'aurait pas pu voir le jour sans la disponibilité et les compétences de Grégoire. Je vais essayer de n'oublier aucun des autres membres passés ou présents du MATIS qui ont, chacun à leur manière, contribué à la bonne ambiance du laboratoire : Jean-Pascal, Bertrand, Nicolas, Isabelle, Fabien, Nicolas, Alexandre, Antonio, Athanasios, Karim, Patrick, Lâmân, Clément, Jean-Pierre, Mélanie, Bahman, Olivier, Frédéric, Nesrine, Adrien, Daniela, Erwann, Corina, Emilie et Lionel.

Enfin, je désire exprimer la plus profonde reconnaissance à tous les membres de ma famille et tout particulièrement à Perrine, mon épouse, et à mes enfants Noé et Juliette qui, par leur affection, m'ont soutenu au cours de ces années, ainsi qu'à mes amis qui ont su me distraire sans trop me demander : “Alors, c'est pour quand cette thèse ?”.

Notations and Abbreviations

DSM	Digital Surface Model
KDS	Kinetic Data Structure
B-Rep	Boundary Representation
GSD	Ground Sample Distance
CSG	Constructive Solid Geometry
Superstructure	Small scale geometric and topologic detail of an object.
$[x : y : z : w] = [\vec{p} : w]$	Homogeneous point coordinates
$[a : b : c : d] = [\vec{n} : d]$	Homogeneous plane coordinates
$\frac{\vec{p}}{w} = (\frac{x}{w}, \frac{y}{w}, \frac{z}{w})$	Cartesian point coordinates ($w \neq 0$) of point $[x : y : z : w] = [\vec{p} : w]$
$\vec{p} = (x, y, z)$	Direction of point $[\vec{p} : w]$
$\vec{n} = (a, b, c)$	Oriented plane normal vector
$[\frac{\vec{p}}{w} : 1]$	Normalized homogeneous point coordinates ($w \neq 0$)
$[\frac{\vec{n}}{\lvert\vec{n}\rvert} : \frac{d}{\lvert\vec{n}\rvert}]$	Normalized homogeneous plane coordinates ($\vec{n} \neq \vec{0}$)
A^T	The transpose of a matrix A
$\vec{X} \cdot \vec{Y} = \vec{X}^T \vec{Y}$	The dot product of the vectors $\vec{X}$ and $\vec{Y}$
$\vec{X} \times \vec{Y}$	The vector product of the vectors $\vec{X}$ and $\vec{Y}$
A_{ij}	The element of the matrix A at the i^{th} column and j^{th} column (starting at 1)
$\det A$ **or** $\lvert A\rvert$	The determinant of the matrix A
$\operatorname{com} A$	The cofactor matrix of the matrix A (*i.e.* the matrix of signed minors)
$\arg\min_{x \in X} f(x)$	The global minimum of f over X, or any one of them if multiple exist
$\arg\max_{x \in X} f(x)$	The global maximum of f over X, or any one of them if multiple exist

Contents

List of Figures

List of Tables

Modélisation 3D de bâtiments

Sommaire

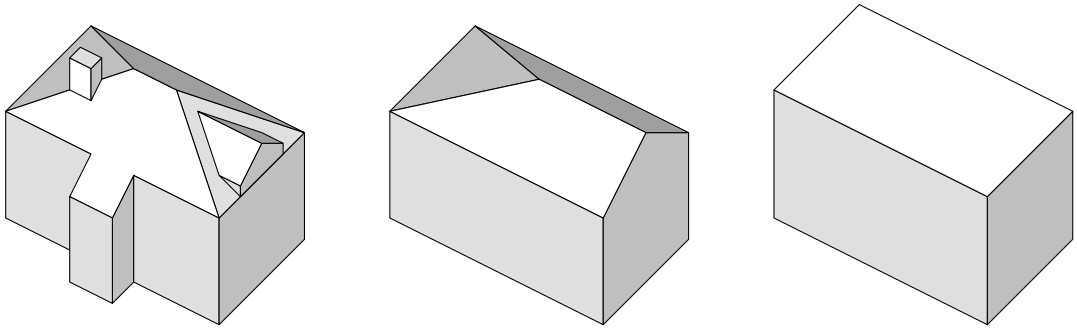
1 Introduction

1.1 Contexte

La modélisation numérique d'une ville en 3D nécessite la création d'une base de données d'objets urbains décrits en 3D (la surface du sol, ses bâtiments, ses rues, sa végétation...). Ces modèles de ville sont utilisés dans le cadre d'études d'urbanisme ou architecturales, pour le tourisme, au sein de simulations physiques (pollution sonore, flux thermiques, transmission radio pour le placement optimal d'antenne de téléphonie mobile), pour la gestion des catastrophes naturelles telles que les inondations, pour la défense et la sécurité civile, pour l'aide à la navigation routière ou pédestre, ou encore pour la robotique.

Cette thèse porte plus particulièrement sur le processus de modélisation des bâtiments. En fonction de la donnée disponible pour la création d'un modèle de ville et de l'application visée, le niveau de détail (accessible ou requis) de description géométrique, radiométrique et sémantique de la ville varie grandement. Les bâtiments peuvent ainsi n'être décrits à une précision métrique qu'à l'aide d'une emprise au sol polygonale et d'une altitude moyenne, ou peuvent à l'inverse représenter à une précision centimétrique chacune des façades et chacun des pans de toit, ainsi que leurs détails géométriques, tels que les fenêtres, les balcons, les cheminées ou les chiens assis. Lorsqu'un modèle de ville est utilisé dans des applications de réalité virtuelle ou augmentée, l'apparence radiométrique de ses objets doit aussi être modélisée à l'aide, par exemple, de textures issues des images aériennes ou terrestres utilisées pour la création du modèle de ville. Enfin, certaines applications nécessitent, au delà d'une simple description géométrique du modèle, d'informations sémantiques telles que la donnée que telle partie du modèle correspond à un mur, une porte, une fenêtre, un pan de toit ou à une cheminée.

Pour créer ou maintenir un modèle de ville, plusieurs types de données peuvent être utilisés. Il s'agit principalement d'imagerie terrestre, aérienne ou satellitaire, et de télémétrie laser (LIDAR)



(a) (b) (c)

FIGURE 1 – Niveaux de détails CityGML pour les bâtiments : 3 (a), 2 (b) et 1 (c).

terrestre ou aérienne. Le niveau de détail accessible pour la modélisation d'une ville dépend directement de la précision et du type de la donnée utilisée et du temps alloué aux interactions humaines de saisie ou de retouche.

CityGML [KGP05] est un modèle de données ouvert pour l'échange et l'archivage de modèles 3D de villes. Il définit 5 niveaux de détail (figure. 1) pour la description d'un bâtiment :

Niveau 0 Les bâtiments ne sont pas décrits.

Niveau 1 Les bâtiments ont des toits plats et des façades à angles droits.

Niveau 2 Les façades et les faces de toits sont décrits avec une précision métrique.

Niveau 3 Les superstructures de toits et de façades telles que les portes, les fenêtres, les balcons, les cheminées, les chiens assis et les verrières sont décrites. La précision géométrique des éléments décrits est centimétrique.

Niveau 4 Au delà d'une description topologique et géométrique plus précise, l'intérieur des bâtiments est aussi modélisé.

Un des objectifs de l'Institut Géographique National (IGN) est la production et la mise à jour de modèles de ville 3D. Sur cet objectif, ses activités de recherche visent à augmenter la précision et à réduire les coûts de production et de mise à jour de ces bases de données. Les processus de production de bases de données équivalentes aux niveaux de détail 0 (BD-Alti) et 1 (BD-Topo) sont maîtrisés à l'IGN et ces bases de données sont disponibles sur toute la France. Le produit Bati3D, correspondant au niveau de détail 2, vient d'être industrialisé suite aux travaux de [Tai05, DT06].

L'objectif de cette thèse est d'accroître la précision géométrique, topologique et sémantique des modèles de bâtiments produits dans un contexte tout-automatique pour atteindre une description des toits équivalente à un niveau de détail 3. De plus nous nous plaçons dans le cadre opérationnel de la mise à jour d'une base de données préexistante à améliorer au vu de nouvelles données plus récentes et/ou plus précises. Dans ce cadre, pour conserver la cohérence de la base de données au cours de cette mise à jour auprès des utilisateurs qui l'exploitent, les bâtiments sont traités individuellement afin de garantir un appariement simple entre l'ancienne et la nouvelle base.

1.2 Données utilisées

Cette thèse suppose qu'un modèle type Bati3D (niveau de détail 2) est préexistant et cherche à le préciser et à le compléter grâce aux informations fournies par un Modèle Numérique de Surface (MNS, figure 2) d'assez grande précision (10 cm par pixel au sol). Un MNS est une image de hauteur brute décrivant une surface sans aucune sémantisation (*i.e.* sans connaissance des objets qu'il représente). Un MNS de cette précision est typiquement produit grâce à des images aériennes ou à un LIDAR aérien. Le LIDAR procure un nuage de point 3D par mesure du temps de parcours aller-retour d'une impulsion lumineuse entre l'avion et la surface de la ville qui est ensuite rééchantillonné

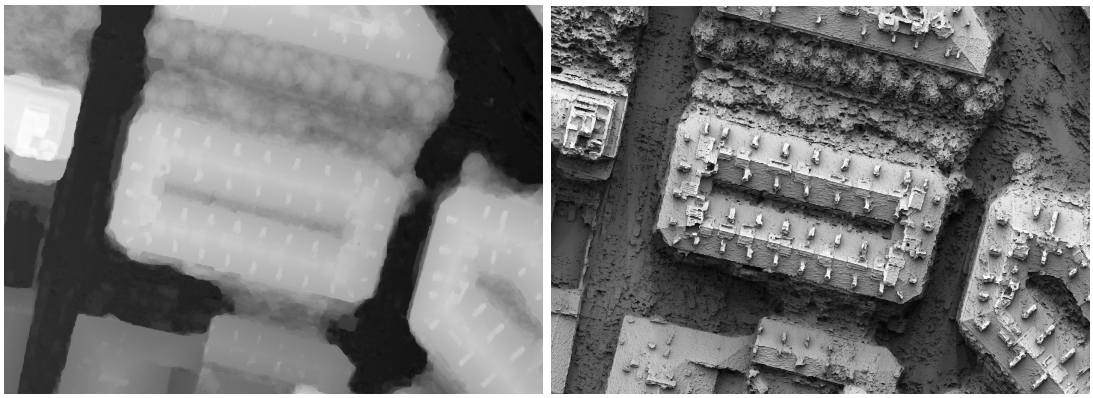


Figure 2 – MNS calculé à partir des images de la figure 3. (à gauche) vue directe de l'image de hauteur (sombre=bas, clair=élevé). (à droite) vue ombrée, souvent plus lisible, de la même surface utilisant un algorithme d'occlusion ambiente.

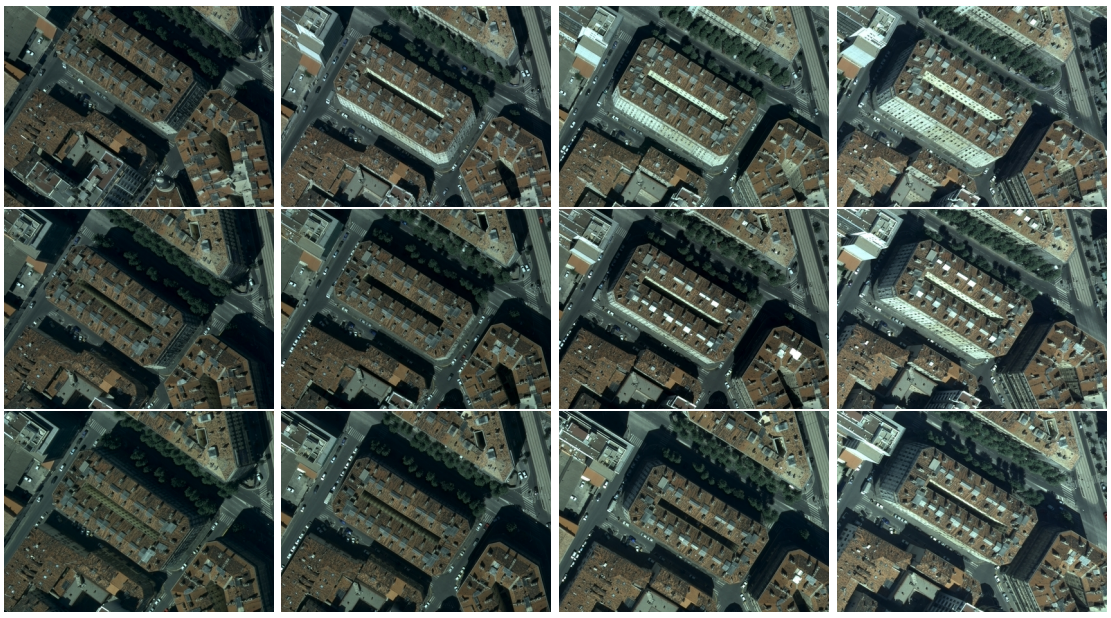


Figure 3 – Imagerie aérienne multivue : Bâtiment vu depuis 12 points de vue aériens différents (Ces imagettes 1186x880 sont extraites d'images aériennes 13824x7680 pixels).

sur la grille régulière d'un MNS. Les MNS produits par imagerie aérienne (figure 3) utilisent le principe photogrammétrique de reconstitution de la profondeur par l'analyse de la déformation due au relief entre deux images ou plus de la même zone vue de points de vue différents.

1.3 Approche retenue

L'amélioration de la modélisation des bâtiments, qui est le but global de cette thèse, peut se diviser en deux sous-objectifs (fig 4) :

1. Les toits représentés dans le modèle initial peuvent être replacés plus précisément sur la donnée de mise à jour, c'est ce que nous nommons le **recalage des pans de toit principaux**.
2. Les objets présents sur les toits tels que les cheminées, chiens assis ou autres superstructures ne sont généralement pas modélisés, il va donc falloir les détecter et les ajouter au modèle 3D décrivant le bâtiment, c'est ce que nous nommons la **reconstruction des superstructures**.

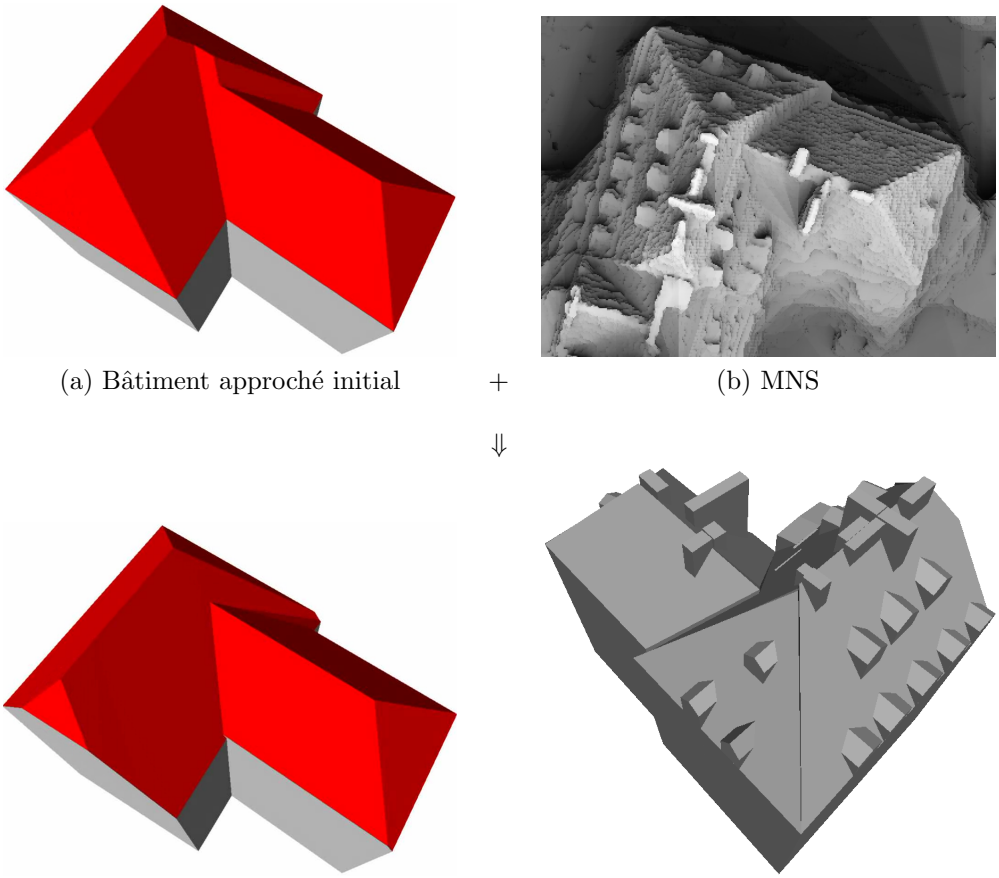
Dans le contexte tout automatique de cette thèse, il n'a pas été réaliste d'attaquer simultanément les problématiques de recalage et de reconstruction des superstructures. L'approche proposée se base donc sur une optimisation alternée de ces deux sous-objectifs en considérant l'autre sous-objectif fixé (fig. 5). En effet, ces sous-objectifs sont fortement interdépendants : un meilleur recalage des pans de toit principaux procure une meilleure modélisation géométrique et facilite donc l'extraction des superstructures par inspection de l'erreur verticale entre le modèle recalé et le MNS. Inversement, les superstructures biaisent l'estimation de la géométrie des pans de toit principaux. La détection des superstructures permet de ne pas prendre en compte ces données perturbatrices dans l'optimisation de la géométrie des pans de toit principaux. La discussion de cette interdépendance et la proposition de cette optimisation alternée est une des principales contributions thématiques de cette thèse.

2 Reconstruction automatique de superstructures de toits

La reconstruction de superstructures de toit (cheminées, chiens assis...) est un sujet de recherche extrêmement récent pour plusieurs raisons. Jusqu'à très récemment, la taille réduite de ces objets rend leur reconstruction et même leur détection ambigüe par manque de données assez précises. De plus leur diversité et leur variabilité constitue un frein certain à leur reconstruction automatique. Enfin la maturation récente des approches visant à reconstruire les bâtiments eux-mêmes semblait préalable à l'étude de la reconstruction de ces superstructures.

La taille de ces superstructures est réduite par rapport à la précision des données aériennes utilisées pour la reconstruction de bâtiment, ce qui conduisit [MV99] à ne reconstruire que les chiens assis de taille conséquente sur des données LIDAR et [LDZPD10] à saisir manuellement la planimétrie des superstructures, n'optimisant automatiquement que leur altitude. Une des principales difficultés limitant leur reconstruction est leur taille et la modélisation imparfaite des pans de toit principaux. Pour contrer cette modélisation imparfaite, [Nan06] adopte une approche novatrice en extrayant des cheminées indépendamment des pans de toit principaux par une approche basée sur une segmentation hiérarchique d'images aériennes suivie d'une extraction de cheminées sur des critères géométriques et radiométriques. Il est à noter que la plupart des recherches en reconstruction de bâtiment considèrent les superstructures comme du bruit sans chercher à comprendre exactement la structure de ces objets. Une de nos contributions est justement d'extraire explicitement les superstructures afin de mieux reconstruire les pans de toit principaux.

Le problème posé dans le chapitre 3 consiste à détecter et à reconstruire automatiquement un nombre inconnu de superstructures de toit étant donné un MNS et un modèle 3D de bâtiment considéré comme parfait en géométrie au niveau des pans de toit principaux. A cette fin, nous proposons une représentation hybride des bâtiments entre un bâtiment polyédrique et des superstructures paramétriques (figure 6). L'intérêt applicatif de cette représentation est qu'elle apporte



(a) Bâtiment approché initial + (b) MNS

⇓

(c) Bâtiment recalé final + (d) Superstructures reconstruites

Figure 4 – Contexte de la thèse : à partir d'un modèle de bâtiment approché (a) et d'un MNS décrivant ce bâtiment (b : vue ombrée), il s'agit de recaler les pans de toit (c) et de reconstruire ses superstructures (d).

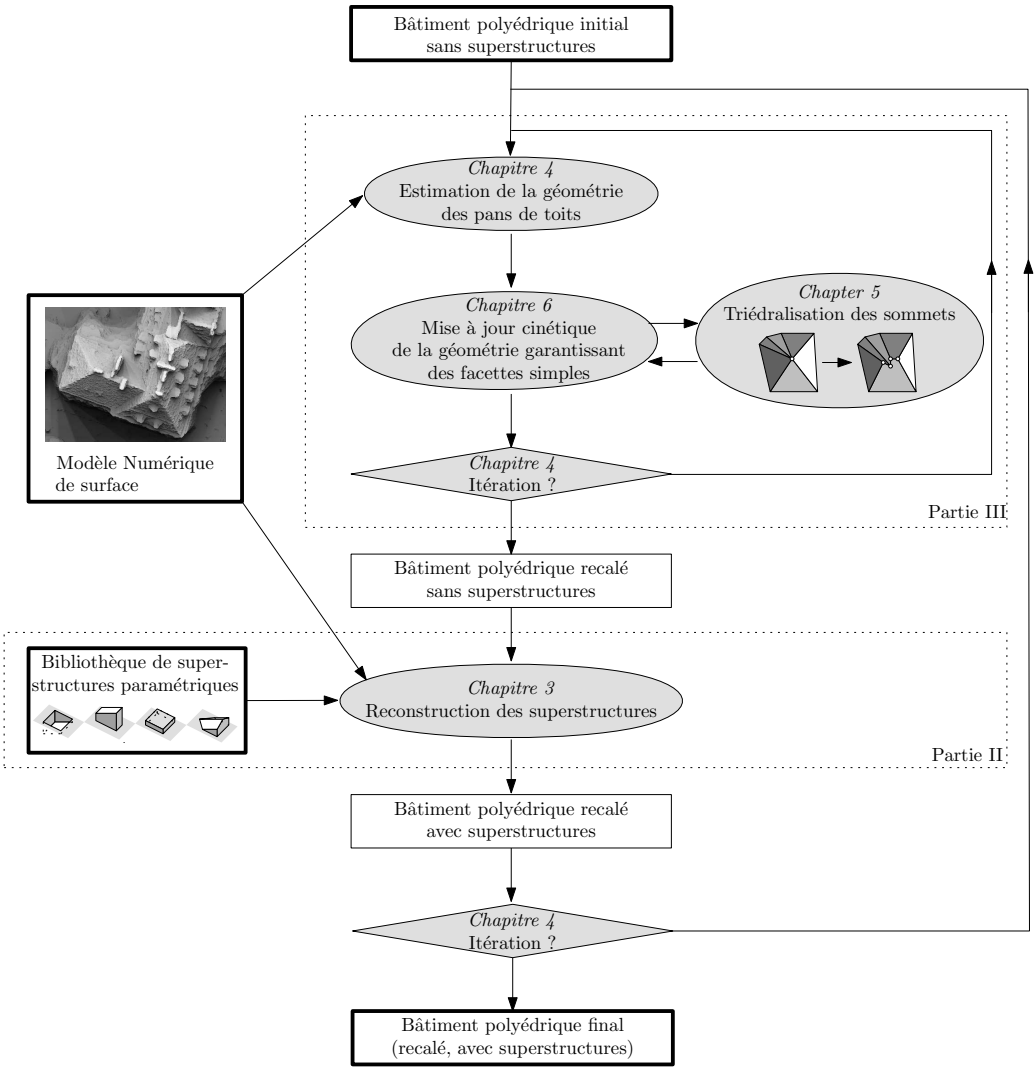


FIGURE 5 – Vue globale de l'approche retenue.

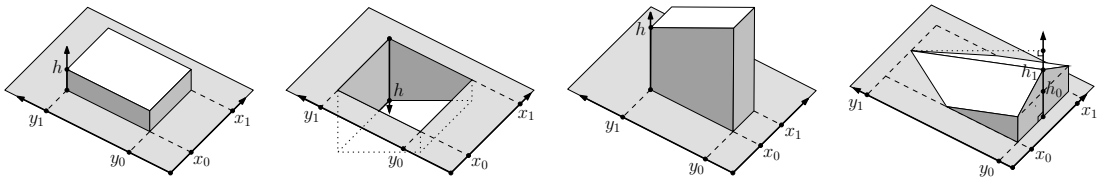


Figure 6 – Modèles paramétriques de superstructures : (a) $Verriere$, (b) $Terrasse$, (c) $Cheminee$ et (d) $ChienAssis$.

de la sémantique : les faces du modèle de bâtiment final sont alors groupées de façon sémantiquement homogène en façades, pans de toit, cheminées etc. Cette représentation hybride procure deux niveaux de détail : avec ou sans superstructure.

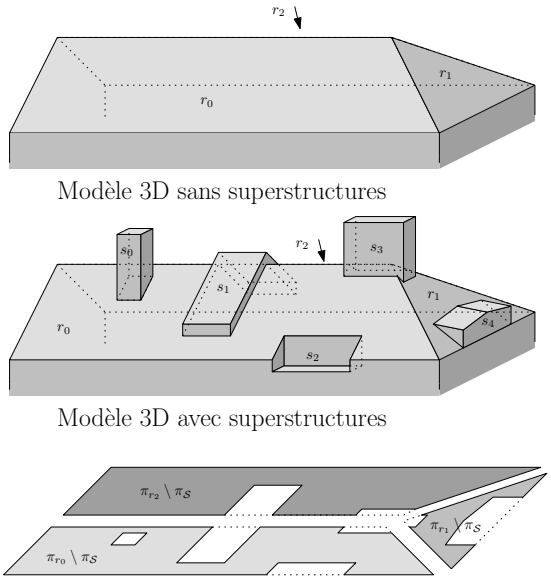


Figure 7 – Représentation hybride d'un bâtiment par une base polyédrique générique munie de superstructures paramétriques contraintes. L'ensemble des données utilisées pour la réestimation d'une face modélisant un pan principal de toit correspond au support 2D de cette face restreint par les supports 2D des superstructures détectées.

Nous définissons la robustesse d'un algorithme comme sa capacité à extraire une information plausible et vraisemblable sur tout type de donnée, quitte à renvoyer des résultats trop simplifiés ou normalisés. Cette notion de robustesse est souvent opposée à la généricité d'un algorithme permettant de retourner des résultats plus complexes, plus proches de la donnée analysée, en contrepartie d'une vraisemblance réduite. Pour le type de données en entrée visé, il s'avère que les pans de toit principaux bénéficient d'observations abondantes, alors que les zones de MNS décrivant les superstructures aparaissent bruitées et relativement peu étendues. La modélisation hybride proposée présente donc un bon compromis entre généricité pour la description du bâtiment principal et robustesse au niveau des superstructures.

L'ensemble des meilleures superstructures est discriminé suivant une formulation énergétique de type MDL (Minimum Description Length, [Ris78]), qui permet naturellement d'exprimer le compromis entre maîtrise de la complexité de la solution et attache aux données. Le terme de complexité est constant par type de superstructure et l'attache aux données correspond à une norme $\mathcal{L}_p$ sur l'erreur verticale entre le MNS et le modèle reconstruit. L'annexe A développe une

spécialisation de la méthode d'optimisation dans le cas de la norme $\mathcal{L}_2$.

La reconstruction des superstructures se déroule schématiquement en trois étapes :

1. Un balayage exhaustif de toutes les superstructures plausibles optimise les paramètres verticaux de description de la structure pour chaque type de superstructure recherchée, et chaque jeu quantifié plausible de paramètres horizontaux.
2. Pour réduire la taille du problème, on filtre les superstructures candidates en ne gardant que les superstructures qui ont un score localement maximum.
3. On se retrouve alors confronté à un problème d'optimisation combinatoire : la sélection de superstructures disjointes, qui se réduit à un problème de clique maximale pondérée.

En pratique, les paramètres verticaux (hauteurs) des superstructures étant à emprise constante sur le MNS, ils peuvent être optimisés de façon continue. Les paramètres horizontaux, eux, (translations, orientation, largeur, longueur) sont discrétisés, menant à une optimisation combinatoire. De plus, afin de palier les problèmes de sur-détection, on fait la supposition que les superstructures sont disjointes en planimétrie : elles ne se chevauchent pas. Cette modélisation ne permet pas de modéliser des superstructures non disjointes. De plus, il est à noter que la classification entre pans de toit et superstructures peut-être ambigüe, principalement pour les grands chiens assis. Enfin, la quantification des paramètres horizontaux peut s'avérer une limite de cette représentation. C'est pourquoi une approche stochastique ne quantifiant pas ces paramètres a été explorée et développée dans l'annexe B.

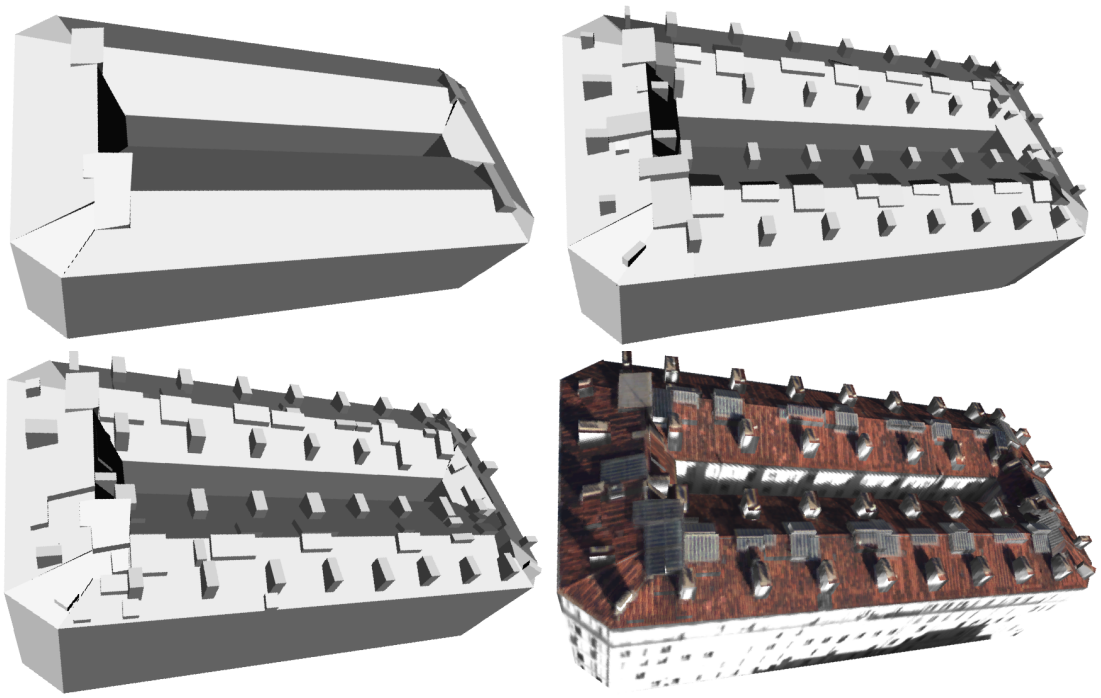


FIGURE 8 – Dans le sens de la lecture : un modèle de bâtiment sans superstructure, ce modèle muni de superstructures saisies manuellement, ce modèle muni de superstructures reconstruites automatiquement et enfin une version texturée par les images aériennes du modèle précédent.

L'approche proposée à l'atout de présenter une bibliothèque de superstructures extensible et l'avantage opérationnel de s'effectuer de façon totalement automatique. De plus la modélisation hybride retenue introduit de la sémantique et permet, outre les applications de visualisation pour lesquelles la méthode proposée semble tout à fait adaptée, de répondre à des problématiques de comptage d'objet : calcul du nombre de chiens assis, souvent mal isolés, dans des contextes de prospection énergétique par exemple. Le point faible de l'approche proposée est sa sensibilité aux

artefacts du MNS, par exemple au pied de discontinuités altimétriques et à la difficulté de reconstruire des superstructures basses telles que des verrières qui ont un niveau de signal comparable au bruit de la donnée en entrée.

Les principales pistes d'extension de cette méthode comprennent l'étude de données alternatives au MNS tel que le LIDAR ou un retour aux images brutes (études des ombres dans la suite de [Nan06], de la radiométrie, reconstruction multivue...) et l'introduction dans l'énergie à minimiser d'un a priori de régularisation tel que l'alignement des cheminées ou la similarité des dimensions des chiens assis. La complexification induite par ce changement d'énergie demanderait une modification de la méthode d'optimisation. Une approche stochastique de type RJ-MCMC [LDZPD10] semble alors indiquée et permettrait de plus de déquantifier l'ensemble des paramètres décrivant les superstructures. Enfin, la phase actuelle d'énumération des superstructures candidates se prêterait bien à un portage sur carte graphique (GPGPU) afin de réduire considérablement les temps de calcul à quantification égale, ou d'améliorer la précision de la reconstruction à temps de calcul équivalent.

3 Recalage cinétique à topologie variable de toits polyédriques

3.1 Recalage de polyèdre à topologie fixée

Le recalage d'un bâtiment sur un MNS consiste en une optimisation d'un polyèdre modélisant ce bâtiment sur ce MNS. Par souci de vraisemblance, nous imposons la non auto-intersection de la surface polyédrique du modèle recalé décrivant le bâtiment. De plus, pour simplifier le problème et par cohérence avec la nature 2.5D du MNS, le modèle polyédrique recherché ne comporte pas de surplomb. Ceci simplifie la contrainte de non auto-intersection de la surface globale en une contrainte de non auto-intersection de chacune des faces polyédriques en empêchant les intersections globales de repliement.

Figure 9 – Recalage de bâtiment à topologie variable (bâtiment filaire projeté sur une orthophotographie).

La problématique d'optimisation d'un maillage sur une observation a suscité de nombreuses recherches. [HDD+93] propose d'optimiser une surface triangulée en modifiant les coordonnées de ses sommets. Appliquée à notre contexte polyédrique, cette approche nécessite une triangulation arbitraire préalable des faces du polyèdre initial afin de se ramener à une surface triangulée où les sommets peuvent bouger librement. En effet, une surface triangulée n'impose plus aucune contrainte géométrique de coplanarité sur les sommets délimitant chaque face. Cette approche est efficace mais elle présente deux principaux inconvénients dans notre contexte. Premièrement, cette triangulation arbitraire des faces fend les façades et les pans de toit de grandes diagonales sans réalité physique ou sémantique. Deuxièmement, cette méthode n'apporte aucune garantie de non auto-intersection de la surface triangulée recalée. Plus récemment, [CSAD04] adopte l'approche duale en n'optimisant non plus la position des sommets mais les équations des plans supportant

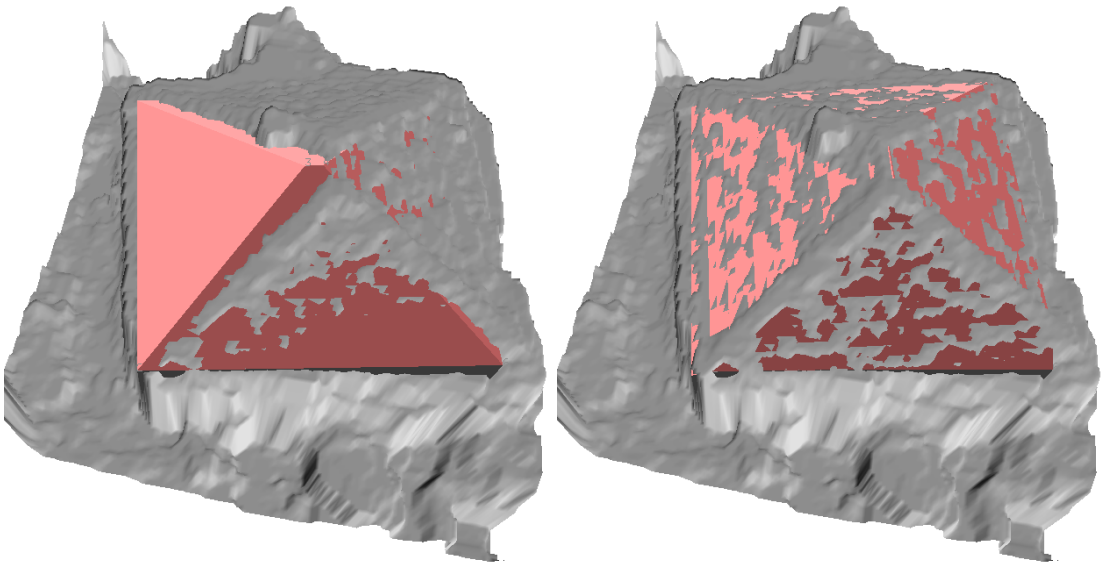


FIGURE 10 – Recalage de bâtiment à topologie variable (visualisation 3D de la surface MNS et du bâtiment).

les faces des polyèdres. Cette approche semble plus appropriée à notre contexte où tant la donnée MNS que la sémantique des modèles portent plus sur ses faces que sur ses sommets. Là encore la surface optimisée ne présente aucune garantie de non auto-intersection. De plus cette méthode ne maintient le polyèdre en cours d'optimisation qu'implicitement à travers une partition de la donnée de recalage. Ainsi, une région correctement détectée peut-être scindée en de multiples faces durant la phase finale d'export polyédrique.

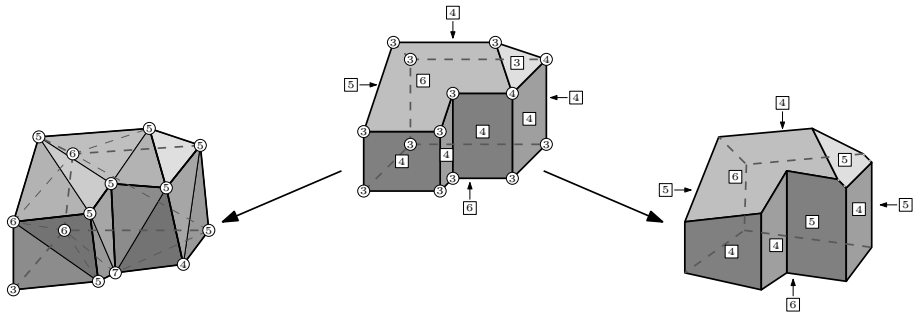


FIGURE 11 – L'optimisation non contrainte d'un polyèdre demande (à gauche) soit une triangulation préalable à une optimisation des coordonnées des sommets [HDD+93], (à droite) soit une triédralisation préalable à une optimisation des équations de plans supportant les faces polyédriques.

La difficulté de cette optimisation polyédrique provient de sa double nature : **combinatoire** au niveau de sa **topologie** (*e.g.* liste des identifiants de sommets adjacents à chaque face) et **continue** au niveau de sa **géométrie** (*e.g.* coordonnées des sommets ou, de manière équivalente, équations des plans supportant chaque face). Une propriété supplémentaire de l'approche proposée est la recherche d'une certaine forme d'hystérésis : le polyèdre recalé doit être le plus topologiquement proche possible du polyèdre initial. Ceci implique des modifications topologiques parcimonieuses pour ne remettre en cause la topologie du bâtiment initial que si nécessaire. Ainsi un bâtiment ambigü sera recalé le plus fidèlement possible à une topologie initiale qui aura peut-être nécessité l'intervention humaine d'un opérateur. Nous ne chercherons pas à expliciter et à quantifier cette notion de distance topologique entre deux polyèdres, ni à la mettre en compétition avec la distance

géométrique en cours d'optimisation. Cette propriété de parcimonie découlera naturellement et implicitement de l'approche cinétique proposée et développée au chapitre 6.

Dans le contexte de la reconstruction de bâtiment, [VT05] propose une méthode de recalage géométrique de bâtiments polyédriques sur des données images. Cette méthode vise à un ajustement géométrique d'un bâtiment topologiquement juste et ne remet donc pas en cause sa topologie. Ainsi, il ne garantit aucune propriété de non intersection sur le polyèdre recalé. Cette méthode ne peut donc pas être appliquée directement dans notre contexte : une topologie initiale erronée doit être remise en cause. A l'inverse, de nombreuses approches de reconstruction de bâtiment optimisent d'abord la géométrie par la recherche de régions planes puis cherchent à retrouver une topologie compatible avec la géométrie détectée. Ainsi, [Tai05] détecte des plans puis effectue une recherche combinatoire d'un bâtiment parmi l'arrangement de ces plans, cette méthode souffrant d'une explosion combinatoire dès que le nombre de plans détecté est conséquent. De son coté, [EAH08] essaie de retrouver la topologie du bâtiment polyédrique à partir d'une segmentation en régions planaires d'un MNS. Ce problème est mal posé et l'approche proposée repose sur des heuristiques soumises à de multiples paramètres difficilement réglables.

Au niveau de nos contributions, le chapitre 4 propose une solution simple au problème du recalage géométrique d'un polyèdre à topologie fixée basé sur une réestimation itérative indépendante des équations de chaque plan supportant chaque face du polyèdre. Cette estimation non-contrainte et indépendante des plans porteurs n'est possible que si tous les sommets sont triédraux (adjacents à 3 faces uniquement). En effet, un sommet non-triédral est localisé à l'intersection de 4 plans ou plus, et cette intersection est en général vide. Le chapitre 5 modifie la topologie initiale du polyèdre afin que le recalage précédent soit aussi applicable lorsque des sommets ne sont pas tous triédraux. Nous introduisons ce problème nouveau de géométrie algorithmique, et le nommons triédralisation. Ces deux chapitres couplés réalisent une méthode duale à une triangulation suivie d'une optimisation des coordonnées des sommets [HDD+93]. Enfin, le chapitre 6 apporte la garantie recherchée en modifiant parcimonieusement la topologie du polyèdre recalé au cours du recalage : il n'aura aucune face auto-intersectante.

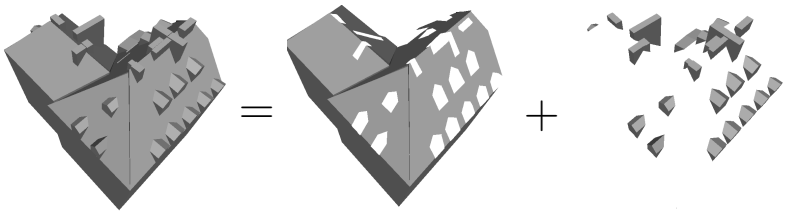


Figure 12 – Les superstructures détectées permettent de réduire les emprises des pans de toit principaux afin de ne les réestimer que sur la donnée ne correspondant pas à des superstructures détectées.

Le recalage d'un polyèdre triédral 2.5D sur un MNS est réalisé itérativement, pour chaque face indépendamment, jusqu'à convergence. La projection verticale d'une face sur le MNS permet de réestimer son plan porteur. C'est à cette étape que les superstructures précédemment détectées sont utilisées pour ne réestimer le plan porteur que sur les régions du MNS ne correspondant pas à des superstructures, réduisant ainsi le biais d'estimation. L'estimation simple proposée minimise l'erreur verticale $\mathcal{L}_p$ intégrée sur la région du MNS dans la projection de la face à réestimer réduite par les projections des différentes superstructures détectées.

Au niveau des extensions possibles de cette phase de réestimation géométrique au cours du recalage, un schéma numérique d'ordre supérieur permettrait d'obtenir des propriétés de convergence numérique améliorée. De plus, le MNS n'intervenant qu'à cette étape de réestimation des plans et non aux étapes de triédralisation et d'évolution cinétique, il est possible d'étendre facilement la méthode proposée à d'autres types de données (images brutes, LIDAR...) et de modifier la mesure d'erreur pour permettre par exemple la réestimation de la position des façades. Enfin, une certaine robustesse pourrait être réintroduite en effectuant une réestimation contrainte des plans porteurs

qui pourraient alors conserver des propriétés de symmétrie, d'horizontalité ou même des sommets non triédraux. Cette dernière extension permettrait de recaler des superstructures détectées comme un ensemble de plans contraints.

3.2 Triédralisation

La notion de triédralisation est un néologisme désignant le problème de scinder un sommet non triédral en un ensemble de sommets triédraux, en introduisant des arêtes entre ces sommets. Nous faisons le parallèle entre le problème de triédraliser un sommet et son problème dual beaucoup plus connu, celui de trianguler une face polygonale.

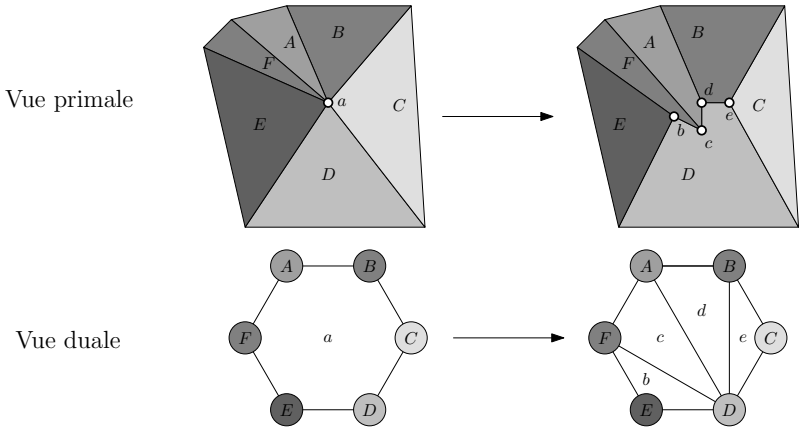


FIGURE 13 – Le problème de triédralisation, dual topologique du problème de triangulation.

Outre cette discussion, la description et l'analyse de la contrainte géométrique souhaitée (ne pas créer d'auto-intersection sur les faces adjacentes), le chapitre 5 présente plusieurs variantes d'algorithme de triédralisation qui ont chacune leur spécificité, de la plus générique à la plus spécifique :

***Winding numbers* :** Cette réduction à un problème de *winding numbers* [For97] est simple et la plus générale mais ne laisse aucun contrôle sur la solution topologique finale.

Coloration d'arrangement de plans : La triédralisation recherchée se trouve parmi l'ensemble des polyèdres généré par l'arrangement des plans réestimés. A partir du calcul de cet arrangement de plans, toute sélection d'un ensemble de ces cellules crée, dans le cas général, un polyèdre triédral. Nous introduisons la notion de décomposabilité du problème de triédralisation d'un polyèdre lorsque la triédralisation de chacun de ses sommets peut-être effectuée indépendamment. Cette propriété de décomposabilité permet de réduire la combinatoire de l'exploration des polyèdres candidats. Par rapport à l'algorithme précédent, n'exhibant qu'une solution unique, cette variante offre l'embarras du choix et nécessiterait pour être applicable dans notre contexte une définition explicite de la distance entre deux topologies.

Découpage d'oreilles : Cet algorithme n'est applicable que sur des problèmes indépendants de triédralisation de sommets. Nous proposons alors de réduire l'espace combinatoire de recherche à celui des triangulations abstraites et ainsi de minimiser la complexité topologique du résultat.

Squelette droit pondéré : Dans le cas particulier d'un sommet surcontraint à triédraliser qui présente une propriété d'extrémalité locale que nous introduisons, le problème de triédralisation se réduit à un problème du squelette droit pondéré.

Dans notre domaine d'application, la valence des sommets est petite, ce qui ne nous a pas poussé à optimiser la complexité asymptotique de cet algorithme. L'algorithme de découpage d'oreilles a

donc été implémenté pour sa relative simplicité et son adéquation aux problèmes de triédralisation indépendants rencontrés suite à des mouvements modérés de plans depuis un polyèdre non auto-intersectant. Couplé à une évolution cinétique, nous allons même voir que l'hypothèse de mouvement modéré tombe d'elle-même.

D'autre part, comme dans tout problème de géométrie algorithmique, il convient de se soucier des cas particuliers. Il s'avère que le traitement de ces dégénérescences ne peut pas être évité par une perturbation géométrique car il existe des problèmes de dégénérescence uniquement dus à la topologie donc pour lesquels une perturbation géométrique ne rompt pas la dégénérescence. Nous discutons le traitement de ces dégénérescences. D'un point de vue théorique, nous prouvons qu'il n'y a pas toujours unicité à un problème de triédralisation et l'existence même d'une solution n'est prouvée que pour un sommet adjacent à 4 plans.

Au final, la formalisation, l'analyse et la résolution de ce nouveau problème de triédralisation permet de triédraliser un polyèdre non-triédral tout en garantissant la non auto-intersection de ses faces pour des mouvement relativement restreints par rapport à une géométrie initiale non auto-intersectante. L'objectif de la section suivante est de permettre des évolutions de plan arbitraires.

3.3 Approche cinétique garantissant la simplicité du polyèdre

Le chapitre 6 permet de garantir la non auto-intersection des faces du polyèdre recalé pour des perturbations arbitraires et non plus seulement pour de petites perturbations de la géométrie des plans supportant chaque face polyédrique. Ainsi, l'approche introduite atteint l'objectif fixé : un bâtiment peut-être recalé de façon toute automatique sur un MNS par des réestimations itératives de ses plans supportant ses faces, tout en garantissant la non auto-intersection de ses faces.

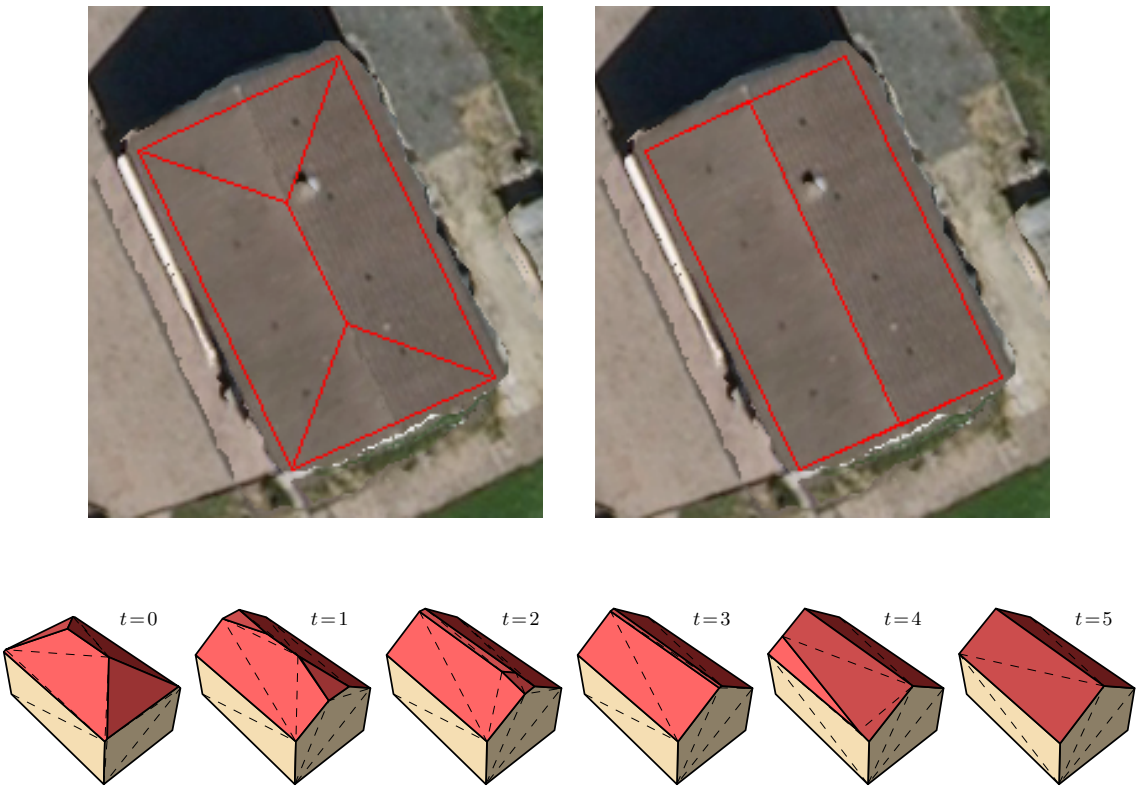


Figure 14 – Evolution cinétique itérative d'un bâtiment.

Pour maintenir cette garantie, il s'agit de modifier de façon implicite et parcimonieuse la topolo-

gie du polyèdre. Pour résoudre ce problème jusqu'alors inexistant dans la littérature, ce manuscrit introduit, présente et discute de multiples approches. Si le problème à traiter peut se réduire à celui du choix d'un coloriage d'un arrangement de plan, il est alors difficile de prendre en compte le polyèdre initial dans la définition intuitive de la modification topologique minimale. L'idée est de contourner cette définition explicite par une modification continue déterministe de la géométrie du polyèdre initial vers la géométrie estimée. Ceci permet d'isoler les problèmes d'auto-intersection et évite ainsi l'écueil des évolutions à pas constants ou adaptatifs qui ne garantissent pas l'individualisation des sous-problèmes. L'approche proposée se base sur les structures de données cinétiques [BGH97]. Cet ensemble d'algorithmes géométriques permet de maintenir une structure de données combinatoire vérifiant une propriété donnée alors que ses éléments subissent une évolution continue. Le chapitre 6 introduit la terminologie nécessaire à l'explication plus poussée de cette méthode et un exemple simple d'algorithme cinétique maintenant une liste de fonction $f_i(t)$ triée au cours du temps. L'algorithme 2D du squelette droit pondéré est un autre exemple d'algorithme cinétique.

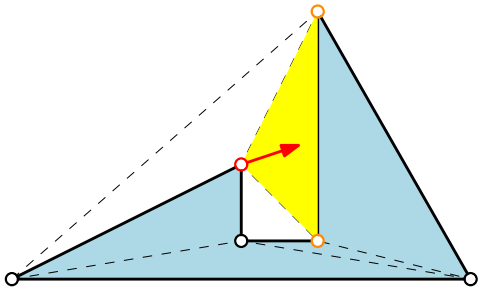


FIGURE 15 – Pour prouver que cette face polygonale reste non auto-intersectante, il suffit d'exhiber une triangulation de son enveloppe convexe qui contient toutes ses arêtes. Le mouvement illustré du sommet rouge menant à une intersection provoquera l'annulation de l'aire du triangle jaune.

Afin de garantir la non auto-intersection des faces, une preuve va être maintenue valide alors que les plans vont évoluer continuement entre leur géométrie initiale et leur géométrie estimée. Afin de prouver qu'un polygone ne s'auto-intersecte pas, il suffit d'exhiber une triangulation contrainte de son enveloppe convexe munie de triangles partageant tous la même orientation et qui contient toutes les arêtes du polygone. Nous allons donc maintenir une telle triangulation 2D pour chaque face du polyèdre à l'intérieur de son plan porteur.

Nous proposons de laisser évoluer l'ensemble des plans simultanément, suivant une interpolation linéaire de leurs coordonnées projectives normalisées. Plus intuitivement, ceci revient à translater de manière uniforme les plans qui ont été réestimé de façon parallèle et à effectuer une rotation des autres plans autour de la droite d'intersection entre le plan initial et le plan réestimé.

Les coordonnées de chaque plan variant en fonction du temps, il est possible d'exprimer la position des sommets du polyèdre en fonction du temps, par des calculs d'intersection. De même, il est possible de calculer l'aire signée de chaque triangle de la preuve de non auto-intersection afin de vérifier sa validité au cours du temps : elle reste valide tant que l'ensemble de ses triangles conserve une aire positive. Nous prouvons que l'aire signée d'un triangle est une fonction rationnelle que nous analysons afin d'optimiser le calcul de ses racines, permettant de détecter les instants (nommés événement) où la preuve doit être mise à jour. Il s'agit alors d'ordonner les racines futures par ordre croissant afin de les traiter dans l'ordre. On utilise un calcul arithmétique en précision arbitraire ce qui permet de ne pas être confronté aux problèmes d'arrondi en virgule flottante et évite donc les incohérences sur les tests géométriques, ou sur l'ordonnancement des événements.

Afin d'initialiser le mouvement des sommets, il faut préalablement triédraliser le polyèdre. Afin de conserver les garanties de non auto-intersection, cette triédralisation n'est pas effectuée par rapport à la géométrie réestimée, mais sur la géométrie initiale déplacée infinitésimalement en direction de la géométrie réestimée.

Les événements peuvent être de plusieurs types. Premièrement, un sommet peut rencontrer une

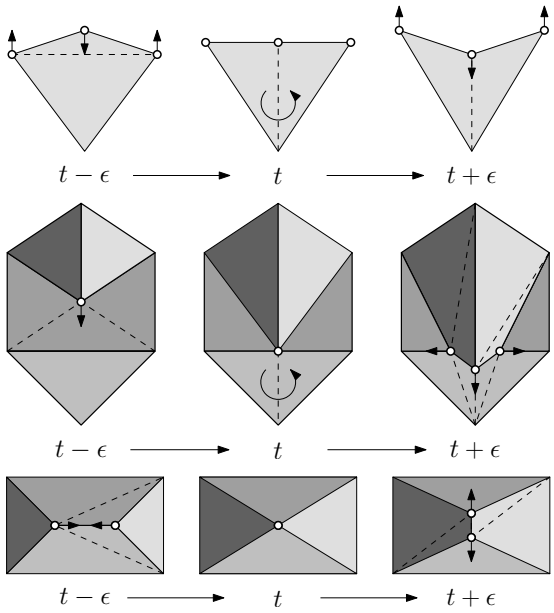


Figure 16 – Les 3 types d'évenements nécessitant une mise à jour de la preuve de non auto-intersection.

arête de triangulation qui n'est pas une arête du polyèdre. Il suffit alors de modifier localement la triangulation et de recalculer et réordonnancer les événements afférents aux nouveaux triangles dans la queue de priorité recensant les événements futurs. Deuxièmement, un ensemble d'événements simultanés peut faire disparaître complètement une face. Dans les autres cas, une collision s'opère entre un sommet du polyèdre et une arête ou un sommet du polyèdre et crée donc un point qui n'est plus triédral instantanément. Il convient alors d'effectuer sa triédralisation juste après l'événement, similairement à la triédralisation initiale, afin de maintenir la garantie de non auto-intersection des faces. Puis les triangulations sont localement mises à jour afin de recréer la preuve de non auto-intersection des faces juste après l'événement et jusqu'au prochain événement ordonnancé.

Pour chaque réestimation géométrique des équations de plan, cette évolution s'effectue donc à pas adaptatifs entre la géométrie initiale à $t = 0$ et la géométrie réestimée à $t = 1$. Les événements d'auto-intersection sont traités un par un, en n'effectuant uniquement les modifications topologiques nécessaires (triédralisation, suppression de face, collision). L'algorithme cinétique proposé permet donc de traiter des mises à jour arbitraires des coordonnées des plans supportant les faces du polyèdre et garantit, tout au long de l'évolution la non auto-intersection des faces. Le polyèdre est donc bien rendu malléable au sens où il peut-être manipulé uniquement par sa géométrie, en laissant sa topologie évoluer parcimonieusement et implicitement. Enfin, des possibilités d'extension sont discutées comme la détection d'auto-intersections globales (telles que des repliements de surface), pour laquelle il n'existe pas de preuve linéaire comme la triangulation contrainte des enveloppes convexes utilisée ici pour la non auto-intersection des faces. Il est à noter que cette partie n'est ni spécifique à notre contexte 2.5D outre la non détection des auto-intersections globales, ni spécifique au type de données utilisées pour réestimer les équations de plan.

D'un point de vue plus théorique, nous proposons ici le premier algorithme cinétique qui n'est pas uniquement justifié par la cohérence temporelle de la structure topologique maintenue. En effet, c'est grâce à cette évolution cinétique continue à changement de topologie implicite que la parcimonie des modifications topologiques a été obtenue.

FIGURE 17 – Zone test sur Amiens, France : (haut) MNS et (bas) modèles 3D de bâtiments filaires sur un fond orthophotographique.

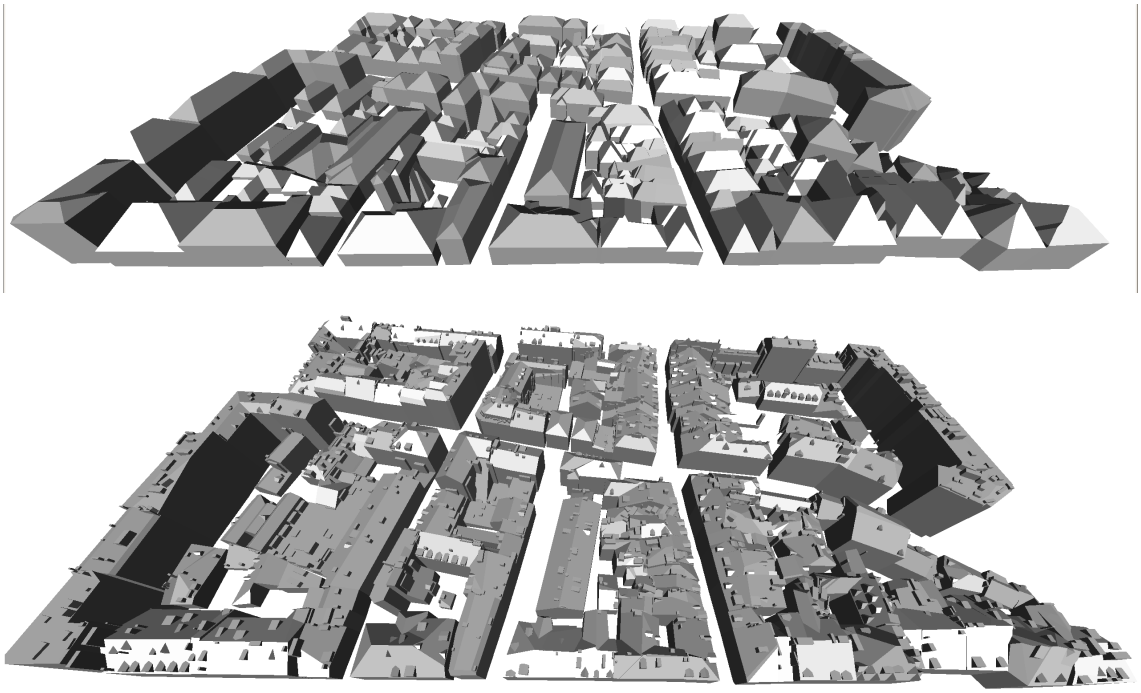


Figure 18 – (haut) Modèle 3D de ville initial et (bas) modèle final (recalage+superstructures).

4 Résultats

L'approche proposée d'optimisation alternée itérative de la géométrie et de la topologie de chaque bâtiment a été implémentée et évaluée sur une zone de centre ville (Amiens, France). Les données utilisées sont un MNS de 300m par 200m et d'une précision de 10cm généré à partir d'images aériennes de même résolution, et une base de données d'environ 300 bâtiments reconstruits automatiquement sur un cadastre saisi manuellement.

Les résultats de recalage et de reconstruction des superstructures ont été évalués qualitativement et quantitativement par rapport à une donnée de référence filaire saisie manuellement en stéréoscopie sur une paire d'images aériennes par un opérateur chevronné. La précision du recalage est évaluée par comparaison géométrique des segments représentant les arêtes faîtières. L'erreur moyenne mesurée de translation est de l'ordre de la dizaine de centimètres verticalement et de la vingtaine de centimètres horizontalement, ce qui est comparable à la précision du MNS utilisé pour le recalage. L'erreur moyenne en orientation (pente et direction) est de quelques degrés. La reconstruction des superstructures s'avère moins robuste sur cette zone de centre ville historique particulièrement difficile. En particulier, en limite de bâtiments mitoyens l'imprécision du MNS sur ces discontinuités altimétriques provoque une surdétection massive de superstructures.

5 Conclusion

Dans un contexte nouveau, où peu de travaux ont cherché à reconstruire un tel niveau de détail, nous proposons une approche toute automatique de mise à jour d'un modèle polyédrique de bâtiment s'attaquant à la fois à la détection et à la reconstruction de ses superstructures de toit, à l'amélioration de la précision géométrique de ses pans de toit principaux et à la garantie de non auto-intersection du modèle recalé. Cette distinction entre reconstruction des superstructures et recalage géométrique ainsi que leur optimisation itérative alternée constituent les principales

contributions thématiques de ce travail.

La partie recalage à modification topologique implicite a été intégrée à la plateforme de production IGN Bati3D et a permis la production d'un jeu test sur une zone d'un kilomètre carré (figure 19).

FIGURE 19 – Recalage sur une zone d'$1km^2$ dans le cadre du projet Terra Numerica du pôle de compétitivité CapDigital.

Nos principales contributions en géométrie algorithmique sont la conception d'un algorithme cinétique garantissant la non auto-intersection des facettes et la formalisation et l'analyse du nouveau problème de triédralisation, ainsi que l'implémentation d'une méthode de résolution de ce problème suivant le canevas de la triangulation par découpage d'oreilles.

Ces travaux ouvrent de nombreuses perspectives, telles que le passage du recalage d'un polyèdre 2.5D à celui d'un polyèdre avec surplomb, permettant son application dans le contexte terrestre de reconstruction de façades et dans de multiples contextes plus éloignés de l'information géographique comme la rétroconception de pièces mécaniques. La qualité de reconstruction des superstructures est suffisante pour des applications de visualisation, mais est encore trop dépendante de la qualité du MNS et de la modélisation des pans de toit principaux pour les applications de comptage. A court terme, la variabilité spatiale de la fiabilité du MNS pourrait être exploitée afin de réduire les surdétections d'objets là où le MNS et/ou le modèle de toit sont entachés d'erreurs. Il serait d'autre part possible de considérer les superstructures, au moment du recalage, comme des ensembles de plans contraints, ce qui permettrait d'affiner leur géométrie par une extension de l'approche proposée. Enfin, d'un point de vue opérationnel, il est envisagé de décliner des approches semi automatiques, en incluant des interactions utilisateur pour guider la reconstruction et le recalage : en un clic, ajout ou suppression de superstructures qui se recaleraient automatiquement, et division ou fusion d'un pan de toit afin d'opérer une modification explicite de la topologie d'un polyèdre au cours de son recalage.

Part I

Introduction

Chapter 1

Introduction

Contents

1.1 Context

A 3D City Model is a database of 3D urban objects (the ground surface, buildings, roads...) that is a digital representation of a city. Typical usages of these databases include urban and landscape planning, architectural design, tourist and leisure activities, environmental simulations, antenna placement for mobile telecommunications, disaster management, defense and security, vehicle and pedestrian navigation, training simulators and mobile robotics.

Depending on the input data and the application, their level of detail (LOD) may vary widely. Buildings may only be represented with parallelepipeds with metric accuracy, or may model in detail all the 3D facets of their roofs and façades or even their interiors, with a centimetric accuracy. Apart from the geometric and topological information, 3D City Models may also contain some semantics, explicitly labelling human understandable objects with meaningful labels such as *roof*, *window* or *chimney*. Moreover, when used in virtual reality applications, the appearance of the objects must also be described with attributes such as colors, transparencies or textures to be more visually appealing.

Two decades of research have driven down the costs of building such databases. The state of the art methods are primarily based on terrestrial, aerial and satellite imagery, and on terrestrial and airborne LIDAR (LIght Detection And Ranging). The attainable LOD of the resulting 3D city model depends directly on the richness of the input data and the time and cost allocated.

The City Geography Markup Language (CityGML) is an open data model and XML-based format for the storage and exchange of 3D City Models (fig. 1.1). It defines 5 levels of detail in the description of a building in a 3D City Model. More details can be found in [KGP05], but, as an illustration to the purpose of this work, here is a brief summary:

LOD0 The ground surface of the city only is modeled, and not the buildings.

LOD1 Buildings are modeled without details: they are rectilinear, have flat roofs.

LOD2 Metric scale 3D facets of the roofs and façades are present.

LOD3 Smaller scale features of the roofs and façades are modeled. The accuracy is centimetric.

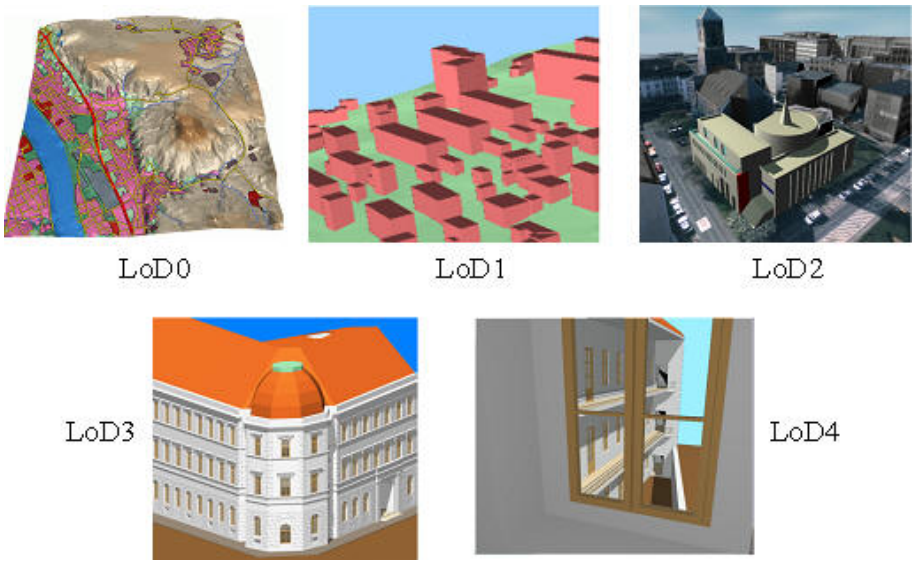


Figure 1.1: CityGML levels of details.

LOD4 Along with a better topological and geometric accuracy, the interior of the building is also modeled.

The *Institut Géographique National* (IGN) is the French national mapping agency. Among its objectives is the production and update of 3D City Models and the research towards more accurate and less expensive generation and maintenance of these databases. The databases corresponding to CityGML's LOD0 and LOD1 are mastered and already available and roughly correspond respectively to the BD-Alti and BD-Topo databases produced by the IGN. Recent works [Tai05, DT06] on building reconstruction are being industrialized by the Bati3D project which corresponds more or less to CityGML's LOD2. The context of this thesis is to continue this trend towards more accurate 3D building models by refining their roof models to the level of geometric, topological and semantic accuracy of CityGML's LOD3.

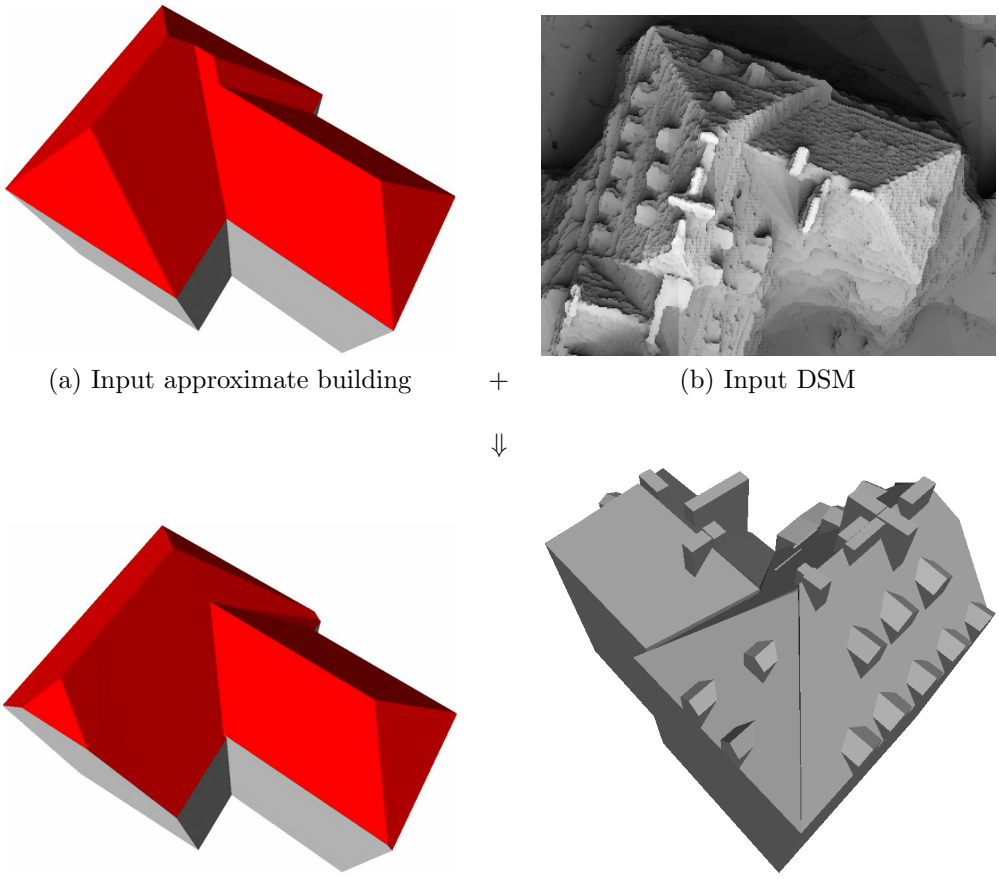
1.2 Objectives

The general objective of this thesis is to take as an input a generalized building model, which is a building model that may contain errors both in geometry and topology, and to improve it relative to an input high resolution (less than 10cm per pixel) Digital Surface Model (DSM) (fig. 1.2). The characteristics of the input data will further be detailed in chapter 2.

This thesis does not tackle the problem of reconstructing a building from scratch. The envisionned application is rather on refining existing CityGML's LOD2 databases or post-processing a building model output by any robust but less accurate reconstruction method, to reach the accuracy of CityGML's third level of detail on the roofs.

This general objective can be divided into the two following sub-objectives:

Reconstruction of missing details: Usually, small scale elements of the roofs have not been considered during the modelling of the building at a lower scale. These elements, such as chimneys, dormer windows or skylights are often omitted. The objective is thus to automatically detect and reconstruct those details, based on the input DSM. Admittedly, façade superstructures such as doors or windows are also very important, in applications such as ground level virtual reality, but they fall outside the scope of this work, primarily because of the input data: a DSM is not well suited to retrieve the overhanging façade superstructures.



(a) Input approximate building + (b) Input DSM

⇓

(c) Output refined building + (d) Output Superstructures

Figure 1.2: Context of the thesis: given an approximate building model (a) and a Digital Surface Model (shaded view b), refine the approximate building model (c) and reconstruct its superstructures.

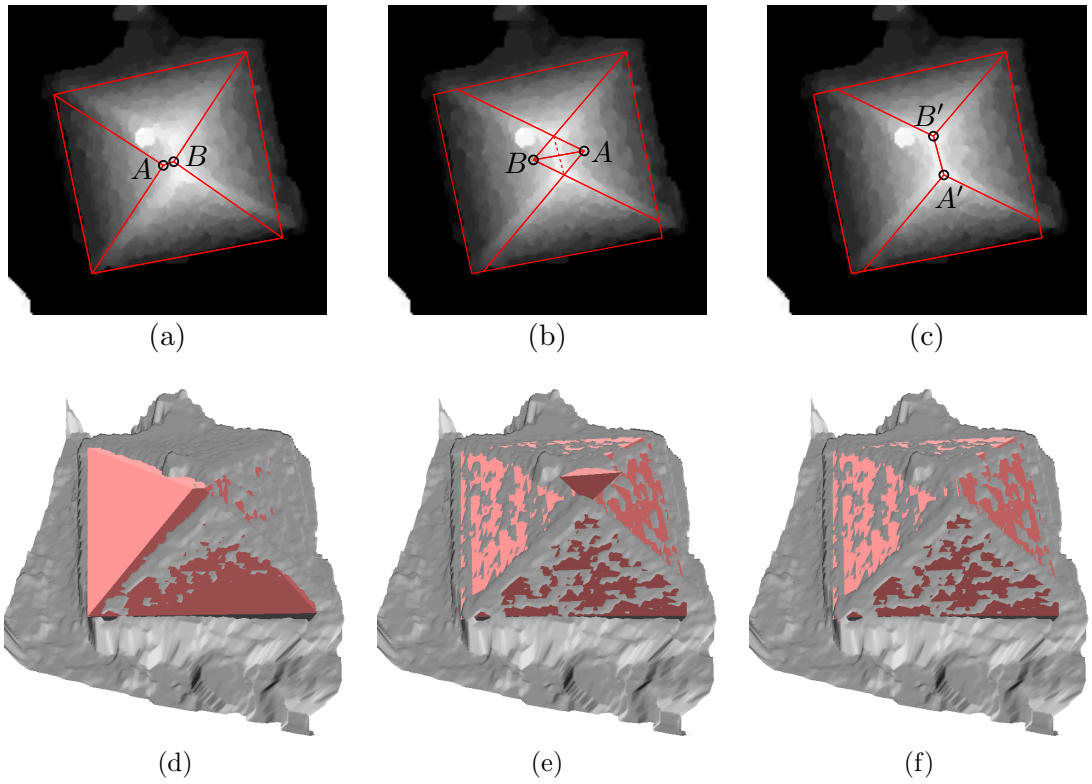
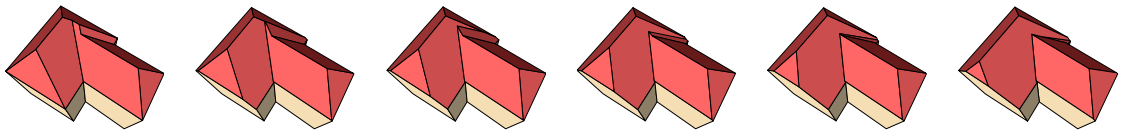


Figure 1.3: A topology-aware fitting may be required. The first row shows 3 distinct building wireframes superimposed on the same DSM image (the brighter, the higher). The second row is a 3D version of the first row: it shows 3 identical DSM surfaces together with 3 distinct building models. The input building model (a-d) has an erroneous topology. Its rooftop edge AB should be topologically flipped to $A'B'$ in order to separate the left and right facets rather than the top and bottom ones. Without modifying the topology of the input building model (a-d), its geometric optimization relative to the DSM will produce a reversed tetrahedron floating above the roof top (b-d). The proposed kinetic framework detects this inversion and modifies implicitely the topology (c-f).

Refining the geometry while guaranteeing a well-formed model: Being a generalization, the input building model is not guaranteed to be exact in geometry (the point coordinates, the face normals...) or even in topology (the combinatorial information: the number of facets and vertices, the number of facets adjacent to a vertex...). The input building may have been reconstructed automatically or manually with many constraints: horizontal roof tops, symmetric roof slopes, edges longer than a predefined threshold... Those constraints are typically used to ensure robustness but they may prevent the building to fit tightly the real-world building or more precisely the observation of the real-world building through the acquired data. Moreover, the input model have usually been created based on less accurate data. The consideration of the high resolution DSM will allow the refinement of the roof geometry. This is typically performed in the literature [VT05] under the assumption that the topology is not to be altered. However even small modifications of the roof plane equations may lead to artefacts in the modeled polyhedron if its topology is not updated, as illustrated in Figure 1.3. The objective is thus to allow the fitting of the geometry while preserving a well shaped 3D Building Model without those artefacts.



(a) (b) (c) (d) (e) (f)

Figure 1.4: The kinetic framework (chapter 6) uses the continuous evolution (a) to (f) to enable a geometric refinement of the base building that applies implicitely the necessary topology modifications. The topological modification of chapter 5 lifted the topological constraint that the front left triangular roof facet was anchored at the same gutter height than the rest of the building (a) so that it can better fit the input DSM in (f).

1.3 Proposed Approach

Chapter 2 presents in more details the input data and the issues involved. It also provides an overview of recent works that are related to our problem.

The optimization of the two formerly stated sub-objectives are intimately linked. For instance, when fitting a 3D Building Model, ignoring the presence of the roof superstructures and treating them as noise will inevitably bias the estimation of the roofs. Likewise, if the geometry and topology of the building without superstructures are too erroneous, the search for superstructures, which are relatively small modifications of the base building, will rather try to compensate the errors in the roof planes rather than reconstructing the actual superstructures.

However, the simultaneous optimization of the two sub-objectives is a difficult problem. Therefore, we have chosen a suboptimal iterative method that alternates between the optimization of each of them, as detailed in chapter 2 :

Addition of superstructures given a fixed roof: The content of chapter 3 is a method to detect and reconstruct roof superstructures on a fixed building based on a DSM. Given the resolution of the DSM and the size of the reconstructed objects, the approach is based on a collection of parametric superstructure types to overcome issues with inaccurate or missing data. These parametric superstructure objects are instantiated to modify locally the geometry of the fixed building, similar to Constructive Solid Geometry (CSG) approaches.

Topology-aware fitting of the geometry: Chapters 4, 5 and 6 expose the fitting to the DSM of the geometry of a 3D Building Model without superstructures. DSM regions corresponding to previously detected superstructures are not taken into account during the estimation of the roof plane equations, preventing them from biasing the estimation. The topology of the building is allowed to change to enhance the fitting in two ways. First, the initial building topology may impose geometric constraints. For instance, when a vertex is adjacent to more than three planes, its adjacent plane equations are constrained to have a non empty intersection. The first topological change, detailed in chapter 5, is to split those extraordinary vertices into vertices adjacent to only 3 facets by introducing small edges or even edges of null length. This process is illustrated in the first row of figure 1.3. The roof edges were initially meeting the façades at their corner, the topological change of chapter 5 allowed to relax this constraint to let them move freely. These topological modifications prevent the topology from constraining the fitting of the geometry, and thus enable the use of the unconstrained building fitting described in chapter 4. The second type of topological change is necessary to keep a well shaped polyhedral model throughout the optimization process. Chapter 6 introduces a kinetic framework to detect and process events where artefacts occur during the fitting (Figure 1.4). For instance, translating a roof facet along its normal may cause the building model to no longer be simple.

Finally, chapter 7 will present the results achieved by the overall proposed approach and their evaluation, before concluding in chapter 8.

Chapter 2

Background and Related Work

Contents

2.1 Introduction

This chapter gives a more extensive background, in order to more clearly define the proposed approach that addresses the stated objective of increasing the LOD of existing 3D building models using very high resolution aerial data.

First, section 2.2 describes the raster input: the real world observations given by very high resolution aerial data. Section 2.3 defines the vector data: the 3D building models that will be modified to increase their LOD. Thus, the building vector data is, within our context, both an input and an output. Then, section 2.4 presents an overview of the previous work in building reconstruction. Finally, section 2.5 details the approach that is proposed in this thesis.

2.2 Aerial Raster Data

Large scale 3D City models are curently generated using sensors carried aboard planes and satellites. Satellite sensors are not current sufficiently accurate to reach the reconstruction level of

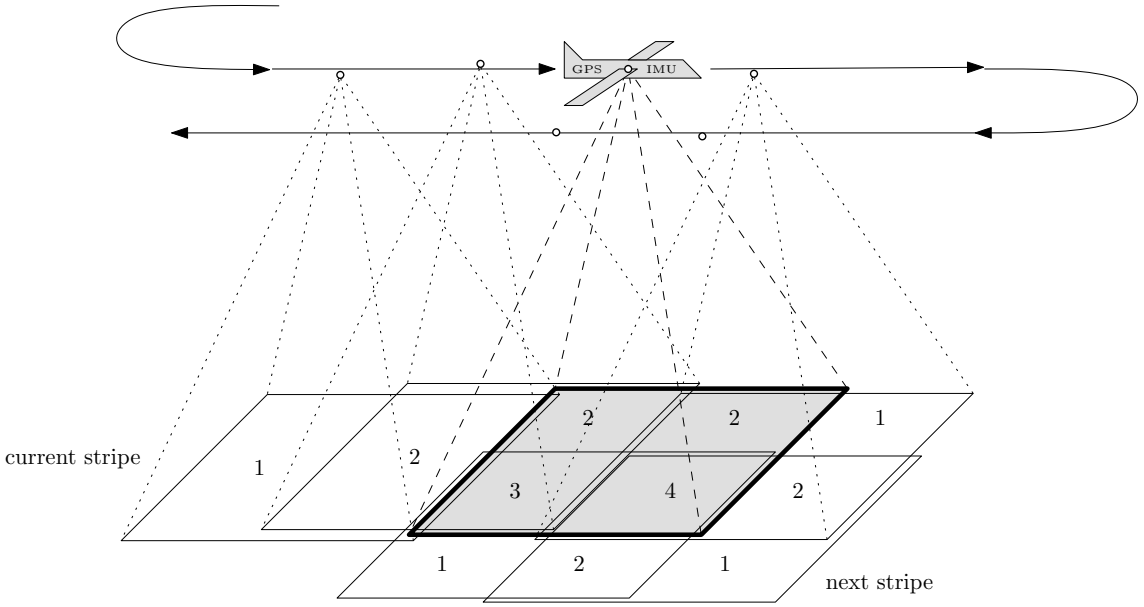


Figure 2.1: Multiview Aerial Imagery. Figures illustrate the number of photos that view the corresponding regions from different airplane locations.

detail aimed by this work (in satellite images, current approximate pixel squares projected on the ground surface mesure around 50cm by 50cm). At a lower scale, drones are also equipped of sensors to collect data at much higher resolutions. Research on this subject is active, but currently, they may not reliably survey large scale cities. On the other hand, specially equiped ground vehicules are used to complement aerial data at higher resolutions or simply to acquire data that was occluded or seen at grazing angles from aerial vehicles such as the façades, but are not relevant to model the roof structures. To reach the coverage and accuracy needed by our objectives, aerial data must then currently be used. We focus here on two types of sensors, namely images and lidar, considering that radar data does not meet our accuracy requirements.

2.2.1 Multiview Aerial Imagery

To acquire aerial images, planes are equiped with nadir (*i.e.* downwards facing) digital cameras that take pictures of the ground along the track of the flight. Thus, such a plane takes multiple images from a strip of the city. To survey an entire city, the plane flies over the city multiple times such that these stripes cover the whole city. The special purpose digital cameras used in survey airplanes currently have an approximate projected resolution on the ground of around 10cm (the Ground Sample Distance, GSD), compared to a GSD around 50cm for satellite imagery. To go a step further, all the points on the ground are guaranteed to be imaged from multiple plane positions, rather than a single one, by overlapping the stripes and taking photos with enough overlap along the track. This allows reconstruction techniques to be applied to recover the shape of the scene from the stereo or multi-view data, as in the binocular human vision system.

Another specificity of this aerial surveying is that the airplanes carries a Global Positioning System (GPS) and an Inertial Measurement Unit (IMU). This gives an approximate position and orientation of the plane and hence of the digital cameras, for each of the photographs taken. The digital cameras are calibrated precisely intrinsically and the camera poses are refined from the initial GPS and IMU guesses using a bundle adjustement method [TMHF00], that makes use of the multiview stereo. The post-processed GPS position accuracy is currently about 50cm, which is sufficient to yield an initial guess that greatly helps avoiding convergence to undesired minima of the bundle adjustement minimization.

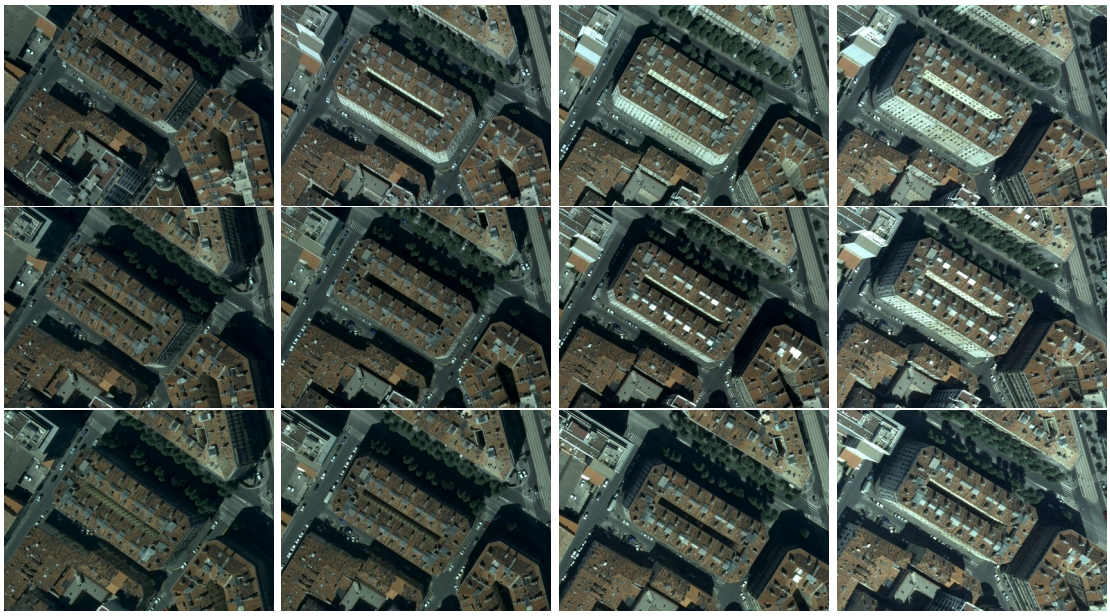


Figure 2.2: 1186x880 convergent image crops showing the same building viewed from 12 different airplane locations. Each uncropped image has 13824x7680 pixels.

The bundle adjustement constraints are either of relative or absolute nature. Within each image, feature points are selected manually or detected automatically, using for instance Scale-Invariant Feature Transform (SIFT) keypoints [Low04]. The relative constraints are given by point correspondences between these feature points across images. These correspondences may be input manually or automatically. The absolute constraints are optional and result from manually pointing in some images a few easily distinguishable georeferenced ground landmarks. This enables computing the successive positions of the plane in an absolute coordinate system relative to the earth, a process known as *georeferencing* the images.

This enables the use of photogrammetric techniques, which are using photographs to perform measurements, as the cameras are fully calibrated (*i.e.* both intriscally and extrinsically): each pixel of each aerial image may be associated with the georeferenced 3D ray along which the pixel gathered its color. More details on the field of digital photogrammetry may be found in [KE02].

Correlation-based Digital Surface Model The literature on shape from stereo (or multiview) is vast. An operational by-product of a set of calibrated aerial images is a Digital Surface Model (DSM), which is a grey-level image, coding a height map (figure 2.3). Sampled on a regular horizontal grid, each pixel value measures the altitude of the scene surface at each grid node. A DSM thus represents a 2.5D surface (*i.e.* without overhead). The most operational approach to generate a DSM is a graph-cut based reconstruction technique [RC98]. This approach has been implemented and is used in production at *IGN* [PDP06, MICMAC]. A competing state of the art approach is based on Semi-Global Matching using Mutual Information [Hir08].

The main advantages of this approach are the coverage and the density of the reconstructed DSM. The DSM can be reconstructed as soon as stereo information is available, and it makes sense to use the approximate image ground pixel size as the grid resolution of the DSM. Given the baseline/depth ratios used in production at *IGN*, urban DSMs are thus currently produced with both an altimetric (*i.e.* vertical) and planimetric (*i.e.* horizontal) accuracy of around 10cm, matching the image GSD. The main disavantages of correlation-based DSMs are their smearing behavior around height discontinuities such as around the façades, and that they may fail or find

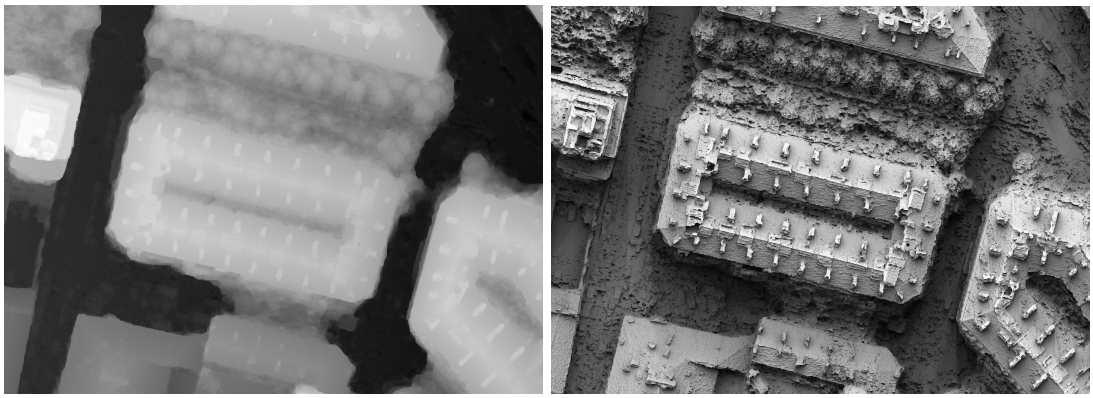


Figure 2.3: DSM generated by correlating images of figure 2.2. (left) Raw view of the height image (the higher, the brighter). (right) Shaded view of the resulting surface using ambient occlusion.

false correspondences when the scene texture is homogeneous or periodic.

2.2.2 Lidar Data

Light Detection And Ranging (Lidar) is an active optical remote sensing technique that principaly measures distances. This sensor beams a laser pulse and uses its time of flight to measure ranges. By varying the direction of the laser pulse across the flight track as the plane flies forward, the Lidar sensor acquires data along a stripe, similar to the image sensors.

Lidar Point Cloud The GPS and IMU systems are used to measure the position and orientation of the plane and then map the range information to a set of 3D points, the Lidar point cloud. Simply put, for each backscattered laser pulse emitted in a given direction with respect to the airplane, the range measurement yields a 3D point at the given distance of the known position of the aircraft in the known direction of the laser pulse. Typical caracteristics of aerial lidar point clouds is an altimetric (*i.e.* vertical) precision of 5cm, and 4 points per squared meters and per overlapping stripe. The planimetric (*i.e.* horizontal) precision is however typically around 40cm, for two reasons. First, contrary to the aerial imagery, it is much more difficult to refine the geopositionning of the airplane, because the laser pulses are sent sequentially, so that the plane has moved between two lidar points, whereas the pixel measure of an image are performed simultaneously. To overcome this, and to increase the density of points per squared meter, the scene is surveyed using multiple overlapping stripes, so that the redundant acquired information is used to refine the trajectory of the plane. Second, the range is not a pointwise measure, it integrates the range of the surface within a cone. The radius of the circular intersection with the ground is typically 40cm. Therefore, the reconstructed point, which is placed in the middle of the cone at the measured distance, is a somewhat integrated view of the real surface.

Lidar Digital Surface Model It is sometimes convenient to resample the lidar point cloud on a grid as a 2.5D surface. This produces a DSM image, as sensed by the lidar sensor, rather than the one produced using multiview stereo techniques. This has the advantage, compared to the relatively unorganized point cloud, that it has a fixed implicit topology given by the pixel adjacencies in the image. This resampling as a DSM allows most approaches relying on correlation-based DSMs, such as our proposed approach, to be applied to Lidar data without any modification.

2.2.3 Digital Surface Models

Digital Surface Models (DSM) may be generated by post-processing the data of various acquisition systems. This allows a unified treatment of these data by considering their DSM only. How does a Lidar DSM compare to a Correlation-based DSM? Broadly speaking, the Lidar DSM is more accurate vertically, and does not suffer from the height discontinuities, the lack of non-periodic texture or specular effects. However, it is much more expensive to achieve the coverage, the sampling density, and the pose estimation accuracy of a correlation-based DSM.

Dealing directly with a DSM without taking the acquired images or lidar datasets is a trade-off that will be taken in this work. This trade-off balances the easy manipulation of the DSM as a regularly sampled 3D surface with the artefacts introduced while generating it from the acquired data. Thus, this choice on the scope of this work has been made to simplify the tackled problem at the cost of the reconstruction accuracy. However, more accurate approaches that use directly the images or lidar data will be discussed as extensions in sections 3.6.2 and 4.6.3.

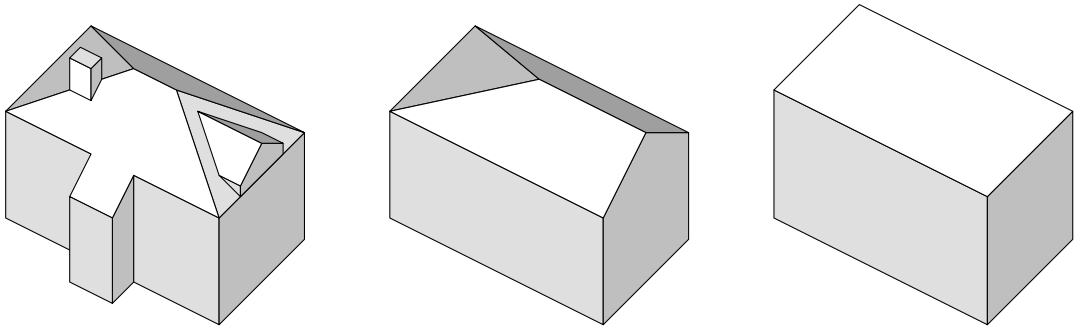
2.3 Vector Data: 3D Building Models

2.3.1 Polyhedral Building Models

3D city models are stored as databases of objects. Each such object contains various attributes, including its 3D geometry and its topology. These objects may model the roads, the buildings, the vegetation, the ground... We focus here on the building models. Man-made objects and specially buildings have fairly restricted properties on their shapes. We do not target the reconstruction or refinement of buildings of special interest, which are relatively few compared to rest of the buildings, and the architecture of which is more complex, and which are curently too complex to reconstruct automatically from aerial data. Apart from some of these buildings, the vast majority of the buildings may be relatively accurately described using only a few simple parametric surfaces (planes, spherical domes, cylinders...). We restrict here these surfaces to be planes, which can be achieved by tesselating the more complex surfaces. However, this decomposition of non-planar surfaces into planes is arbitrary as it conveys no semantics on the building surface. To prevent this artificial tesselation, some model-based building reconstruction approaches allow specific types of non-planar surfaces such as elliptic roofs [LDZPD10]. However, this assumption of piecewise-planar buildings does not appear to be too restrictive in practice. This leads the current 3D city models to model buildings as polyhedra.

2.3.2 Generalization

A 3D building model is, by definition, only a modelized view of the real world. It can thus not expect to be an exact representation. Therefore, there must be a balance between the complexity of the representation of the real world and its accuracy (figure 2.4). This means that, for instance, there is no perfect polyhedral building representation. A building may be represented at various levels of details from a prismatic extrusion of its vertical footprint with a horizontal plane, as in [LN98], up to higher levels of detail with higher geometric accuracy, which model much more topological features of its polyhedral surface, such as all the individual roof planes, their superstructures, the doors and windows, the gutter of roof overhead... Given the resolution of our input data, the roof tiles, for instance, will not be described individually. Building surface features smaller than about 10cm will be generalized and hence discarded or agglomerated in the building polyhedral representation. The typical databases that we are targeting to refine the geometry and topology of their buildings present features that have a dimension of at least 1m, even if their accuracy of the modeled features is much better. These are the city models that are created routinely nowadays. The roof elements that are typically missing in such databases are what we



(a) (b) (c)

Figure 2.4: Building model generalization: these building models represent the same building at various levels of detail, from the detailed model (a) featuring roof superstructures and a non-rectangular footprint to a generalized model (b) down to the box shaped model (c).

call roof **superstructures**. The representation of these roof superstructures will be discussed in chapter 3.

2.4 Building Reconstruction

The various building reconstruction approaches that have been developped may be analyzed using multiple criteria: the type of input data, the reconstruction strategy and the degree of user interaction.

2.4.1 Input Data

The type of input data determines what is reasonable to reconstruct. Building reconstruction approaches typically deal with aerial, satellite or terrestrial imagery (monocular, stereo or multiview), lidar data, a DSM, vector databases, or combinations of them.

Approaches based only on a **single aerial image**, such as [LN98, KP09], have to constrain the buildings to remain robust without the more direct 3D cues of stereo or multivue images, or of lidar data. In [LN98], buildings are limited to simple prismatic building models laid on a flat ground. In [KP09], a limited library of building footprint templates is introduced, and matched against the single panchromatic image. **Single terrestrial image** building reconstruction approaches only focus on the façades [KTS+09, MZWVG07], as the building roofs are usually merely visible. These monocular settings face difficult ambiguities that may be solved by **Multiview imagery** or **3D lidar data**.

[BR06b] combines the high image sampling and the robust accuracy of lidar to segment and reconstruct planar roof patches. A complete building reconstruction system fusing both image and lidar datasets seems promising. However, such a system has not been published yet.

Other approaches are based on a **single DSM only** [EAH08, LDZPD10]. We saw that a DSM is computed from image or lidar data. Basing reconstructions on the DSM only and discarding its input image or lidar data is a process that may convey a deformed view of the reality due to an intermediate optimization. It however provides a regularly sampled 3D surface which does not present the ambiguities of image correlations. Its main advantage is to enable metric measurements relative to a building model that are both unambiguous and inexpensive to compute.

There has not been much work on using **jointly terrestrial and aerial data**, since [FZ03]. The difficulties arising from the huge change of scale between aerial and terrestrial data usually

lead to perform an aerial reconstruction that is subsequently refined with the terrestrial data. One such difficulty is the relative pose estimations between terrestrial and aerial images. In urban contexts, detecting road marks in both aerial images [TP09] and terrestrial images [Soh08] yields tie points to successfully estimate the relative image poses [TSP06]. Using jointly terrestrial and aerial datasets would however lead to a more accurate and consistent reconstruction.

Prior to building reconstruction, another difficult problem is building detection and focalisation. Operational approaches rely either on user input or on a **2D vector database** to get the building footprint [DT06]. Without such a 2D database, building footprints may be extracted from a DSM using, as in the stochastic approach [ODZ07]. In our case, the building focalisation is given by the input 3D database of coarse buildings, however the detection problem will appear at the superstructure level.

2.4.2 Degree of User Interaction

Semi-automatic approaches rely to some extent on operator input [FM05, DT06, KZG06]. This human intervention is necessary in some contexts to introduce the prior of human knowledge where it is not easily expressible algorithmically, in order to solve ambiguities. For instance, in [DT06] the operator provides a simple high level hint about the general shape of each building, through the number of hipped or gabled roof terminations.

By contrast, fully automatic approaches are attractive since they may be performed off-line [JPDPM00, SMG02, SV04, Tai05], thereby drastically reducing the production costs. However, their robustness depends heavily on the quality of the input data and on the adopted strategy.

2.4.3 Strategies

Building reconstruction strategies may caricaturally be decomposed into parametric and generic approaches.

Parametric - Top/Down - Model-based When the input data is not sufficient by itself or too ambiguous to recover the shape of a building, prior knowledge must be introduced to provide a robust reconstruction that meets the expectations of the final user. This knowledge is generally introduced by restricting the class of allowable building reconstructions. This has been done by imposing that the roof is a single flat facet (prismatic models) [LN98], or that the building is the union of simple parametric models, out of a library of possible models (gabled roof, hipped roof, flat roof...) [LDZPD10]. More complex models are given by a grammar of shapes [MZB$^+$08] or a CSG approach

These model-based approaches are often called top/down approaches as they explore the search space by trying to fit constrained building models to the data rather than constructing the building models from features detected in the data. The top/down approaches have a high imaginative power as they can hallucinate missing data thanks to the a priori knowledge they introduced. They are however strongly limited by the restricted set of building that may be reconstructed.

Generic - Bottom/Up - Feature-based On the other hand, if the input data can be trusted to provide all the necessary pieces of information for a robust reconstruction, little a priori knowledge has to be introduced. Then a generic approach starts by inspecting the raw data to detect features such as 3D segments [BZ00, TD02] and planar patches for almost all feature-based approaches. Then it raises its level of abstraction by combining these features to create a polyhedral surface. These generic approaches are often called bottom/up due to this constructive or aggregative approach. These approaches have a high descriptive power as they tend to assume few restrictions on the reconstructible buildings. However, they fail if the detected features are not sufficient to recover

the geometry of the building. This failure may translate into a surface with holes corresponding to undetected planar patches [EAH08]. Alternatively, it may translate into an unreallistic surfaces, if they are closed using the available detected features only. All in all, these approaches are generally accurate but their robustness suffers greatly from feature underdetections.

2.5 Proposed Approach

First of all, this work aims at designing a fully automatic approach. Since we cannot rely on any user input, a primary concern is to process robustly the input data. The two interrelated objectives of refining the roof plane geometry and adding roof superstructures cannot rely on the input data in the same way. The roof refinement has plenty of data at hand and thus each roof plane may be optimized freely without fearing robustness issues. On the other hand, the roof superstructures have a small footprint in the input DSM. Moreover the DSM acuracy is much worse around their small discontinuities.

The disparity of the information available in the data for the 2 objectives led us to treat them individually with different approaches: Chapter 3 introduces a library of parametric superstructures to reconstruct them using a top/down approach, whereas chapters 4, 5 and 6 model generically as the building as an relatively unconstrained polyhedron. The input building polyhedron to be refined are used as a hint to turn its re-estimated planes into a new polyhedron, making it a bottom/up process.

2.5.1 Iterative Optimization

The proposed approach to optimize both the roof superstructures and the roof plane geometries is to alternately optimize one while keeping the other fixed. Figure 2.5 summarizes the overall approach. The input polyhedral building is used to initialize what the algorithm maintain, a polyhedral building modified by a set of parametric roof superstructures. Initially no superstructures are present. Then the topology-aware geometry refinement step (part III) fits the planes of the input polyhedron to the input DSMs while keeping its facets simple and relaxing various geometric constraints. The second block (part II) discards any previously reconstructed superstructures and reconstructs new superstructures given the fixed building polyhedron that has just been updated. After one such iteration, a polyhedral building with superstructures has been generated, so the process may terminate. However, a few iterations are likely to improve the results by feeding back to the roof plane estimation step the classification of DSM pixels as superstructures. This prevents the use of these outliers in the plane estimations, and thus reduces the plane estimation bias. Then, after updating the building polyhedron with more accurate planes, superstructures will be better reconstructed.

This alternate optimization of these 2 aspects of the building polyhedron with superstructures is admittedly suboptimal. However, chapter 7 will show that it makes the problem tractable and that it works well in practice. Chapter 7 will further illustrate that usually, only a few iterations are needed before the process converges.

2.5.2 Superstructure Reconstruction

The proposed reconstruction of small missing topological features is detailed in chapter 3. The proposed approach consider the base polyhedral building fixed and, using a top/down approach, matches parametric superstructure models to explain the disparity between the input DSM and the fixed base building polyhedron.

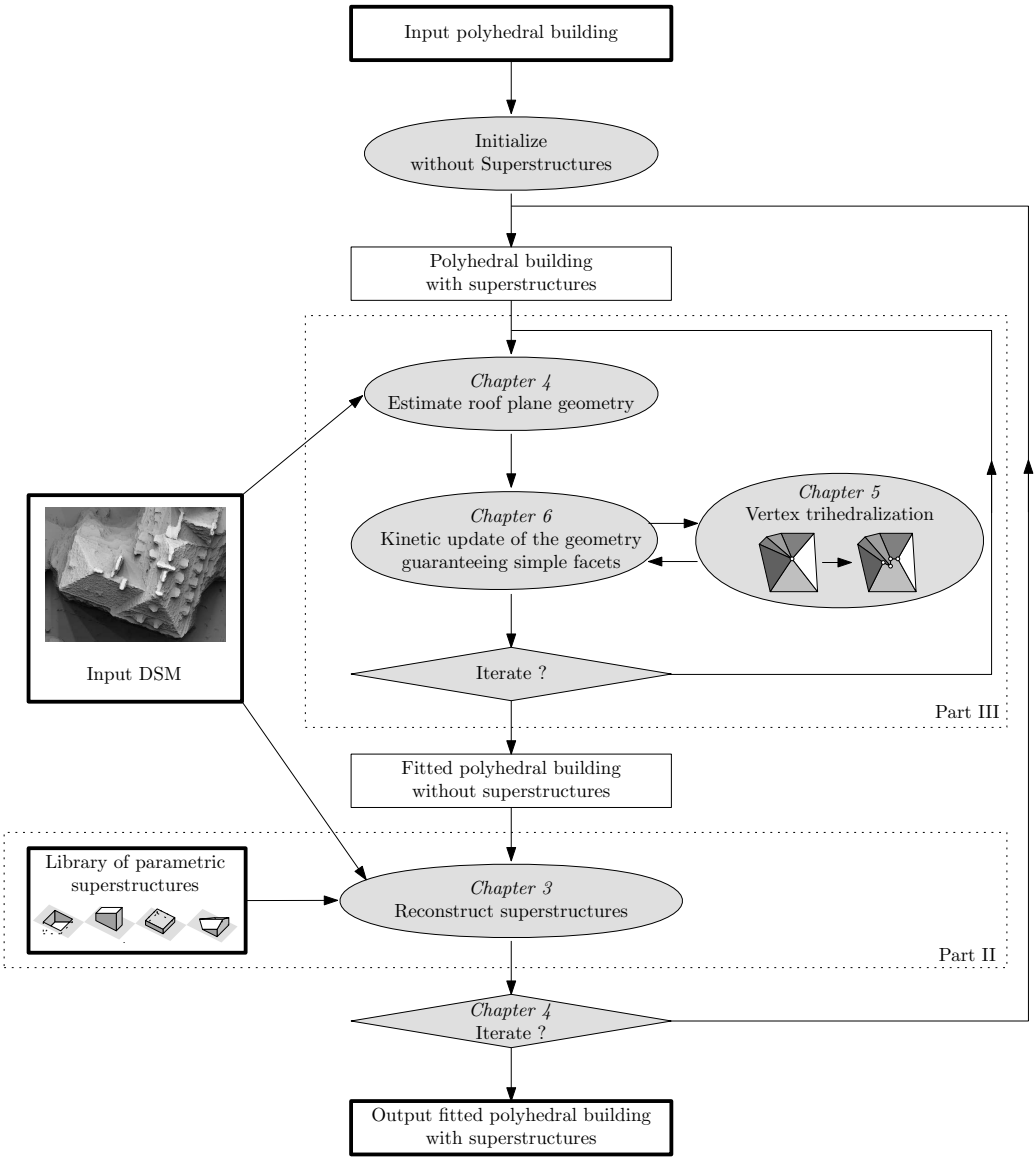


Figure 2.5: Overview of the proposed approach.

2.5.3 Topology-aware Geometry refinement

Alternating with superstructure reconstructions, the base polyhedron is optimized considering these previously reconstructed superstructures fixed. If the fitted polyhedron remains combinatorially identical (*i.e.* its topology is kept fixed), its fitting has in general to be carried out as a constrained optimization, to ensure that the vertex points of a facet remain coplanar and that facet planes adjacent to a vertex have a non-empty intersection. These constraints, denoted the **topology-induced constraints**, may either be included in the energy and thus be only approximately verified or be inherently part of the parameterization of the problem and thus exactly enforced, as in [VT05].

Chapter 4 describes a simple fitting approach for polyhedral topologies that disregards these constraints. Its main contribution is to use the superstructure previously reconstructed to only estimate the plane equations using DSM points that have not been classified as superstructures. This estimation provides, for each facet of the polyhedron a new supporting plane that improves the fit, *i.e.* an updated plane geometry of the polyhedron.

Using the plane estimates described in chapter 4, chapters 5 and 6 perform the necessary topological modifications to keep well-defined vertices and self-intersection free facets. Since the plane estimation is unconstrained, vertices adjacent to more than 3 planes are likely to not have a well-defined point location. Chapter 5 details the process of splitting these vertices into vertices with well-defined point locations, which we define as a **trihedralization**. Chapter 6 introduces a new kinetic approach to interpolate between the input and the fitted polyhedral plane geometry, while guaranteeing that its facets remain simple. Events during the interpolation where facets become self-intersecting are detected and handled using the trihedralization problem of chapter 5.

Putting everything together, the building model undergoes a continuous morphing from the input building model to a better fitting one. Chapter 4 takes advantage of the detected superstructures to provide the geometric evolution of the roof supporting planes, unaware of any topology-induced constraint. To be able to start the morphing, chapter 5 removes the initial topology-induced constraints and guarantees that the polyhedron is not immediately self-intersecting. Finally, chapter 6 ensures that, throughout the morphing, the building polyhedron keeps well-defined vertices and self-intersection free facets.

2.6 Conclusion

This chapter has described the kind of input data used in our work, namely aerial DSMs and 3D building models. Then our objectives were analyzed through a review of recent works on building reconstruction. Finally, the proposed overall approach has been detailed, alternating the reconstruction of the superstructures (part II) and the fitting of the roof planes (part III) as an iterative process.

Part II

Automatic Roof Superstructure Reconstruction

Chapter 3

Automatic Roof Superstructure Reconstruction

Contents

3.1 Introduction

3.1.1 Context

A superstructure is defined as an element of a building that is only relevant at high levels of detail (LOD). It is a small scale feature relative to the whole building. Concerning roof superstructures, it can be a chimney, a dormer window, some glass roof or skylights, a roof terrace, an air conditioner placed on the roof, an antenna, pipes on the top of a factory... Since the input data is given as a 2.5D

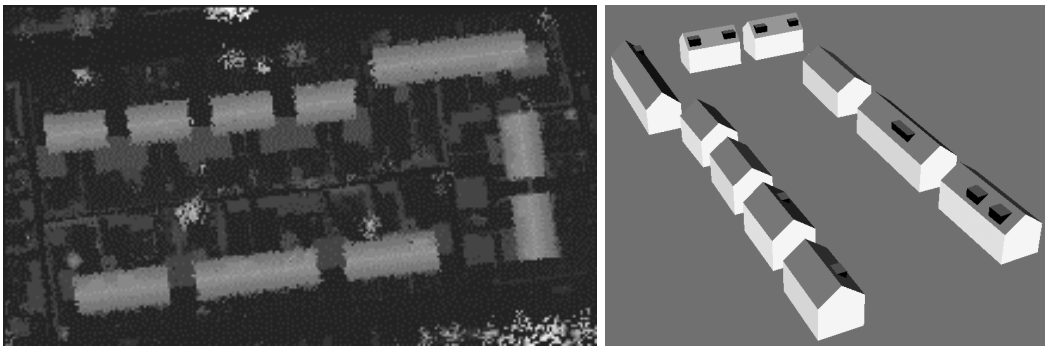


Figure 3.1: Input lidar DSM and buildings with roof superstructures reconstructed in [MV99].

DSM, the present work does not and cannot address the reconstruction of façade superstructures like windows, doors, balconies, arches, door staircases... This chapter assumes that a fairly accurate polyhedral representation of the building without superstructures is already at hand.

3.1.2 Related Work

3.1.2.1 Roof Superstructure Reconstruction

To our knowledge, [MV99] is the first work to explicitely reconstruct the roof superstructures. They proposed a single type of superstructure: dormer windows with a horizontal rectangular roof. However, they mentioned dormer windows with a gable roof as an extension. Their approach is based on moments of the laser point cloud and is hierarchical. The first pass reconstructs the base buildings and the second pass detects the sufficiently large outlier regions and fits the 4 parameters of their dormer windows to each of these regions. The input lidar point cloud only contained around 20 measured points per superstructure. Thus, the reconstruction of smaller superstructures like chimneys was unrealistic and the number of parameters of the reconstructed dormer was to remain low (Fig. 3.1).

[Nan06] tackles the detection and scale-free reconstruction of chimney-like cuboidal superstructures out of a single aerial image. The outline of the process is as follows. First, a hierarchichal segmentation of the aerial image yields image region hypotheses for superstructure facets and the shadow region projected on the roof. These regions are then grouped together to find associations of 3 visible superstructure facet regions and their shadow region that are meaningful both radiometrically and geometrically. The reconstruction is then performed for each such group of image regions up to the unknown scaling due to the monocular setting. This method performs particularly well in a restricted setting where the superstructures are bright on a dark roof with clearly distinguishable facets. It further requires that the shadow is entirely cast on a single roof plane, possibly preventing a robust detection of superstructures near a gutter of roof or with a shadow spreading over multiple roof planes.

In [LDZPD10], a model-based building reconstruction approach is applied to the reconstruction of superstructures. However, even if this method is automatic, it relies on unchallenged input 2D building (and superstructure) footprints. The results are convincing but only address the easier altimetric reconstruction problem, rather than the difficult superstructure detection and planimetric delineation problems, which are performed manually.

Apart from [BBPDM07], an extension of which is presented in this chapter, and [DB08] presented in appendix B, we are not aware of another work directly focusing on reconstructing roof superstructures. However, with the increasing resolution of the input data, recent papers, such as [EAH08], that use a bottom-up approach are able to detect and often reconstruct the large superstructures. They are however based on detecting features such as the superstructure planes.

Therefore they fail to detect all the facet planes necessary to perform a faithful reconstruction.

3.1.2.2 Building Reconstruction from Satellite Imagery

The detection and reconstruction of roof superstructures from very high resolution aerial data (10cm per pixel) is a problem that share many similarities with the detection and reconstruction of whole buildings from satellite imagery. Whereas the availability of very high resolution aerial imagery is relatively new and thus dealing with roof superstructures is a new field of research, there has been much work on the reconstruction of building from satellite imagery, as discussed in section 2.4.

One distinction however is that, whereas the ground is considered to be fairly smooth and flat when reconstructing buildings from satellite data, this assumption no longer holds for roof superstructures. Superstructures are supported by the base roof planes that may be themselves difficult to reconstruct, and these roofs feature discontinuities and slope changes. Furthermore, the superstructure geometries themselves are higly correlated with their supporting roof planes. This means that trying to reconstruct superstructures on an erroneous base roof is likely to give poor results.

3.1.2.3 Façade Reconstruction

Reconstruction roof superstructures is a problem that appears related to façade element reconstruction. In both cases, it involves reconstructing fine details that modify the planar approximation of the base roof plane or of the wall façade.

It appears that the façade elements present much more variations than roof elements but tend, in general, to be more organized. The presence of variations is especially true for elements of the first floor, which include store windows. However, whereas façades have an intrinsic orientation given by horizontal and vertical lines, it is more ambiguous to assign an orientation to a roof facet. All the façade superstructures are oriented along those horizontal and vertical directions, which reduces greatly the search space. On the other hand, roof superstructures tend to be aligned with interior and exterior walls of the building. In rectilinear buildings we can reasonably assume that the interior walls are along one of the two orthogonal directions of the façades. However there may be some ambiguities in non-rectilinear buildings. Furthermore, the prior that elements tend to be aligned and to have equal dimensions is stronger on the façade elements than on the roof elements. All in all, compared to roofs, façades contain generally more elements, these elements vary more from one building to another than roof elements, and they are more structured (*e.g.* vertical/horizontal orientation, alignments and symetries). The strong vertical and horizontal priors of the façade elements helps greatly by resampling the input data (image or lidar data) in the geometry of the approximated wall plane, a process called rectification.

As in the reconstruction of whole buildings, there is schematically two approaches. Either features are extracted from the data to build up the façade elements, or a strong prior knowledge is used to decompose the façade down to its individual elements.

Feature-based approaches To take advantage of the alignment of façade elements, a feature-based approach is to accumulate boundary evidences vertically and horizontally (*e.g.* [SB03]). These accumulations let the boundary alignements stand out and allow the detection and coarse reconstruction of the façade elements. The placement and dimensions of these façade elements are then usually locally optimized to better fit the image or lidar data.

Grammar-based approaches The façade structure has lead works [MZWVG07, KTS+09, BR06a] that define a grammar to describe the façade and the layout of its elements. For in-

stance, a simple grammar would decompose a façade vertically into floors, then horizontally into tiles. Then each tile may refer to a wall, a window or a door. Based on such a grammar, a model-based reconstruction of the façade searches within the possible realizations of the grammar the derivation that fits best the input data. This optimization has been performed [BR06a] using a stochastic approach based on Reversible Jump Monte Carlo Markov Chains (RJ-MCMC). The ealier approach of [DTC04] uses this stochastic optimization on a simpler grammar, which does not encode explicitely the alignements and symmetries. It simply considers that each wall features a set of primitive such as doors, windows or columns. These alignement and symmetry properties are however encoded into the likelihood of the façade configuration. As model-based approaches, the downside of the grammar-based approaches is their inherent restriction of the modelizable façades.

3.1.3 Proposed Approach

This chapter tackles the following problem:

Problem Statement

How to automatically detect and reconstruct, using a DSM, an unknown number of roof superstructures, on a 3D building model, given that the input building model is assumed to be a good approximation of main roof planes of the true building?

Our approach to solve this problem is detailed in the subsequent sections: section 3.2 introduces the building representation that formally defines the search space: the set of buildings with superstructures that are considered as possible reconstructions. Then, the purpose of section 3.3 is to choose an energy function that will be able to value each building model with superstructures. Section 3.4 details how the building that minimizes the energy is **automatically** determined within the context of this chapter where the façades and roof facets are kept fixed and the superstructures have to be detected and reconstructed. Before concluding the chapter, sections 3.5 and 3.6 present the results achieved by our method and discuss the design choices and possible extensions.

3.2 3D Building Model Representation $\mathcal{B} = (\mathcal{R}, \mathcal{S})$

In this section we detail how the 3D building model is represented in our algorithms. This defines the search space: the continuous set with variable dimensions of buildings $\mathcal{B}$ with superstructures in which the "best" building is searched for. The exact definition of the "best" building within this set, given an input DSM, is detailed in section 3.3. However this choice of representation is not particularly tailored to be used with the energy defined in section 3.3. The very same representation can be used with other energy functions, such as an energy that evaluates the quality of a 3D building model using the correlation between images instead of the distance to a DSM, or a Lidar point cloud.

Superstructures are defined as disjoint local geometric modifications of a base building, leading to a two-part representation: a building $\mathcal{B}$ is described by a general polyhedron $\mathcal{R}$ (figure 3.2.a) representing the base building (roofs and façades) on one side and a set $\mathcal{S}$ of modifying roof superstructures on the other (figure 3.2.b). The reconstruction of the façade superstructures, like doors, windows or balconies is outside the scope of this work, they will thus not be represented. This is a rudimentary form of a grammar based representation [MWH$^+$06]. To model more complex interactions between the superstructures and the base model, a more complex grammar based representation should be introduced.

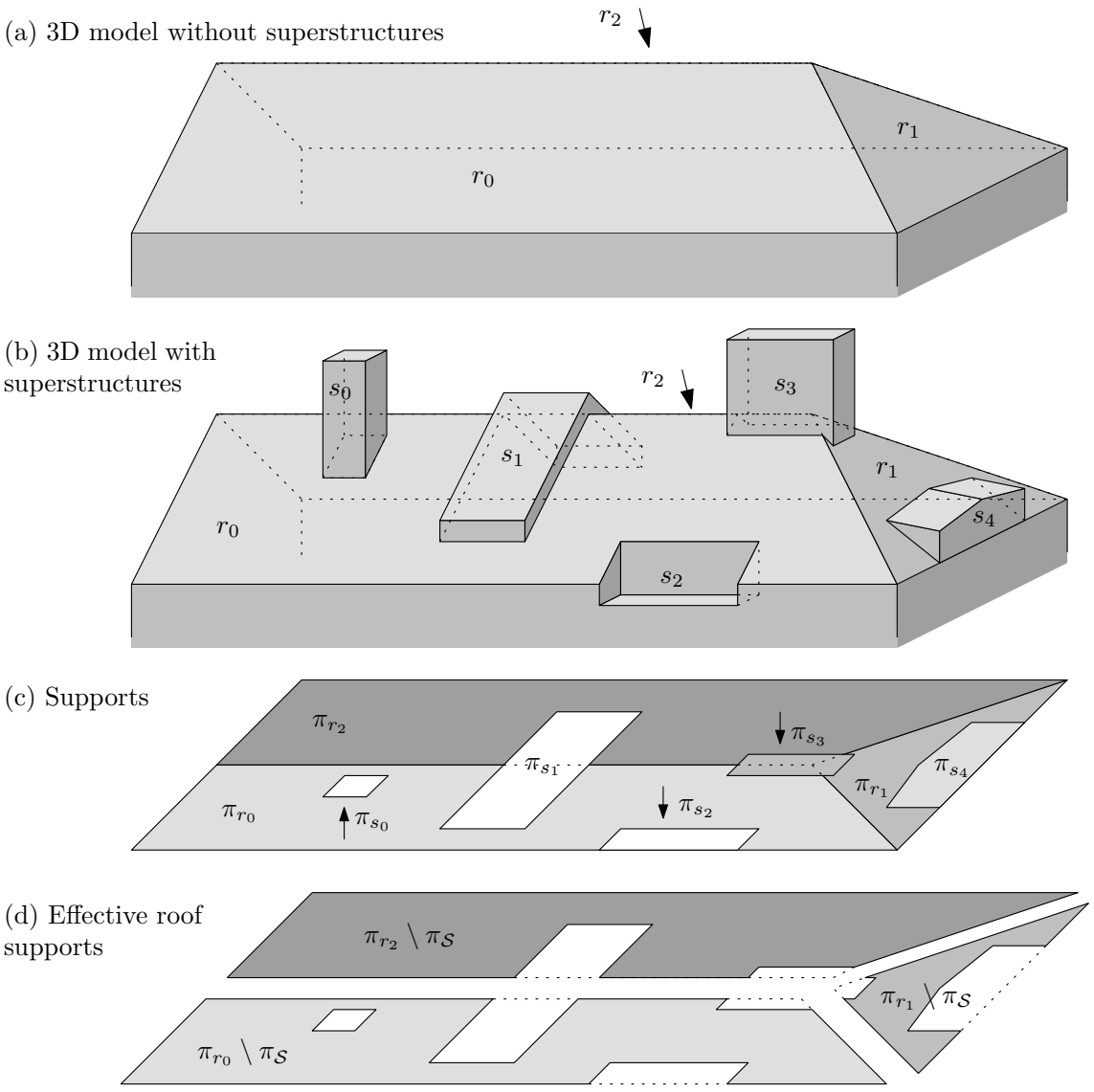


Figure 3.2: (a) A 3D building model with 3 roof planes $\mathcal{R} = \{r_0, r_1, r_2\}$ and no superstructures, (b) the same building model modified by 5 superstructures $\mathcal{S} = \{s_0, s_1, s_2, s_3, s_4\}$, (c) the supports (*i.e.* vertical projections) of the roofs (π_{r_i}) and of the superstructures (π_{s_i}), and (d) the effective supports of the roofs ($\pi_{r_i} \setminus \pi_{\mathcal{S}}$), which are the projections of the parts of the roofs r_i that are not altered by any superstructure.

3.2.1 Polyhedral Base Building $\mathcal{R}$

This section defines polygons and polyhedra, which are ubiquitous primitives for modeling man-made objects. Then, we discuss our choice of modeling buildings without superstructures using polyhedra. The representation of these polyhedra is then discussed and the ???????

3.2.1.1 Polygon Definitions

Whereas points, (line) segments, lines, and planes are common geometric primitives in $\mathbb{R}^3$, 2D or 3D polygons with or without holes deserve a definition to lift any possible ambiguity.

Definition 1 (2D Polygon)**.** *A **2D polygon** is a closed piecewise-linear curve in $\mathbb{R}^2$ described by a circular chain of 2D segments, or equivalently by a circular list of 2D points.*

Definition 2 (Simple 2D Polygon)**.** *A 2D polygon is **simple** iff it is defined by a self-intersection free curve.*

A simple polygon has a well-defined bounded interior region. Thus, it models a connected region of $\mathbb{R}^2$ enclosed by a single piecewise-linear boundary. Simple polygons are topologically equivalent to a disk. Depending on the context, a simple 2D polygon may refer to either its boundary curve or its interior surface.

Definition 3 (2D Polygon with holes)**.** *A **2D polygon with holes** is defined by a distinguished polygon, named the outer boundary, and a set of 2D polygons modeling the hole boundaries.*

Definition 4 (Simple 2D Polygon with holes)**.** *A 2D polygon with holes is **simple** iff its boundary polygons are simple and disjoint from each other, and the outer boundary polygon encloses the hole boundary polygons.*

Simple 2D polygons with holes are thus an extension of simple 2D polygons to model a region with disconnected piecewise linear boundaries. The 2D polygon definitions may now be extended to 3D by considering 2D polygons inside an arbitrary supporting 3D plane instead of the plane $\mathbb{R}^2$. 3D polygons with holes are defined likewise.

Definition 5 (3D Polygon)**.** *A 3D polygon is a closed piecewise-linear planar curve in $\mathbb{R}^3$ described by a circular chain of coplanar 3D segments, or equivalently by a circular list of coplanar 3D points.*

Definition 6 (3D Polygon with holes)**.** *A 3D polygon with holes is defined by a distinguished 3D polygon, named the outer boundary, and a set of 3D polygons modeling the hole boundaries, such that all these polygons are coplanar.*

The definition of simple 3D polygons with or withoPerrinette18 holes follow trivially from their 2D counterpart by restricting the ambient space $\mathbb{R}^3$ to the plane supporting the 3D polygon. An alternative extension of 2D polygons to 3D may not require the coplanarity of the 3D polygon. Except in section 5.1.1.2, 3D polygons with or without holes mentionned in this thesis will be assumed to be coplanar, following the definitions above.

3.2.1.2 Polyhedron Definitions

Definition 7 (Polyhedron)**.** *A **polyhedron** is a bounded, connected and piecewise-planar manifold surface without boundary in $\mathbb{R}^3$, described by a set 3D polygons (without holes).*

Definition 8 (Simple polyhedron)**.** *A polyhedron is **simple** iff its 3D polygons are simple and define a self-intersection free surface.*

To comment on this definition, it implies that a simple polyhedron defines a bounded connected volume without interior volumetric holes. This defintion does not

From the geometry of the polyhedral surface, the combinatorial topological elements of a polyhedron and their supporting geometric elements can be defined:

Definition 9 (Facet)**.** *A **facet** models a maximal non-trivial connected planar region of a polyhedral surface, and is supported by a **3D polygon (possibly with holes)**.*

Definition 10 (Edge)**.** *An **edge** models a maximal non-trivial connected 3D **segment** of the boundary of a facet polygon, and is supported by a 3D **line**.*

Definition 11 (Vertex)**.** *A **vertex** models an endpoint of an edge segment, and is supported by a 3D **point**.*

The assumption that its boundary is manifold (*i.e.* restricting the neighborhood of any boundary point to the boundary surface yields a topological disc), prevents the consideration of degenerate polyhedra that have non-manifold points on their surface, such as distinct edges intersecting or overlapping.

3.2.1.3 Polyhedral Building Modeling

The vast majority of buildings may be well approximated using piecewise-planar surfaces with large planar facets. Relatively few complex buildings feature non piecewise-planar surfaces. The rarity and the complexity of these buildings of special interest make them suitable to more accurate and more costly semi-automatic reconstruction techniques rather than the fully automatic method developped here. Apart from these complex buildings, we consider that surfaces that are not piecewise-planar may be approximated using planar primitives. The buildings considered in this thesis may then be modeled generically with a polyhedron.

The facets of a building with unmodeled superstructures measure approximately from $1m$ to $100m$. At this metric scale of a building without superstructures, we consider that the data at hand for each facet is of sufficient quality to relax all the constraints but the verticality of the façades. Namely, no geometric assumption is made about any symmetry, orthogonality or edge horizontality, except for the verticality of the facets describing the façades.

Within our context, a polyhedron is used to model each building that will be processed independently. The interface between the air and a building is a surface with a boundary at the intersection of the ground or neighboring buildings. A polyhedron, modeling a building, is virtually closed by extending down the façade facets and intersecting them with a bottom-facing plane that approximates the ground level around the building, because it is easier to deal with polyhedral surfaces without boundaries. This virtual plane will not be fitted to the input DSM. Furthermore, it will be set virtually so low that it does not interact with facets that are not modelling the building façade.

3.2.1.4 Polyhedron Representation

The description of a polyhedron is two-fold:

Its topology is the combinatorial part. It can be given by the incidence graph linking its combinatorial elements (vertices, edges and facets): each edge is linked to its 2 endpoint vertices and its 2 adjacent facets. For instance, if the polyhedral facets have no holes, the topology may be encoded using the well-known halfedge data structure [Wei85, Ket99].

Its geometry is the continuous part, that makes the abstract polyhedral topoloy concrete by giving the polyhedron a geometrical embedding. It can be given by the (x, y, z) coordinates of the points that support its vertices.

A halfedge data structure describes the polyhedral geometry using the vertex point locations and the polyhedral topology encoding the adjacency relations between combinatorial vertex/edge/facet objects. This data structure is compact and allows a convenient exploration of the surface using pointer to adjacet facets, oriented edges or vertices. It may be extended to facets with holes by registering all the boundary polygons in each facet record.

This kind of Boundary Representation (B-Rep) is generally opposed to Constructive Solid Geometry (CSG), where a solid is modeled using Boolean operators on elementary solids. For instance, the building decomposition into a base polyhedron $\mathcal{R}$ and a set of modifying superstructure elements $\mathcal{S}$ can be viewed as a CSG decomposition into its superstructures and a polyhedron modeling the base building.

CSG is generally preferred because it conveys more semantics [FKL+98]. However, CSG representations do not explicitly store the surface of the object and rather emphasize the polyhedral volume. They are thus less adapted to reconstruction applications that aim at fitting the object surface to some data. In our case, it makes sense to define the superstructures using a CSG formulation as modification of a base building polyhedron, and to export the resulting polyhedron in B-Rep to get explicitly the polyhedron boundary in order to evaluate simply its goodness of fit relative to the input data.

The Nef polyhedron approach [GHH+03] seems to present an interesting trade-off between B-Rep and CSG. It models polyhedra using boolean operations on 3D half-spaces. It is thus CSG-based with half-space primitives. However the simplicity of these primitives allows to explore the Nef polyhedron surface easily, as when using a B-Rep, with operations such as exploration of the vertices adjacent to a facet, edges adjacent to a vertex. However Nef polyhedra model a broader class of polyhedral surfaces that may be unbounded and non-manifold, which is out of our scope. Handling these surfaces complexifies the handling of Nef polyhedra. Therefore, the B-Rep is preferred.

Using a Boundary Representation of the input base polyhedron, we abusively identify the polyhedron $\mathcal{R}$ with its set of 3D-polygons $\{r\}$ when using the notation $r \in \mathcal{R}$. Similarly, we denote by $(x, y, z) \in \mathcal{R}$ a point of the volume inside the polyhedron $\mathcal{R}$.

3.2.1.5 Surfaces S as functions $z_S : \pi_S \to \mathbb{R}$, with $\pi_S \subset \mathbb{R}^2$

A polyhedron $\mathcal{R}$ defines the surface S of its boundary. To be able to directly compare a surface to a DSM, a tool is needed to express it in the heightfield geometry of the DSM. A surface can be cast as a function that provides the highest elevation z of its points (x, y, z) that have the given x and y coordinates. This can be seen as intersecting the surface with vertical lines. When the intersection is empty, the function is not defined, and when there is at least one intersection point, the function value is the elevation of the highest elevation point. The **height function** (fig. 3.3) of a surface S, denoted z_S is formally defined as:

$$z_S(x, y) = \max \{z \in \mathbb{R} \quad / \quad (x, y, z) \in S\} \tag{3.2.1}$$

We also introduce the **support** π_S of a surface S, which is the definition domain of z_S. More intuitively, this is the set of 2D points that results from a vertical projection of the surface.

$$\pi_S = \left\{(x, y) \in \mathbb{R}^2 \quad / \quad \exists z \in \mathbb{R}, \quad (x, y, z) \in S\right\} \tag{3.2.2}$$

This height function $z_S : \pi_S \to \mathbb{R}$ is not only used to compare a polyhedron to a DSM, but also to express the geometric modifications of the different superstructure types (section 3.2.2) and to compare a DSM to a building with superstructures.

Considering a surface S using its height function only is not injective. As only the maximum elevation is encoded in z_S, the lower parts of the surface are not described by S. If $\mathcal{R}$ is a polyhedron, then $z_{\mathcal{R}}$ is piecewise linear and its discontinuities may correspond to either the support

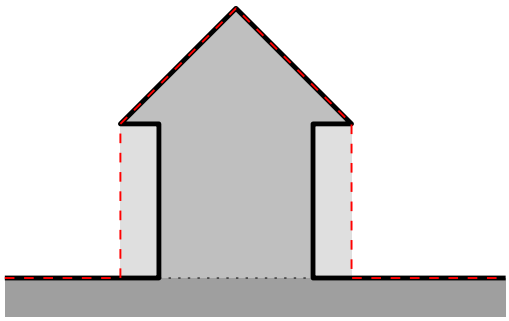


Figure 3.3: The bold surface represents a section of a polyhedral scene. The building is closed at the bottom using an accessory facet (dotted). The corresponding height function (dashed) has no overheads.

of its vertical facets (the building façades) or edges of bottom-facing facets. As a DSM does not present any information about overhead parts, they will not be reconstructed in the proposed method: the only bottom-facing facet of the reconstructed building models will be a virtual facet that closes the building by intersecting the bottom of its façades. Note however, that the base polyhedral building may contain overhanging parts. The height function view of such a building acts as if the volumes below these overhanging parts were filled in.

3.2.2 Parametric Superstructures $\mathcal{S}$

Contrary to the polyhedron $\mathcal{R}$ that describes the main roof planes and the façades at roughly the metric scale, the modeling scale of the superstructures is rather decimetric than metric. This means that the currently available data is not sufficient anymore to allow a reconstruction of those superstructures without any prior knowledge. Furthermore, superstructures appear to be less variable than roofs, at least at the expected reconstruction scale of this work. This is why prior knowledge is introduced here using a finite collection $T = \{Chimney, Terrace, GlassRoof, Dormer\}$ of parametric superstructure types, illustrated in figure 3.4. This list of superstructure types has been designed after surveying the superstructures present on the roofs of multiple European cities [dB07]. The proposed superstructures are fairly generic, and the list is easily extensible as it will be discussed in section 3.6.

The superstructures are modeled with an unordered set $\mathcal{S}$. Each superstructure $s \in \mathcal{S}$ is defined by the tuple $(\vec{\phi}, \theta, \tau, \mathcal{R})$, where:

- $\mathcal{R}$ is the base building modified by the superstructure.
- $\tau \in T = \{Chimney, Terrace, GlassRoof, Dormer\}$ is the superstructure type.
- θ is a 2D rectangle that specifies the position, scale and orientation of the support π_s of the surface modified by the superstructure. It is the oriented bounding box of the support of the superstructure along a given orientation. By extension, we also denote by π_s the support of a superstructure. For superstructures s that have a rectangular support, $\theta = \pi_s$.
- $\vec{\phi} \in \mathbb{R}^{d_\tau}$ is a vector of altimetric parameters ϕ_i that specifies each of the d_τ vertical degrees of liberty of superstructures of type τ.

The DSM data is structured as a 2.5D grid. To take advantage of the regular density of the DSM and design an efficient algorithm, a discrete subset $\Theta_{\tau,\mathcal{R}}$ of all the 2D rectangles is introduced to describe the plausible θs given the base building $\mathcal{R}$ and the superstructure type τ, whereas the set $\Phi_{\theta,\tau,\mathcal{R}} \subseteq \mathbb{R}^{d_\tau}$ of the plausible altimetric parameters $\vec{\phi}$ is kept continuous. Please note that the subscripts in the notations $\Theta_{\tau,\mathcal{R}}$ and $\Phi_{\theta,\tau,\mathcal{R}}$ document elements that are required by the definition of these sets.

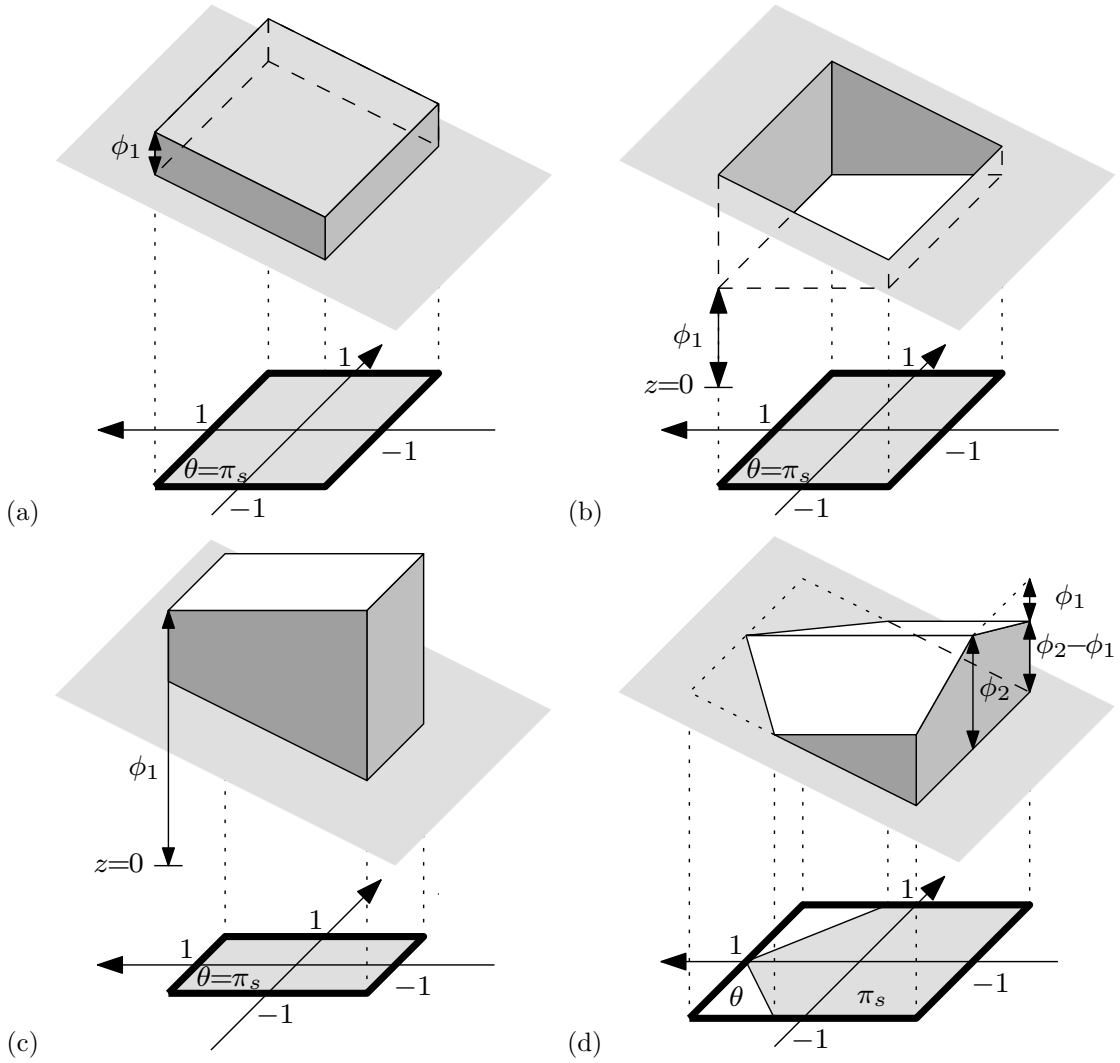


Figure 3.4: 4 simple superstructure types: (a) *GlassRoof*, (b) *Terrace*, (c) *Chimney* and (d) *Dormer* window. They are illustrated here on a single roof plane. Below each superstructure is its bold rectangle θ, and its light-gray π_s support.

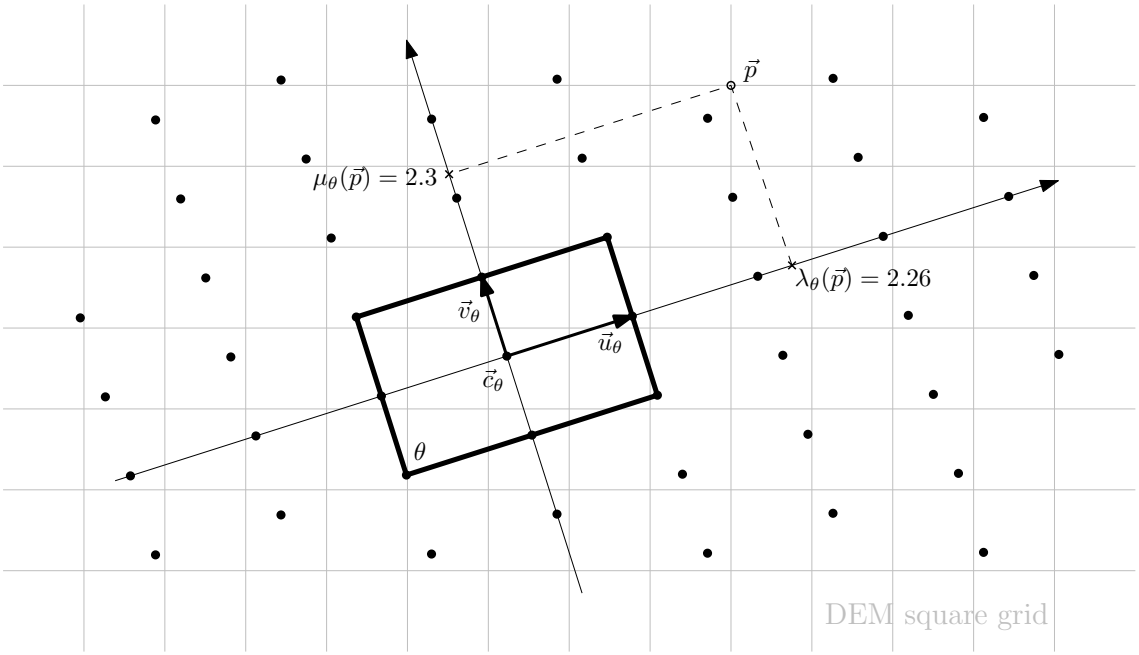


Figure 3.5: The rectangle θ is given by its center $\vec{c}_\theta$ and the vectors $\vec{u}_\theta$, $\vec{v}_\theta$. It defines the local coordinates $(\lambda_\theta(\vec{p}), \mu_\theta(\vec{p}))$ of any point $\vec{p}$. Dots are the grid of points with integer coordinates.

3.2.2.1 Superstructure geometry

We only consider here 2.5D superstructures that modify a 2.5D base building, meaning that any vertical ray intersects the building surface or the surface of the building modified by a set of superstructures only twice, the lower intersection being with the virtual bottom facing ground facet. The geometry of a 2.5D surface is entirely specified by a heightmap $(x, y) \mapsto z(x, y)$ defined over a given support. The heightmap of a building $\mathcal{R}$ is modified locally by a superstructure s:

$$z_{(\mathcal{R},\{s\})}(x,y) = \begin{cases} z_{\mathcal{R}}(x,y) & \text{if } (x,y) \in \pi_{\mathcal{R}} \setminus \pi_s \\ z_s(x,y) & \text{if } (x,y) \in \pi_s \end{cases} \tag{3.2.3}$$

To specify the heightmap z_s in a way that is independent by translation, horizontal rotation and scaling, a local frame $(\vec{c}_\theta, \vec{u}_\theta, \vec{v}_\theta)$ is introduced based on the rectangle θ (Fig. 3.5) where :

- $\vec{c}_\theta$ is the 2D center of the rectangle and the origin of the local frame coordinate system.
- $\vec{u}_\theta$ and $\vec{v}_\theta$ are two orthogonal 2D vectors such that vertices of the rectangle θ are the four points $(\vec{c}_\theta \pm \vec{u}_\theta \pm \vec{v}_\theta)$ and that the frame $(\vec{c}_\theta, \vec{u}_\theta, \vec{v}_\theta)$ is direct. This makes θ the $[-1, 1]^2$ square within its local frame coordinate system.

It should be noted that the definition of $\vec{u}_\theta$ and $\vec{v}_\theta$ from a rectangle θ presents an ambiguity. If $(\vec{c}_\theta, \vec{u}_\theta, \vec{v}_\theta)$ is a valid frame, so is $(\vec{c}_\theta, \vec{v}_\theta, -\vec{u}_\theta)$, $(\vec{c}_\theta, -\vec{u}_\theta, -\vec{v}_\theta)$ and $(\vec{c}_\theta, -\vec{v}_\theta, \vec{u}_\theta)$. To disambiguate the specification of the local frame coordinate from a given rectangle, we impose that $z_{\mathcal{R}}(\vec{c}_\theta + \vec{v}_\theta)$ is higher than $z_{\mathcal{R}}(\vec{c}_\theta - \vec{v}_\theta)$, $z_{\mathcal{R}}(\vec{c}_\theta + \vec{u}_\theta)$ and $z_{\mathcal{R}}(\vec{c}_\theta - \vec{u}_\theta)$. In ambiguous cases where $z_{\mathcal{R}}(\vec{c}_\theta + \vec{u}_\theta) = z_{\mathcal{R}}(\vec{c}_\theta + \vec{v}_\theta)$, we will consider separately the two possible frames as two different candidate rectangles, when it impacts the geometry (*i.e.* with the dormer superstructure types).

The local frame $(\vec{c}_\theta, \vec{u}_\theta, \vec{v}_\theta)$ allows the expressions of a 2D point $\vec{p}$ within the frame coordinates (λ, μ), and conversely to generate the 2D point, denoted $\vec{p}_\theta(\lambda, \mu)$ from the frame coordinates.

$$\lambda_\theta(\vec{p}) = \frac{(\vec{p} - \vec{c}_\theta) \cdot \vec{u}_\theta}{\vec{u}_\theta \cdot \vec{u}_\theta} \quad , \quad \mu_\theta(\vec{p}) = \frac{(\vec{p} - \vec{c}_\theta) \cdot \vec{v}_\theta}{\vec{v}_\theta \cdot \vec{v}_\theta} \qquad \text{and} \qquad \vec{p}_\theta(\lambda, \mu) = \vec{c}_\theta + \lambda \vec{u}_\theta + \mu \vec{v}_\theta$$

Type $\tau \in T$	Height $z_s(\vec{p})$: $\theta \to \mathbb{R}$	Support $\pi_s \subset \mathbb{R}^2$
Chimney	ϕ_1	rectangle θ
Terrace		
GlassRoof	$z_{\mathcal{R}}(\vec{p}) + \phi_1$	
Dormer	$z_{\mathcal{R}}(\vec{p}) - \lvert\lambda_\theta(\vec{p})\rvert\,\phi_1 + \frac{1-\mu_\theta(\vec{p})}{2}\phi_2$	$\{\vec{p} \in \theta \;/\; \lvert\lambda_\theta(\vec{p})\rvert \leq \frac{1-\mu_\theta(\vec{p})}{2}\frac{\phi_2}{\phi_1}\}$

Table 3.1: The geometry of each superstructure type.

Corners of the rectangle θ are thus simply the four points $\vec{p}_\theta(\pm 1, \pm 1)$. Those frame coordinates (λ, μ) are used to express, in table 3.1, the geometric modifications of a superstructure in the local frame $(\vec{c}_\theta, \vec{u}_\theta, \vec{v}_\theta)$ of the 2D rectangle θ.

- A *Chimney* or a *Terrace* are both modeled using a horizontal rectangle linked to the roof with vertical facets. The reason for separating those superstructures into two types is that their sets of plausible rectangles θ and altimetric parameters $\vec{\phi}$ are disjoint: generally speaking, a terrace is below the roof with a much bigger area than the chimneys that are above the roof.
- *GlassRoof*s are vertical offsets of the roof above it with a rectangular support. Thus, they share their slopes with the underlying roofs. This allows to model skylights that are contained in a single roof facet, but also those that overlap the rooftop.
- Concerning *Dormer* windows, we assume that dormer windows do not span multiple roof facets, thus, each dormer belongs to a single roof facet $r \in \mathcal{R}$. The roof top of a dormer can be constructed by linking the back point on the roof $(\vec{c}_\theta + \vec{v}_\theta, z_r(\vec{c}_\theta + \vec{v}_\theta))$ to the top point of the dormer façade $(\vec{c}_\theta - \vec{v}_\theta, z_r(\vec{c}_\theta - \vec{v}_\theta) + \phi_2)$, that is at a distance ϕ_2 above the roof r. $(\phi_2 - \phi_1)$ is simply the length of the vertical edges of the dormer façade.

3.2.2.2 Non-overlapping assumption

Furthermore, we denote by $(s_0 \not\cap s_1)$ the relation that states whether superstructures s_0 and s_1 have disjoint supports :

$$s_0 \not\cap s_1 \iff area(\pi_{s_0} \cap \pi_{s_1}) = 0 \tag{3.2.4}$$

Throughout this work we assume that the superstructures have disjoint supports:

$$\forall s_0, s_1 \in \mathcal{S}, \quad s_0 = s_1 \text{ or } s_0 \not\cap s_1 \tag{3.2.5}$$

Under this assumption, the geometry of a polyhedral building $\mathcal{R}$ modified by an unordered set of superstructures $\mathcal{S}$ is well-defined:

$$z_{(\mathcal{R},\mathcal{S})}(x, y) = \begin{cases} z_{\mathcal{R}}(x, y) & \text{if } (x, y) \in \pi_{\mathcal{R}} \setminus \bigcup_{s \in \mathcal{S}} \pi_s \\ z_s(x, y) & \text{if } \exists s \in \mathcal{S} \text{ such that } (x, y) \in \pi_s \end{cases} \tag{3.2.6}$$

Assuming no superstructure overlaps, there is no need to define the combination of two superstructure geometries. Thus, the superstructure set $\mathcal{S}$ does not need to be ordered. This is required by the optimization algorithm that handles the superstructures as an unordered set.

3.2.2.3 Discrete set $\Theta_{\tau,\mathcal{R}}$

The set of all the rectangles of the 2D plane is continuous with 5 degrees of freedom : 2 for the position, 1 for the orientation and 2 for the dimensions. To discretize this set, first a discrete set of principal directions is computed from the building $\mathcal{R}$, then a 2D square grid is constructed along this direction to quantify both the position and the dimensions of the candidate rectangles.

A possible approach could have been to quantize the orientations with a fixed angular step of a few degrees ($\vec{u} = \left(\cos(\frac{2k\pi}{N}), \sin(\frac{2k\pi}{N})\right), k = 1 \dots N$ for some integer N). The drawback of this quantization is that the orientations are either oversampled, leading to a huge number of superstructure hypotheses and thus extremely slow computing times, or undersampled, leading to inaccurate or erroneous superstructure reconstructions. For simple buildings, a principal component analysis (PCA) of the building footprint gives good estimates of the orientation of its superstructures, but it can fail even for rectilinear buildings (*e.g.* buildings with façades that are either parallel or orthogonal), like L-shaped buildings. Therefore we decided to use a more extensive set of orientations computed from the building $\mathcal{R}$. Each facet of the roof $r \in \mathcal{R}$ may determine a few preferred orientations (see figure 3.6.b).

- The horizontal orientations of the 3D segments forming the edges of the 3D polygon r. This includes orientations of the façades for roof facets that are neighboring one through one of their edges.
- For roof facets r that are not horizontal, there is a unique orientation defined only by its supporting plane without taking into account its boundary. This orientation is the horizontal orientation of the horizontal 3D lines contained in the plane supporting the roof facet.

Some robustness and speed-up has been achieved by only considering a subset of these orientations. A practical choice has been to consider only one orientation per facet. This is the orientation in the former set of orientations that is the closest from the latter: we only consider, for each roof facet, the orientation given by one of its edges that has the smallest slope. This edge is most likely representing a rooftop or a rain gutter.

Now that a finite set of orientations has been defined, the positions and dimensions of the rectangles have to be quantized. The idea is to resample the DSM along each of the preferred orientations and allow only axis aligned rectangles in the resampled DSMs that have integer pixel coordinates.

To construct a 2D square grid from an orientation, one need two more elements: the grid quantization step r in each direction, and the position of one of its vertices $\vec{c}$. The quantization step r is chosen isotropic (equal in both direction) and left as a parameter of the algorithm. A reasonable choice is the horizontal resolution of the fitted DSM. However, there is a computing time/reconstruction accuracy trade-off that may be used to compute less accurate superstructures in shorter time by selecting a greater grid step. The position of a vertex is chosen so that this vertex lies on the edge that originated the orientation. This snapping to the façade or edge introduce a prior that a superstructure near a façade or a roof feature tends to be aligned with that feature. This reduces some reconstruction artefacts and produces simpler and more visually appealing reconstructions.

Given a normalized orientation $\vec{u}$, a grid quantization step r, and any one point of the grid $\vec{c}$, a rectangle θ_0 can be constructed from the frame $(\vec{c}_{\theta_0}, \vec{u}_{\theta_0}, \vec{v}_{\theta_0}) = (\vec{c}, r\vec{u}, r\begin{pmatrix}-u_y\\u_x\end{pmatrix})$. This rectangle defines a regular grid of points:

$$\vec{p}_{\theta_0}(\mathbb{Z}^2) = \left\{\vec{p}_{\theta_0}(i,j) = \vec{c}_{\theta_0} + i.\vec{u}_{\theta_0} + j.\vec{v}_{\theta_0} \quad / \quad (i,j) \in \mathbb{Z}^2\right\}$$

Conversely, the indices (i, j) of a point of the grid can be computed using the local frame coordinate system of θ_0: $(i, j) = (\lambda_{\theta_0}(\vec{p}), \mu_{\theta_0}(\vec{p}))$. The rectangle θ_0 is called the reference rectangle of the grid $\vec{p}_{\theta_0}(\mathbb{Z}^2)$. Simply put, a 2D point, an orientation and a quantization step define a resampling of the DSM. The rectangle θ_0 coresponds to a 2 by 2 pixel axis aligned rectangle in the resampled DSM and $(\lambda_{\theta_0}(\vec{p}), \mu_{\theta_0}(\vec{p})$ is the vector from $\vec{c}_{\theta_0}$ to $\vec{p}$ expressed in pixels of the resampled DSM (Fig. 3.5).

There is an infinite number of rectangles that can be generated from the reference rectangle θ_0

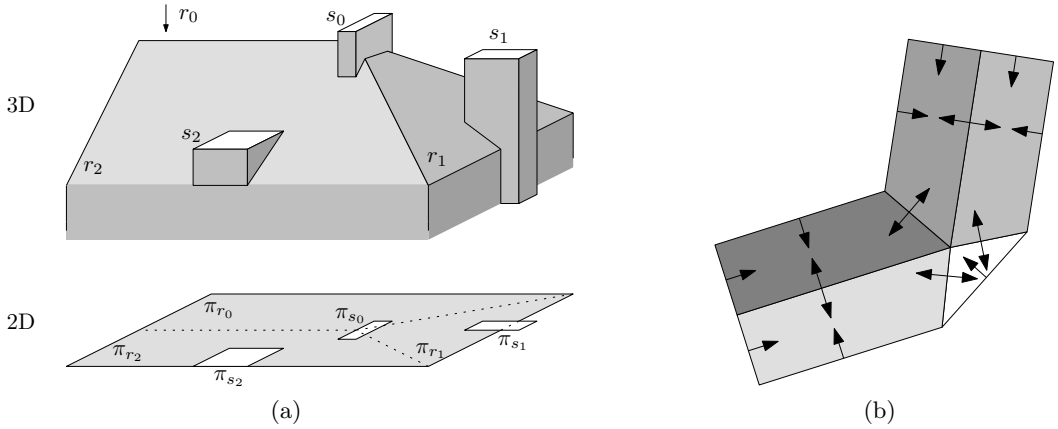


Figure 3.6: (a) A 3D building model and the 2D supports of its facets. Contrary to the *Chimney* s_1, the *Chimney* s_0 and the *Dormer* s_2 verify the building support inclusion property. And s_2 is the only superstructure to span a single roof facet and thus to verify the unique facet support inclusion property. The top view of a building model (c) illustrates the influence of each of its roof edges on the orientation of each roof facet.

by applying a translation and a scaling such that its four corners have integer coordinates in the reference frame defined by θ_0: $\lambda_{\theta_0}, \mu_{\theta_0} \in \mathbb{Z}$. Each such rectangle is identified by its indices $(i_0, j_0, i_1, j_1) \in \mathbb{Z}^4$ that provide the frame coordinates of its four corners: (i_0, j_0), (i_0, j_1), (i_1, j_1) and (i_1, j_0). For instance a rectangle θ has indices $(-1, -1, +1, +1)$ relative to itself. The rectangle θ of indices (i_0, j_0, i_1, j_1) relative to the rectangle θ_0 is defined by:

$$\begin{aligned}
\vec{c}_\theta &= \vec{c}_{\theta_0} + \frac{i_0 + i_1}{2}.\vec{u}_{\theta_0} + \frac{j_0 + j_1}{2}.\vec{v}_{\theta_0} \\
\vec{u}_\theta &= \frac{i_1 - i_0}{2}.\vec{u}_{\theta_0} \\
\vec{v}_\theta &= \frac{j_1 - j_0}{2}.\vec{v}_{\theta_0}
\end{aligned}$$

And the relation between the frame coordinates are:

$$\begin{aligned}
\lambda_\theta(\vec{p}) &= \frac{2.\lambda_{\theta_0}(\vec{p}) - (i_0 + i_1)}{i_1 - i_0} \\
\mu_\theta(\vec{p}) &= \frac{2.\mu_{\theta_0}(\vec{p}) - (j_0 + j_1)}{j_1 - j_0}
\end{aligned}$$

Each superstructure type τ comes with some restrictions on the candidate rectangles θ in order to only consider a finite set of indices (i_0, j_0, i_1, j_1) and thus a finite set of rectangles $\Theta_{\tau,\mathcal{R}}$. This set of candidate rectangles $\Theta_{\tau,\mathcal{R}}$ for each superstructure type τ is simply the finite subset of rectangles generated by the reference rectangles that satisfy the following restrictions of type τ:

Building support inclusion ($\pi_s \subset \pi_\mathcal{R}$): A restriction on θ that is common to all superstructure types τ is that the support π_s of a superstructure $s = (\vec{\phi}, \theta, \tau, \mathcal{R})$ must fall inside the support of the building $\pi_\mathcal{R}$ for every plausible vector $\vec{\phi} \in \Phi_{\theta,\tau,\mathcal{R}}$. This restriction is easy to verify for superstructures that have a support that does not depend on $\vec{\phi}$. This is the case in the proposed library of superstructures for chimneys, roof terraces and glass roofs: they have a rectangular support, so $\pi_s = \theta$ does not depend on $\vec{\phi}$. For instance, the protruding chimney s_1 of figure 3.6.a is not allowed in our model.

Orientations remain local The prefered orientations of a roof facet are given by its surrounding edges (see figure 3.6.b, and section 3.2.2.3). Because every orientation is generated by a single

Type	Constraints on θ : $\Theta_{\tau,\mathcal{R}} = \{\theta \subseteq \pi_{\mathcal{R}}\}$		
$\tau \in T$	$2\|\vec{u}_\theta\|$	$2\|\vec{v}_\theta\|$	$\min(2\|\vec{u}_\theta\|, 2\|\vec{v}_\theta\|)$
Chimney	$[0.3m, 5.0m]$	$[0.3m, 5.0m]$	$[0.3m, 1.0m]$
GlassRoof	$[1.0m, 5.0m]$	$[1.0m, 4.0m]$	
Terrace	$[2.0m, 4.0m]$	$[1.0m, 3.0m]$	
Dormer	$[1.0m, 5.0m]$	$[1.0m, 7.0m]$	

Table 3.2: Indicative minimum and maximum dimensions used for each superstructure type.

edge of the roof, neighboring at most 2 roof facets r_1, r_2, the position of θ relative to $\pi_{r_1} \cup \pi_{r_2}$ is a meaningful consideration. The regularization introduced by snapping the orientation of θ to the orientation of an edge makes more sense if it remains local. For instance in the proposed library, all superstructures must have their support π_s intersect the supports of the roof facets r_1, r_2 that neighbor the edge that generated the superstructure orientation: $area\,(\pi_s \cap (\pi_{r_1} \cup \pi_{r_2})) > 0$.

Unique facet support inclusion ($\exists r \in \mathcal{R}\ s.t.\ \forall r' \in \mathcal{R} \setminus \{r\}, area\,(\pi_s \cap \pi_{r'}) = 0$)**:** A superstructure type may even disallow one of its superstructure to span multiple roof facets (for architectural reasons or because the resulting geometric modification is no longer intuitive if the superstructure spans multiple facets). This is the case, in the proposed library of superstructures, for the *Dormer* window type of superstructures. For instance, the dormer s_2 of figure 3.6.a verifies this property contrary to the chimney s_0 that spans the roof facet supports of r_0, r_1 and r_2.

Dimensions: Each superstructure type τ must define the plausible dimension of its superstructures which is equivalent to giving the minimum and maximum dimensions of the rectangle θ. Because its dimensions are simply $2\|\vec{u}_\theta\|$ and $2\|\vec{v}_\theta\|$, in practice, this requirement is implemented by giving the intervals minimum and maximum values of $2\|\vec{u}_\theta\|$ and $2\|\vec{v}_\theta\|$. The minimum and maximum values typically used in this work are given in table 3.2. For example, plausible roof terraces are between 2 and 4 meters along the preferred direction $\vec{u}_\theta$ and between 1 and 3 meters in the orthogonal direction $\vec{v}_\theta$. The extra constraint on the chimneys insures that the width of the chimney stays below a meter. The dimensions given in table 3.2 are purely indicative. They intend to be loose constraints so that they do not rule out a real superstructure. They admittedly introduce many parameters, but those have a physical meaning. Thus if a reconstruction is not satisfactory because one of those parameters is too tight, it is easy to understand which parameter to tune and how. For instance, if a city contains buildings with very wide dormer windows, one only has to increase the maximum allowable $2\|\vec{u}_\theta\|$ value of *Dormer*s to make those wide dormer windows plausible.

3.2.2.4 Continuous set $\Phi_{\theta,\tau,\mathcal{R}}$

The last element required to determine a superstructure within our representation, given the base building $\mathcal{R}$, the superstructure type τ and the rectangle θ is the vector of continuous parameters $\vec{\phi}$. As documented in table 3.1, each type τ defines the dimension d_τ of the vector $\vec{\phi}$ for superstructures of this type. Furthermore each superstructure type τ restricts the reconstructed superstructures to have their $\vec{\phi}$ vector within a plausible set $\Phi_{\theta,\tau,\mathcal{R}}$ that is context sensitive (depending on the geometry of the roof $\mathcal{R}$ and the position and scale θ of the candidate superstructure).

The sets $\Phi_{\theta,\tau,\mathcal{R}}$ are given in table 3.3. More verbosely, the constraints can be expressed as follows:

- A *Chimney* is higher than 0.5m above the roof. It is also higher than 30cm above the highest point of the roof within a 3 meter radius to ensure a suitable flow of air.
- A *Glassroof* is between 20cm and 80cm above the roof, so that it is discernable from the roof but not too high above it.
- A *Terrace* is lower than 2m below the roof, so that a door can be placed to access the terrace.
- A *Dormer* has a maximum façade height ϕ_2 of at least 1m, to be able to fit a window. The condition $\phi_1 \leq \phi_2$ ensures that the oriented bounding box of the dormer is θ. The constraints on a *Dormer* window involve two angles, α and β. $\tan\beta$ is the slope of the roof top of the dormer and $\tan\alpha$ is the average of the two slopes $\tan\alpha_1$ and $\tan\alpha_2$ of the top edges of the façade, as illustrated in figure 3.7. If the roof is horizontal along the direction $\vec{u}_\theta$, then $\alpha = \alpha_1 = \alpha_2$ which yields a symmetric superstructure. The top of the façade is lower than the intersection of the roof and the top edge of the dormer ($\tan\beta \leq 0$) but the slope of the top edge is not too steep($\tan\beta \geq -1$). The superstructure is convex($\phi_1 \geq 0m$) but the slope of the top edge of the façade is not too steep($\tan\alpha \leq 2$). Finally, the support of a *Dormer* is constrained to overlap only one roof facet, which implies that the upperbound on $\frac{\phi_2}{\phi_1}$ might be strictly lower than 1.

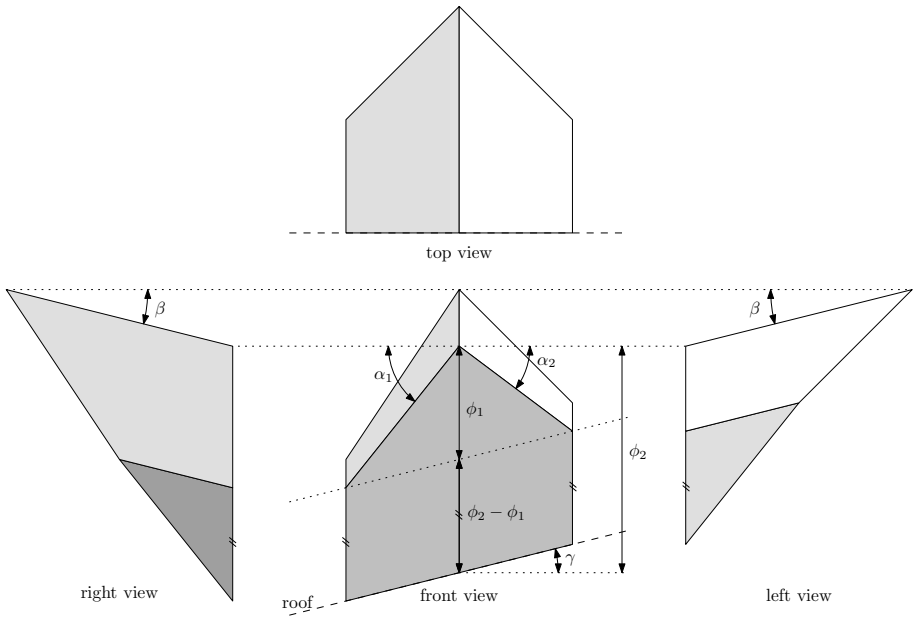


Figure 3.7: Dormer window parameterization: the angles α_1, α_2 and β in a *Dormer* window, illustrated on a roof that has an exagerately great slope $\tan\gamma$ along the direction $\vec{u}_\theta$.

3.2.3 Discussion

Let us compare our representation with the alternative design choice of modeling the building and its superstructures with a single polyhedron. That is, the geometric superstructure modifications are applied to the base building polyhedron, resulting in a polyhedron only described by a set of 3D polygons without any labelling. By deferring those geometric modifications until it is necessary, for visualization or manipulation of the building with the superstructures, the proposed representation has the following advantages:

Semantics: Each 3D polygon is labelled either as a roof facet, as the top facet of a given chimney, as a left side façade of a dormer window... Furthermore, the number of each superstructures is readily available. For instance, computing the total window area of all the dormers is crucial to model the heat transfer of a roof. Such a specific query can directly be answered, with a good approximation, by returning the total area of the dormer front façade polygons.

Type $\tau \in T$	$\dim(\Phi_{\theta,\tau,\mathcal{R}}) = d_\tau \in \mathbb{N}$	$\Phi_{\theta,\tau,\mathcal{R}} = \left\{ \vec{\phi} = (\phi_i)_{i=1\ldots d_\tau} \in \mathbb{R}^{d_\tau} \right\}$ that satisfy the following constraints:
Chimney	1	$\phi_1 \geq \max\left(\max_{\vec{p}\in\theta}(z_{\mathcal{R}}(\vec{p})) + 0.5m, \max_{\{\vec{p}/dist(\vec{p},\theta)\leq 3m\}}(z_{\mathcal{R}}(\vec{p})) + 0.3m \right)$
Terrace	1	$\phi_1 \leq \max_{\vec{p}\in\theta}(z_{\mathcal{R}}(\vec{p})) - 2m$
GlassRoof	1	$0.2m \leq \phi_1 \leq 0.8m$
Dormer	2	$0m \leq \phi_1 \leq \phi_2, \quad 1m \leq \phi_2$ $\tan\alpha = \frac{\phi_1}{\|\vec{u}_\theta\|} \leq 2$ $-1 \leq \tan\beta = \frac{\phi_2 + z_{\mathcal{R}}(\vec{c}_\theta - \vec{v}_\theta) - z_{\mathcal{R}}(\vec{c}_\theta + \vec{v}_\theta)}{2\|\vec{v}_\theta\|} \leq 0$ $\exists r \in \mathcal{R}\ s.t.\ \forall r' \in \mathcal{R} \setminus \{r\}, area\,(\pi_s \cap \pi_{r'}) = 0$

Table 3.3: The set $\Phi_{\theta,\tau,\mathcal{R}}$ used for each superstructure type τ is the intersection of all the above mentionned constraints.

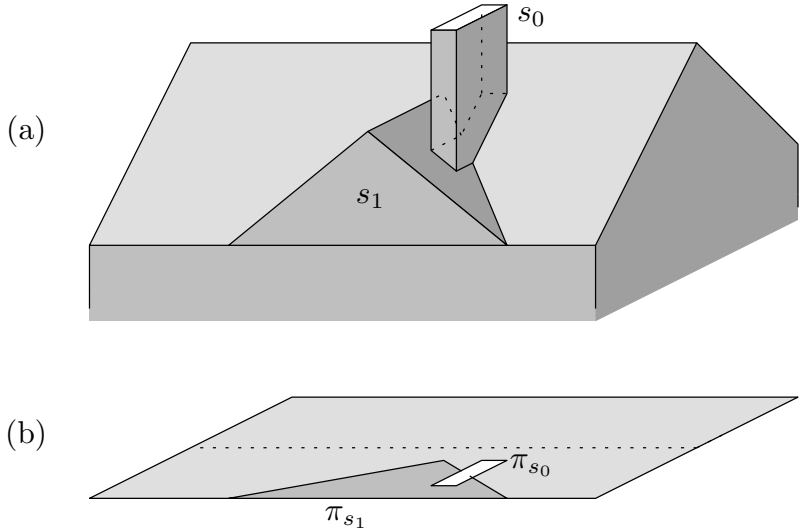


Figure 3.8: A 3D building model (a) and the supports of its facets (b). The chimney s_0 overlaps the large dormer s_1.

Level of detail (LOD): This representation allows the user to export a 3D model of the building at various levels of details. The lower level of detail is just the base building $\mathcal{R}$, while the highest one is the polyhedron resulting from the application of the modification of all the superstructures in $\mathcal{S}$. While the generation of lower LOD representations of a 3D polyhedral model has been studied [May99, Kad02, Kad06], this process would not have been so simple with a plain polyhedron.

Genericity/Robustness trade-off: The hybrid representation allows to give enough expression power when a generic polyhedron can be used to fit robustly the input data while constraining the representation of fine scale details with a library of parametric models where the fit would not have been carried out robustly because of insufficient or noisy data. In the future, this scale limit between objects that can or cannot be reconstructed robustly without incorporating priors will go down because of increasing data acquisition accuracy.

However, this representation of a building with superstructures have the following limitations:

No overlaps: Modeling the superstructures as an unordered set of parametric objects imposes that the superstructures do not overlap. In practice, this constraint is not too restrictive:

when a building may be interpreted as a base building with two overlapping superstructures, as in figure 3.8, one of the superstructures is certainly big enough to be modeled as a part of the roof. By integrating the facets of this big superstructure in the base building polyhedral representation, the overlap conflict is resolved. For instance, considering figure 3.8, a first pass would reconstruct the large dormer s_1. The large facets of the dormer would be treated as roof facets in a second superstructure reconstruction pass to reconstruct the chimney s_0.

Arguable definition of the roof/superstructure classification: The limit is artificial when considering whether some roof facets form a big dormer window or are just some part of the main roof. A more hierarchical approach could be designed where the order of the superstructures $\mathcal{S}$ has a meaning. The polyhedron modeled by a polyhedral building modified by an ordered set of non-necessarily disjoint superstructures will then be the polyhedron $\mathcal{R}$ modified iteratively with all the superstructures of $\mathcal{S}$ in the given order. First, this would avoid issues with overlapping superstructures. Second, those roof modifications could also be used to model any kind of geometric modifications, rather than just adding a superstructure.

Discretization of $\Theta_{\tau,\mathcal{R}}$: The discretization of the positions, and horizontal scales of a superstructure is only required by the superstructure candidate generation of section 3.4.1. For instance, the stochastic diffusion scheme introduced in the appendix B, which is a variant of the section 3.4.1, does not require a discretization of $\Theta_{\tau,\mathcal{R}}$.

3.3 Energy Formulation

To be able to evaluate the coherence of a model with the fitted data, it is a common practice to introduce a function that maps every 3D building model to a real number. The score of a building model $\mathcal{B}$ is denoted $E(\mathcal{B})$ and refered to as its energy to convey some physical intuition. This scoring function yields a total order on the building models that is used to distinguish the best model according to this score, which is the building model that minimizes the energy $E(\mathcal{B})$ within the set, described in section 3.2, of all the representable models.

3.3.1 Minimum Description Length

We choose a Minimum Description Length (MDL) approach [Ris78]. The score $E(\mathcal{B})$ of a building model $\mathcal{B}$ is thus expressed as the sum of an error term with respect to the DSM and a complexity term:

$$E(\mathcal{B}) = \underbrace{D(\mathcal{B})}_{\text{Data/Model Error}} + \underbrace{L(\mathcal{B})}_{\text{Model Complexity}} \tag{3.3.1}$$

This approach is well suited to search spaces with varying complexity, such as ours. It achieves a balance between an overfit of the data by an overly complex model and a poor fit of the data by a simplistic model. The score $E(\mathcal{B})$ is based on the information theoretic description length of the input data using the building model $\mathcal{B}$. $L(\mathcal{B})$ is the bitlength of the description of the 3D bulding model itself, and $D(\mathcal{B})$ is the bitlength of the description of the disparity between the fitted data and the model.

3.3.2 Model Complexity $L(\mathcal{B})$

Assuming the independence of those disjoint superstructures, the description of a building model $\mathcal{B}$ is just the concatenation of the description of its polyhedral part $\mathcal{R}$ and the descriptions of all its superstructures $s \in \mathcal{S}$, yielding the following additive equation on the bit lengths:

$$L(\mathcal{B}) = L(\mathcal{R}) + L(\mathcal{S}|\mathcal{R}) = L(\mathcal{R}) + \sum_{s \in \mathcal{S}} L(s|\mathcal{R}) \tag{3.3.2}$$

The notation $L(s|\mathcal{R})$ emphasizes the fact that the definition of a superstructure s is relative to a given base roof $\mathcal{R}$. The additive nature of this formulation without terms involving more than one superstructure is a design choice required by the optimization algorithm used in this chapter. Another possibility could have been to drop the independence assumption to allow cross terms like binary terms. Those terms could introduce some regularization by rewarding sets of superstructures that are aligned or regularly spaced, or have similar parameters $\vec{\phi}$ (same height, slope, depth...).

3.3.2.1 Roof Description Length $L(\mathcal{R})$

In this chapter, the polyhedral part $\mathcal{R}$ of the building model $\mathcal{B} = (\mathcal{R}, \mathcal{S})$, which represents the main roof planes of $\mathcal{B}$, remains fixed. So its description length $L(\mathcal{R})$ remains constant, and is thus ignored during the superstructure reconstruction step.

3.3.2.2 Superstructure Description Length $L(s|\mathcal{R})$

A superstructure is described by its type $\tau \in T$, its rectangular approximate support $\theta \in \Theta_{\tau,\mathcal{R}}$ and a vector of continuous parameters $\vec{\phi} \in \Phi_{\theta,\tau,\mathcal{R}}$.

$$\begin{aligned} L(s|\mathcal{R}) &= L((\vec{\phi}, \theta, \tau, \mathcal{R})|\mathcal{R}) && (3.3.3)\\ &= L(\tau|\mathcal{R}) + L(\theta|\tau, \mathcal{R}) + L(\vec{\phi}|\theta, \tau, \mathcal{R}) && (3.3.4)\\ &= -\log p_\tau + \log|\Theta_{\tau,\mathcal{R}}| + 12.d_\tau && (3.3.5) \end{aligned}$$

The probability p_τ may be tuned to reflect the proportion of each superstructure type τ in a typical building. Of course this notion of an average proportion of superstructure is rather vague as it will vary greatly with multitude of factors like the country of interest, the region or the building style. A uniform probability is used by default, when this kind of data is not available, resulting in a $-\log p_\tau = -\log \frac{1}{|T|} = \log|T|$ term.

Concerning the second term $\log|\Theta_{\tau,\mathcal{R}}|$, all the possible rectangular approximate support θ are assumed to be equiprobable within the discrete set $\Theta_{\tau,\mathcal{R}}$. Further architectural knowledge may be embedded here to introduce priors on the placement and dimension of superstructures based on the roof $\mathcal{R}$ and its type τ. This could refine or even replace by softer priors the loose hard prior introduced in section 3.2.2.3.

Finally, the continuous parameter vector $\vec{\phi} \in \Phi_{\theta,\tau,\mathcal{R}} \subset \mathbb{R}^{d_\tau}$ is coded using 12 bits for each of the d_τ dimensions, quantizing real values into 4096 values. That leads to an unrestrictive centimetric quantization, given that typical intervals in $\Phi_{\theta,\tau,\mathcal{R}}$ span only a few meters.

3.3.3 Error Term $D(\mathcal{B})$

3.3.3.1 Pixel independence assumption

The error term $D(\mathcal{B})$ measures the bit length needed to code the vertical difference between the ideal DSM predicted by the building model $\mathcal{B}$ and the input DSM, for each pixels of the input DSM within the support of the building $\pi_{\mathcal{B}}$. This assumption that the error of the input DSM values relative to the model $\mathcal{B}$ are independent is a common practice to simplify the evaluation of this error term $D(\mathcal{B})$. However this independence assumption is clearly an approximation: it does not take into account the usual regularization introduced by the DSM generation algorithms such as [RC98].

3.3.3.2 Additive Noise Model

We further assume that the independent vertical pixel errors $(z_{DSM} - z_{\mathcal{B}})$ are identically distributed according to a probability $P(z_{DSM} - z_{\mathcal{B}})$ of the following form,

$$P(z_{DSM} - z_{\mathcal{B}}) = C_{p,\sigma}.e^{-\frac{|z_{DSM}-z_{\mathcal{B}}|^p}{p.\sigma^p}} \tag{3.3.6}$$

where $p > 0$ is a positive integer, σ is the prior standard deviation of the noise and $C_{p,\sigma}$ is the normalizing factor. Applying Shannon's theorem, the minimum coding length of this probability P is given by the sum of a constant $\log C_{p,\sigma}$ and the $\mathcal{L}_p$ norm of the error:

$$-\log\left(P(z_{DSM} - z_{\mathcal{B}})\right) = \log C_{p,\sigma} + \frac{|z_{DSM} - z_{\mathcal{B}}|^p}{p.\sigma^p} \tag{3.3.7}$$

For instance, a normal error distribution assumption will be coded using a $\mathcal{L}_2$ norm.

3.3.3.3 Error Term Derivation

The assumed independence of the DSM values allows the direct summation of the pixelwise error code length over arbitrary sets of pixels. Pixels of the DSM correspond to points in 3D, where the height is given by the pixel value and the horizontal coordinates are given by the 2D point $\vec{p}_{DSM}(i,j)$ that is generated by the pixel indices (i,j) according to the geometry of the DSM. When considering the summation of the single pixel error term $-\log\left(P(z_{DSM} - z_{\mathcal{B}})\right)$, over a set of pixels that map to 2D points $\vec{p}_{DSM}(i,j)$ inside a bounded 2D area $A \subset \mathbb{R}^2$, the error term $D_A(\mathcal{B})$ is obtained:

$$D_A(\mathcal{B}) = |A \cap \vec{p}_{DSM}(\mathbb{Z}^2)|.\log C_{p,\sigma} + \sum_{A \cap \vec{p}_{DSM}(\mathbb{Z}^2)} \frac{|z_{DSM} - z_{\mathcal{B}}|^p}{p.\sigma^p} \tag{3.3.8}$$

Finally, the error term $D(B)$ can be written as the error code length over the whole area $\pi_{\mathcal{B}}$. Since a superstructure support is not allowed to overlap the outside of the building support, the support of the building is given by the support of its roof planes: $\pi_{\mathcal{B}} = \pi_{\mathcal{R}}$. :

$$D(\mathcal{B}) = D_{\pi_{\mathcal{B}}}(\mathcal{B}) = D_{\pi_{\mathcal{R}}}(\mathcal{B}) = |\pi_{\mathcal{R}} \cap \vec{p}_{DSM}(\mathbb{Z}^2)|.\log C_{p,\sigma} + \sum_{\pi_{\mathcal{R}} \cap \vec{p}_{DSM}(\mathbb{Z}^2)} \frac{|z_{DSM} - z_{\mathcal{B}}|^p}{p.\sigma^p} \tag{3.3.9}$$

3.3.4 Fixed Roof Additive Reformulation

Within this chapter, the polyhedral part $\mathcal{R}$ of the building representation stays fixed. This constraint will lead to a reformulation of the energy $E(B)$. We introduce the background hypothesis $\mathcal{B}^0 = (\mathcal{R}, \emptyset)$: this is the building that has the roof planes $\mathcal{R}$ but no superstructures. By rearranging the terms, using the pixel error independence assumptions, and superstructure non-overlap, the resulting terms can then be written as:

$$\begin{aligned}
D(\mathcal{B}) &= D_{\pi_{\mathcal{B}}}(\mathcal{B}) = D_{\pi_{\mathcal{R}}}(\mathcal{R}) + D_{\pi_{\mathcal{S}}}(\mathcal{S}) - D_{\pi_{\mathcal{S}}}(\mathcal{R}) && (3.3.10)\\
&= D(\mathcal{B}^0) + D_{\pi_{\mathcal{S}}}(\mathcal{S}) - D_{\pi_{\mathcal{S}}}(\mathcal{R}) && (3.3.11)\\
&= D(\mathcal{B}^0) + \sum_{s \in \mathcal{S}} \left(D_{\pi_s}(s) - D_{\pi_s}(\mathcal{R})\right) && (3.3.12)\\
L(\mathcal{B}) &= L(\mathcal{R}) + L(\mathcal{S}|\mathcal{R}) && (3.3.13)\\
&= L(\mathcal{B}^0) + \sum_{s \in \mathcal{S}} L(s|\mathcal{R}) && (3.3.14)
\end{aligned}$$

The energy $E(\mathcal{B})$ can then be turned into an additive energy with only a constant term $E(\mathcal{B}^0)$ depending on the building without superstructure $\mathcal{B}^0$ and unary terms $\Delta E(s)$ for each superstructures $s \in \mathcal{S}$:

$$E(\mathcal{B}) = E(\mathcal{B}^0) - \sum_{s \in \mathcal{S}} \Delta E(s) \tag{3.3.15}$$

$$\text{with} \quad \Delta E(s) = D_{\pi_s}(\mathcal{R}) - D_{\pi_s}(s) - L(s|\mathcal{R}) \tag{3.3.16}$$

$\Delta E(s)$, defined in equation 3.3.16, encodes the benefit of adding the superstructure s to the building reconstruction compared to the building without this superstructure. The higher $\Delta E(s)$ is, the better the superstructure s modifies the building to fit the data and the lower gets the overall energy $E(\mathcal{B})$ (hence the minus sign in 3.3.15).

3.4 Optimization

Now that the search space has been defined in section 3.2 and that the objective function, introduced in section 3.3, has been derived into a simpler additive score function in subsection 3.3.4 when the roof $\mathcal{R}$ is fixed, it is time to explore the search space to select the model that minimizes the energy function.

3.4.1 Generation of Superstructure Hypotheses

Given the limited extent of a building relative to the desired accuracy of the reconstructed superstructures, it is tractable to perform a brute-force search by enumerating and evaluating a set of plausible superstructures. Using a grid quantization step $r = 0.1m$, typical problem sizes involve less than 10^4 discrete translations, corresponding to roof surfaces of less than $100m^2$, and less than 10^4 admissible dimensions, all types considered. Finally, the typical number of type/rectangle couples (θ, τ) is less than 10^8.

Algorithm 1 sketches the exhaustive exploration of the discretized part of the search space (the types T and the rectangles $\Theta_{\tau,\mathcal{R}}$). The search space, restricted by the given superstructure type τ and discrete parameter θ, is reduced to a continuous set $\Phi_{\theta,\tau,\mathcal{R}}$ of parameters. The estimation of the best continuous vector $\vec{\phi}_{\max}$ is carried out by the function $estimate\_vector_p(\Phi_{\theta,\tau,\mathcal{R}}, \theta, \tau, \mathcal{R}, DSM)$ that finds the vector $\vec{\phi} \in \Phi_{\theta,\tau,\mathcal{R}}$ that maximizes the benefit ΔE with a given DSM, roof $\mathcal{R}$, type τ and rectangle θ. The resulting superstructure $s = (\vec{\phi}_{\max}, \theta, \tau, \mathcal{R})$ is validated (*i.e.* added to the list of superstructure hypotheses) if its introduction in the building model lowers the overall energy of the building ($\Delta E(s) > 0$).

Algorithm 1 $Hypotheses(\mathcal{R}, step, DSM)$

```
H ← ∅
for all τ ∈ T do
  for all θ ∈ Θ_{τ,R} do
    φ⃗_max ← estimate_vector_p(Φ_{θ,τ,R}, θ, τ, R, DSM)
    if ΔE(φ⃗_max, θ, τ, R) > 0 then
      H ← H ∪ {(φ⃗_max, θ, τ, R)}
return H
```

Let us now detail the general $\mathcal{L}_p$ optimization of $\vec{\phi}_{\max}$ performed by the $estimate\_vector_p$ function, before presenting the speed ups achievable when $p = 2$.

3.4.1.1 Estimation of $\vec{\phi}_{\max}$: $\mathcal{L}_p$-Norm

Given the fixed roof $\mathcal{R}$ and every superstructure type τ, the set $\Theta_{\tau,\mathcal{R}}$ of all candidate discrete supports is computed and for each support $\theta \in \Theta_{\tau,\mathcal{R}}$ the continuous parameter vector $\vec{\phi} \in \Phi_{\theta,\tau,\mathcal{R}}$ is estimated continuously to maximize the benefit $\Delta E(s)$. Since $L(s|\mathcal{R})$ and $p.\sigma^p$ do not depend on $\vec{\phi}$, they can be canceled out of the optimization of $\Delta E(s)$, yielding the following definition of $\vec{\phi}_{\max}$:

$$\Delta E(s) = \frac{\sum_{\pi_s \cap \vec{p}_{DSM}(\mathbb{Z}^2)} |z_{DSM} - z_s|^p - |z_{DSM} - z_{\mathcal{R}}|^p}{-p.\sigma^p} - L(s|\mathcal{R}) \tag{3.4.1}$$

$$\text{Thus,} \quad \vec{\phi}_{\max} = \underset{\substack{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}} \\ s=(\vec{\phi},\theta,\tau,\mathcal{R})}}{\arg\max} \left(\Delta E(s)\right) \tag{3.4.2}$$

$$= \underset{\substack{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}} \\ s=(\vec{\phi},\theta,\tau,\mathcal{R})}}{\arg\min} \left(\sum_{\pi_s \cap \vec{p}_{DSM}(\mathbb{Z}^2)} |z_{DSM} - z_s|^p - |z_{DSM} - z_{\mathcal{R}}|^p \right) \tag{3.4.3}$$

z_s is a heightfield that determines the geometry of the superstructure s. According to table 3.1, the heightfield z_s of the 4 proposed superstructure types can be written as an affine combination of $\vec{\phi}$-independent heightfields, where the coefficients are scalars that do not depend on the position $\vec{p}$. The coefficients of the affine combination are simply the coordinates of the vector $\vec{\phi}$. Thus the pointwise error $(z_{DSM} - z_s)(\vec{p})$ may be expressed as the dot product $(z_{DSM} - z_{\vec{\phi},\theta,\tau,\mathcal{R}})(\vec{p}) = \vec{e}_{\theta,\tau,\mathcal{R},DSM}(\vec{p}) \cdot \begin{pmatrix}1\\ \vec{\phi}\end{pmatrix}$, where $\vec{e}_{\theta,\tau,\mathcal{R},DSM}(\vec{p}) : \mathbb{R}^2 \to \mathbb{R}^{d_\tau+1}$ is called the error vector field. The error vector fields of the proposed superstructure types are :

$$\begin{aligned}
\vec{e}_{\theta,Chimney,\mathcal{R},DSM}(\vec{p}) &= (z_{DSM}(\vec{p}), -1) \\
\vec{e}_{\theta,Terrace,\mathcal{R},DSM}(\vec{p}) &= (z_{DSM}(\vec{p}), -1) \\
\vec{e}_{\theta,GlassRoof,\mathcal{R},DSM}(\vec{p}) &= ((z_{DSM} - z_{\mathcal{R}})(\vec{p}), -1) \\
\vec{e}_{\theta,Dormer,\mathcal{R},DSM}(\vec{p}) &= \left((z_{DSM} - z_{\mathcal{R}})(\vec{p}), |\lambda_\theta(\vec{p})|, \frac{\mu_\theta(\vec{p}) - 1}{2}\right)
\end{aligned}$$

This finally allows the following definition of $\vec{\phi}_{\max}$:

$$\vec{\phi}_{\max} = \underset{\substack{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}} \\ s=(\vec{\phi},\theta,\tau,\mathcal{R})}}{\arg\min} \left(\sum_{\pi_s \cap \vec{p}_{DSM}(\mathbb{Z}^2)} \left| \vec{e}_{\theta,\tau,\mathcal{R},DSM} \cdot \begin{pmatrix}1\\ \vec{\phi}\end{pmatrix} \right|^p - |z_{DSM} - z_{\mathcal{R}}|^p \right) \tag{3.4.4}$$

If the support π_s does not depend on $\vec{\phi}$ either, as it is the case with the proposed *Chimney*, *Terrace* and *GlassRoof* types of superstructures where $\pi_{(\vec{\phi},\theta,\tau,\mathcal{R})} = \theta$, the $D_{\pi_s}(\mathcal{R})$ term becomes constant with respect to $\vec{\phi}$ and the expression defining $\vec{\phi}_{\max}$ can be further simplified as:

$$\text{If } \pi_{(\vec{\phi},\theta,\tau,\mathcal{R})} = \theta, \qquad \vec{\phi}_{\max} = \underset{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}}}{\arg\min} \left(\sum_{\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)} \left| \vec{e}_{\theta,\tau,\mathcal{R},DSM} \cdot \begin{pmatrix}1\\ \vec{\phi}\end{pmatrix} \right|^p \right)$$

$\mathcal{L}_p$ minimization with $p \neq 2$ is typically performed via an iterative weighted least square minimization. For efficiency reasons, the optimization is carried disregarding the constraints. The unconstrained minimum is then projected onto the constraints to get a possibly suboptimal minimum, that is acceptable in practice.

This minimization is rather costly, as the optimization is iterative and that $|\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)|$, the number of observations, is proportional to the area of the superstructure. It is however considered to be a robust metric when $1 \leq p < 2$, and is typically used with $p = 1.2$.

3.4.1.2 Estimation of $\vec{\phi}_{\max}$: $\mathcal{L}_2$-Norm

When using the $\mathcal{L}_p$ norm with $p = 2$, the optimization amounts to a constrained least square minimization. While the $\mathcal{L}_2$ norm is known to over-penalize errors, and is thus less robust than $\mathcal{L}_p$ norms with $1 \leq p < 2$, its minimization can be carried out in constant time for the superstructure types that have a rectangular support(*Chimney*, *GlassRoof* and *Terrace* in our library). This is also true for superstructures that have a support that can be written as the disjoint union of a constant set of aligned rectangles. The basic idea is to preprocess the input to be able to answer in constant time queries that ask for the sum of a given quantity over a rectangular support.

This preprocessing involves resampling various moments of order 0, 1 and 2 the geometry and the DSM to the geometry of the grid generated by the reference rectangle of each preferred orientation. Then a cumulative version of each resampled image is computed, to be able to answer rectangular queries in constant time. Further details are presented in the appendix A.

However, for superstructure types that have a more complicated support, that is not a union of a constant number of rectangles and/or that is varying with $\vec{\phi}$, like the *Dormer*s, the estimation can not be carried out in constant time. But some approximations are introduced in appendix A to estimate $\vec{\phi}_{\max}$ in time linear to the perimeter of the superstructure support.

This ends the description of the estimation of the $\vec{\phi}_{\max}$ vector. Now that a set of superstructure hypotheses has been reconstructed, it is time to select within this set the best subset of disjoint superstructures.

3.4.2 Selection of Disjoint Superstructures

The next step is to minimize the total error of the reconstructed building with superstructures, by selecting within the $2^{|\mathcal{H}|}$ subsets of the superstructure hypotheses $\mathcal{H}$ (Algorithm 1) the subset that minimizes the energy while having only pairwise disjoint superstructures.

The energy has been reformulated in subsection 3.3.4 as a simple sum of unary terms when the base building $\mathcal{R}$ remains fixed. Given that the term $E(\mathcal{B}^0) = E(\mathcal{R}, \emptyset)$ is a fixed constant, the resulting selection problem is:

Best Disjoint Superstructure Set

Given a set of superstructure hypotheses $\mathcal{H}$, a non-overlapping relation $\not\cap$ and positive benefits $\Delta E(s) \in \mathbb{R}^+$ for each superstructure $s \in \mathcal{H}$, the Best Disjoint Superstructure Set problem may be stated as computing:

$$\underset{\mathcal{S} \in \not\cap(\mathcal{H})}{\arg\max} \left(\sum_{s \in \mathcal{S}} \Delta E(s) \right)$$

with the disjoint subsets $\not\cap(\mathcal{H}) = \{\mathcal{S} \subset \mathcal{H} / \forall s_0, s_1 \in \mathcal{S}, s_0 = s_1 \text{ or } s_0 \not\cap s_1\}$.

This problem is an instance of the following Maximum Weighted Clique problem [BBPP99], which is NP-hard [GJ79]:

Maximum Weighted Clique (MWC)

Given an undirected node-weighted graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, w)$ with a set of nodes $\mathcal{N}$, undirected unweighted edges without loops $\mathcal{E} \subseteq \{(n_1, n_2) \in \mathcal{N} \times \mathcal{N} / n_1 \neq n_2\}$ and positive node weights $w(n) \in \mathbb{R}^+$ for each node $n \in \mathcal{N}$, the Maximum Weighted Clique problem may be stated as computing:

$$\underset{\mathcal{C} \in Cliques(\mathcal{G})}{\arg\max} \left(\sum_{n \in \mathcal{C}} w(n) \right)$$

with the set of cliques $Cliques(\mathcal{G}) = \{\mathcal{C} \subseteq \mathcal{N} / \forall n_0, n_1 \in \mathcal{C}, n_0 = n_1 \text{ or } (n_0, n_1) \in \mathcal{E}\}$.

One can easily see that the equivalence between the two problems is obtained by constructing a graph where the nodes are the candidate superstructures $\mathcal{N} = \mathcal{H}$, the positive weights are given by $w(s) = \Delta E(s)$ and the undirected edges E link nodes that satisfy the symmetric $\not\wedge$ relationship.

This Maximum Weighted Clique optimization is performed using *cliquer*[Öst02, NÖ03] a state of the art general Maximum Weighted Clique solver. A variant of this algorithm, exposed in appendix C, that was tailored to make use of the specific structure of the graph induced by the $\not\wedge$ relationship and the weights $\Delta E(s)$ has been developed. The basic idea is to introduce an upperbound of the maximum clique weight of any MWC subproblem by relaxing both :

the intersection-free constraint: Instead of requiring disjoint superstructures, we only require the less restrictive constraint that the sum of the support areas of the selected superstructures is less than the area of the union of the supports of all the competing superstructure hypotheses.

the selection constraint: The binary selection of each superstructure in $\{0, 1\}$ is replaced with a floating point selection weight in $[0, 1]$. This allows selecting only a percentage of a superstructure to benefit from the corresponding percentage of its benefit ΔE.

Solving this relaxed problem is easy and provides an upperbound of the unrelaxed Maximum Weigted Clique problem, which may allow pruning the exploration of any MWC subproblem, based on the weight of the best clique found so far. However, while keeping the optimality of the solution, the computing time was similar and there was no obvious way further improve it in a more efficient variant. Further details about the theory and implementation of the Maximum Weighted Clique problem can be found in appendix C.

3.4.3 Local Maxima Filtering

We propose to apply a local maxima filter to the candidates output by the exhaustive search of section 3.4.1, before the selection of the reconstructed superstructures in section 3.4.2. This filtering discards a hypothesis if a neighboring hypothesis has a strictly better benefit ΔE. The notion of neighborhood between two superstructures is defined by the following neighborhood relation on their rectangles: two rectangles are neighbors if they share the same orientation and if they differ only by a small scaling and/or translation. Their indices (i_1, j_1, k_1, l_1) and (i_2, j_2, k_2, l_2), relative to the same reference rectangle θ_0 must differ by at most 1 quantization step: $\max(|i_2 - i_1|, |j_2 - j_1|, |k_2 - k_1|, |l_2 - l_1|) \leq 1$. In other words, two rectangles are neighbors if a rectangle can be transformed into the other by translating each of its edges by, at most, one sampling *step* along its normal direction.

By applying such a filter, the solution is no longer guaranteed to be optimal, according to the optimized energy. There are mainly two reasons to consider only a subset of all the detected superstructures: the reduction of the problem size and the prevention of multiple detections.

3.4.3.1 Problem Size Reduction

It is too computationally intensive in practice to consider all the superstructure hypotheses enumerated in section 3.4.1 during the optimization of the previous section.

The most obvious way to reduce the number of candidates, is to increase the resolution of the sampling *step* of the rectangles in $\Theta_{\tau,\mathcal{R}}$. However, this impacts the robustness of the detection and the accuracy of the reconstruction. The alternate approach is to only keep superstructures that are locally "good" superstructures. This is what the local maxima filtering approach performs.

A third problem reduction approach is detailed in appendix B. It avoids the exhaustive reconstruction of the superstructure of the discrete set $\{\Theta_{\tau,\mathcal{R}}, \tau \in T\}$, by first detecting regions of interest and then reconstructing a single superstructure candidate for each type $\tau \in T$ and each region of interest.

3.4.3.2 Filtering Prevents Multiple Detections

While reducing the size of the maximum weighted clique problem, this local maxima filtering limits multiple detection artefacts. Input DSMs are typically regularized to reduce their noise. While this gives satisfactory results on flat or low curvature surfaces, the sharp height discontinuities of the small features that are the superstructures, may suffer from delocalization and smearing. Even if a real superstructure is well reconstructed, its representation in the input DSM presents a transition region around the support of the superstructure where the DSM height is not exactly the roof height but is biased towards the superstructure height. For instance, Figure 3.9 shows how the imperfections of the DSM (illustrated in red) are interpreted as small chimneys neighboring the real chimneys. In Figure 3.10, the local maxima filter has been applied, preventing the reconstruction of these erroneous chimneys.

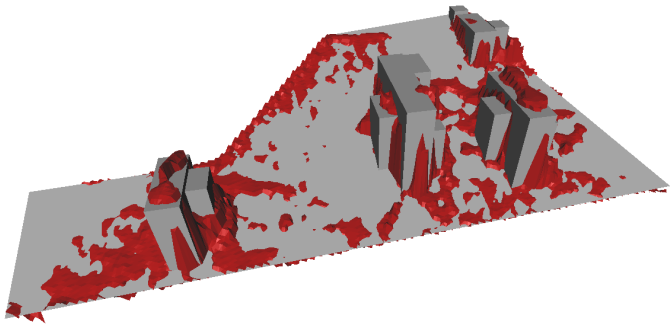


Figure 3.9: Multiple detections without local maxima filtering: small chimneys are reconstructed around the real chimneys to take into account the imperfections of the DSM (shown in red).

3.4.3.3 Implementation

While this filtering is conceptually a post process of the exhaustive search detailed in Algorithm 1, it is implemented with the single pass version of Algorithm 2. This implementation allows first to improve the memory costs and locality by keeping buffers and online filtering and second to get a more data sensitive performance with the $\mathcal{L}_p$ metric using a lazy evaluation of the benefits ΔE.

This buffered implementation processes independently rectangles of $(\bigcup_{\tau \in T} \Theta_{\tau,\mathcal{R}})$ that do not share the same orientation (and reference rectangle θ_0). The roof is then scanned along the direction where it is the most extended, among the two preferred directions $\vec{u}_{\theta_0}$ and $\vec{v}_{\theta_0}$. The notations $\lambda_{\mathcal{R}} = \lfloor \max_{\pi_{\mathcal{R}}}(\lambda_{\theta_0}) \rfloor - \lceil \min_{\pi_{\mathcal{R}}}(\lambda_{\theta_0}) \rceil$ and $\mu_{\mathcal{R}} = \lfloor \max_{\pi_{\mathcal{R}}}(\mu_{\theta_0}) \rfloor - \lceil \min_{\pi_{\mathcal{R}}}(\mu_{\theta_0}) \rceil$ are shorthands to

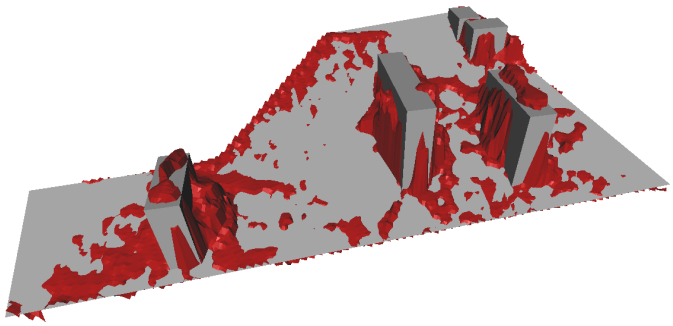


Figure 3.10: Single detection with local maxima filtering: the small chimneys of figure 3.9 have been filtered out and are no longer reconstructed around the real chimneys.

Algorithm 2 *FilteredHypotheses*$(\mathcal{R}, step, DSM)$

for all orientations, given by a reference rectangle θ_0 **do**
  for $i = \lceil \min_{\pi_\mathcal{R}}(\lambda_{\theta_0}) \rceil$ to $\lfloor \max_{\pi_\mathcal{R}}(\lambda_{\theta_0}) \rfloor$ **do**
    for all j, k, l, such that the rectangle θ, of indices (i, j, k, l) relative to θ_0, is in $(\bigcup_{\tau \in T} \Theta_{\tau,\mathcal{R}})$ **do**
      {Compute the best superstructure $s_{\max}(\theta) = \arg\max_{\vec{\phi},\tau} \left(\Delta E(\vec{\phi}, \theta, \tau, \mathcal{R})\right)$}
      $\Delta E_{\max}(\theta) \leftarrow 0$
      $s_{\max}(\theta) \leftarrow \emptyset$
      for all superstructure type $\tau \in T$, such that $\theta \in \Theta_{\tau,\mathcal{R}}$ **do**
        $\vec{\phi}_{\max} \leftarrow estimate\_vector_p(\Phi_{\theta,\tau,\mathcal{R}}, \theta, \tau, \mathcal{R}, DSM, \Delta E_{\max}(\theta))$
        if $\Delta E(\vec{\phi}_{\max}, \theta, \tau, \mathcal{R}) > \Delta E_{\max}(\theta)$ **then**
          $s_{\max}(\theta) \leftarrow (\vec{\phi}_{\max}, \theta, \tau, \mathcal{R})$
          $\Delta E_{\max}(\theta) \leftarrow \Delta E(s_{\max}(\theta))$
      {Maintain the local maxima buffer $s_{\max}$.}
      for all rectangle θ' previously considered that is neighboring θ **do**
        if $\Delta E_{\max}(\theta') < \Delta E_{\max}(\theta)$ **then** $s_{\max}(\theta') \leftarrow \emptyset$
        if $\Delta E_{\max}(\theta) < \Delta E_{\max}(\theta')$ **then** $s_{\max}(\theta) \leftarrow \emptyset$
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{s_{\max}(\theta) \quad / \quad \theta$'s first index is $i - 1\}$
    Clear buffered benefits $\Delta E_{\max}$ and local maxima $s_{\max}$ that have a first index of $i - 1$
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{s_{\max}(\theta) \quad / \quad \theta$'s first index is $\lfloor \max_{\pi_\mathcal{R}}(\lambda_{\theta_0}) \rfloor\}$
  Clear buffered benefits $\Delta E_{\max}$ and local maxima $s_{\max}$
return $\mathcal{H}$

the integer sizes of the roof $\mathcal{R}$ along the orientations $\vec{u}_{\theta_0}$ and $\vec{v}_{\theta_0}$. For simplicity, Algorithm 1 and the remainder of this paragraph assume that $\lambda_\mathcal{R} > \mu_\mathcal{R}$ and thus iterate over i instead of iterating over j, to minimize the size of the buffers $\Delta E_{\max}$ and $s_{\max}$. They keep track, for the previous iteration $i - 1$ and the current iteration i, of all the benefits $\Delta E_{\max}(\theta)$ and of all the hypotheses $s_{\max}(\theta)$, that are, so far, local maxima. This buffered implementation thus keeps only in memory at most $(2. \min(\lambda_\mathcal{R}, \mu_\mathcal{R}))$ superstructures - instead of all the $(\lambda_\mathcal{R}.\mu_\mathcal{R}.|T|)$ superstructures - for each different size $(k - i, l - j)$ of the plausible rectangles of $(\bigcup_{\tau \in T} \Theta_{\tau,\mathcal{R}})$.

The estimation of $\vec{\phi}_{\max}$ is performed lazily in Algorithm 2 by taking into account the best benefit $\Delta E_{\max}(\theta)$ computed so far of superstructures of different types but that share the same rectangle θ. Comparatively to the implementation of Algorithm 1, the function $estimate\_vector_p$ now takes an extra argument $\Delta E_{\max}(\theta)$. This argument is a lowerbound on the benefit of the estimated superstructure, assuming that it is local maximum. This lowerbound is used in the function $estimate\_vector_p$ when the estimation $\phi_{\max}$ is expensive, but there exists an inexpensive way to compute an upperbound to the maximum attainable benefit $(\max_{\vec{\phi} \in \Phi_{\theta,\tau,\mathcal{R}}} \Delta E(\vec{\phi}, \theta, \tau, \mathcal{R}))$.

If the upperbound is lower than the required lowerbound, then the expensive estimation of $\phi_{\max}$ is skipped, yielding a data sensitive estimation. The superstructure of the first type considered is estimated with a lowerbound of 0, ensuring a positive benefit and an early termination of the estimation of $\vec{\phi}_{\max}$ if no such superstructure exists.

An upperbound on the benefit ΔE can be inexpensively computed and tightened while building the system that will be minimized to estimate $\vec{\phi}_{\max}$. Because $L(s|\mathcal{R})$ does not vary with $\vec{\phi}$, by using the inclusion $\pi_{(\vec{\phi},\theta,\tau,\mathcal{R})} \subseteq \theta$ for all $\vec{\phi}$, and the notation $\pi_\cap = \bigcap_{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}}} \pi_{(\vec{\phi},\theta,\tau,\mathcal{R})}$ for the intersection of all the plausible supports, we can derive:

$$\begin{aligned}
&\max_{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}}} \Delta E(\vec{\phi},\theta,\tau,\mathcal{R}) = \max_{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}}} \left(\frac{\sum_{\pi_s\cap\vec{p}_{DSM}(\mathbb{Z}^2)} |z_{DSM}-z_s|^p - |z_{DSM}-z_{\mathcal{R}}|^p}{-p.\sigma^p} - L(s|\mathcal{R}) \right) \\
&\leq \max_{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}}} \frac{\sum_{\pi_s\cap\vec{p}_{DSM}(\mathbb{Z}^2)} |z_{DSM}-z_{\mathcal{R}}|^p}{p.\sigma^p} - L(s|\mathcal{R}) - \min_{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}}} \frac{\sum_{\pi_s\cap\vec{p}_{DSM}(\mathbb{Z}^2)} (|z_{DSM}-z_s|^p)}{p.\sigma^p} \\
&\leq \frac{\sum_{\theta\cap\vec{p}_{DSM}(\mathbb{Z}^2)} |z_{DSM}-z_{\mathcal{R}}|^p}{p.\sigma^p} - L(s|\mathcal{R}) - \frac{\sum_{\pi_\cap\cap\vec{p}_{DSM}(\mathbb{Z}^2)} \min_{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}}} (|z_{DSM}-z_s|^p)}{p.\sigma^p} \qquad (3.4.5)
\end{aligned}$$

Since all the terms $\left(\min_{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}}} (|z_{DSM}-z_s|^p)\right)$ are positive, a first upperbound can be computed in constant time as $\left(\frac{\sum_{\theta\cap\vec{p}_{DSM}(\mathbb{Z}^2)}|z_{DSM}-z_{\mathcal{R}}|^p}{p.\sigma^p} - L(s|\mathcal{R})\right)$, if a cumulative preprocessing of $|z_{DSM}-z_{\mathcal{R}}|^p$ has been performed like in section 3.4.1.2. It can then be tightened by subtracting the $\min_{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}}} (|z_{DSM}-z_s|^p)$ terms while they are evaluated to build the system to be minimized. Whenever, during the construction of the system, the upperbound falls below the input lowerbound, the expensive iterative $\mathcal{L}_p$ minimization may be skipped.

For the *Chimney*, *GlassRoof* and *Terrace* superstructures, the support is always equal to θ, thus $\pi_{(\vec{\phi},\theta,\tau,\mathcal{R})} = \pi_\cap = \theta$, which tightens the upperbound and simplifies its evaluation: all pointwise minimum residuals in $\pi_\cap = \theta$ may be subtracted.

3.5 Results

The generic $\mathcal{L}_p$ and optimized $\mathcal{L}_2$ metrics have been implemented. The $\mathcal{L}_p$ metric with $p < 2$ is more adapted than the $\mathcal{L}_2$ metric to the correlated, non-Gaussian noise typically present in correlation DSMs, but the computation and optimization of the latter is much faster. The computing time is dominated by the optimizations of the specific parameters ϕ. The roof plane of figure 3.11 contains 300 000 superstructure candidates with benefit $\Delta E(s) > 0$ and 100 locally maximum superstructures. Computing times typically range from a few seconds to a few minutes with the $\mathcal{L}_2$ metric, with roughly a 50% overhead for the $\mathcal{L}_p$ metrics, on a single Intel Xeon 1.60GHz CPU core. However, timings are very sensitive to the input data:

1. If the input roof planes are erroneous, our approach is not suitable, as it is likely to yield poor results after a long computing time. The reason is that the roof will have to be covered with utilitary superstructures that do not correspond to real superstructures but which only purpose is to correct the erroneous main roof height.
2. If some superstructures are very large, such as in 3.11, the DSM will present a large deviation from the roof plane and thus, the estimations of the altimetric parameters will not be pruned in these large areas (section 3.4.3.3).

Fortunately, the large majority of roofs supports only small-scale isolated superstructures, experimentally yielding the aforementionned computing times for an 0.1m horizontal quantization of $\Theta_{\tau,\mathcal{R}}$ and a vertically projected roof surface of about $100m^2$.

A ground truth has been generated by an operator using the images of the building of figure 3.13. This ground truth contains 46 chimneys and 40 glass roofs. Then the automatic $\mathcal{L}_2$ reconstruction

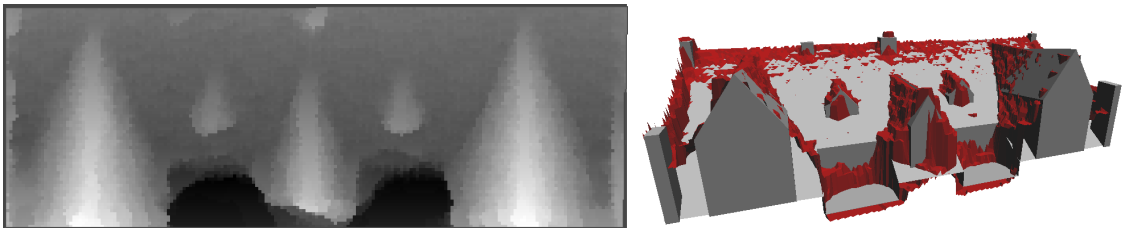


Figure 3.11: A DSM (left) and its reconstruction with a 3D-triangulation that represents the DSM (right).

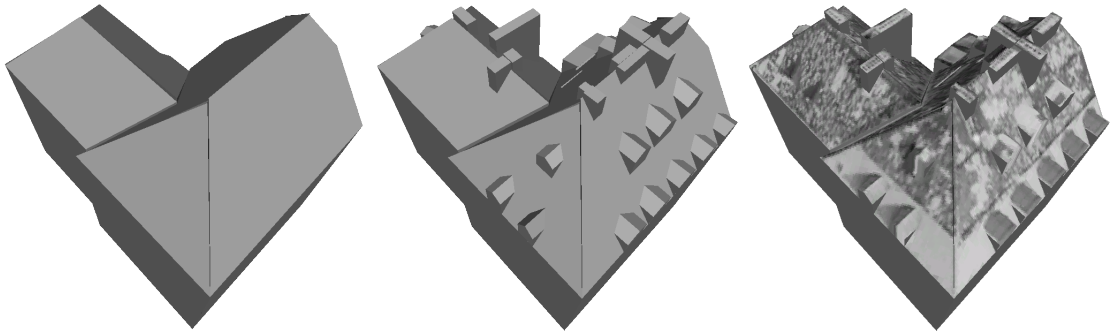


Figure 3.12: The input model (left), the reconstructed building (center) and its textured version (right) where each polygon is textured by the most front facing aerial image.

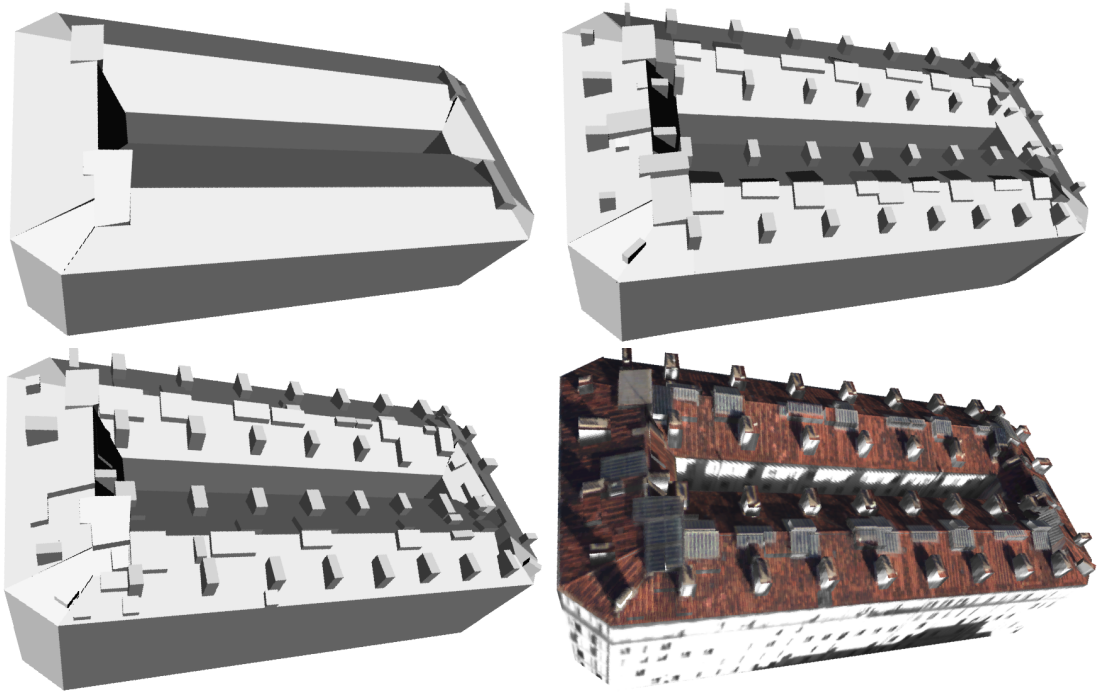


Figure 3.13: The input model, the ground truth reconstructed manually, the reconstructed building and its textured version.

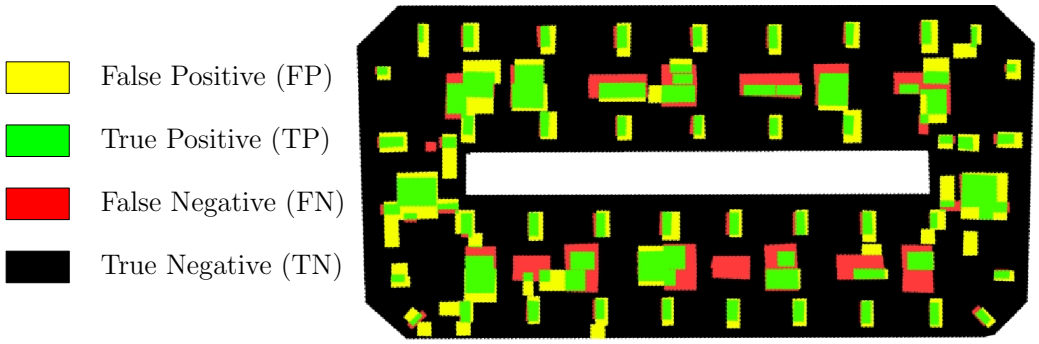


Figure 3.14: Evaluation of the classification of the DSM pixels as pixels of the superstructure supports.

has been performed.The surfacic false alarm rate is of 11% (9 overdetected glass roofs) and the detection rate is of 85% (1 underdetected chimney and 12 glass roofs) without taking into account the labeling errors : 5 chimneys and 7 glass roofs have been assigned the wrong superstructure type. Processing whole buildings at once rather than one roof plane at a time allows the reconstruction of superstructures that are contained in multiple roof planes, like chimneys crossing the rooftop. The false positive (FP) regions in figure 3.14 illustrate the general overestimation of the support, whereas the true negative (TN) regions are mainly caused by under-detection of superstructures. The under-detection is due to the difficulty to separate the imperfections of the DSM from the glass roof models which are allowed to have only a small height h.

$\mathcal{L}_2$ Detection	TP	FP	TN
Chimney	45	0	1
GlassRoof	28	9	12
Terrace	0	0	0
Dormer	0	0	0
superstructures	73	9	13

Table 3.4: Evaluation of the detection of the $\mathcal{L}_2$ detection on figures 3.13 and 3.14: the false alarm rate is of 11% and the detection rate is of 85%.

The table 3.4 investigates, on an object by object basis, if a detected superstructure corresponds to a true superstructure of the ground truth. It does not take into account the labeling errors : 5 chimneys have been incorrectly detected as glass roofs and 7 glass roofs have been detected as chimneys. This were mainly caused by the deficiencies in the DSM generation that are not taken into account in our method. The true negative (TN) chimney in table 3.4 is due to 2 true glass roofs and a true chimney that are detected as a single larger glass roof. The false positive (FP) glass roofs are due to the small volume of this object type and to the regularization of the DSM : small glass roofs may be detected near chimneys to take into account the inaccuracies of the DSM.

With a DSM resolution of 25cm, medium-sized and large superstructures are reconstructed correctly such as the 3 dormer windows of figure 3.15. But smaller structures are altered and hard to distinguish from noise. Coherent regions due to the regularization of the DSM are spuriously reconstructed as small superstructures. At these resolutions, the small superstructures should be disabled in the superstructure library or the MDL noise parameter σ has to be tuned up, because their signal is comparable to the imperfections of the DSM.

However, there appear to be a major drawback that increases computing times drastically as the $\Theta_{\tau,\mathcal{R}}$-defining discretization *step* of the explored superstructures decreases. This evolution is asymptotically of the order of $O(step^{-4})$ given that it corresponds to the size increase of the sets $\Theta_{\tau,\mathcal{R}}$ of rectangular supports ($O(step^{-2})$ quantized locations and $O(step^{-2})$ quantized dimensions).

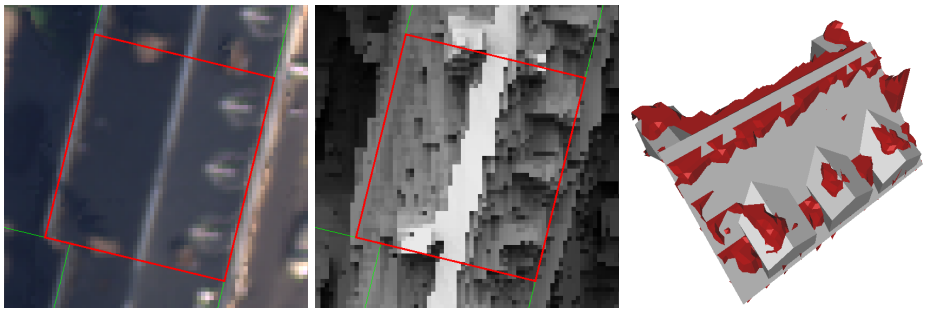


Figure 3.15: An image with 25cm resolution, its shaded DSM (same resolution), and the reconstructed building with a DSM triangulation.

Furthermore, the $O(step^{-4})$ assymptotic computing time is valid for constant-time per-hypothesis optimizations. This is however only true for the $\mathcal{L}_2$ error metric and specific superstructure types (see appendix A). In practice, the empirical exponent has been measured around -5.16 for the $\mathcal{L}_2$ metric, and -5.72 for the $\mathcal{L}_1$ metric (table 3.5). This yields that the presented approach is well-suited for superstructure reconstruction resolutions of 10cm or more, but that finer resolutions would need exagerately high computing times. Appendix B will present and discuss non-exhaustive variants that are more efficient (but presumably less robust) in these finer cases.

Horizontal quantization	$\mathcal{L}_2$	$\mathcal{L}_1$
0.7m	8.02s	9.59s
0.6m	12.94s	16.01s
0.5m	35.9s	47.6s
0.4m	105.98s	154.31s
0.3m	476.67s	845.76s
0.2m	5167.84s	12435.5s

Table 3.5: Sample superstructure reconstruction computing times for the difficult roof facet of figure 3.11. (Intel Xeon 1.60GHz CPU core)

When the data is of good quality, the prior noise may be set to zero ($\sigma \to 0$). This has the effect of making the model description length $L(\mathcal{B})$ negligible relative to the error term $D(\mathcal{B})$, yielding a parameter-less error-driven reconstruction. Nevertheless, setting $\sigma > 0$ becomes useful to prevent over-fit with noisier data.

3.6 Discussion

3.6.1 Library Extensibility

The library is extensible and the algorithm behaves linearly with the size of the number of superstructure types. To introduce a new superstructure type, one has just to provide 5 elements:

- d_τ : the dimension of the continuous parameter vector $\vec{\phi}$.
- $z_s(\vec{p})$: this height function provides the geometry and topology of a 2.5D superstructure.
- π_s : the support of a superstructure given its parameters $\theta, \vec{\phi}$ and the roof $\mathcal{R}$.
- $\Theta_{\tau,\mathcal{R}}$: the constraints on the allowable oriented bounding rectangles θ.
- $\Phi_{\theta,\tau,\mathcal{R}}$: the constraints on the allowable continuous parameters $\vec{\phi}$.

Nothing prevents superstructures to have non-planar faces ! For instance a dome may be introduced using a heightfield $z_s(\vec{p}) = \phi_1 + \phi_2\sqrt{1-\lambda(\vec{p})^2+\mu(\vec{p})^2}$ defined over a support $\pi_s =$

$\{\vec{p} \ / \ \lambda(\vec{p})^2 + \mu(\vec{p})^2 \leq 1\}$, with $d_\tau = 2$. Reasonnable restrictions on $\Theta_{\tau,\mathcal{R}}$ and $\Phi_{\theta,\tau,\mathcal{R}}$ would be to ensure that the dome has a a plausible size, is convex ($\phi_2 \geq 0$) and above the roof($\phi_1 \geq \max_{\vec{p}\in\pi_s}(z_{\mathcal{R}}(\vec{p}))$. If only half-spherical domes are allowed, then the restrictions $\|\vec{u}_\theta\| = \|\vec{v}_\theta\| = \phi_2$ are to be enforced.

3.6.2 Future work

The exhaustive search for superstructures developed in this chapter is conservative, in order not to miss superstructures. As a drawback it is slower than approaches that first detect zones of interests and focus on those zones. Two variations of the method developed in this chapter that uses one such approach are developed in Appendix B.

Possible extensions include :

Input data: lidar, images... Rather than using only the DSM, one could rather use the images directly and/or some lidar data to reconstruct the superstructures. A possibility is to use this other kind of data to fit reconstructed superstructures as a postprocess of the method developed in this chapter. Alternatively, the data term could use, for instance, image correlation, image features or distances to a lidar point cloud to measure how close is a building model with superstructures to the reality.

Regularization between superstructures There is currently no regularization between the different superstructures. One may want to add the prior that neighboring superstructures tend to share the same heights, width, length and slopes, that they are likely to be aligned and that groups of superstructures are often evenly separated. The easiest way to introduce this kind of prior is as a post-processing step, where similar superstructures are detected in order to snap their parameter to enforce the prior. This would require the introduction of some parameters controling the extent of the snapping. Furthermore, this regularization may move the superstructures and violate the disjoint support requirement.

On the other hand, this prior may be introduced beforehand in the search space or at the energy level. For instance, instead of searching for individual superstructures, regular groups of superstructures may be searched directly. The introduction in the energy of this regularization prior is likely to break the reduction of the minimization to a maximum weight clique problem. Methods based on Reversible Jump Monte Carlo Markov Chains (RJMCMC), already used in the building reconstruction applicative context [ODZ07], seem to be a good candidate to optimize the resulting more complex energies.

Dequantization of $\Theta_{\tau,\mathcal{R}}$ The discretization of the set of rectangles is required by the exhaustive search exposed in this chapter. However, it introduces an approximation in the horizontal localization of the superstructures. The stochastic diffusion scheme of Appendix B does not discretize the set of superstructure bounding boxes: they are drawn continuously at random in the neighborhood of a candidate superstructure.

Discretization of $\Phi_{\theta,\tau,\mathcal{R}}$ On the contrary, one may want to discretize all the superstructure parameters, including the altimetric parameters. This could be useful if the energy to estimate $\vec{\phi}_{max}$ is expensive to optimize. The altimetric accuracy of the estimation of $\vec{\phi}_{max}$ cannot be higher than the accuracy of the DSM, which is around $0.1m$ within the context of this thesis. Since each component of the plausible intervals $\Phi_{\theta,\tau,\mathcal{R}}$ span only a few meters, $\Phi_{\theta,\tau,\mathcal{R}}$ may be quantized into a reasonnably small number of discrete $\vec{\phi}$ values, provided that its dimension d_τ is not too high.

More precisely, this discretization is not useful for superstructures that have a constant-time parameter estimation (applying the optimizations of appendix A). In other cases, the evaluation

of a small number of quantified parameter values may be beneficial compared to an iterative continuous estimation.

Shadows Shape from shadows is an entire field of research. In our context, the time of acquisition is available, hence the exact position of the sun. As the images are fully calibrated, if the geometry of a plane is assumed to be exact, it is possible to get some knowledge on the silhouette of the superstructure casting a shadow on the given plane, as seen from the sun. However, this approach needs to access directly the images which is not necessary with our DSM-only approach. A major cave-at in such an approach is the interdependence between the superstructures and the roof. Superstructure shadows possibly span multiple roof planes, they may be cast on other superstructures, or even outside the roof surface.

Parallel computing The generation of the superstructure hypotheses and its estimation of the altimetric parameters are highly data-parallel tasks. We think that the use of General-Purpose computations on Graphics Processing Units (GPGPU) may significantly lower the computing costs of this processing step, which currently dominates the overall computing time. It is however unclear how to take advantage of this parallel processing power to solve the superstructure selection problem, as the maximum weighted clique problem is less obviously parallelizable.

3.7 Conclusion

The proposed method achieves a reasonably fast detection and reconstruction of buildings with roof superstructures using only a DSM, an initial building model without superstructures and an easily extensible collection of parametric models defining the available superstructure types. This approach gives convincing results and is fully automatic with 10cm data using the parameters $n = 2$ and $\sigma \to 0$. This method could be extended to lidar scanned point clouds.

Improving the geometric accuracy would require the direct use of the images rather than processing only the DSM. Such algorithms could either fit DSM-produced models to the images as a post-process as in [SB03] or produce those models directly from the images. The energy could be reformulated as a Bayesian energy that handle interactions between the superstructures and the roof planes, to introduce stronger priors such as alignments or to model the imperfections of the DSM.

Within this chapter, the input building is assumed to perfectly fit the input data. In practice, the input 3D model has an inaccurate geometry and even its topology may be erroneous: planes may be missing, small edges may be collapsed to a vertex adjacent to four or more planes... As the superstructure is a small modification of the base roof, if the roof itself is imprecise, the reconstructed superstructures may be arbitrarily erroneous. For instance, if the roof is biased by 30cm below the real roof, reconstructing superstructures is likely to produce a lot of low $GlassRoof$s.

The following part introduce a strategy to refine the geometry of the input polyhedral building model using the DSM, while relaxing the geometric constraints induced by the topology and updating the topology so that the polyhedron facets remain self-intersection free.

Part III

Topology-Aware Kinetic Fitting of Polyhedral Roofs

Chapter 4

Fixed Topology 3D Building Model Fitting

Contents

4.1 Introduction

As discussed in chapter 3, buildings are modeled using a closed polyhedron modified by a set of superstructure elements. To reach the objective of increasing the LOD of a building using an input DSM, we proposed an iterative approach (see chapter 2). We discussed that the input lower LOD building was likely to (1) miss fine detail superstructures and (2) have an improvable geometry. The overall approach is then to alternately optimize these two interdependent aspects, considering the other one is fixed. Chapter 3 dealt with the detection and reconstruction of superstructures given a fixed base polyhedron. On the other hand, this chapter and the two following ones provide a method to refine the geometry of the base polyhedral building, and its topology as necessary, given a DSM and superstructures that have already been reconstructed in a previous step.

4.1.1 Related Work

The polyhedron fitting problem has already been addressed with the restriction that the topology of the polyhedron was kept fixed, for instance in [VT05] within our applicative context. More precisely, [VT05] fits an input building model to a set of aerial images, aligning image gradients with the building wireframe projections. Furthermore, it automatically detects and enforces geometric constraints such as perpendicularity, orthogonality or symetry. It however keeps the polyhedral topology fixed and handles the resulting topology-induced geometric constraints similarly to the detected and enforced purely geometric constraints.

We recall that a fixed-topology optimization process is allowed to change the geometry of the polyhedron (the point and the plane coordinates) but not the incidence relations between its vertices, edges and facets. The topological property that a vertex is adjacent to more than 3 planes implies the geometric constraint that these planes have a non empty intersection. Likewise a facet bounded by more than 3 vertices imposes that their supporting points are all coplanar. These topology-induced geometric constraints cause a fixed-topology polyhedron optimization to be constrained in order to get a well-defined result.

These constraints are trivial to detect in the input topology and easy to take into account in the optimization process. However, they translate some kind of singularity of the polyhedron topology, which is in our context the result of some prior that a roof or façade facet is exactly coplanar, or that multiple roof or façade facets meet exactly at the same location. This prior knowledge is typically introduced when reconstructing buildings from unsufficiently accurate data. This introduces a necessary generalization of the building to get some robustness in the reconstruction. Since our goal is to refine a building model using a new and more accurate dataset, we would like to reduce the generalization of the input model, by discarding the topology-induced constraints. This chapter concentrates on the optimization of an unconstrained building, where all these constraints have been relaxed, leaving a selective relaxation of these constraints as an extension (see section 4.6.2).

The classical approach to avoid these constraints is to triangulate the facets of the optimized polyhedron, yielding an unconstrained optimization, where the unknowns are the point coordinates [HDD+93]. We argue, in section 4.4.1, that a dual approach is better suited to our problem. This *trihedralization* approach, developped in chapter 5, yields an alternate unconstrained optimization, where the unknowns are the plane coordinates, by splitting vertices adjacent to more than 3 facets.

Using the unconstrained plane estimates is not straightforward, as the unconstrained fitting of the polyhedral geometry is likely to yield self-intersecting or non-coplanar facets or undefined vertices at the intersection of planes that have an empty intersection. The issue is then to retrieve a polyhedral topology that yields a polyhedron with coplanar self-intersection free facets and well-defined vertex locations, that is as close as possible to the initial polyhedral topology. A fruitful approach [JPDPM00, TD04, LPK07] to build a polyhedron from its set of supporting planes uses the 3D arrangement of these planes [Grü71] (section 4.3.1). However, it is unclear how to take

intuitively and effectively the initial polyhedral topology into account.

[EAH08] addresses a related problem in the context of building reconstruction from a DSM. The proposed approach is a two stage process. First the DSM is partitionned into almost planar regions and then a polyhedron is exported from the partition. The DSM partition is performed by iteratively detecting planes using the well-known RANSAC technique [FB81]. Each partition region corresponds to an estimated plane, yielding the reconstructed polyhedral plane geometry. Furthermore, the partition induces a preliminary polyhedral topology: one can construct a polyhedron from the image topology of the partitionned DSM, by identifying partition regions with polyhedral facets, boundaries between regions with polyhedral edges, and boundary endpoints with vertices. For instance, a square of 2 by 2 pixels generates a vertex if its pixels belong to 3 or 4 distinct partition regions. Vertices are then located by intersecting their adjacent planes and edges are constructed by linking vertex locations. However, this preliminary topology of the partition is not readily usable as a polyhedral topology. Multiple complications may occur. Facets may self intersect, vertices adjacent to 4 facets are likely not well-defined, and *loop* partition regions surrounded by a single outside region define no vertex and therefore may not be exported. [EAH08] proposed to keep the semantics of the partition regions as unsplitted planar facets of the polyhedron. To achieve this goal, Small topological corrections are performed on the preliminary topology until it yields a well-defined self-intersection free polyhedron. These updates are rule-based and require many fine-tuned parameters. A vertical plane is added if the intersection edge of two neighboring planar regions is too far from the boundary between the two regions. The other local topological modifications are used to make the topology, coupled with the estimated plane geometry, refer to an acceptable polyhedron. Compared to our problem, the RANSAC-based partition yields the target plane geometry and an initial preliminary topology. Then a rule-based methodology is used to make the resulting polyhedron well-defined and self-intersection free.

A related approach is the variational shape approximation approach [CSAD04] that is able to fit an approximate polyhedral surface to an input polyhedral mesh. The basic idea is likewise to partition the input dataset into planar regions. The approach is intrinsically iterative and the result is reached when the partition converges. A partition update is carried out by first fitting a plane to each approximately planar partition region. Then a region growing algorithm is used to update the mesh partition according to the fitted planes. At convergence, this yields a partition of the input mesh with an approximate supporting plane for each partition region. The final polyhedron is then an export of the partition, that is only generated as a final processing step. It avoids the difficulties encountered by [EAH08] by not directly relying on the preliminary topology of the partition and the estimated planes to reconstruct the polyhedron. Instead, vertices of the partitioned mesh are defined as the barycenter of the intersections of triplets of adjacent planes. This reduces to the simple intersection of 3 planes for a vertex adjacent to 3 partition regions. However such a barycenter does not generally lie on the planes of its adjacent facets. Therefore the planarity of the partition regions is sacrificed, requiring the triangulation of their facets. Last but not least, the resulting facets may self intersect.

4.1.2 Overall Topology-aware Fitting Approach

The main idea of the topology-aware plane geometry refinement step introduced in this part of the dissertation is to design an algorithm where the polyhedron is always readily available as in [VT05], and let the polyhedron define the input data partition rather than having to export the polyhedron as a post process from the partition, as in [CSAD04]. We propose a framework that let the polyhedral topology be implicitly handled as in [CSAD04, EAH08], but that further guarantees self-intersection free facets and well-defined vertices. Furthermore, the algorithm should be robust without the fine parameter tuning of [EAH08].

To meet this objective, the proposed approach is primarily built on top of a partition of the input data, recycling ideas from [CSAD04, EAH08]. The polyhedron with self-intersection free facets is maintained explicitly throughout the optimization, in order to avoid its problematic final

export. Conversely, we propose to export the DSM partition from the maintained polyhedron. Therefore, the topology of the partition is intrisically compatible with the topology and geometry of a self-intersection free polyhedron.

The next two chapters contain our two main contributions to the field of computational geometry. They jointy let the topology vary implicitely, while guaranteeing a building model without self-intersecting facets. In order to make the optimization unconstrained, the necessary modifications of the topology, so that the geometry of the fitted polyhedron is not constrained by its topology, will be developed in chapter 5. Then, a new kinetic framework will be introduced in chapter 6 to keep self-intersection free polyhedral facets throughout the optimization.

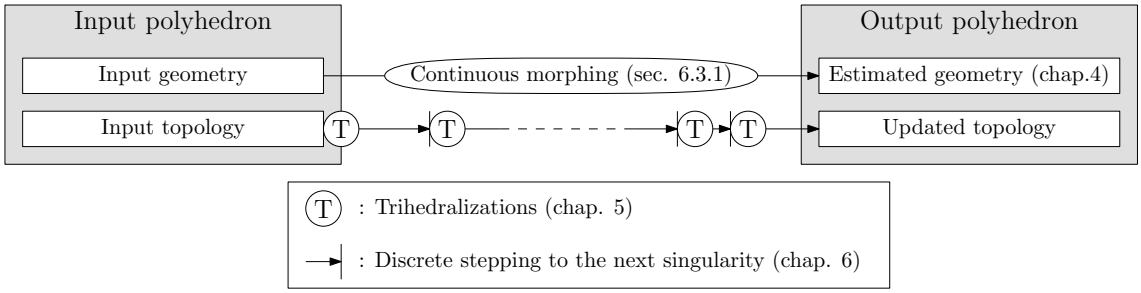


Figure 4.1: Overview of a single topology-aware fitting step.

Figure 4.1 illustrates a single polyhedron fitting step of the proposed approach, which is to be iterated until convergence of the fitting. Given the current input polyhedron, a refined supporting plane is estimated for each polyhedral roof facet (chapter 4). The supporting plane are then set to evolve continuously from the current to the re-estimated planes (section 6.3.1). In order to maintain a topology yielding self-intersection free facets, the input topology is modified as required during this continuous morphing. The trihedralization process of chapter 5 modifies the initial input topology to discard the input topology-induced constraints. The framework developped in chapter 6 is responsible for detecting intermediate self-intersections, that may be handled using the trihedralization process.

Now that the overall approach of part III has been detailed, we now turn to the material developped in this chapter, namely, the re-estimation of the plane geometry for each fitting step.

4.1.3 Outline

The usage of previously reconstructed superstructures to prevent them from biasing the roof plane estimation is the main contribution of this chapter. A straightforward fixed topology unconstrained optimization method is modified to take into account the superstructures that may have already been reconstructed at a previous iteration. The knowledge of the DSM regions that have been segmented as superstructure supports is used to give a more accurate estimation of the roof plane equations.

This chapter addresses the following issue:

Problem Statement

Assuming that the geometry of the polyhedral building model without superstructures is not constrained by its topology, how to use already reconstructed superstructures, to refine its geometry relative to a DSM, while keeping its topology fixed?

This chapter introduces the basic concepts and the modelization of the unconstrained opti-

mization framework. Section 4.2 presents shortly the concepts and tools of the oriented projective geometry that are required by this chapter and the two following ones. Then section 4.3 introduces the higher level geometric objects that are plane arrangements in 3D, their coloring, and the concept of polyhedron duality. Section 4.4 details how to reestimate the supporting planes of a polyhedron, disregarding topological constraints. Finally, section 4.6 discusses the proposed approach.

4.2 Oriented Projective Geometry

The geometry of Cartesian three-space $\mathbb{R}^3$ is greatly simplified by using the oriented projective space $\mathbb{T}^3$. An in-depth presentation of this space $\mathbb{T}^3$ can be found in [Sto91]. It retains the powerful representation and unification properties of the unoriented projective space $\mathbb{P}^3$ while preserving the orientation and separability of the Cartesian space $\mathbb{R}^3$.

Obviously, this section does not cover all the concepts of oriented projective geometry, but our intent is to introduce the notations and provide the required tools.

4.2.1 Primitives

Points: The homogeneous coordinates $\vec{P} = [x : y : z : w]$, or $[\vec{p} : w]$ with $\vec{p} = (x, y, z)$, refer, if $w \neq 0$, to the 3D point of Cartesian coordinates $\frac{\vec{p}}{w} = (\frac{x}{w}, \frac{y}{w}, \frac{z}{w})$. The multiplication of the homogeneous coordinates $\vec{P}$ by any non-zero factor leaves the Cartesian 3D point unchanged. The points $[\vec{p} : 0]$ refer to points at infinity in the direction $\pm\vec{p}$.

Planes: A plane equation is given by the homogeneous coordinates $\vec{N} = [a : b : c : d]$, or $[\vec{n} : d]$ with $\vec{n} = (a, b, c)$. It refers to the planar set of points $\vec{P}$ that verify $\vec{N} \cdot \vec{P} = ax + by + cz + dw = 0$, which is equivalent, if $w \neq 0$, to the Cartesian equation $\frac{\vec{p}}{w} \cdot \vec{n} + d = 0$. The normal of the plane is thus encoded in $\vec{n} = (a, b, c)$ and, if $\vec{n}$ is normalized, d represents the signed distance separating the plane $\vec{N}$ from the origin. The multiplication of the homogeneous coordinates $\vec{N}$ by any non-zero factor also leaves the plane unchanged. The direction of the normal vector $\vec{n}$ induces an orientation of the plane $\vec{N}$: a plane $\vec{N}$ multiplied by a negative factor thus parameterizes the same plane but with a reversed orientation. There is only one unoriented plane that does not contain any finite vertex: its coordinates are $[\vec{0} : d]$. This is the plane at infinity.

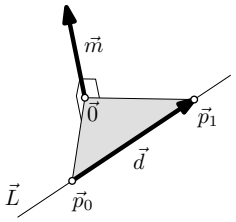


Figure 4.2: The Plücker coordinates of a line $\vec{L} = [\vec{d} : \vec{m}]$ passing through 2 cartesian points $\vec{p}_0$ and $\vec{p}_1$: $\vec{d} = \vec{p}_1 - \vec{p}_0$ and $\vec{m} = \vec{p}_0 \wedge \vec{p}_1$.

Lines: The Plücker coordinates, introduced by Julius Plücker in the 19th century, are a homogeneous parameterization of the 3D lines. The Plücker coordinates are composed of 6 coordinates, usually denoted using two 3D vectors: $\vec{L} = [\vec{d} : \vec{m}]$. The set of lines of the 3D Cartesian space $\mathbb{R}^3$ is 4-dimensional. One of the 2 extra degrees of freedom is due to the homogeneous nature of the Plücker coordinates: multiplying the Plücker coordinates by a scaling factor does not change the described geometric line. However a negative factor reverses its orientation. The second extra

degree of freedom is lost due to the quadratic constraint $\vec{d} \cdot \vec{m} = 0$. Geometrically, the vector $\vec{d}$ encodes the direction vector of the line and $\vec{m}$ the moment of the line: it is the normal of the plane that passes through the line and the origin (or the null vector if the line passes through the origin). Its magnitude encodes the distance of the line to the origin (fig 4.2).

4.2.2 Constructions

Matrix notations: The various constructions involve the computation of determinants of square matrices A, denoted $\det(A)$, or simply $|A|$ for readability when there is no confusion with the absolute value. Composite matrices $A = \begin{bmatrix}\vec{A}_1 & \vec{A}_2 & \dots & \vec{A}_n\end{bmatrix}$ are built by juxtaposing n vectors of equal size m as columns of the composite n by m matrix A.

Definition 12. *The $(i,j)^{th}$ cofactor $cof_{ij}(A)$ of a square matrix A of size n is its signed minor:*

$$cof_{ij}(A) = (-1)^{i+j} \begin{vmatrix} A_{1,1} & \cdots & A_{i-1,1} & A_{i+1,1} & \cdots & A_{n,1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ A_{1,j-1} & \cdots & A_{i-1,j-1} & A_{i+1,j-1} & \cdots & A_{n,j-1} \\ A_{1,j+1} & \cdots & A_{i-1,j+1} & A_{i+1,j+1} & \cdots & A_{n,j+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ A_{1,n} & \cdots & A_{i-1,n} & A_{i+1,n} & \cdots & A_{n,n} \end{vmatrix}$$

Definition 13. *The matrix of the cofactors of A is denoted* $\mathrm{com}(A)$*:* $(\mathrm{com}(A))_{ij} = cof_{ij}(A)$

Point construction from 3 Planes: The homogeneous coordinates $\vec{P} = [x : y : z : w]$ of the intersection point of 3 planes $\vec{N}_0$, $\vec{N}_1$, $\vec{N}_2$ with projective coordinates $[a_i : b_i : c_i : d_i]_{i=0\dots2}$ are computed using the cofactors $cof_{1j}(N)$ of the first column of the 4 by 4 matrix $N = \begin{bmatrix}\vec{0} & \vec{N}_0 & \vec{N}_1 & \vec{N}_2\end{bmatrix}$:

$$\vec{P} = [\vec{p} : w] = [cof_{11}(N) : cof_{12}(N) : cof_{13}(N) : cof_{14}(N)] \quad (4.2.1)$$

$$= \begin{bmatrix} + \begin{vmatrix} b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 \\ d_0 & d_1 & d_2 \end{vmatrix} : - \begin{vmatrix} a_0 & a_1 & a_2 \\ c_0 & c_1 & c_2 \\ d_0 & d_1 & d_2 \end{vmatrix} : + \begin{vmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ d_0 & d_1 & d_2 \end{vmatrix} : - \begin{vmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 \end{vmatrix} \end{bmatrix} \quad (4.2.2)$$

These are, up to their signs, the determinants of the 4 possible 3 by 3 submatrices of the 3 by 4 matrix $\begin{bmatrix}\vec{N}_0 & \vec{N}_1 & \vec{N}_2\end{bmatrix}$. It follows that the Cartesian point coordinates $\frac{\vec{p}}{w}$ of a trihedral vertex are rational functions in terms of the 12 plane coefficients $(a_i, b_i, c_i, d_i)_{i=0\dots2}$ of the 3 adjacent facets. Using the notation $\mathrm{com}(A)$ for the matrix of cofactors of A, $\vec{P}$ may be succinctly rewritten as:

$$\vec{P} = \left(\mathrm{com}\begin{bmatrix}\vec{0} & \vec{N}_0 & \vec{N}_1 & \vec{N}_2\end{bmatrix}\right) \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

A vertex is well-defined if and only if its 4 point coordinates are not all null. Remarkably, $w = cof_{1,4} = -\begin{vmatrix}\vec{n}_0 & \vec{n}_1 & \vec{n}_2\end{vmatrix}$ only depends on the plane normals. Thus, we retrieve the property that the intersection point of three planes is well-defined and finite if and only if $w \neq 0$: the normals of the planes are linearly independent. Degenerate intersections translate into an undefined resulting point $\vec{P} = [0 : 0 : 0 : 0]$. By its construction using the matrix of cofactors, the vector $\vec{P}$ is orthogonal to the 3 input planes, thus $\vec{P} \in \vec{N}_0 \cap \vec{N}_1 \cap \vec{N}_2$

Plane construction from 3 Points: Using the point-plane duality, by the symmetry of the containment relationship of a point on a plane ($\vec{N} \cdot \vec{P} = 0$), the same formula allows the computation of the plane coordinates of a 3D triangle:

$$\vec{N} = \left(\mathrm{com}\begin{bmatrix}\vec{0} & \vec{P}_0 & \vec{P}_1 & \vec{P}_2\end{bmatrix}\right) \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

Remark. *As a sanity check, we can check that those construction formulae are consistent by computing the supporting planes $\vec{N}'_0\vec{N}'_1\vec{N}'_2\vec{N}'_3$ from the intersection points $\vec{P}_{123}\vec{P}_{023}\vec{P}_{013}\vec{P}_{012}$ of a tetrahedron described by its 4 plane coordinates $\vec{N}_0, \vec{N}_1, \vec{N}_2$ and $\vec{N}_3$:*

$$\begin{aligned}\begin{bmatrix}\vec{N}'_0 & \vec{N}'_1 & \vec{N}'_2 & \vec{N}'_3\end{bmatrix} &= \operatorname{com}\begin{bmatrix}\vec{P}_{123} & -\vec{P}_{023} & \vec{P}_{013} & -\vec{P}_{012}\end{bmatrix} \\ &= \operatorname{com}\left(\operatorname{com}\begin{bmatrix}\vec{N}_0 & \vec{N}_1 & \vec{N}_2 & \vec{N}_3\end{bmatrix}\right) \\ &= \begin{vmatrix}\vec{N}_0 & \vec{N}_1 & \vec{N}_2 & \vec{N}_3\end{vmatrix}^2 \begin{bmatrix}\vec{N}_0 & \vec{N}_1 & \vec{N}_2 & \vec{N}_3\end{bmatrix}\end{aligned}$$

The cofactor matrix relations are due to a simple arrangement of the construction relations in a 4 by 4 matrix. The 4-dimensional identity $\operatorname{com}(\operatorname{com} A) = (\det A)^2 A$ shows that the 2 matrices of plane coordinates are equal up to a positive scaling factor, which is non null if the tetrahedron is not degenerate. Thus, the $\vec{N}_i$ and $\vec{N}'_i$ planes model the same oriented planes.

Line defined by a Point and a Direction: The Plücker coordinates of the line passing through a point $\vec{P} = [\vec{p} : w]$ in the direction of a 3D vector $\vec{n}$ are given by:

$$\vec{L} = \left[\vec{d} : \vec{m}\right] = [w\vec{n} : \vec{p} \wedge \vec{n}]$$

Line defined by 2 Points: The Plücker coordinates of the line passing through a disjoint pair of points $\vec{P}_0 = [\vec{p}_0 : w_0]$ and $\vec{P}_1 = [\vec{p}_1 : w_1]$ are given by:

$$\vec{L} = \left[\vec{d} : \vec{m}\right] = [w_0\vec{p}_1 - w_1\vec{p}_0 : \vec{p}_0 \wedge \vec{p}_1]$$

These 6 Plücker coordinates are, up to their sign, the determinants of the 6 possible 2 by 2 submatrices of the 4 by 2 matrix $[\vec{P}_0\ \vec{P}_1]^T = \begin{bmatrix} x_0 & y_0 & z_0 & w_0 \\ x_1 & y_1 & z_1 & w_1 \end{bmatrix}$.

Intersecting line of 2 Planes: The duality of the projective coordinates is obvious in the symmetry of the construction of the line passing through two points and the line at the intersection of two planes $\vec{N}_0 = [\vec{n}_0 : d_0]$ and $\vec{N}_1 = [\vec{n}_1 : d_1]$:

$$\vec{L} = \left[\vec{d} : \vec{m}\right] = [\vec{n}_0 \wedge \vec{n}_1 : d_0\vec{n}_1 - d_1\vec{n}_0] \tag{4.2.3}$$

Equivalently, these 6 Plücker coordinates are also, up to their sign, the determinants of the 6 possible 2 by 2 submatrices of the 4 by 2 matrix $[\vec{N}_0\ \vec{N}_1]^T = \begin{bmatrix} a_0 & b_0 & c_0 & d_0 \\ a_1 & b_1 & c_1 & d_1 \end{bmatrix}$.

Corollary 4.1. *Let $\vec{N}_0, \vec{N}_1, \vec{N}_2$ and $\vec{N}_3$ be four planes and let $\vec{P}_{123}$ and $\vec{P}_{023}$ be the points at the respective intersections of the planes $\vec{N}_1, \vec{N}_2, \vec{N}_3$ and $\vec{N}_0, \vec{N}_2, \vec{N}_3$ (Figure 4.3). Then the line $\vec{L}(\vec{P}_{123}, \vec{P}_{023})$ that passes through the points $\vec{P}_{123}$ and $\vec{P}_{023}$, is also the intersection line $\vec{L}(\vec{N}_2, \vec{N}_3)$ of the planes $\vec{N}_2$ et $\vec{N}_3$. Thus, there are two ways to compute the Plücker coordinates of this line. The two homogeneous representations $\vec{L}(\vec{P}_{123}, \vec{P}_{023})$ and $\vec{L}(\vec{N}_2, \vec{N}_3)$ of the same geometric line are equal up to the scaling factor $\begin{vmatrix}\vec{N} & \vec{N}_0 & \vec{N}_1 & \vec{N}_2\end{vmatrix}$:*

$$\vec{L}(\vec{P}_{123}, \vec{P}_{023}) = \begin{vmatrix}\vec{N}_0 & \vec{N}_1 & \vec{N}_2 & \vec{N}_3\end{vmatrix} \vec{L}(\vec{N}_2, \vec{N}_3)$$

Proof. Given that the proofs of the 6 scalar relations follow the same derivation, let us only prove the last equation: $(x_{123}y_{023} - x_{023}y_{123}) = (\det N)(c_3d_2 - c_2d_3)$, with the 4 by 4 matrix $N = \begin{bmatrix}\vec{N}_0 & \vec{N}_1 & \vec{N}_2 & \vec{N}_3\end{bmatrix}$.

If $\det N = 0$, then the points $\vec{P}_{123}$ and $\vec{P}_{023}$ are linearly dependant, which proves that $\vec{L}(\vec{P}_{123}, \vec{P}_{023}) = \vec{0}$. Assuming now that N is invertible, the 4 planes $\vec{N}_i$ define 4 distinct intersection points: $\begin{bmatrix}\vec{P}_{123} & -\vec{P}_{023} & \vec{P}_{013} & -\vec{P}_{012}\end{bmatrix} =$

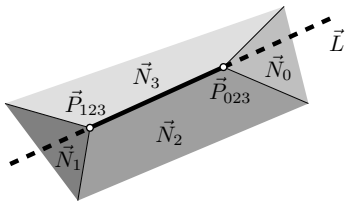


Figure 4.3: Corollary 4.1 notations: the line $\vec{L}$ that passes through 2 points $\vec{P}_{123}$ and $\vec{P}_{023}$ is also at the intersection of the 2 planes $\vec{N}_2$ et $\vec{N}_3$.

$\operatorname{com} N$. Given that $N^T \operatorname{com} N = (\det N)I$ (denoting by I the 4 by 4 identity matrix), it follows that:

$$N^T \begin{bmatrix} \vec{P}_{123} & -\vec{P}_{023} & \vec{P}_{013} & -\vec{P}_{012} \end{bmatrix} = (\det N) I$$

Thus, modifying the last 2 columns,
$$N^T \begin{bmatrix} \vec{P}_{123} & -\vec{P}_{023} & \begin{matrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{matrix} \end{bmatrix} = \begin{bmatrix} \det N & 0 & c_0 & d_0 \\ 0 & \det N & c_1 & d_1 \\ 0 & 0 & c_2 & d_2 \\ 0 & 0 & c_3 & d_3 \end{bmatrix}$$

Applying the determinant,
$$(\det N)(-x_{123}y_{023} + x_{023}y_{123}) = (\det N)^2(c_2d_3 - c_3d_2)$$

Dividing by $(-\det N)$,
$$(x_{123}y_{023} - x_{023}y_{123}) = (\det N)(c_3d_2 - c_2d_3)$$

□

4.2.3 Measures

Point-Plane Signed Distance: To measure the signed distance of a finite point $\vec{P} = [\vec{p} : w]$ to an oriented plane $\vec{N} = [\vec{n} : d]$, one has to compute the dot product of the normalized point and plane coordinates:

$$Distance(\vec{P}, \vec{N}) = \frac{\vec{P}}{w} \cdot \frac{\vec{N}}{|\vec{n}|}$$

This extends the incidence test $\vec{P} \cdot \vec{N} = 0$.

Triangle Signed Area: The signed area of a finite triangle with points $\vec{P}_i = [\vec{p}_i : \vec{w}_i]$ can be computed relative to the given normal $\vec{n}$ of the oriented plane $\vec{N} = [\vec{n} : d]$ that supports the triangle:

$$Area(\vec{N}, \vec{P}_0, \vec{P}_1, \vec{P}_2) = \frac{\begin{vmatrix} \vec{n} & \vec{p}_0 & \vec{p}_1 & \vec{p}_2 \\ 0 & w_0 & w_1 & w_2 \end{vmatrix}}{2|\vec{n}| w_0 w_1 w_2}$$

Proof. The proof uses the cartesian relation that the vector product $\left(\frac{\vec{p}_1}{w_1} - \frac{\vec{p}_0}{w_0}\right) \wedge \left(\frac{\vec{p}_2}{w_2} - \frac{\vec{p}_0}{w_0}\right)$ computes a vector collinear with the normal $\vec{n}$, the length of which is twice the signed area of the triangle:

$$\left(\frac{\vec{p}_1}{w_1} - \frac{\vec{p}_0}{w_0}\right) \wedge \left(\frac{\vec{p}_2}{w_2} - \frac{\vec{p}_0}{w_0}\right) = 2\ Area(\vec{N}, \vec{P}_0, \vec{P}_1, \vec{P}_2)\ \frac{\vec{n}}{|\vec{n}|}$$

$$\Rightarrow \quad \vec{n} \cdot \left(\frac{\vec{p}_1}{w_1} - \frac{\vec{p}_0}{w_0}\right) \wedge \left(\frac{\vec{p}_2}{w_2} - \frac{\vec{p}_0}{w_0}\right) = 2\ Area(\vec{N}, \vec{P}_0, \vec{P}_1, \vec{P}_2)\ \frac{\vec{n} \cdot \vec{n}}{|\vec{n}|}$$

$$\Rightarrow \quad \frac{1}{w_0 w_1 w_2} \begin{vmatrix} \vec{n} & \vec{p}_0 & \vec{p}_1 & \vec{p}_2 \\ 0 & w_0 & w_1 & w_2 \end{vmatrix} = 2|\vec{n}|\ Area(\vec{N}, \vec{P}_0, \vec{P}_1, \vec{P}_2)$$

$$\Rightarrow \quad Area(\vec{N}, \vec{P}_0, \vec{P}_1, \vec{P}_2) = \frac{\begin{vmatrix} \vec{n} & \vec{p}_0 & \vec{p}_1 & \vec{p}_2 \\ 0 & w_0 & w_1 & w_2 \end{vmatrix}}{2|\vec{n}| w_0 w_1 w_2}$$

□

By considering a horizontal plane $\vec{N}_h = [0:0:1:d]$, and denoting by (x_i, y_i) the cartesian 2D coordinates of a point $\vec{P}_i$, the well-known 2D formula may be retrieved:

$$Area(\vec{N}_h, \vec{P}_0, \vec{P}_1, \vec{P}_2) = \frac{1}{2}\begin{vmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{vmatrix}$$

Tetrahedron Signed Volume: The signed volume of a finite tetrahedron with points $\vec{P}_{i=0\ldots3}$ and planes $\vec{N}_{i=0\ldots3}$ is given equivalently by:

$$\begin{aligned} Volume(\vec{P}_0, \vec{P}_1, \vec{P}_2, \vec{P}_3) &= \frac{\left|\vec{P}_0 \quad \vec{P}_1 \quad \vec{P}_2 \quad \vec{P}_3\right|}{6w_0w_1w_2w_3} \\ Volume(\vec{N}_0, \vec{N}_1, \vec{N}_2, \vec{N}_3) &= \frac{\left|\vec{N}_0 \quad \vec{N}_1 \quad \vec{N}_2 \quad \vec{N}_3\right|^3}{6\left|\vec{n}_1 \quad \vec{n}_2 \quad \vec{n}_3\right|\left|\vec{n}_0 \quad \vec{n}_2 \quad \vec{n}_3\right|\left|\vec{n}_0 \quad \vec{n}_1 \quad \vec{n}_3\right|\left|\vec{n}_0 \quad \vec{n}_1 \quad \vec{n}_2\right|} \end{aligned}$$

The sign depends on the ordering of the points or planes.

4.2.4 Predicates

Definition 14 (Predicate)**.** *A predicate is a function which takes a structured set of geometric objects as input and produces one of a discrete set of outputs.*

A predicate is the basic geometric tool that queries the geometric configuration of the input geometric objects. The output typically coresponds to the sign of a function of the coordinates of the input geometric objects. For instance, is a point inside, on or outside a sphere? The *in sphere* predicate evaluates the sign of the difference of the squared sphere radius and the squared distance from the sphere center to the query point: positive is inside, negative is outside and null is on the spherical surface.

Point Above Plane Predicate: To test if a finite point $\vec{P} = [\vec{p} : w]$ is above a plane $\vec{N} = [\vec{n} : d]$, it suffices to evaluate the sign of the following expression:

$$Above(\vec{P}, \vec{N}) = sign\left(\frac{\vec{P}\cdot\vec{N}}{w|\vec{n}|}\right) = sign\left(\frac{\vec{P}\cdot\vec{N}}{w}\right)$$

If the sign is null ($\vec{P}\cdot\vec{N} = 0$), then the point lies within the plane. If it is strictly positive, the point in the halfspace delimited by the plane pointed by the plane normal. Otherwise, it is in the other halfspace.

This formula illustrates the fact that we are, here, not interested in the orientations of the points (the sign of their homogeneous coordinates w), but only on the orientations of the planes.

Triangle Orientation Predicate: The orientation of the finite triangle $\vec{P}_0\vec{P}_1\vec{P}_2$ on the oriented plane $\vec{N} = [\vec{n} : d]$ is determined by the sign of its signed area:

$$Orientation(\vec{N}, \vec{P}_0, \vec{P}_1, \vec{P}_2) = sign\left(\frac{\begin{vmatrix} \vec{n} & \vec{p}_0 & \vec{p}_1 & \vec{p}_2 \\ 0 & w_0 & w_1 & w_2 \end{vmatrix}}{w_0w_1w_2}\right) \tag{4.2.4}$$

A strictly positive, null or strictly negative sign denotes respectively a direct, aligned or indirect triangle with respect to the orientation defined by $\vec{n}$.

Proof. Using the definition of the area and the assumption that, since the points are finite, the plane is finite too($\vec{n} \neq \vec{0}$):

$$Orientation(\vec{N}, \vec{P}_0, \vec{P}_1, \vec{P}_2) = sign\left(Area(\vec{N}, \vec{P}_0, \vec{P}_1, \vec{P}_2)\right) = sign\left(\frac{\begin{vmatrix} \vec{n} & \vec{p}_0 & \vec{p}_1 & \vec{p}_2 \\ 0 & w_0 & w_1 & w_2 \end{vmatrix}}{2|\vec{n}|w_0 w_1 w_2}\right)$$

□

4 Points Coplanarity/4 Planes Meet 4 points $\vec{P}_{i=0\ldots3}$ are coplanar or the intersection of 4 planes $\vec{N}_{i=0\ldots3}$ is not empty if and only if they define a degenerate tetrahedron of volume 0. Assuming finite points, this is equivalent to

$$\left|\vec{P}_0 \quad \vec{P}_1 \quad \vec{P}_2 \quad \vec{P}_3\right| = \left|\vec{N}_0 \quad \vec{N}_1 \quad \vec{N}_2 \quad \vec{N}_3\right| = 0$$

2 Lines Coplanarity: 3D Lines are either skew or coplanar. To determine whether two lines $\vec{L}_0 = [\vec{d}_0 : \vec{m}_0]$ and $\vec{L}_1 = [\vec{d}_1 : \vec{m}_1]$ are coplanar, one has to check whether the quantity $\vec{m}_0 \cdot \vec{d}_1 + \vec{m}_1 \cdot \vec{d}_0 = 0$. When the lines are skew, the sign of $\left(\vec{m}_0 \cdot \vec{d}_1 + \vec{m}_1 \cdot \vec{d}_0\right)$ indicates the direction of crossing: positive if a right-handed screw takes L_0 into L_1, else negative. This permuted dot product $\vec{m}_0 \cdot \vec{d}_1 + \vec{m}_1 \cdot \vec{d}_0$ is denoted by $\vec{L}_0 \odot \vec{L}_1$. The quadratic Plücker relation $\vec{m} \cdot \vec{d} = 0$ ensures that a line is coplanar with itself.

Remark. *A straightforward derivation proves that* $\vec{L}(\vec{P}_0, \vec{P}_1) \odot \vec{L}_1(\vec{P}_2, \vec{P}_3) = \left|\vec{P}_0 \quad \vec{P}_1 \quad \vec{P}_2 \quad \vec{P}_3\right|$*: a tetrahedron is degenerate if and only if the supporting lines of two non adjacent edges of the tetrahedron are coplanar.*

4.3 Polyhedra and Plane Arrangements

Now that the basic geometric elements have been introduced, higher level geometric object can be defined: 3D plane arrangements and polyhedra.

4.3.1 Plane Arrangements

An intuitive description of a plane arrangement [Grü71] in 3D is using a volume of clay (representing $\mathbb{R}^3$), successively cut into halves along each 3D plane. This produces many convex polyhedral pieces of clay, which are the cells of the 3D plane arrangement. The **plane arrangement** generated by a set of 3D planes is a combinatorial structure that describes the topology of the partition of the space $\mathbb{R}^3$ induced by successive cuttings by these planes. Its combinatorial elements are:

3D cells are the maximal subsets of $\mathbb{R}^3$ that are not intersecting any plane.

2D facets are the maximal subsets of one of the planes that are not intersecting any other plane.

1D edges are the maximal subsets of a line, which is the intersection of 2 (or more) planes, that are not intersecting any other plane.

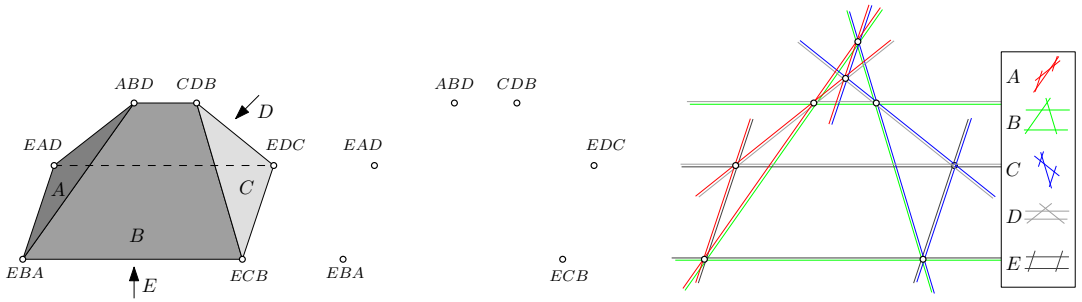
0D vertices are the non-empty intersections of subsets of the planes, that are reduced to a single point.

A 3D plane arrangement is not degenerate if the intersection of its planes are in general position (*i.e.* the intersection of k planes has dimension $(3-k)$ if $k \leq 3$ and is empty if $k \geq 4$). Assuming the arrangement is not degenerate, its combinatorial elements can be counted. An arrangement of n planes has at most $\sum_{i=3-k}^{3} \binom{i}{3-k}\binom{n}{i}$ combinatorial elements of degree k, which translates into $\frac{n(n-1)(n-2)}{6}$ vertices, $\frac{n(n-1)^2}{3}$ edges, $\frac{n^3-n^2+2n}{2}$ facets and $\frac{n^3+5n+6}{6}$ cells (at most $\frac{n^3-6n^2+11n+6}{6}$ of

which are bounded). The maximal number of combinatorial elements is reached when the arrangement is not degenerate [EOS86]. Thus, except for high degeneracies, the size of the arrangement is $O(n^3)$. It is in general difficult to represent a partition of $\mathbb{R}^3$ on a flat sheet of paper. Nevertheless, figure 4.4.c illustrates the vertices, and edges of a simple arrangement of 5 planes only.

4.3.2 Polyhedron Duality

Section 3.2.1.2 defined a polyhedron as a bounded volume with a piecewise-planar boundary. Its boundary representation can be subdivided into its continuous geometry and its combinatorial topology. The topology describes an abstract polyhedron without any geometric embedding. A facet is only a topological primitive, its geometric counterpart is its supporting plane. Likewise, a vertex is only a topological primitive too, and its geometric location is given by a point in 3-space. Finally, the same relation holds between the topological edges and their supporting 3D lines.



A 5-sided Polyhedron with facets supported by the planes A, B, C, D and E. Points are denoted with the 3 intersecting planes.

(a)

Primal geometry
Point coordinates:
$\vec{P}_{EBA}$=[+1:−2:0:1], $\vec{P}_{ECB}$=[+1:+2:0:1], $\vec{P}_{EDC}$=[−1:+2:0:1], $\vec{P}_{EAD}$=[−1:−2:0:1], $\vec{P}_{ABD}$=[0:$-\frac{1}{2}$:2:1], $\vec{P}_{CDB}$=[0:$\frac{1}{2}$:2:1]

(b)

Dual geometry
Plane coordinates:
$\vec{N}_A$=[0:−4:3:−8], $\vec{N}_B$=[+2:0:1:−2], $\vec{N}_C$=[0:+4:3:−8], $\vec{N}_D$=[−2:0:1:−2], $\vec{N}_E$=[0:0:−1:0]

(c)

Figure 4.4: (a) A polyhedron with 5 facets A, B, C, D and E. Its geometric embedding may either be given by (b) the point coordinates of its vertices (which generates a point cloud) or (c) the plane coordinates of its facets (which generates a plane arrangement).

To fully describe a polyhedron, the point coordinates of all its vertices may be given, as in figures 4.4.a and 4.4.b. Then, the equations of the planes that support each of the polyhedron facets are simply a by-product of the point coordinates of the vertices adjacent to each facet, assuming that these points are indeed coplanar and not aligned. Figure 4.5 shows a polyhedron (b) and two variants (a) and (c) that are the result of, respectively, geometric and topological modifications of the polyhedron (b).

4.3.2.1 Duality

A number of computational geometry problems may be considered using a second viewpoint: their **dual** problem. Basically, the 3D duality reverses the roles of the points and the planes, and of the vertices and the facets.

The 4D homogeneous vector of the plane coordinates are considered as points in the dual space. On the other hand the 4D homogeneous coordinate vector of a point are viewed as plane coordinates: this dual plane is the set of all the dual points (*i.e.* planes) that passes through the initial point. By opposition to the **dual** view of the problem, the initial problem is called **primal**. Using the symmetry of the projective geometry formalism, the duality is seamless and the dual of a dual problem is indeed the primal one. Lines $[\vec{m} : \vec{d}]$ are mapped onto lines $[\vec{d} : \vec{m}]$ by swapping the $\vec{m}$ and $\vec{d}$ parts of their coordinates.

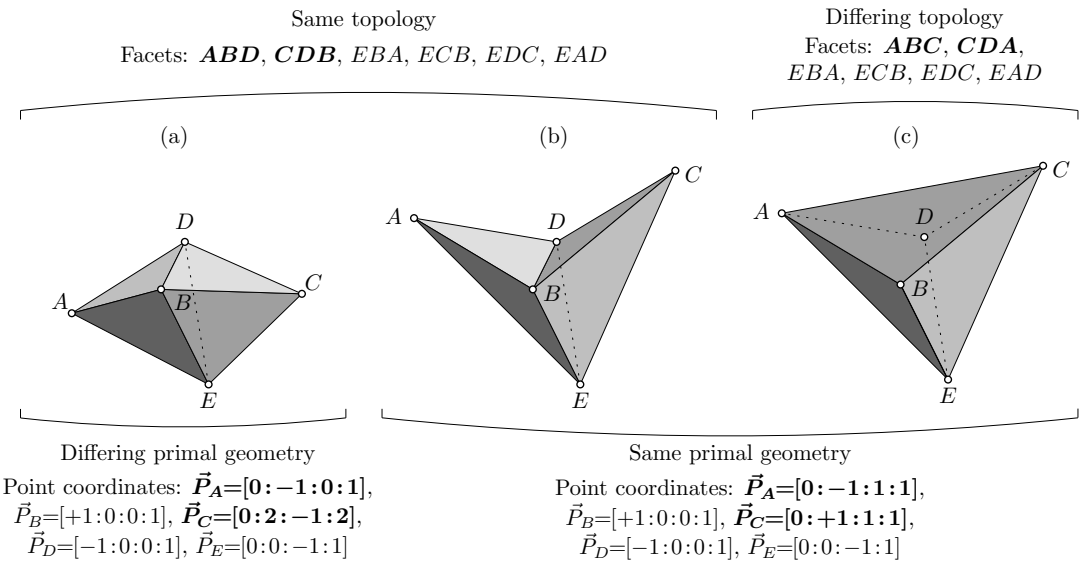


Figure 4.5: Three 6-sided simple polyhedra with vertices A, B, C, D and E. (a) and (b) have the same topology. (b) and (c) have the same point coordinates(*i.e.* the same **primal** geometry). The vertices are located using their homogeneous point coordinates $\vec{P}$ (section 4.2).

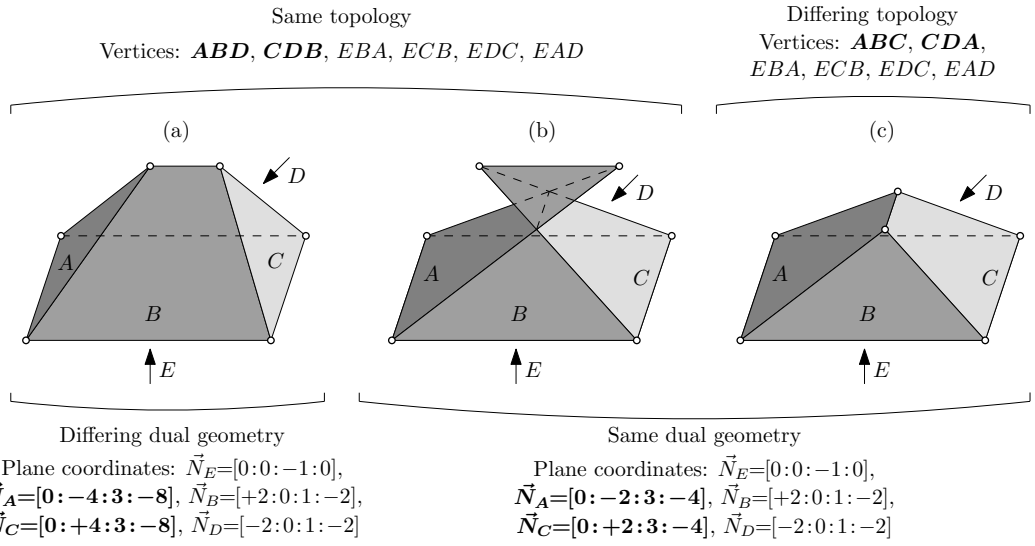


Figure 4.6: Three 5-sided polyhedra with facets A, B, C, D and E. (a) and (b) have the same topology. (b) and (c) have the same plane coordinates(*i.e.* the same **dual** geometry). (a) and (c) are simple, and (b) is an example of a polyhedron that is not simple: the polygonal facet supported by the plane B is auto-crossing (and so is the one supported by the plane D). The facets are located using their homogeneous plane coordinates $\vec{N}$ (section 4.2).

The dual topology can be seen as keeping the same adjacencies between the vertex, edge and facet topological objects, and only considering vertex nodes as facet nodes and vice-verse. As an example one can note that the triangular facets of figure 4.5 and trihedral vertices of figure 4.6 have the same letter labels. The topologies of figures 4.5 and 4.6 are dual.

To give a geometric embedding to the abstract polyhedron defined by a topology, the **dual geometry** may be given instead of the primal geometry: *i.e.* the equations of the planes that support the facets of the polyhedron (Figure 4.4.a and 4.4.c). Then, the point coordinates of each vertex may be computed by intersecting the supporting planes of its adjacent facets, assuming that the intersections of these planes are not degenerate.

4.3.3 Arrangement Coloring

Instead of the 3D point cloud of the primal geometry(Figure 4.4.b), the dual geometry defines a set of 3D planes(Figure 4.4.c). The topology of a polyhedron described by its dual geometry then only states which facets of the plane arrangement are part of a polyhedron facet, or which connected set of 3D cells of the arrangement are inside the closed polyhedron. Thus an alternate description of the geometry of a polyhedron is a 0-1 coloring of the cells of the arrangement of the supporting planes of its facets. Without loss of generality, a cell tagged 0 is outside the polyhedron and cells tagged 1 are inside its volume. Cells are said to be connected if they share an adjacent facet. We define the following properties over 0-1 colorings of a given arrangement of planes:

Bounded: A coloring is bounded if the unbounded cells are tagged 0. A bounded coloring thus defines a bounded piecewise linear surface.

Hole-free: A coloring is hole-free if all its bounded cells are connected to an unbounded cell of the same color using a path of identically colored cells.

Compatible: A coloring is said to be **compatible** with the plane orientations if no facet normal, given by the first three supporting plane coordinates $\vec{n} = (a, b, c)$, points from a 0-colored cell to a 1-colored cell.

Such colorings of a partition of $\mathbb{R}^3$ are handy. For instance the maximum flow class of surface reconstruction algorithms [RC98] can be seen as computing a coloring of the partition of a bounded volume in $\mathbb{R}^3$. Furthermore, a 3D partition coloring has the nice property to always describe an orientable surface.

4.3.4 Polyhedron Properties

4.3.4.1 Topological only properties

Some properties may be defined without any knowledge of the polyhedron geometry. To state geometric properties based on the topology alone, provided that the unknown geometry is not pathological, some of these properties invoke a non-degenerate geometry assumption. In particular, such an assumption ensures that the intersection of respectively 2, 3 and 4 or more planes is respectively a line, a point and an empty set. We define the following **topological** properties on polyhedron elements:

- The **valence** of a vertex is the number of facets that meet at this vertex. It maps to the number of edges of the corresponding dual facet in the dual polyhedron.
- A **trihedral vertex** is a vertex adjacent to exactly 3 facets - its valence is 3. It thus maps to a triangular face in the dual polyhedron. If the geometry of the polyhedron is not degenerate, the point coordinates of one of its vertices may be uniquely determined from the plane coordinates of its adjacent facets, if and only if this vertex is trihedral.
- A **triangulated facet** is a face that is subdivided by recursively splitting it with edges (called **diagonals**) between 2 of its vertices until the facet is made of triangles only. Those subdivision edges are called **soft** edges as opposed to the initial non-subdividing **hard** edges

of the polyhedron. Note that this is only the definition of an abstract triangulation (see section 5.4.1: there is no requirement on the underlying geometry, such as a consistent orientation of the triangles.
– An edge of a polyhedron may be defined by its 2 vertices or its 2 incident facets. A **non-degenerate edge** is an edge for which the only facets that are adjacent to both vertices of the edge are its incident facets. It follows that, a degenerate edge is an edge that is constrained to have a null length because of the topology only, without having to specify further the geometry of the polyhedron. A polyhedron can always be modified to describe the same surface without degenerate edges, yielding a simpler representation.

4.3.4.2 Geometric and Topological properties

The following properties also involve **geometric** requirements:
– A vertex is **under-constrained** if the dimension of the intersection of the supporting planes of its adjacent facets is strictly positive. The point coordinates of those vertices cannot be uniquely defined using the topology and the dual geometry. In dual space, this corresponds to a facet defined by aligned points.
– A vertex is **over-constrained** if the intersection of the supporting planes of its adjacent facets is empty. Those vertices have undefined point coordinates using the topology and the dual geometry. This is the dual setup of a facet defined by non-coplanar points.
– A vertex is **well-defined** if it is neither under-constrained, nor over-constrained: the intersection of the supporting planes of its adjacent facets is a unique point (possibly at infinity). Disregarding geometric degeneracies, a vertex is well-defined if and only if it is trihedral, as the unique intersection point of its three adjacent supporting planes in general position.
– A **simple facet** is a facet of a polyhedron, the geometry of which is a simple (*i.e.* self-intersection free) planar 3D polygon (possibly with holes), where a 3D polygon is a 2D polygon embedded in an arbitrary plane (see section 3.2.1.1).

A vertex is caracterized by the rank of the N by 4 matrix formed by concatenating its N adjacent supporting plane coordinates. A respectively under-constrained, over-constrained or well-defined vertex corresponds to a rank that is respectively less, greater or equal to 3. The coordinates of a well defined vertex can be computed using any 3 of its adjacent planes that do not have linearly dependant coordinates.

A **trihedral, well-defined, simple, triangulated or non-degenerate polyhedron** is defined by extension when all its qualified primitives (vertices, edges or facets) satisfy the property. In a more general setting, a simple polyhedron not only require its facets to be simple, but also to have a **proper intersection**, that is, the intersection of the 3D polygons with holes supporting any pair of facets must be equal to the set of segments supporting the edges they share. Within our context however (see section 3.2.1.2), the modeled polyhedra have a single bottom-facing facet that is set sufficiently low as to have a proper intersection with all the other facets. This implies that the facets that model a building have a proper intersection. An extension to more general polyhedra is discussed in section 6.6.2.

If a vertex is well-defined but not trihedral, there is a constraint between the plane equations of the planes that support the polyhedron facets adjacent to the non-trihedral vertex. Namely, for each 4-tuple of distinct adjacent facets that are incident to a common vertex, the determinant of their 4 homogeneous plane coordinates is null. Those constraints are geometric constraints induced by topological singularities. When the geometric embedding of a polyhedron is given by its primal geometry, this concurrency constraint is verified by construction.

The other type of such constraints, which is the dual of the previous one, occurs when a facet is delimited by more than 3 vertices. This topological singularity induces the following geometric constraints: all their points must be coplanar. When the geometric embedding of a polyhedron is given by its dual geometry, the coplanarity constraint is verified by construction.

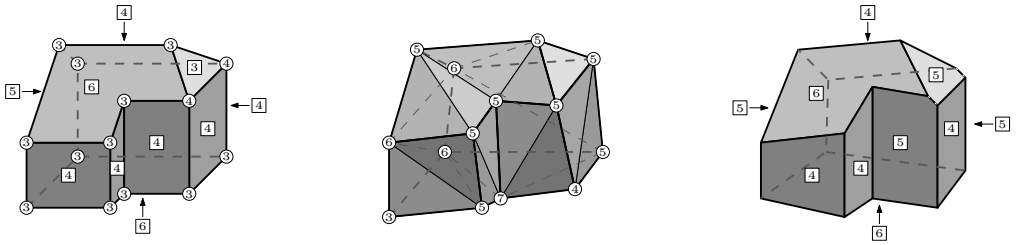
4.4 Dual Geometry Refinement

The purpose of this section is to reestimate the supporting planes of a polyhedron (*i.e.* its dual geometry), disregarding topological constraints. Before the formulation of the minimized energy, we describe how the polyhedron is represented so that it has no constraints induced by the topology, but yet its points are coplanar by construction. Then, the following section details the iterative fitting algorithm, before the discussion of the last section.

4.4.1 Using the Dual Polyhedron

We argue that relaxing the topology-induced geometric constraints gives more degrees of freedom, and thus allows a better fit of the building polyhedron. These degrees of freedom are gained at the cost of a reduced robustness. In a context where the fitted data is accurate enough, this topology-induced robustness is not required and has the only property of hindering a good fit. For instance, these topology-induced constraints may translate a prior that made sense at the original level of generalization, but are now too simplistic at the desired LOD. In this context, we are facing an alternative between two means to make the optimization unconstrained:

- The **primal** approach (fig. 4.4.b) is to optimize the point coordinates under the geometric constraint that vertices of each facet remain coplanar. Without those geometric constraints, the point coordinate estimation would not guarantee coplanar points in each facet. Thus, all the facets would have to be triangulated to get a topology consistent with the primal geometry. This approach is thus preferred when there are only a few vertices per facet and that the minimization criterion involves more naturally the point coordinates than the plane coordinates.
- The proposed **dual** approach (fig. 4.4.c) is to optimize the plane coordinates directly, so that the points are coplanar by construction. The drawback is that a polyhedron described by its dual geometry has a singular topology if its vertices are not trihedral, and that this singular topology implies geometric constraints on the planes supporting those non-trihedral vertices. To avoid this restriction, all the vertices have to be trihedral, which is the exact dual of the having triangular facets. This approach is thus preferred when there are many vertices per facet and that the minimization criterion involves more naturally the plane coordinates than the point coordinates.



(a) Constrained Input Building

(b) Each facet is adjacent to [3] vertices.

(c) Each vertex is adjacent to ③ facets.

Figure 4.7: Triangulated Vs Trihedral: (a) The numbers of vertices per facet (shown in squares) are typically greater than vertex valences (shown in circles) in a polyhedron that models a building. (b) Unconstrained perturbation of the vertex locations after triangulating all the facets. (c) Unconstrained perturbation of the plane equations after making all the vertices trihedral.

Both parameterizations are equivalent, but the primal parameterization emphasizes the point locations over the plane equations and thus the vertex locations over the planar region supports. On the other hand, the dual parameterization using the plane equations emphasizes the 2D planar regions and consider the edge and point locations as by-products of the plane equations and the polyhedron topology. We advocate to use the dual geometric representation for the following reasons:

Semantics: First, we advocate that the semantics of a polyhedron modeling a building are more naturally expressed on facets than on vertices. For instance, relaxing the coplanarity constraints using a triangulation (fig. 4.7.b) generates a polyhedron that feels more complex and thus less acceptable than relaxing the meeting of multiple planes using a trihedralization (fig. 4.7.c). The perturbation of the plane equations of a trihedralized building yields a more intuitive approximation of the building than the perturbation of the vertex locations of a triangulated building.

Convenience: To be robust, the minimized energy uses the bulk of data of the DSM relative to the polyhedral surface rather than only a sparse set of DSM points relative to the polyhedron vertices. This enables an independent estimation of the plane coordinates under the simplifying assumption that the boundaries of the facet supports remain fixed between optimization steps. Using the primal parameterization of a triangularized polyhedron, the error term depends, for each triangle, on the locations of its 3 adjacent vertices. Then, moving a vertex modifies the error term of the minimized energy on all its adjacent triangles. Thus the fixed boundary assumption is not sufficient to make the optimization of the vertex locations independent. As a sidenote, enforcing the verticality of the façades amounts to estimating a plane equation with $c = 0$ in the dual case, whereas in the primal approach, an equality constraint has to be introduced on the x and y coordinates of the adjacent vertices of the triangular facet. It seems then more appropriate to use the parameterization that gives direct access to the plane coordinates.

Therefore, we propose to model the building geometry using their supporting planes rather than their point locations. Before continuing, note that the geometric constraints induced by the topology are disregarded in this section, yielding an unconstrained fitting. The algorithm that performs the trihedralization of a polyhedron, and by doing so makes the fitting unconstrained, is detailed in chapter 5.

4.4.2 Minimized Energy

The energy that is minimized during the geometric refinement of a polyhedron $\mathcal{R}$ is the energy $E(\mathcal{R}, \mathcal{S})$, introduced in section 3.3 (equations 3.3.1 and 3.3.9), of a building $\mathcal{B} = (\mathcal{R}, \mathcal{S})$. The set of superstructures $\mathcal{S}$ is initialy empty ($\mathcal{S} = \emptyset$), and is afterward the result of a previous detection step in the overall building refinement system that alternately reconstructs its superstructures and refines the building polyhedral geometry, taking into account its previously reconstructed superstructures. We recall briefly the minimized energy and the meaning of the terms involved:

$$E(\mathcal{R}, \emptyset) = L(\mathcal{R}) + |\pi_{\mathcal{R}} \cap \vec{p}_{DSM}(\mathbb{Z}^2)|. \log C_{p,\sigma} + \sum_{\pi_{\mathcal{R}} \cap \vec{p}_{DSM}(\mathbb{Z}^2)} \frac{|z_{DSM} - z_{\mathcal{R}}|^p}{p.\sigma^p} \tag{4.4.1}$$

- $L(\mathcal{R})$ is a measure of the complexity of the polyhedron $\mathcal{R}$.
- $\pi_{\mathcal{R}}$ is the support of the roof surface, which is its vertical projection on an horizontal plane (section 3.2.1).
- $\pi_{\mathcal{R}} \cap \vec{p}_{DSM}(\mathbb{Z}^2)$ are the point samples of the DSM inside $\pi_{\mathcal{R}}$.
- $C_{p,\sigma}$ is a normalizing constant that depends only on p and σ
- z_{DSM} is the height of the DSM at a point sample.
- $z_{\mathcal{R}}$ is the height of the roof polyhedron.
- p defines which $\mathcal{L}_p$ metric is used.
- σ is the prior standard deviation of the DSM noise.

4.4.3 Fitting Algorithm

The proposed method to minimize this energy is to iterate optimization steps until either a maximum number of iterations are performed, or the computation time has been too long, or the

quality of the fitting is not increasing sufficiently, meaning that the optimization has converged. Thus, there is no guarantee to obtain a global minimum, but a local minimum close to the approximate input model appears to be a good guess in practice.

The trihedralization step will increase the complexity of non-trihedral polyhedra (*i.e.* which have at least one vertex that is not trihedral), but, once all the vertices are trihedral, the fitting algorithm itself keeps the topology fixed. We did not model the complexity increase due to the relaxation of the topology-induced geometric constraints. Thus $L(\mathcal{R})$ is considered constant during the fitting. The energy minimized at each iteration (equation 4.4.1) is thus:

$$E(\mathcal{R},\mathcal{S}) = \text{Constant}_1 + \text{Constant}_2 . \left(\sum_{(\pi_\mathcal{R} \setminus \pi_\mathcal{S}) \cap \vec{p}_{DSM}(\mathbb{Z}^2)} |z_{DSM} - z_\mathcal{R}|^p \right) \tag{4.4.2}$$

where Constant_1 includes the complexity term $L(\mathcal{R})$, the constant $|\pi_\mathcal{R} \cap \vec{p}_{DSM}(\mathbb{Z}^2)|.\log C_{p,\sigma}$ and the vertical error in the superstructure support regions $\sum_{\pi_\mathcal{S} \cap \vec{p}_{DSM}(\mathbb{Z}^2)} \frac{|z_{DSM} - z_\mathcal{R}|^p}{p.\sigma^p}$. The minimization of $E(\mathcal{R},\mathcal{S})$ then simply reduces to the minimization of the vertical error in the effective roof support region $(\pi_\mathcal{R} \setminus \pi_\mathcal{S})$, where the superstructure supports are not taken into account.

A second simplification arises by neglecting the modifications of the linear regions of $z_\mathcal{R}$ during each estimation iteration. The minimization of $E(\mathcal{R},\mathcal{S})$ is then performed as the independent $\mathcal{L}_p$ minimization of the plane coordinates of each top facing roof facet r over its effective support $(\pi_r \setminus \pi_\mathcal{S})$. For a top facing roof plane $r \in \mathcal{R}$, assuming the normalization $c_r = 1$, $(z_{DSM} - z_r) = (a_r x + b_r y + d_r + z_{DSM}) = (a_r, b_r, d_r) \cdot (x, y, 1) + z_{DSM}$ is an affine combination of the plane coordinates. During an optimization step, the data is partitionned by the current 3D model: the DSM is partitionned by a vertical projection of the 3D model into regions that correspond to points of facets belonging to a common supporting roof plane. The plane equations $[a : b : 1 : d]$ are then estimated independently using the $\mathcal{L}_p$ estimator over the corresponding, possibly not connected, region of the DSM, using only DSM samples outside the superstructure supports.

Algorithm 3 $Fitting(\vec{P}_{1\ldots n}, \mathcal{T}, DSM, \pi_\mathcal{R} \setminus \pi_\mathcal{S})$

Require: The initial point locations of the polyhedral vertices $(\vec{P}_{1\ldots n})$.
Require: The topology $\mathcal{T}$ of the polyhedron and the fitted DSM.
Require: The effective roof plane region $\pi_\mathcal{R} \setminus \pi_\mathcal{S}$.
Require: The function $Estimate(i, \vec{P}_{1\ldots n}, \mathcal{T}, DSM, \pi_\mathcal{R} \setminus \pi_\mathcal{S})$ that estimates the i^{th} plane equation $\vec{N}_i$ according to the DSM over a subregion of the effective roof support $\pi_\mathcal{R} \setminus \pi_\mathcal{S}$ that corresponds to facets supported by the plane $\vec{N}_i$.

repeat
  for each supporting plane $\vec{N}_i$ **do** $\vec{N}_i \leftarrow Estimate(i, \vec{P}_{1\ldots n}, \mathcal{T}, DSM, \pi_\mathcal{R} \setminus \pi_\mathcal{S})$ (sec. 4.4)
  if it is the first iteration **then**
    for each vertex v **do** Modify the topology $\mathcal{T}$ around v to relax its topology-induced constraints. (chap. 5)
  for each vertex $\vec{P}_i$ **do** $\vec{P}_i \leftarrow$ intersection of its adjacent planes. (sec. 4.2.2)
until convergence.

Schematically, this iterative process may be described using algorithm 3. In practice, the input geometry does not verify exactly the topology-induced constraints, due to round-off errors. For instance, when the polyhedron geometry is described by the point coordinates, facets are likely to have points that are only nearly but not exactly coplanar. The proposed approach does not depend on the initial geometry of the polyhedron, but only the initial topology and the target dual geometry. Thus, there is no need to estimate the initial dual geometry from the initial primal geometry. The initial primal geometry is sufficient to partition the input DSM in order to estimate the target dual geometry.

Using the supports of the previously detected superstructures to restrict the fitted DSM sample data to regions that have not been detected as superstructures is the key to reduce the bias they introduced in the earlier plane fitting steps.

4.5 Results

This section illustrates the iterative mimization approach coupled with superstructure reconstructions. We chose a simple building with a single slanted roof facet, to avoid the topological issues that will only be handled in the next chapters. This single roof facet features however many superstructures : terraces and small and large scale dormers. These superstructures represent about 50% of the total main roof facet surface. This level of outliers yield a poor superstructure-unaware fit of the main roof plane. Figure 4.8 shows the benefits of the iterative approach. (b) The first iteration allows the detection of the terraces and of the large dormers, leaving the small dormers undetected due to the erroneous initial roof facet geometry. (c) Then, detecting these large superstructures directly allows for a better fit of the main roof plane. The second iteration then detects all the superstructures. (d) Finally the third iteration slightly refines the geometry of the roof and of the superstructures.

The reconstructed superstructures of the first iterations are subsequently discarded and are only useful to refine the roof plane geometry prior to the last superstructure reconstruction step. That is why the superstructure reconstruction horizontal quantization may be increased to reduce the computing time of the first iterations and finally decreased at the last iteration that computes the final superstructures. Likewise, the first iterations use a $\mathcal{L}_2$ error metric whereas the last iteration uses a more robust $\mathcal{L}_{1.2}$ error metric. This greatly speeds up the computation without noticeably degrading the reconstruction quality. Namely, the 3 iterations were performed in respectively 37.25s, 35.85s and 12435.5s, on a single Intel Xeon 1.60GHz CPU core. We may note that the computing time explodes when the planimetric accuracy is increased. This example is extreme since the huge superstructure *blobs* in the DSM yield many superstructure hypotheses before the filtering, compared to roofs with the usually more isolated and smaller superstructure blobs.

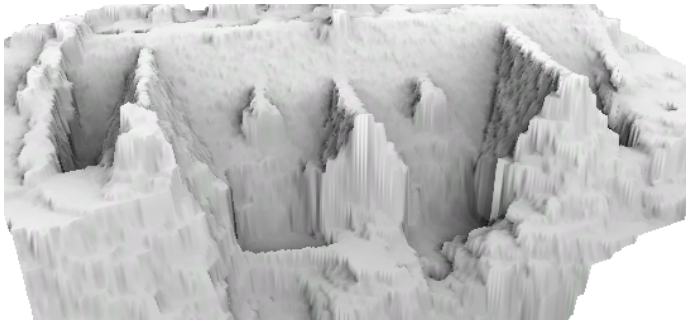
4.6 Extensions

This chapter presented an approach to fit a fixed-topology polyhedral building surface to a DSM, given the *outlier* superstructure regions. A straightforward extension is to improve the stability and accuracy of the optimization scheme (section 4.6.1), which would also enable the fitting of vertical facets. Then section 4.6.2 discusses a modification of the complexity term of the minimized energy to only selectively relax topological and geometric constraints, ensuring that the meaningful constraints are guaranteed throughout a constrained optimization. Finally, section 4.6.3 discusses the data term of the minimized energy, such as the usage of alternative input data such as LIDAR points or aerial images.

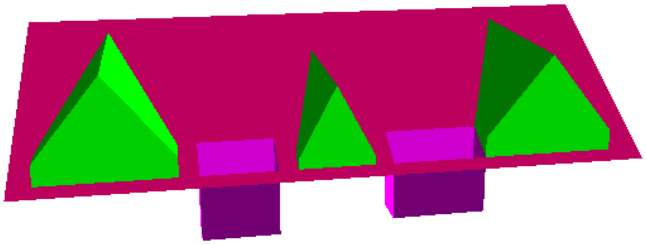
4.6.1 Numerical Scheme

Higher order numerical schemes could be designed to more robustly converge to a minimum. Likewise, it is easy to compute the gradient of the energy. Thus, a numerical scheme could use this property.

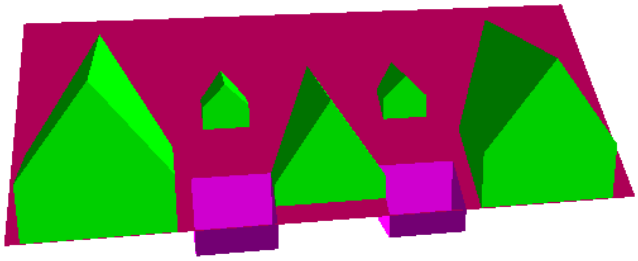
Furthermore, the described optimization estimates the plane coordinates independently. Thus it does not directly take into account the fact that the boundary separating two roof supports is a function of the coordinates of their supporting planes. Taking into account the dependence in the estimated plane coordinates of the planar region supports would lead to a better and more stable optimization.



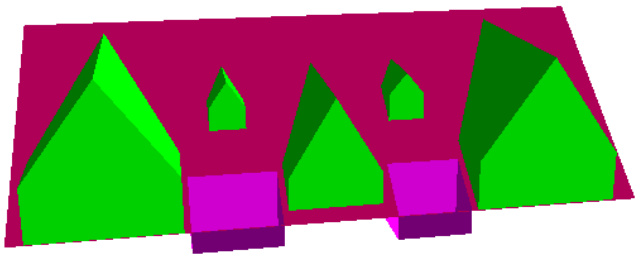
(a)



(b)



(c)



(d)

Figure 4.8: Iterative roof plane fitting using detected superstructure outliers. (a) The input DSM (ambient-occlusion shaded 3D view). (b) input main roof (red) and detected superstructures (step=0.5m). (c) Re-estimated roof and subsequent second superstructure reconstruction results at step=0.5m. (d) final roof estimation followed by a superstructure reconstruction at a step of 0.2m.

A limitation of this geometric optimization step is its impossibility to reconstruct and optimize vertical or bottom-facing roof planes. Although vertical facets of the polyhedron are not handled as a special case in our dual representation of polyhedron (they are supported by planes with a coordinate $c = 0$), it should be noted that the proposed geometric optimization is not able to move those vertical facets, since they have empty supporting regions in the 2.5D DSM.

However, even if the vertical facets have a degenerate support in the DSM, moving them modifies the supporting regions of their adjacent planes. An optimization scheme that takes into account the dependance of the supports relative to the neighboring plane parameters will thus be able to move and fit the vertical façade facets.

4.6.2 Selective Constraint Relaxation

The proposed approach gets rid of all the input topological singularities and does not incorporate the induced complexity increase in the minimized energy. This would inevitably involve adding one or more parameters to be able to value the complexity increase of relaxing a constraint in the $L(\mathcal{R})$ term of the minimized energy. The minimization problem would thus have to be able to chose which topological singularities are worth being preserved and perform a optimization constrained by the topological singularities that are not broken. The constrained optimization could be performed using a method similar to the one used in [VT05].

Such an approach would allow the relaxation of topological singularities, in a way that is a mix of the point coordinate parameterized primal approach and the plane equation parameterized dual approach. For instance, it would allow to let some facets coplanar and some other vertices adjacent to more than 3 planes, or even to ensure geometric properties such as symmetries and orthogonalities. The MDL framework is well suited to quantify the loss in the complexity term $L(\mathcal{R})$ due to a set of constraints. The difficulty of the implementation of this extension is on the optimization of the combinatorial decision of which singularities to preserve, and which to relax.

If all superstructures are merged into the building polyhedron to create a new polyhedron, one may be interested in fitting this resulting polyhedron directly to the DSM. It would keep the nice feature that DSM points detected as superstructure would not be used to reestimate the roof planes. But it would also enable the continuous optimization of the discretized parameters of the superstructure models (similar to appendix B). The geometries of the superstructure models of the proposed library may easily be translated in terms of constraints on their plane equations : horizontality, parralelism, orthogonality and distances between points or their relative position.

To go a step further, facets where the planar fit is poor would benefit from being split in two or more planar facets. Likewise, depending on the application, it may be interesting to merge nearly coplanar planes. These split (respectively merge) decisions would balance a better (respectively worse) fit of the data with an increased (respectively decreased) model complexity. One can view the superstructure reconstruction process of chapter 3 as a facet split process where the topology and geometry of the new facets introduced are given by the parametric superstructure library.

4.6.3 Alternative Input Data and Energies

This algorithm does not rely on the regular nature of the DSM grid points. It is thus easily adaptable to any other explicitely given input data, such as LIDAR data, with a careful dealing of the possible planar inhomogeneity of the lidar points to prevent estimation bias.

Similar to the discussion about the superstructure energies, the proposed approach is not strictly restricted to energies that measure the vertical error relative to some input data. A significant feature is that the optimization process maintains explicitly a polyhedron, thus any energy that values the goodness of fit of a polyhedron may be used. For instance, one may want to fit the polyhedron to higher level geometric primitives such as 3D segments.

An alternative possible error term could no longer rely on partitionning the data with a vertical projection. A Voronoï decomposition of the three-space, would partition the data according to the closest 3D polygon of a polyhedron facet. This would allow to handle the façades as regular facets. They could then be reestimated just like the roof facets. However we suppose that the strong architectural prior on the verticality of the façades should still be taken into account during the façade reestimation. Whereas the DSM sampling is homogeneous in planimetry, it is no longer homogeneous in 3D, when a Voronoï decomposition of $\mathbb{R}^3$ is used. The solution seems to segment a surface defined by the DSM rather than its point cloud. A simple surface defined by the DSM is the quad mesh of the point cloud such that there is an edge between each interior pixel and its 4 nearest neighbors. To get a piecewise planar surface, each quad may be divided into 2 triangles, using one of the two diagonals. The smoothest choice is to divide each quad using the diagonal that minimizes the area of the resulting 3D triangles.

Obviously, an interesting extension is to fit directly the polyhedron to aerial images, rather than preprocessing them as a correlation-based DSM. An input DSM is easy to manipulate and gives convincingly good optimization results. We argue that an optimization relative to the DSM is desirable before an image based optimization, because of its robustness and its good results. From this intial DSM approximation, we believe that an image based optimization is the way to extract the most accurate data out of aerial images.

4.7 Conclusion

This chapter introduced an unconstrained fitting of the polyhedral part of the building that takes into account the previously detected superstructures to remove the bias they introduced in the roof plane estimations.

Now that the fitting approach based on an iterative energy minimization, but unaware of the topological constraints, has been described, the relaxation mechanism that removes these topological constraints, given the target dual geometry of the first iteration, has to be introduced (chapter 5).

Furthermore, the boundary representation of polyhedron does not guarantee that the described surface is self-intersection free and thus encloses coherently the piecewise planar volume of a polyhedron. The trihedralization steps are local and require that the trihedralization problems are independent. However, the input topology may not only be simplified, which may not only require trihedralizations, but may also contain more complex erroneous topological modifications. Chapter 6 introduces a kinetic framework that interpolates the fitted polyhedron from the initial to the target dual geometry, while guaranteeing that it remains self intersection-free.

In a nutshell, this chapters plans the itinerary from the initial dual geometry to a dual geometry that fits the input data. Chapter 5 transforms the initial topology to be able to start the journey on this planned itinerary. Finally, chapter 6 modifies the topology along the way so that the polyhedron stays self-intersection free.

Chapter 5

Polyhedron Trihedralization

Contents

5.1 Introduction

Building reconstruction techniques tend to introduce constraints to increase their robustness. Some of these constraints are only of geometric nature : horizontal edges, vertical facets, right angles, symmetric slopes... To allow a better fit of the input data, they may be relaxed by modifying the geometry of the building polyhedron as a post process, while keeping its topology fixed. Chapter 4 has presented such a fitting process: it updates the plane equations of its facets (and thus indirectly the point coordinates of its vertices) without any modifications of the adjacency relationships between vertices, edges and facets.

However, the input topology itself may be erroneous. In general, the topology of the input building model may be arbitrarily different from the topology of a polyhedron that models the real building at the expected level of detail. Among these differences, two kinds of simplifications modify slightly the polyhedron geometry to simplify its topology. Nearly coplanar facets may be merged into a single facet (figure 5.1), resulting in a facet bounded by many vertices, or vertices connected by small edges may be merged into a single vertex (figure 5.2), resulting in a vertex adjacent to many facets. More general topological modifications are handled in chapter 6.

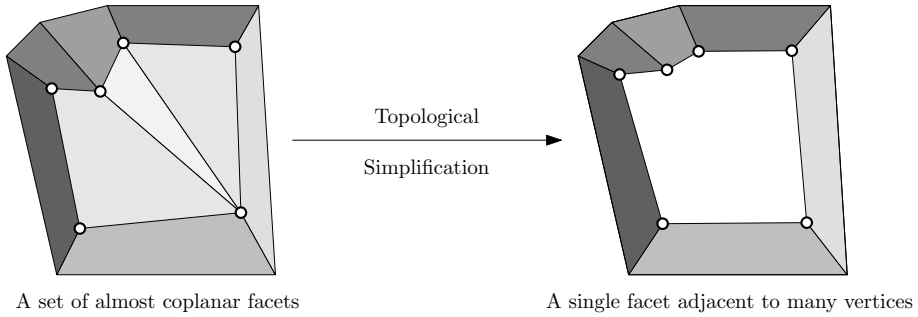


Figure 5.1: A set of almost coplanar facets are merged into a single facet.

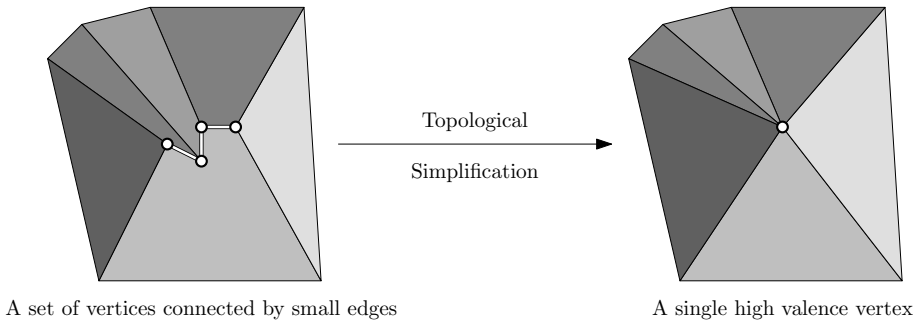


Figure 5.2: A set of vertices connected by small edges is collapsed to a single vertex.

The topological simplifications imply geometric constraints. If they are not verified by the ideal model (assuming it exists), keeping a fixed polyhedral topology will hinder a good fitting of the polyhedron. Without modifying the topology, the reconstruction may then give poor results. Thus an accurate reconstruction may have to undo these topological simplifications. An unconstrained fitting, such as the one presented in chapter 4, must undo some of these simplifications to guarantee that the resulting polyhedron is well-defined: vertices bounding a facet are supported by coplanar points and facets adjacent to a vertex are supported by meeting planes.

Depending on the geometric parameterization of the polyhedron, either facet simplifications (fig. 5.1) or vertex simplifications (fig. 5.2) must be undone. These two possibilities, which were

already presented in section 4.4.1, are discussed more precisely in the following two sections.

5.1.1 Polyhedron Triangulation

Undoing the first type of topological simplifications (figure 5.1) until all facets are triangular is required to move freely its vertices while keeping the semantics that facets are planar. It is thus required to perform an unconstrained optimization of the polyhedron vertex locations. If the facets are *almost* planar, then a planar facet triangulation is sufficient (section 5.1.1.1), otherwise, a much more complex triangulation of the non-planar facets is required (section 5.1.1.2).

5.1.1.1 Independent Planar Facet Triangulations

If the polyhedral vertices have been slightly perturbated from locations where they form a simple polyhedron with planar facets, triangulating the polyhedron amounts to independently triangulating the simple planar 3D polygons of the non-pertubated polyhedron. If the vertex displacements are small enough from the planar facet configuration, the self-intersection free polyhedral surface with planar triangulated facets remains self-intersection free after the perturbation. From section 3.2.1.2, the 3D polygon that supports a planar facet may have polygonal holes. Then, the triangulation of a polyhedron reduces to solving independently the triangulation problem of one 2D polygon (possibly with holes) per facet.

More precisely, a polyhedron is said to be a perturbation of a simple polyhedron with coplanar facets if both:

1. Each of its facets has a simple orthogonal projection (*i.e.* a plane exists such that the orthogonal projection of the facet boundaries onto this plane forms a simple planar 3D polygon, possibly with holes).
2. The independent triangulations of each orthogonal projection applied to the perturbated polyhedron yield a simple triangulated polyhedron.

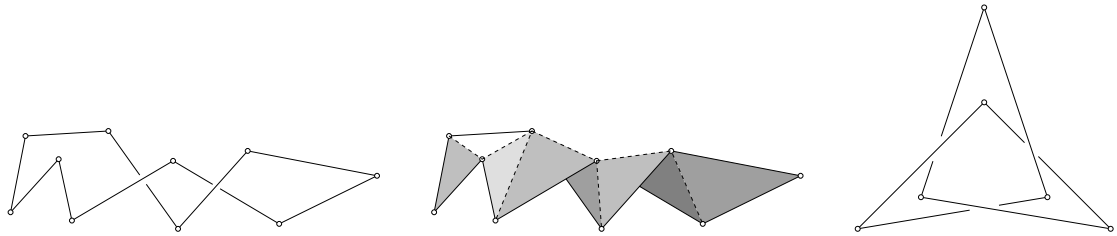
See [BDE96] for computing a projection plane that yields a simple orthogonal projection, when such a plane exists.

The triangulation of a simple 2D polygon is a well-known problem [dBCvKO08]. A triangulation is a decomposition of the area bounded by a polygon (possibly with holes) into triangles that are consistently oriented, such that they do not intersect each other except at their boundaries. A 2D polygon with holes with n vertices in total can be triangulated in $O(n \log n)$ time with an algorithm that uses $O(n)$ storage [dBCvKO08]. In particular, a 2D polygon without holes can be triangulated in linear space and time [Cha91].

In the context of chapter 4 with a polyhedron parameterized using its vertex locations, a triangulation of its facets relaxes the topology-induced geometric constraints on the vertex locations (namely, the coplanarity of the vertices of each facet). It further guarantees that if the vertex displacements are small enough, an input self-intersection free polyhedral surface remains self-intersection free. In the more general case, where facets have no orthogonal projections or where the triangulation problems are not independent, which may result from the vertex-parameterized optimization sketched in the previous chapter, the triangulation problem is however much more complex.

5.1.1.2 Non-planar Facet Triangulation

Before turning to our trihedralization problem, this section assesses the difficulty of a more general triangulation problem. We focus on polyhedra with facets that have a single topological boundary (no facet holes) but that may not have any orthogonal projection. The issue is the difficulty of the problem of triangulating a 3D polygon that is not necessarily planar:



(a) Simple non-planar 3D polygon (b) Triangulation of (a) (c) Knotted non-planar 3D polygon

Figure 5.3: Non-planar 3D polygon triangulation.

Definition 15 (Non-planar 3D polygon)**.** *A non-planar 3D polygon is a piecewise linear closed curve in* $\mathbb{R}^3$*.*

Definition 16 (Simple non-planar 3D Polygon)**.** *A non-planar 3D polygon is simple if it is self-intersection free.*

3D polygons may now be re-defined using the definition of non-planar 3D polygons:

Definition 17 (3D Polygon)**.** *A 3D polygon is a non-planar 3D polygon contained in a plane.*

Triangulating a simple planar 3D polygon reduces to triangulating a simple 2D polygon. However, in the general case, where the simple 3D polygon is not planar, it may not even be triangulable! Deciding whether it is triangulable is $\mathcal{NP}$-complete [BDE96]. We refer the reader to [BDE96] for the definitions of unknottedness and simple perspective or spherical projections, in order to introduce a necessary and a sufficient condition for triangulability:

Theorem 5.1 (3D Polygon Triangulability : Necessary Condition)**.** *The 3D polygon must be unknotted to be triangulable.*

Theorem 5.2 (3D Polygon Triangulability : Sufficient Condition)**.** *A 3D polygon that has a simple orthogonal, perspective or spherical projection is triangulable.*

Finally, before turning to the dual trihedralization approach, this section showed that the polyhedron triangulation problem is easy and well understood when the vertices have undergone a small displacement from a simple polyhedral configuration with planar facets, but that this problem in the general case is much more difficult and still leads to open questions such as a simple condition for triangulability that is both necessary and sufficient.

5.1.2 Polyhedron Trihedralization

A triangulation is needed when the polyhedron geometry is given using its point locations. This chapter develops concepts and tools that address the dual problem occurring when the polyhedron geometry is given using its plane equations. The previous chapter does not take into account the constraints induced by the topology in its geometry refinement step, since it is unconstrained. Thus, the independently reestimated plane support of each facet of the polyhedron are likely to produce over-constrained vertices.

This means that the polyhedron must be modified to be trihedral, so that the locations of its vertices are well defined using the updated dual geometry (the new plane equations). This may be done by splitting the vertices that are adjacent to more than 3 facets into trihedral vertices. This undoes the topological simplification of figure 5.2.

In section 4.4.1, we argued that a plane-based polyhedral parameterization was more semantically suited to our building reconstruction problem. This chapter thus explores the following **trihedralization** problem:

Polyhedron trihedralization

Given a plane-parameterized polyhedron (possibly featuring over-constrained vertices), modify parsimoniously its topology so that the modified polyhedron is trihedral and simple.

The parsimony constraint will only be precisely formalized in a restricted class of vertex trihedralization problems (section 5.4). At least, it imposes that an already well-defined and simple polyhedron should be returned unmodified. Note also that searching for a polyhedron with trihedral vertices rather than well-defined vertices amounts to assuming that the plane intersections are not degenerate.

The trihedralization of a well-defined high-valence vertex is dual to the simple planar 3D polygon triangulation. It is even simpler: its trihedralization introduces null length edges, such that all the vertex splittings describes the same polyhedral surface and are thus equally valid. However, the primal orthogonal projection trick is no longer helpful in the dual vertex trihedralization problem. The validity of this trick is based on the fact that a polyhedron with triangulated planar facets remains simple even if its vertices are slightly displaced. This is however false in the dual problem: considering a trihedralization of a polyhedron with well-defined high-valence vertices, infinitesimal perturbations of its plane equations may result in self-intersections. For instance, given a well-defined high-valence vertex, we just argued that any trihedralization is valid as they yield null length edges. An infinitesimal perturbation of its adjacent supporting planes may however yield self-intersections. The trihedralization problem thus seems more related to the much more complex non-planar triangulation problem.

This chapter is organized as follows. Section 5.2 shows how to adapt a winding-number based approach to solve this trihedralization problem. Then section 5.3 extends the search space of the winding-number based approach by providing an alternative approach based on a plane arrangement coloring. Compared to the winding-number based approach, this approach explores all the polyhedra supported by the input planes, such that the polyhedron obtained by the most parsimonious topological modification is now considered. Concurrently, section 5.3 also provides a sufficient condition under which the polyhedron trihedralization problem may be decomposed into independent vertex trihedralization problems. Section 5.4 discusses the vertex trihedralization problem. It finally formalizes the parsimony constraint in this independent vertex trihedralization context, reducing the problem to searching an abstract triangulation for each over-constrained vertex. Section 5.5 proposes an ear-cutting based algorithm to solve the vertex trihedralization problem. Then, section 5.6 introduces reductions between the vertex trihedralization problem and the weighted straight skeleton problem. Finally, section 5.7 discuss the trihedralization unicity and existence and concludes the chapter.

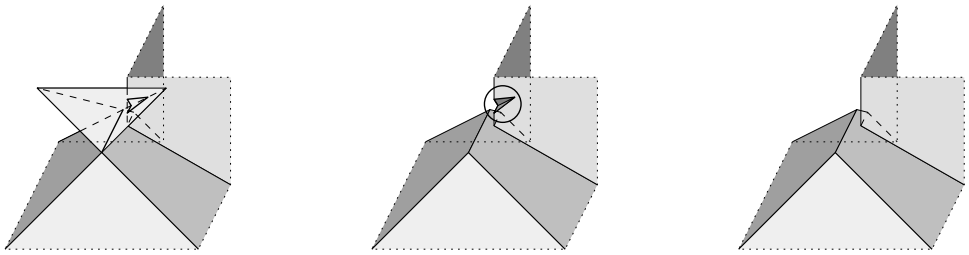
5.2 Winding Number-based Trihedralization

To get a better intuition of the trihedralization problem, we present how an existing approach based on winding numbers may be adapted to effectively provide a well-defined simple polyhedron out of an over-constrained polyhedron (such as one produced by the optimization of chapter 4).

The self-intersection free trihedral polyhedron maintenance problem appears in [For97] in the context of plane movements due to a rounding of their plane coefficients, rather than due to an optimization. Their approach uses symbolic perturbation to avoid non-manifoldness issues. It

(a) (b)

Figure 5.4: (a) Winding numbers on an oriented 2D piecewise linear curve. (b) The resulting self-intersection free curve after thresholding the winding numbers.



(a) (b) (c)

Figure 5.5: (a) A 3D self intersecting polyhedral surface. (b) The resulting self-intersection free surface after thresholding the winding numbers features a small dent of 2 facets (circled). (c) An alternative self-intersection free trihedralization without the small dent.

does not handle degeneracies, so that every trihedral vertex is considered to remain finite and well-defined. Thus, the trihedral polyhedron with an updated plane geometry (*i.e.* with rounded-off plane coefficients) defines a finite well-defined but possibly self-intersecting surface.

Their approach relies on winding numbers to update the topology of the polyhedron so that the polyhedron becomes simple. A bounded trihedral (possibly self-intersecting) polyhedron defines a closed oriented surface. It thus partitions $\mathbb{R}^3$ into multiple connected volumes. For instance, a non-empty self-intersection free polyhedron defines two complementary volumes, the *outside* and *inside* volumes. However, in presence of self-intersections, the polyhedron may define more than 2 connected volumes. The winding numbers corespond to an affectation of an integer number to each connected volumes that is consistent with the polyhedral surface and orientation. The consistence involved ensures that whenever the surface bounds a volume A on one side of a surface point and a volume B on its other side, with a surface normal going from A to B, the winding number of A is one more than the winding number of B. See figure 5.4 for winding numbers in 2D. Since they are defined up to a constant, the winding number of the unbounded connected volume may be set to 0. For instance, the winding numbers of a simple polyhedron would be respectively 1 and 0 for the *inside* and *outside* volumes. To produce a self-intersection free polyhedron from a possibly self-intersecting one (figure 5.5.a), the winding numbers are thresholded. Volumes with a positive winding number are the bounded volume inside the resulting self-intersection free polyhedral surface (figure 5.4.b in 2D, 5.5.b in 3D). All in all, this approach takes an arbitrary trihedral (possibly self-intersecting) polyhedron and outputs a simple trihedral polyhedron.

The problem of making a non-trihedral polyhedron trihedral after its planes have moved while keeping it simple (*i.e.* self-intersection free), is a variant of the problem of keeping a trihedral polyhedron self-intersection free after moving its planes. To reduce our trihedralization problem to the problem of maintaining a self-intersection free trihedral polyhedron, the input polyhedron may be made trihedral disregarding any self-intersection free requirements. This may be done

by applying an arbitrary, possibly self-intersecting, trihedralization of the high valence vertices beforehand, *i.e.* by splitting arbitrarily the over-constrained vertices into trihedral vertices. Note that the plane reestimations described in chapter 4 are performed recursively such that, assuming no geometric degeneracies, if the first iteration requires a trihedralization, the following iterations face this second problem of maintaining a self-intersection free trihedral polyhedron.

This approach is efficient and, using an arbitrary (*i.e.* possibly self-intersecting) trihedralization preprocessing step, solves the trihedralization problem, up to the parsimony constraint: even if a simple trihedral polyhedral is returned unchanged, the topological complexity of the resulting polyhedron is not directly minimized. For instance, the small dent in figure 5.5.b is not mandatory and a parsimony argument may be used to prefer the trihedral surface that does not feature this dent (figure 5.5.c).

We would like to restrict the topological complexity of the trihedralization (*i.e.* the number of vertices). The geometry of the trihedralization is completely determined by the 3D planes of the input problem instance. All the geometry of the problem is then captured by the arrangement of the input 3D planes: *e.g.* the vertices of the trihedralized polyhedron are located on vertices of the input plane arrangement. Whereas the winding number thresholding approach provides a single 0-1 coloring of the coarse partition of $\mathbb{R}^3$ induced by the polyhedral surface, a 0-1 coloring of the arrangement of planes supporting the polyhedral facets has many more degrees of freedom. It can be seen as a generalization of the winding number based approach as the plane arrangement is a refinement of the winding number partition.

5.3 Plane Arrangement Coloring-based Trihedralization

The arrangement of 3D planes captures the topology of the intersection of a set of planes (section 4.3.1). It thus plays a central role in the trihedralization problem. Since the introduction of new supporting planes is not allowed (no auxiliary planes: section 5.4.2.1), the trihedralization of the polyhedron is equivalent to a compatible coloring of its plane arrangement (see section 4.3.3).

The number of bounded cells in an arrangement of n planes is $O(n^3)$. This means that a solution of a trihedralization problem is a polyhedron out of the $O(2^{n^3})$ sets of polyhedra produced by 0-1 colorings of the bounded cells of the arrangement of the facet supporting planes. While the initial geometry provides the plane arrangement, the initial topology must drive the arrangement coloring to yield a resulting topology that is as combinatorially close as possible to the initial topology.

This partition coloring approach to reconstruct a polyhedral surface is well-known in computer vision. For instance, in [RC98, TD04, LPK07], a bounded subset of $\mathbb{R}^3$ is partitionned using respectively a set of cells of the arrangement of a set of detected planes, a regular grid of voxels, or a Delaunay tetrahedralization. Then a coloring of those regions is computed by optimizing a given criterion, be it based among others, on vertical errors, photoconsistency, visibility, regularization or on a mix of these terms.

Our trihedralization problem does not however rely on minimizing an energy based on some input data, but instead tries to make the simplest topological changes based on the input polyhedral topology. While the winding number based approach readily provides a compatible coloring of the plane arrangement, it has no degree of freedom. The issue is then how to use the combinatorial degrees of freedom of the plane arrangement coloring approach to provide a compatible coloring that induces a simple polyhedron with minimal topological changes from the input polyhedron. All the compatible colorings may be enumerated, so if the minimal topological change criterion were formally defined, a brute force enumeration and examination of all the compatible arrangement colorings would solve our problem.

We do not attempt to define here what a minimal topological change means in the general case and restrict ourselves to a class of polyhedron trihedralization problems which may be decomposed into independent vertex trihedralization problems. To this end, we now introduce the definitions

required to characterize this reduced class of polyhedron trihedralization problems.

5.3.1 Decomposability Assumption

To formulate an assumption that makes the trihedralization of a whole polyhedron decomposable into independent vertex-centered trihedralization subproblems, we now propose to define the **global arrangement** of a polyhedron, the **local arrangement** and the **zone** of one of its vertices and to extend the definition of the 3D polygonal supports (possibly with holes) to over-constrained facets.

5.3.1.1 Global and Local Arrangements

Definition 18 (Global Arrangement)**.** *The global arrangement of a polyhedron is the arrangement of all the planes supporting its facets.*

Definition 19 (Local Arrangement)**.** *The local arrangement of a vertex is the arrangement of the planes supporting its adjacent facets.*

A non-trihedral vertex of the initial polyhedron should be at the intersection of $k > 3$ planes. The arrangement of these k planes after the geometry update is called the **local arrangement** of the vertex. If the local arrangement is not highly degenerate (*i.e.* with a single vertex), the vertex location is not well-defined. The arrangement of all the polyhedral facet supporting planes, is called the **global arrangement**, and is by definition a refinement of the partition defined by the local arrangements of any of its vertices. In other words, each cell of the local arrangement corresponds to a union of cells of the global arrangement. These local arrangements capture the local geometry of the trihedralization problem around each polyhedral vertex.

5.3.1.2 Vertex Zones

The zone of a vertex may be informally defined as the region of $\mathbb{R}^3$ that is swept by the surface resulting from the various topological trihedralizations of this vertex. By the definition of a plane arrangement, a polyhedral surface supported by a set of planes may necessarily be expressed as a union of facets, edges and vertices of the arrangement of these planes. If a polyhedral vertex may be trihedralized locally (*i.e.* considering only the ring of adjacent planes and adjacent edges, treated as infinite rays), its local arrangement encodes the local geometry of the problem and the polyhedral surface away from the overconstrained vertex may simply be modelized by a circular list of unbounded facets of its local arrangement. Therefore, its trihedralization reduces to selecting the bounded elements of its local arrangement that are part of the resulting trihedralized surface.

Definition 20 (Vertex Zone)**.** *The zone of a polyhedral vertex is defined as the union of the bounded elements (vertices, edges, facets and cells) of the local arrangement of this vertex.*

Disregarding geometric degeneracies, note that a vertex is over-constrained if and only if its zone is larger than a single point (*i.e.* Its local arrangement contains at least one bounded cell). A pointwise zone then yields a well-defined point. For instance, being adjacent to 3 planes, the zone of a non-degenerate trihedral vertex is necessarily reduced to a point. When trihedralizing a polyhedron featuring an overconstrained vertex, the local arrangement of this overconstrained vertex provides all the possible vertex, edge and facet supports of the resulting well-defined surface in the neighborhood of this trihedralized vertex. Thus, the vertex zones model the regions of uncertainty through which the various possible trihedralized surfaces have to pass.

5.3.1.3 Facet Supports

We extend the definition of a facet support in presence of overconstrained vertices. An overconstrained vertex has, by definition, no well-defined point location. However, we propose to give it one for each facet adjacent to it. Considering a particular facet, each of its adjacent vertices may then be supported by a well-defined point, yielding a facet supported by a well-defined 3D polygon (possibly with holes). Disregarding geometric degeneracies, we propose to introduce the facet-dependant location of a possibly overconstrained vertex.

Definition 21 (Facet-dependant Vertex Location)**.** *The facet-dependant location of a vertex V adjacent to a facet supported by a plane P_i is the intersection point $(P_{i-1} \bigcap P_i \bigcap P_{i+1})$, where $(P_j)_{0 \leq j < n}$ is the circular list of planes supporting the facets adjacent to V.*

As a remark, when the vertex is trihedral, the definition above is no longer dependant on a distinguished adjacent facet, and it reduces to the well-defined location of a vertex defined by the plane geometry of its adjacent facets. Furthermore, section 5.4.3 will discuss the case of geometric degeneracies.

These facet-dependant vertex locations provide well-defined facet supports even in presence of over-constrained vertices. More precisely, the resulting over-constrained facet support is the topologically simplest support among all the supports resulting from the various trihedralizations of its adjacent vertices.

Definition 22 (Restricted Facet Support)**.** *The restricted support of a possibly over-constrained facet is the 3D polygon (with holes) resulting from subtracting the interior of the zones of all the polyhedral vertices from its support.*

Subtracting the interior of the zones rather than the zones themselves ensure that the resulting point set is a 3D polygon (with holes) that includes its boundary edges and vertices. For instance, well-defined vertices have a pointwise zone and thus an empty zone interior. Thus the restricted facet support of a well-defined facet is itself.

Definition 23 (Restricted Polyhedral Surface)**.** *The restricted polyhedral surface is the union of the restricted facet supports.*

The restricted polyhedral surface has well-defined point locations. However, this surface is not strictly a polyhedron given that it may have boundaries in the vicinity of its over-constrained vertices (*i.e.* at the vertex zone boundaries). The restricted polyhedral surface of a possibly over-constrained polyhedron is a polyhedral surface that is common to all the possible polyhedron trihedralizations that are performed locally around the over-constrained vertices. In other words, if the trihedralization problem is decomposed into independent vertex trihedralization problems, then the restricted polyhedral surface is common to all the recomposed solutions. Conversely, a volume may be defined that encloses all the possible polyhedra resulting from independent over-constrained vertex trihedralizations:

Definition 24 (Polyhedral Zone)**.** *The polyhedral zone of a possibly over-constrained polyhedron P is the union of the restricted polyhedral surface of P and the zones of the vertices of P.*

The polyhedral zone represents the region of uncertainty which may not trivially be classified as inside or outside the final trihedralized polyhedron. A particularly interesting case is the polyhedral zone being topologically equivalent to the input polyhedron. The polyhedral zone is then only a thickened version of the input polyhedron around over-constrained vertices and a simple 2D manifold away from them. Next section defines more formally this assumption.

5.3.1.4 Decomposable Trihedralization

The definitions above allow us to introduce the following definition:

Assumption 5.1 (Decomposable Polyhedron Trihedralization)**.** *A polyhedron trihedralization problem is said to be* ***decomposable*** *if its vertex zones are disjoint from each other and its restricted polyhedral surface is self-intersection free.*

Disjoint zones ensure that the subproblems are independent and the self-intersection free restricted polyhedral surface ensures that the restricted polyhedral surface itself does not cause any self intersection, leaving the search for possible self-intersections to within the vertex zones.

Decomposing the global trihedralization topological search into subproblems local to each vertex zone limits the possible topological modifications. Thus, this defines a restricted class of trihedralization problems where the *minimal topological changes* required in our problem statement will finally be formalized. From now on, the considered trihedralization problems are decomposable. Before section 5.4 further investigates a restricted set of vertex trihedralization subproblems, next section uses the above decomposition within the context of the arrangement coloring framework.

5.3.2 Arrangement Coloring-based Decomposable Trihedralization

Considering a decomposable trihedralization problem, cells of the global arrangement are of two types: they either fall entirely inside the zone of a particular vertex or are completely outside all vertex zones. To provide a coloring of the global arrangement, both types of cells, respectively named **zonal** and **non-zonal** cells, have then to be colored.

5.3.2.1 Coloring the Non-Zonal Cells

Assuming a decomposable trihedralization problem, the polyhedral zone partitions $\mathbb{R}^3$ into 3 region types: the polyhedral zone itself, **interior** regions, and **exterior** regions. The cells of the global arrangement may be labelled as zonal (currently uncolored), interior (1-colored) or exterior (0-colored) cells. Non-zonal cells are either labelled interior or exterior depending on the orientation of the planes supporting the restricted facet supports. The coloring of all the non-zonal cells surrounding a vertex zone provides the context necessary to perform locally the overconstrained vertex trihedralization.

More practically, constructing and coloring all the non-zonal cells is not relevant, as computing the restricted polyhedral surface and coloring the non-zonal cells adjacent to zonal cells only gives enough context to perform independently the decomposed vertex trihedralizations (*i.e.* to color the zonal cells).

5.3.2.2 Coloring the Zonal Cells

The trihedralization of a high valence vertex reduces to a compatible 0-1 coloring of the global cells of its zone, given a coloring of its adjacent non-zonal global cells (fig. 5.6). By construction, the winding number thresholding approach of section 5.2 produces such a compatible coloring. It is however not able to enumerate all the compatible colorings to minimize its resulting topological complexity. For instance, the winding number based coloring may be updated to a hole-free compatible coloring by iteratively "filling in the holes" (by reversing the coloring of their cells). This proves that such a hole-free compatible coloring exists, at least in non highly degenerate setups where the winding number approach is not applicable.

Coloring each global arrangement cell inside the vertex zone independently is admittedly redundant, as planes supporting non-adjacent facet are likely not to be relevant locally. Interestingly, the local arrangement may however be too coarse a segmentation. Figure 5.7 shows a vertex zone where some unbounded cells of the local arrangement contain both 0-colored and 1-colored global cells adjacent to the vertex zone. These more global trihedralization subproblems may still be

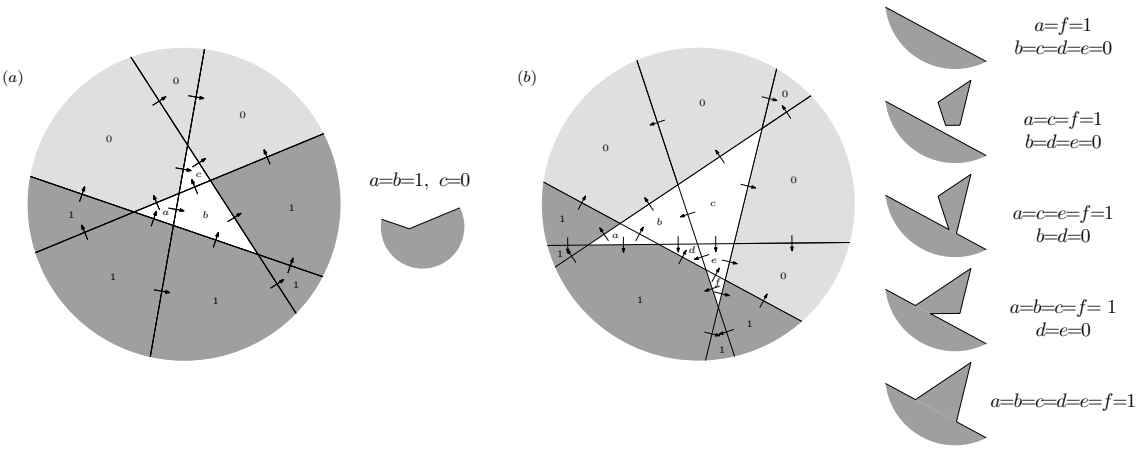


Figure 5.6: 2D planar sections of two distinct local arrangements with compatibly colored unbounded cells and uncolored bounded cells. (a) An arrangement and the only compatible coloring of its bounded cells. (b) This more complex arrangement section admits 5 compatible colorings, the second not being hole-free.

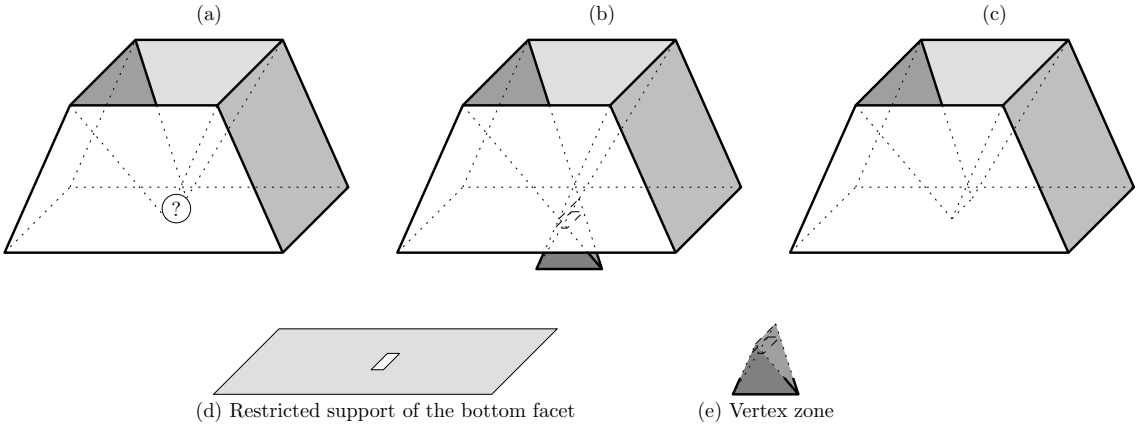


Figure 5.7: (a) A trihedralization problem of a polyhedron with an over-constrained vertex. (b) and (c) are two possible trihedralizations. (d) shows the restricted facet support of the bottom facet, with a rectangular hole cut out by the tetrahedral vertex zone (e).

solved independently and locally but such a vertex zone coloring not only relies on the arrangement of the adjacent planes but on the arrangement of all the planes supporting a facet support that intersects the vertex zone.

5.3.3 Locality Assumption

The decomposability assumption 5.1 enables the reduction of a polyhedron trihedralization problem into independent vertex trihedralization subproblems. This section focuses on a subset of these vertex trihedralization problems, namely the so-called **local** vertex trihedralization problems, that satisfy the following assumption:

Assumption 5.2 (Locality)**.** *A vertex zone is local (i.e. yields a local vertex trihedralization problem) iff its intersection with the restricted polyhedral surface is connected (i.e. is a point, a segment, or a single piecewise linear loop).*

This assumption implies that a local vertex zone produces a single hole in the restricted polyhedral surface. It is thus possible to define the single circular list of supporting planes adjacent to the vertex zone. Figure 5.7.e illustrates a vertex trihedralization problem that is not local: it features two disconnected intersections (dashed: a segment and a planar quadrangle). This assumption is termed *local* as it forces the vertex trihedralization problem to be local on the restricted polyhedral surface.

More practically, as the global arrangement is a refinement of each local arrangement, the unbounded cells of a local arrangement induce a partition of the set of non-zonal cells adjacent to the vertex zone. Non-zonal cells adjacent to a local vertex zone are then colored consistently relative to the local arrangement cell partition: the set of non-zonal adjacent cells included in an unbounded cell of the local arrangement is identically colored. Therefore, local vertex trihedralizations may be solved by coloring the local arrangement bounded cells, rather than all the global cells of the vertex zone.

5.3.4 Discussion

This section defined the proposed decomposability and locality assumptions. These assumptions clearly identify simpler classes of trihedralization problems. The proposed arrangement coloring framework gave some intuition about these assumptions and illustrated how these assumptions simplify the trihedralization algorithms sketched in this section:

No assumption: all the zonal cells are colored at once.

Decomposability: the zonal cells are colored independently within each vertex zone.

Decomposability and locality: the bounded local arrangement cells are colored independently within each vertex zone (which is a coarser partition of the vertex zones, defined locally.)

Assuming a decomposable problem, the arrangement coloring approach may be used to enumerate all the hole-free compatible trihedralizations. We are however interested in minimizing the increase of topological complexity of the resulting trihedralization. By introducing a topological simplicity criterion, it would then be possible to output the simplest trihedralization among the enumerated ones. Thus this approach may be viewed as an extension of the winding number approach to meet our combinatorial simplicity criterion. Future work may use the arrangement coloring induced by the winding number thresholding to drive an efficient computation of a suitable 0-1 coloring.

5.4 Local Vertex Trihedralization

This section further analyzes the local vertex trihedralization subproblem and explores how the decomposability and locality assumptions may be used to provide a straightforward meaning to our goal of computing a self-intersection free trihedral polyhedron while minimizing its topological complexity. Considering a local vertex trihedralization problem, the minimization of the topological complexity is introduced here by restricting the allowable output topologies. To achieve this goal, a framework is needed to efficiently explore the set of topologies of minimal complexity.

A similar problem of searching for the topology of a trihedral polyhedron has been addressed in [YL89], in order to infer the topology of the occluded parts of a polyhedron given by its silhouette only. The polyhedral silhouette gives a circular list of the geometry-less occluded facets. The problem is then similar to ours: find the missing topology of a polyhedron that is supposed to be trihedral and without any interior facet. Relative to [YL89], our contribution is a more precise and fruitful description of this problem in terms of an abstract triangulation.

5.4.1 Abstract Triangulations

Given a polygon, its triangulation amounts to computing a combinatorial structure that satisfies some geometric requirements: the so-called **abstract triangulation**, . This is the combinatorial part of a triangulation, disregarding its geometrical embedding. A triangulation is said to be a **geometric realization** of an abstract triangulation if their combinatorial structures are isomorphic. An abstract triangulation is a special case of a 2-dimensional abstract simplicial complex, as defined, for instance, in [DLM05]. To distinguish between an object and its abstract counterpart, non-abstract objects are referred to as **concrete**.

An **abstract vertex** v is defined as a combinatorial object without any geometrical embedding. It allows the definition of a geometry-less abstract polygon.

Definition 25 (Abstract Polygon)**.** *An abstract polygon is a circular list $V = (v_0 \dots v_{n-1})$ of n abstract vertices v_i.*

To simplify notations, the vertex indices are defined modulo the polygon size n : $v_{i+1} = v_{(i \mod n)}$. The vertex list $(v_0 \dots v_{n-1})$ of an abstract polygon is said to be circular since its circular reorderings $(v_k \dots v_{k+n-1}) = (v_k \dots v_{n-1}, v_0 \dots v_{k-1})$ are considered to refer to the same abstract polygon. The reversed abstract polygon $(v_{n-1} \dots v_0)$ is however different and refer to the abstract polygon $(v_0 \dots v_{n-1})$ after a topological orientation reversal.

Abstract polygons have by definition no geometric embedding. However, for illustrative purposes, the abstract polygons of the figures of this manuscript are given the geometric embedding of regular (*i.e.* equilateral and equiangular) convex polygons. This convex geometric embedding is chosen so that any abstract triangulation yields a self-intersection free (concrete) triangulation. Let us now define the triangles, edges and triangulations in this topology-only context.

Definition 26 (Abstract Triangle)**.** *An abstract triangle is an abstract polygon of size $n = 3$.*

Definition 27 (Abstract Edge)**.** *An edge of an abstract polygon is an ordered pair $[v_i, v_{i+1}]$ of consecutive vertices in the abstract polygon vertex list $(v_0 \dots v_{n-1})$.*

This defines oriented edges: $[v_i, v_{i+1}]$ is an abstract edge of the polygon $(v_0 \dots v_{n-1})$ but $[v_{i+1}, v_i]$ is not (except for degenerate polygons of size $n = 2$). Note that $[v_{n-1}, v_n] = [v_{n-1}, v_0]$ is an abstract edge of the polygon $(v_0 \dots v_{n-1})$, due to the circular ordering.

Definition 28 (Opposite Abstract Edges)**.** *2 edges $[v_i, v_{i+1}]$ and $[w_j, w_{j+1}]$ of 2 abstract polygons $(v_0 \dots v_{n-1})$ and $(w_0 \dots w_{m-1})$ are opposite of each other if $v_i = w_{j+1}$ and $w_j = v_{i+1}$.*

Definition 29 (Adjacent Abstract Polygons)**.** *2 abstract polygons are adjacent if an edge of one abstract polygon is the opposite of an edge of the other abstract polygon.*

Definition 30 (Abstract Diagonal edge)**.** *A diagonal edge of an abstract polygon is an (oriented) abstract edge $[v_i, v_j]$ between non-consecutive abstract vertices of the polygon $|i-j| \neq 1 \mod n$.*

An abstract diagonal edge $[v_i, v_j]$ with $i < j$ **decomposes** an abstract polygon $(v_0 \dots v_{n-1})$ into two abstract subpolygons $(v_0 \dots v_i, v_j \dots v_{n-1})$ and $(v_i \dots v_j)$. Abstract diagonals $[v_i, v_j]$ with $i > j$ induce the same decomposition as their opposite edge $[v_j, v_i]$. The two subpolygons resulting from a diagonal decomposition are adjacent since one has an edge $[v_i, v_j]$ and the other an edge $[v_j, v_i]$.

Definition 31 (Abstract Polygon Triangulation)**.** *An abstract triangulation of an abstract polygon $(v_0 \dots v_{n-1})$ of size $n \geq 3$ is a collection of abstract triangles (v_i, v_j, v_k) with $0 \leq i < j < k < n$ obtained by a recursive diagonal decomposition of the initial abstract polygon until its subpolygons are all triangular.*

Note that these definitions are topological only and that no geometric properties are required. The geometric realization of an abstract polygon triangulation is thus not guaranteed to yield a partition of the (concrete) polygon into disjoint (concrete) triangles. However, the topological part of a (concrete) polygon triangulation is always an abstract polygon triangulation.

Definition 32 (Abstract boundary Edges)**.** *An abstract boundary edge of an abstract polygon triangulation is an edge of the triangulated abstract polygon.*

By extension, an edge of an abstract polygon triangulation refers to an edge of one of its abstract triangles. Thus triangulation edges are either diagonal or boundary edges. Moreover, whereas a diagonal edge has its opposite in the abstract triangulation, the abstract edge opposite to a boundary edge does not exist in the abstract triangulation.

Corollary 5.3. *An abstract triangulation of an abstract polygon of size n contains $n-2$ abstract triangles and $n-3$ diagonals.*

Corollary 5.4. *2 adjacent triangles of an abstract triangulation have together exactly 4 distinct abstract vertices : the 2 vertices of their common abstract edge and the third vertices of both abstract triangles.*

Definition 33 (Abstract Triangle Fan)**.** *The triangle fan of an abstract vertex v_i inside a triangulation of an abstract polygon $(v_0 \dots v_{n-1})$ is the set of all abstract triangles that contain the vertex v_i inside the abstract triangulation. Vertex v_i is called the **apex** of its triangle fan.*

The abstract triangle fan of v_i may thus be given by the (non-circular) ordered list $(w_0 \dots w_m)$ of (abstract) vertices adjacent to v_i such that (v_i, w_j, w_{j+1}) is a triangle of the abstract triangulation for all $j < m$. $[v_i, w_j]$ is then a boundary edge if $j = 0$ or m and a diagonal edge otherwise.

Counting the Abstract Triangulations: The number $T(n)$ of possible triangulations of a convex polygon with n vertices is related to the Catalan numbers $C(n) = \frac{1}{n+1}\binom{2n}{n}$, by the identity $T(n) = C(n-2)$. This series grows exponentially and its first values are 1 (triangular case), 2 , 5, 14 (quadrangular, pentagonal and hexagonal cases, shown in figure 5.8), 42, 132, 429, 1430, 4862, 16796, 58786, 208012 [Slo09] ...

Now that the abstract triangulation of an abstract polygon is properly defined, next section presents how it may be used in our local vertex trihedralization context.

5.4.2 Local Vertex Trihedralizations as Abstract Triangulations

Figure 5.9 illustrates a solution to a local vertex trihedralization problem both in primal and dual space. By duality, triangles map to trihedral vertices. Thus, trihedralizing a vertex appears

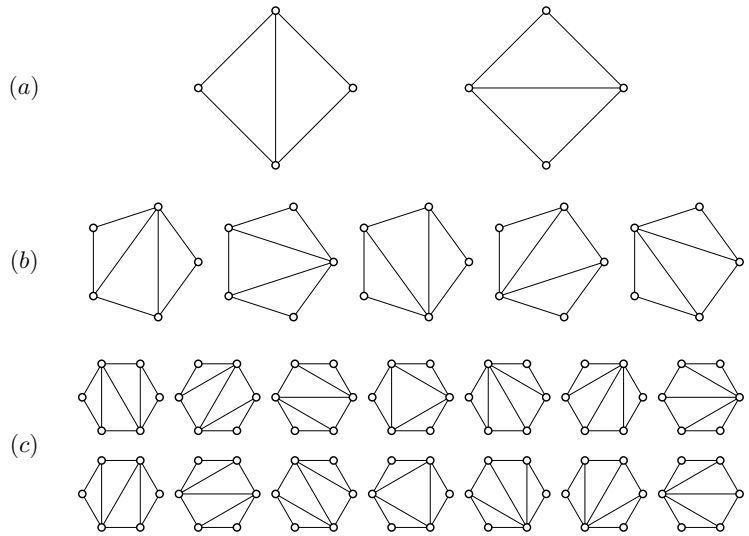


Figure 5.8: Counting the abstract triangulations for abstract (a) quadrangles, (b) pentagons and (c) hexagons.

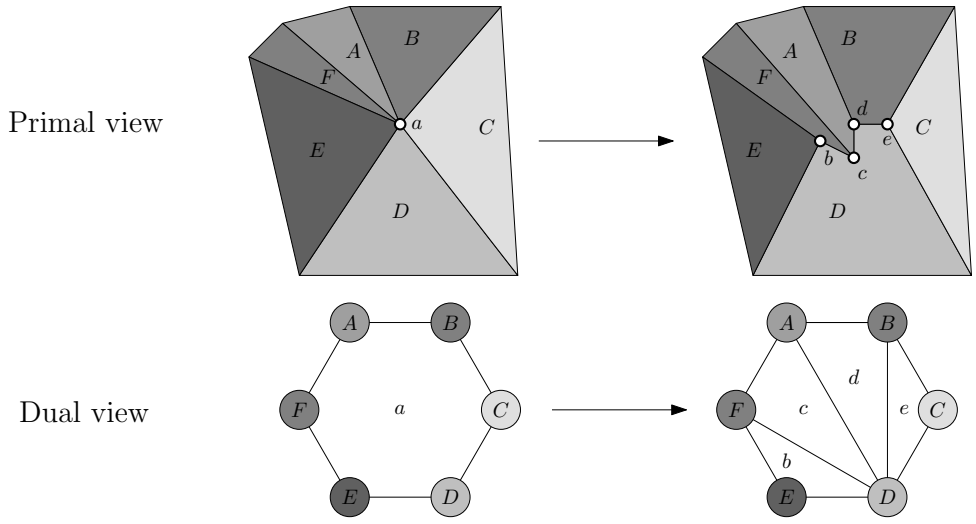


Figure 5.9: (Primal view) Trihedralization of the vertex a of valence 6, adjacent by 6 facets $ABCDEF$ into 4 trihedral vertices $bcde$. (Dual view) Triangulation of the facet a, adjacent to 6 vertices $ABCDEF$, into 4 triangles $bcde$.

to be the dual operation of triangulating a facet. The local vertex trihedralizations thus appear to be, at least topologically, the dual of the facet triangulations, in the sense that they compute the same topological object: an abstract triangulation. For instance, splitting a vertex into two vertices is the dual operation of splitting a facet into 2 facets with a diagonal.

Given assumptions 5.1 and 5.2, the polyhedron trihedralization problem may then be decomposed into the following independent local vertex trihedralization problems:

Local Vertex Trihedralization

Compute an abstract triangulation of the dual polygon representing the circular list of plane equations around the possibly over-constrained vertex, such that the resulting polygonal supports of its adjacent facets are self-intersection free.

Note that this problem formulation does not check for improper intersections and only checks that the individual facet supports do not self-intersect. Dealing with improper intersections will be discussed in section 5.4.4, together with considerations on the more general vertex trihedralization problem that does not assume that vertex zones are local.

The next section comments the minimal topological complexity property of the abstract triangulation, while the following section translates the geometric requirement of self-intersection free facet in terms of the abstract triangulation.

5.4.2.1 Minimal Topological Complexity

The abstract triangulations have minimal complexity since they do not introduce any new abstract vertex. Such an extra abstract vertex may have been recycled from a neighboring facet supporting plane, yielding an interior facet, or may come from an entirely new geometry (an auxiliary plane). By disallowing auxiliary vertices with both new and recycled geometry, the search space of the trihedralization is then exactly the set of the $T(n)$ abstract triangulations of the abstract polygon formed by the circular list of adjacent facets around the high valence vertex.

No Interior Facets Figure 5.10 shows a trihedralization that uses no new auxiliary plane, but uses a supporting plane for two distinct abstract vertices of the abstract triangulation. The dual combinatorial structure is thus not an abstract triangulation. C_1 is on the boundary of the triangulated (dual) polygon, but C_2, on the other hand, is an interior (dual) vertex (*i.e.* is not on the boundary). To minimize the combinatorial complexity of the trihedralization, the facets resulting from the trihedralization are required to be connected. To achieve this, the trihedralization may not create new facet boundary cycles. A facet boundary cycle maps to a circular list of adjacent triangles. The (dual) vertex common to all the triangles of such a circular list is an interior (dual) vertex. Thus, to minimize the combinatorial complexity of the trihedralization, one must prevent the creation of new facet boundary cycles by disallowing interior (dual) vertices.

No Auxiliary planes Casting this problem as finding an abstract triangulation ensures that no auxiliary new planes will be introduced as in figure 5.11. The vertices of the abstract triangulation only refer to existing input facets (condition 1), and no auxiliary dual vertex, or so-called *Steiner* vertex, is created. The resulting surface is supported only by the input planes. Thus the number of planes, which is a measure of the geometric complexity of the polyhedron, is kept constant.

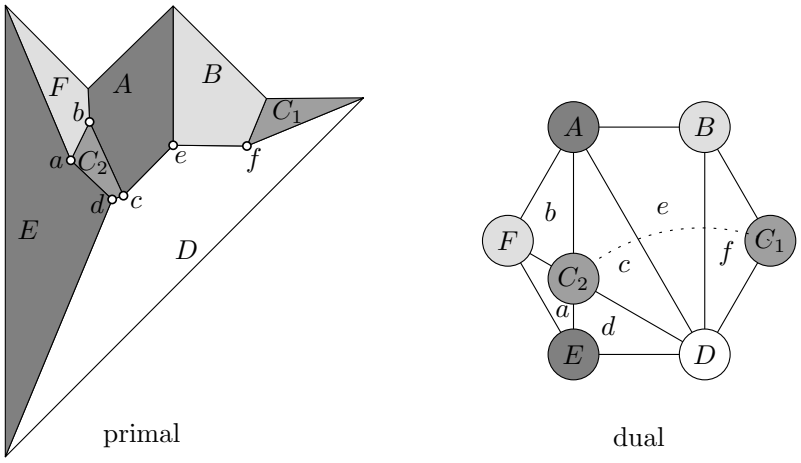


Figure 5.10: The plane C supports 2 disjoint connected facets: C_1 touches the boundary as do the other facets A, B, D, E and F, whereas C_2 is an auxiliary interior (dual) vertex of the trihedralization.

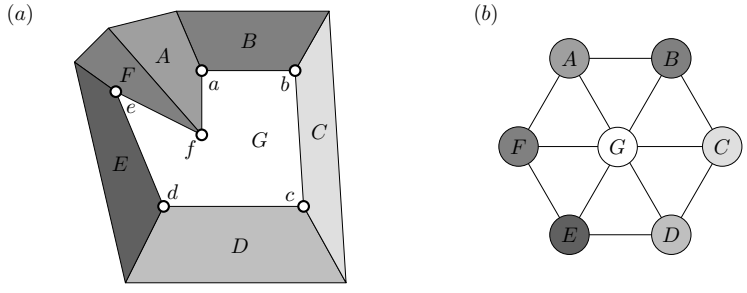
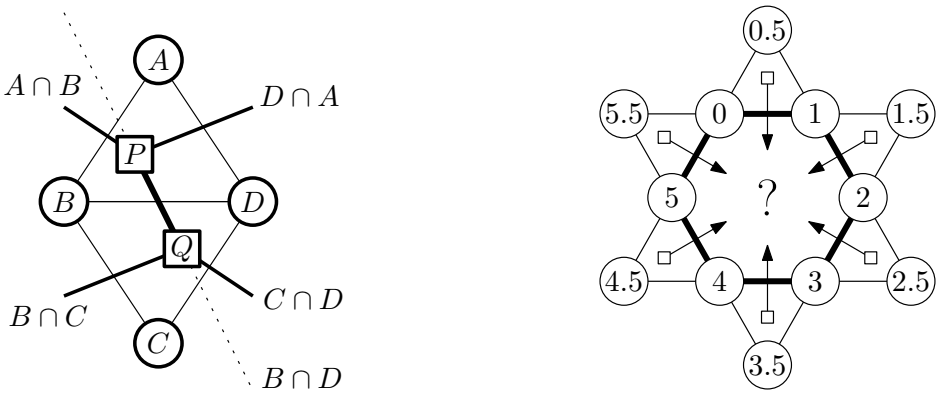


Figure 5.11: The (a) primal and (b) dual views of a trihedralization of $ABCDEF$ that uses a new auxiliary plane G.



(a) (b)

Figure 5.12: Topological view of the trihedralization: (a) Given 2 adjacent dual triangles ABD and BCD (where dual vertices are denoted by their supporting planes), the geometric embedding of the dual edge BD is the 3D segment PQ, which is supported by the 3D line $B \cap D$ and bounded by the endpoints $P = A \cap B \cap D$ and $Q = B \cap C \cap D$. (b) The topology of the trihedralization problem: one needs the outer dual polygon (light edges), to compute the trihedralization of the inner (bold) dual polygon, to take into account the orientation of the incoming rays.

5.4.2.2 Geometric Requirements

This section focuses on the geometric requirements that an abstract triangulation must satisfy to yield a valid local vertex trihedralization.

If the resulting facets are simple, then edges that link newly created vertices adjacent to the same facet must not intersect. Such edges map, through duality, to diagonal edges of the abstract triangulation of the formerly over-constrained vertex. The geometry of such an edge is defined by the geometric embedding (fig. 5.12.a) of the 4 distinct dual vertices (*i.e.* facets) of its 2 adjacent triangles (see corollary 5.4), by means of their 4 plane equations. It is supported by the line at the intersection of the plane supporting the two dual vertices of the diagonal, and bounded by the intersections of this line with the planes of the 2 neighboring dual vertices.

To compute the trihedralization, the facet simplicity geometric requirement has to be rephrased in terms of the abstract triangulation. The following requirement is a first attempt that is necessary but not sufficient to prove that the resulting facets are simple.

Theorem 5.5 (Necessary Geometric Requirement)**.** *If an abstract triangulation yields a trihedralization with simple facets only, then no non-consecutive pair of diagonals in a triangle fan of the abstract triangulation are supported by intersecting segments.*

A dual vertex is a facet, and a pair of diagonals adjacent to this vertex refers to a pair of edges on the boundary of this facet. If these diagonals are adjacent to a common dual triangle, then, they are consecutive along the boundary of the facet, and thus have to intersect at one of their endpoints, which is the geometric embedding of the dual triangle. Otherwise, this pair of diagonals refers to edges that are adjacent to the same (primal) facet but have no (primal) vertex in common. If these edges are supported by a pair of intersecting segments, then the facet is not simple, and the trihedralization is thus invalid.

However, this requirement is not sufficient, as it only prevents the newly created edges to intersect, and does not check for intersections with the segments that link the new vertices to the old adjacent vertices. These edges are dual to the non-diagonal edges of the abstract triangulation. By the definition of an abstract triangulation, such a non-diagonal edge $[v_i, v_{i+1}]$, which is an input (dual) edge of the abstract polygon is adjacent to a single abstract triangle (v_i, v_{i+1}, v_j). The

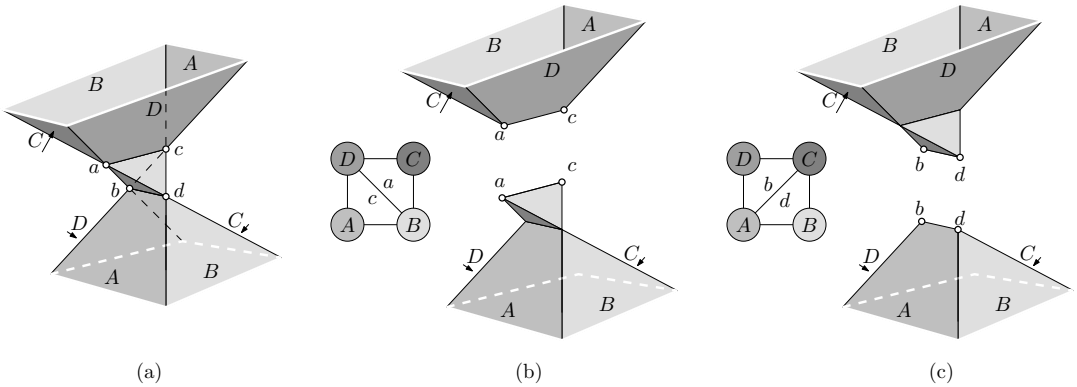


Figure 5.13: Trihedralization is ambiguous with the 1-ring of supporting planes only: (a) 4 input planes $ABCD$ in the order of appearance of the adjacent facets around an over-constrained vertex of valence 4. (b) and (c) show the 2 possible abstract triangulations of this dual polygon. Each possible abstract triangulation is illustrated with 2 geometric embeddings, depending on the adjacent vertices being above or below the tetrahedron $abcd$ defined by the planes A, B, C and D.

geometric embedding is however not fully defined. The decomposability assumption 5.1 allows to consider that the location of the adjacent old vertex of the polyhedron is at infinity, making the segment locally look like a 3D half line. However, to check for intersections, we must know which of the 2 possible half lines has to be considered. The abstract triangle and the plane coordinates P_i, P_{i+1} and P_j of its abstract vertices only define the 3D line $P_i \cap P_{i+1}$ supporting the half-line, and its endpoint $P_i \cap P_{i+1} \cap P_j$. The direction of the half-line is given by the point location of the vertex that is adjacent to the over-constrained vertex of the polyhedron along the considered edge.

For instance, a vertex of valence 4 has 2 possible abstract triangulations, depending on the position of its single diagonal. It creates a single diagonal, and thus a single new segment. With a single segment, the above geometric requirement is always true. If it were sufficient, any of the 2 trihedralizations would be valid. Figure 5.13 shows that condition 5.5 is not sufficient and that the position of the neighboring vertices must be taken into account.

In order to handle planes only and to unify the geometric predicates, the location of such a vertex, adjacent to the over-constrained vertex through an edge between two facets supported by two planes denoted P_i and P_{i+1}, is given by a third plane $P_{i+\frac{1}{2}}$ that intersects the line $P_i \cap P_{i+1}$ at the adjacent vertex location. As a result, the input of the trihedralization problem is not directly the dual polygon of the high valence vertex, denoted $(P_i)_{0 \leq i < n}$, but rather the dual polygon $(P_{\frac{i}{2}})_{0 \leq i < 2n}$, where the edges $[P_i, P_{i+1}]$ of $(P_i)_{0 \leq i < n}$ have been added as diagonals to the dual polygon $(P_{\frac{i}{2}})_{0 \leq i < 2n}$, as illustrated in figure 5.12.b. This forms a circular list of abstract (dual) triangles, representing the circular list of adjacent vertices, such that each pair of consecutive triangles share a (dual) vertex, representing the planar facet between the two vertices. Furthermore, $(P_{\frac{i}{2}})_{0 \leq i < 2n}$ is an ordered supset of $(P_i)_{0 \leq i < n}$, and an abstract triangulation of $(P_i)_{0 \leq i < n}$ can be extended to create an abstract triangulation of $(P_{\frac{i}{2}})_{0 \leq i < 2n}$ by introducing the triangles $(P_i, P_{i+\frac{1}{2}}, P_{i+1})$. Given an input abstract triangulation T of $(P_i)_{0 \leq i < n}$, this abstract triangulation of $(P_{\frac{i}{2}})_{0 \leq i < 2n}$ is unique and is denoted here as the **extension** of T to $(P_{\frac{i}{2}})_{0 \leq i < 2n}$.

Concerning the vertex trihedralization figures illustrating this chapter, the geometric embedding of the trihedralization is illustrated in the primal domain using the polyhedral surface itself, and the dual view of the trihedralization only shows the abstract triangulation of the dual polygon of the adjacent facets, rather than cluttering the figure with its extension.

Theorem 5.6. *Let us consider a local vertex trihedralization problem given by an extended dual polygon $(P_{\frac{i}{2}})_{0 \leq i < 2n}$, and an abstract triangulation of this dual polygon. Assuming no geometric degeneracies (abstract triangles refer to well-defined finite points and abstract diagonals to edge*

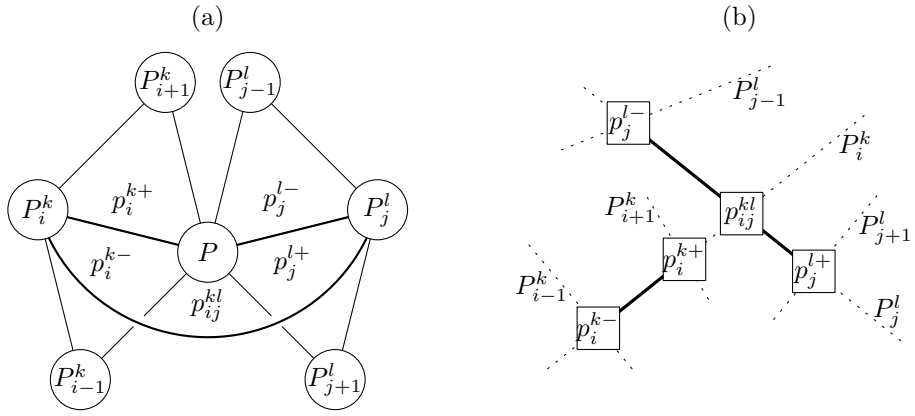


Figure 5.14: Coplanar segments intersection test. (a) The dual abstract triangles involved. (b) Primal 2D view within the plane P that contains both segments $[p_i^{k-}, p_i^{k+}]$ and $[p_j^{l-}, p_j^{l+}]$, and the point $p_{ij}^{kl} = P \cap P_i^k \cap P_j^l$. Planes are represented by their (line) intersection with P.

segments with positive lengths), the following statements are equivalent:

i. The resulting geometric realization of the abstract triangulation has simple facets.

ii. The extended dual polygon $(P_{\frac{i}{2}})_{0 \leq i < 2n}$ satisfies the geometric requirement 5.5.

The proof of this theorem is simply given by translating the geometric requirement on the extended abstract triangulation. It only states that no pair of non-consecutive boundary edges adjacent to a new vertex may intersect, including the edges linking a new to an old vertex.

Coplanar Segments Intersection Test The geometry of the problem of deciding whether 2 coplanar segments intersect is given by 7 planes $P, P_{i-1}^k, P_i^k, P_{i+1}^k, P_{j-1}^l, P_j^l, P_{j+1}^l$, as illustrated in figure 5.14. The input segments $[p_i^{k-}, p_i^{k+}]$ and $[p_j^{l-}, p_j^{l+}]$ are coplanar and their common supporting plane is denoted P. The planes P_i^k and P_j^l define, together with P, the 3D lines supporting the segments. Planes $P_{i-1}^k, P_{i+1}^k, P_{j-1}^l$ and P_{j+1}^l respectively define, along the supporting lines, the endpoints $p_i^{k-}, p_i^{k+}, p_j^{l-}$ and p_j^{l+}. $p_{ij}^{kl} = P \cap P_i^k \cap P_j^l$ is the intersection of the segment supporting lines. The segments intersect if and only if p_{ij}^{kl} is contained in both segments. Thus, the intersection test reduces to 4 point-in-half-line tests. We detail one of them, namely whether the point p_{ij}^{kl}, supported by the line $P \cap P_i^k$, is contained in the half-line along the segment $\{p_i^{k-} + \lambda(p_i^{k+} - p_i^{k-}), \lambda \geq 0\}$. This is tested by considering whether p_i^{k+} and p_{ij}^{kl} are on the same side of P_{i-1}^k, which is the *Above* predicate introduced in section 4.2.4. Since segments are closed, a p_{ij}^{kl} contained in P_{i-1}^k is considered contained in the half-line. Thus, considering an *Above* predicate that returns respectively -1,0 and 1 values in the *below*, *contained* and *above* cases, and given that the segments are not degenerate, the closed half line containment test is simply $Above(p_{ij}^{kl}, P_{i-1}^k) Above(p_i^{k+}, P_{i-1}^k) \geq 0$. The 4 tests together define the intersection predicate:

$$\begin{aligned} Intersect(P, P_{i-1}^k, P_i^k, P_{i+1}^k, P_{j-1}^l, P_j^l, P_{j+1}^l) \quad &= \quad Above(p_{ij}^{kl}, P_{i-1}^k) Above(p_i^{k+}, P_{i-1}^k) \geq 0 \\ &\& \quad Above(p_{ij}^{kl}, P_{i+1}^k) Above(p_i^{k-}, P_{i+1}^k) \geq 0 \\ &\& \quad Above(p_{ij}^{kl}, P_{j-1}^l) Above(p_j^{l+}, P_{j-1}^l) \geq 0 \\ &\& \quad Above(p_{ij}^{kl}, P_{j+1}^l) Above(p_j^{l-}, P_{j+1}^l) \geq 0 \end{aligned}$$

This concludes the translation of our primal local vertex trihedralization problem into the dual problem of finding an abstract triangulation that fulfills a specified geometric criterion. Next section extends this section in the presence of geometric degeneracies.

5.4.3 Handling Degeneracies

5.4.3.1 Well-posed Local Vertex Trihedralization Problems

It is now time to introduce the geometric condition for the local vertex trihedralization problem to be well-posed:

Definition 34 (Well-posed Vertex Trihedralization Problem)**.** *A trihedralization problem* $(P_{\frac{i}{2}})_{0\leq i<2n}$ *is well-posed if, denoting the homogeneous coordinates of a plane* P_i *by* $\vec{P}_i$ *and its normal by* $\vec{n}_i$*:*

- $\vec{n}_{\frac{i}{2}} \neq \vec{0}$ *: Planes are well defined.*
- $\vec{n}_i \wedge \vec{n}_{i+1} \neq \vec{0}$ *: No two consecutive planes may be parallel.*
- $|\,^{\vec{n}_i}\ ^{\vec{n}_{i+\frac{1}{2}}}\ ^{\vec{n}_{i+1}}\,| \neq 0$ *: For all* i*,* $P_i \cap P_{i+\frac{1}{2}} \cap P_{i+1}$ *is a well-defined finite point.*

We define a problem as **ill-posed** if it is not well-posed. $\vec{n}_i \neq \vec{0}$ ensures that adjacent planes are well defined and contain finite points. $\vec{n}_i \wedge \vec{n}_{i+1} \neq \vec{0}$ ensures that consecutive adjacent facets have an edge supported by a well-defined intersection line. $|\,^{\vec{n}_i}\ ^{\vec{n}_{i+\frac{1}{2}}}\ ^{\vec{n}_{i+1}}\,| \neq 0$ ensures that the geometry of the polyhedron away from the over-constrained vertex is well-defined and finite. From now on, trihedralization problems are assumed to be well-posed.

5.4.3.2 Polyhedral Representation Regularization

The representation of a polyhedral surface using its plane equations and the facet/edge/vertex adjacencies is not unique. One of its representations, denoted **regular**, is the one that places vertices and edges exactly at the locations of their corresponding type of discontinuity, when describing the continuous and piecewise planar boundary of the polyhedron. The necessary regularizations are the removal of edges between coplanar facets, of edges of null length and of under-constrained vertices. These simplifying operations, applied recursively, converge to the topology of the regular representation of the same polyhedral surface.

Edges between two coplanar facets may be removed by merging their adjacent facets, and edges of null length may be removed by merging the adjacent vertices. Likewise, a vertex with less than 3 different adjacent facets can safely be removed without altering the described polyhedron: isolated points (valence 0) and point in the interior of a facet (valence 1) are simply removed. Points lying on an edge (valence 2) are under-constrained and may also be removed by merging the two adjacent edges. Once all edges have a well-defined supporting line and vertices have a valence 3 or more, two types of regularizations are still needed. First, higher valence vertices of the representation may still be under-constrained, due to the possibly singular geometry of its adjacent planes. Second, edges between well-defined points may have null length.

Our application does not require the computation of a regular representation of a polyhedron, but only of its facets. In order to express the geometric requirement of the trihedralization in pathological cases, the facets resulting from a tentative trihedralization must be described using a polygonal chain with well-defined and finite endpoints and edges of positive length.

5.4.3.3 Degenerate Trihedralizations because of the Topology only

To begin with, we detail the example case of figure 5.15 that shows that these degeneracies occur in practice even if the set of supporting planes is in general position (*i.e.* its arrangement is simple). The trihedralization problems occurring within our building reconstruction context may contain multiple facets supported by the same planes. Figure 5.15 shows a trihedralization problem where a supporting plane appears twice in the circular list of planes supporting the facets adjacent to the high valence abstract. If a diagonal of the abstract triangulation links two facets supported by the same plane, then the two dual triangles (*i.e.* trihedral vertices) adjacent to this diagonal

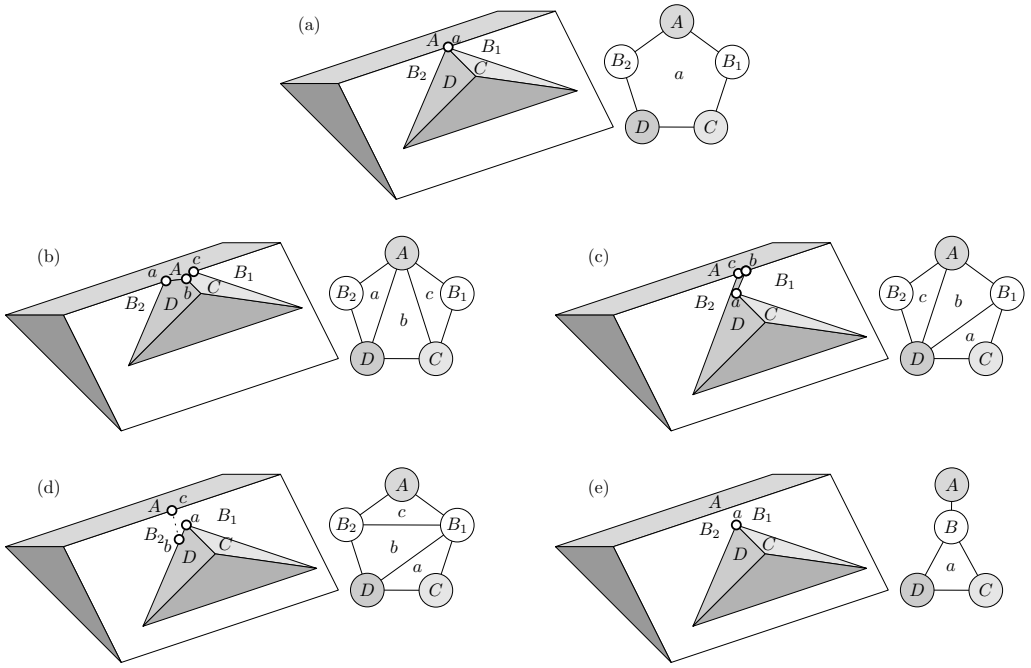


Figure 5.15: Degenerate trihedralization due to its topological setup: (a) A vertex a of valence 5 is adjacent twice to the plane supported by B. (b), (c) and (d) show the result of 3 abstract triangulations out of the 5 possible ones. The two omitted abstract triangulations (c') and (d') are symmetric of cases (c) and (d). Trihedralizations (c), (c'), (d) and (d') define the same geometry and may be regularized to the triangulation (e) by merging the dual vertices B_1 and B_2: cases (c) and (c') contain a degenerate edge $[b, c]$ of null length, whereas cases (d) and (d') contain 2 under-defined vertices b and c of valence 2, generated by dual triangles containing a dual edge between the 2 copies of plane B. The illustrated positions of b and c in (d) is thus arbitrary along their supporting lines. Likewise, points b and c should have been collocated in (c). Depending on the variation of the dual geometry, the trihedralization algorithm must choose which of the 2 polyhedra (b) and (e) contains only simple facets.

of the abstract triangulation will be under-constrained, since two of their defining planes will be equal.

Out of the 5 abstract triangulations of figure 5.8.b:
- Two have a diagonal between the facets that have the same plane support as in (d),
- Two have two adjacent (dual) triangles, such that the (dual) vertices they do not share have the same plane support, yielding a degenerate edge, that is an edge that has null length because of the topology only, as in (c),
- the last one (b) has none of these topological singularities.

Flipping an edge in a primal triangulated polyhedral surface adds to or removes from the described volume the tetrahedron defined by the two former and the two new triangles created by the flip. This property is extensively used in [BDH96] when reducing the 2D Delaunay triangulation computation to the one of the 3D convex hull of the 2D points undergoing the parabolic lifting map of [ES86]. Flipping an edge of the triangulation essentially amounts to adding (or removing) a tetrahedron from the lifted 3D surface.

A similar dual property is also true for edge flips of dual triangulations. A tetrahedron can similarly be defined by the two deleted and two new triangles. An edge flip alters the polyhedral surface by applying its symmetric difference with the tetrahedron surface: it removes from the surface the regions that are also contained in the tetrahedron, and adds to the surface the tetrahedron regions that were not in the surface. However, if the tetrahedron is degenerate (*i.e.* its volume is null), then the edge flip does not modify the surface.

This proves that 4 of the 5 abstract triangulations define the same regularized polyhedral surface, since they can be generated from each other by flipping edges corresponding to degenerate tetrahedra: for instance, edge $[A, D]$ from case (c) to case (d) and edge $[D, B_1]$ from case (d) to its symmetric case (d'). This analysis proves that assuming that the set of supporting planes is in general position is not sufficient to avoid degeneracies.

5.4.3.4 Degenerate Trihedralization Problem Definitions

We introduce the notion of a degenerate trihedralization problem:

Definition 35 (Degenerate Trihedralization Problem)**.** *A trihedralization problem $(P_{\frac{i}{2}})_{0\leq i<2n}$ is degenerate if it is ill-posed or if there exists $0 \leq i < j < k < n$ such that $P_i \cap P_j \cap P_k$ is not a finite point and P_i, P_j and P_k are distinct supporting planes.*

This means that a non-degenerate problem is well-posed and that any of its abstract triangles either contains 2 or 3 facets supported by the exact same plane or maps to a well-defined finite point. This definition does not consider problems with the topology induced degeneracies (*i.e.* with a supporting plane occurring multiple times around the over-defined vertex) as degenerate. Note that, apart from the topology induced degeneracies, a trihedralization of a non degenerate problem might not be a regular representation because of edges of null length. To avoid edges of null length at the level of the input geometry of the problem, we introduce the notion of a strongly non-degenerate trihedralization problem.

Definition 36 (Strongly Non-degenerate Trihedralization Problem)**.** *A trihedralization problem $(P_{\frac{i}{2}})_{0\leq i<2n}$ is strongly non-degenerate if it is not degenerate and if there exists no $0 \leq i < j < k < l < n$ such that $P_i \cap P_j \cap P_k \cap P_l$ is a finite point and P_i, P_j, P_k and P_l are distinct supporting planes.*

Equivalently, a trihedralization problem is strongly non-degenerate if and only if it is well-posed and the arrangement of the planes $(P_i)_{0\leq i<n}$ is simple, without taking into account multiple occurrences of the same plane.

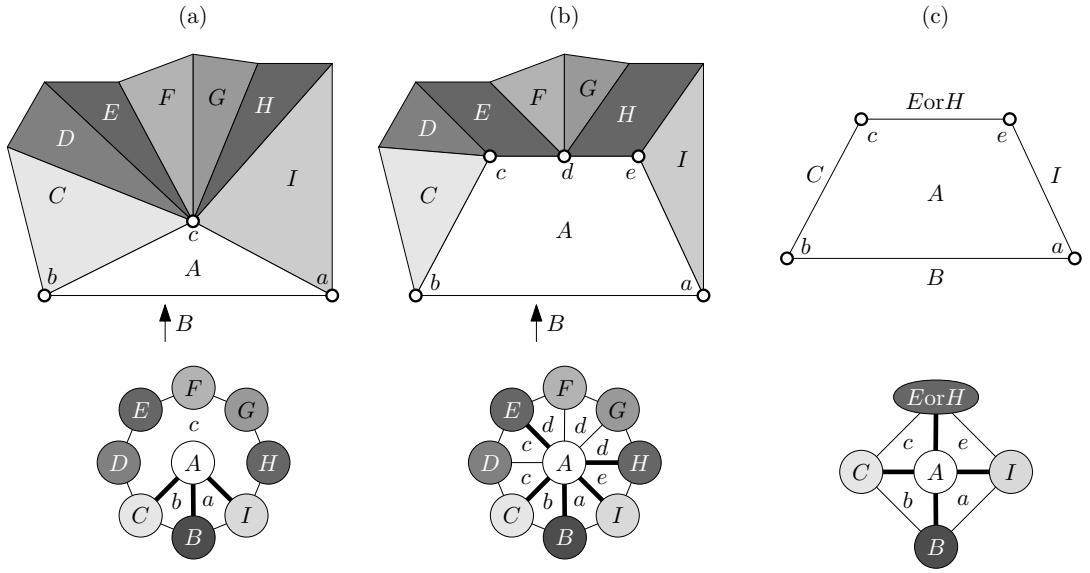


Figure 5.16: Facet regularization. (a) A polyhedron with 9 facets, with a vertex c of valence 8. (b) Its trihedralization after an update with a degenerate dual geometry : A, C, D and E have a new point c as a non empty intersection, $A \cap E$ and $A \cap H$ is the same 3D line, and A, E, F, G and H intersect on point d. The abstract triangulation shows the dual triangles that refer to collocated points on the boundary of facet A. (c) shows a regular representation of the facet A extracted from the abstract triangulation (b).

5.4.3.5 Regular Representation of the Facets

The previous section drives us to generalize the computation of a regular representation of the polyhedron facets from an abstract triangulation with a possibly degenerate input dual geometry. When no degeneracy occurs, the regular representation of a polyhedral facet boundary is straightforwardly given by its corresponding triangle fan in the abstract triangulation, which translates to a polygonal chain of segments bounding the facet. We introduce an algorithm that outputs these polygonal chains as an ordered list of neighboring facets. Figure 5.16 illustrates this approach, by giving a regular representation of the facet A after the trihedralization of the point c. Because of the geometric degeneracies of the updated plane geometry, the triangle fan around A after the trihedralization contains redundant dual triangles. The idea is to only use a subset of the outgoing dual edges of A, namely $BCEI$, or equivalently $BCHI$.

The filtering of the neighboring facets defining the boundary of a given facet A goes as follows.

1. The circular list of dual edges is initialized using the outgoing dual edges of A (*e.g.* $BCDEFGHI$ in the abstract triangulation 5.16.b).
2. All its neighbors are tested for parallelism against A. If a neighboring facet is parallel, but not equal to A, or if it is equal but has an opposite orientation, then the abstract triangulation is considered invalid. All its equal neighbors are merged together and their circular list of neighbors as well. This handles the case of dual edges between coplanar facets.

Then the other kinds of degeneracies have yet to be handled, namely, 3 planes intersecting on a line (*i.e.* their intersecting point is not well-defined) and 4 planes intersecting on a point (*i.e.* they cannot define a non-trivial edge). To filter the circular list of adjacent planes $(P_i)_{0 \leq i < n}$, we propose to filter recursively the list until no degeneracies are left. First, if a dual triangle (P_i, A, P_{i+1}) yields an infinite point, then the abstract triangulation is considered invalid. Otherwise, if this point is undefined, then the dual edges (A, P_i) and (A, P_{i+1}) are supported by the same 3D line. They are thus redundant and any of the two planes P_i or P_{i+1} may be filtered out. Now that all the dual triangles (P'_i, A, P'_{i+1}) of the filtered list $(P'_i)_{0 \leq i < n'}$ are located at well-defined finite points,

edges of null length are searched. This is done by computing the determinant of the 4 by 4 matrix $[\,A\ P'_{i-1}\ P'_i\ P'_{i+1}\,]$. If such a determinant is null, then P'_i can be filtered out, as happened to planes D, F and G in figure 5.16. This filtering creates a new dual triangle (P'_{i-1}, A, P'_{i+1}) that must be checked for degeneracy. Figure 5.16 shows such a degenerate new triangle (E, A, H), created after the planes F and G had been filtered out.

The algorithm outputs a filtered list of neighboring facets $(P'_i)_{0\leq i<n'}$. The correctness of the algorithm is based on the filtering of the planes until the following families of homogeneous plane coordinates are full-rank : planes of an edge (A, P'_i), planes of a dual triangle (A, P'_i, P'_{i+1}), and planes of two consecutive triangles in the filtered list $(A, P'_{i-1}, P'_i, P'_{i+1})$.

The regularization of the facet boundary representation patches the theorem 5.6 when degeneracies are present. Its geometric requirements must be not be checked directly on each resulting adjacent facet but on their filtered representation.

5.4.4 Discussion

This section precisely introduced the proposed local vertex triangulation problem, its topological search space and its geometric requirements. Three assumptions were made that were not crucial for the polyhedron trihedralization decomposition into independent vertex trihedralizations.

First, the geometric requirement introduced in this section only checks for the facet simplicities but not for the improper intersections. This choice is driven by the scope of this thesis. Although the trihedralization problem statement is general, it will only be used in this thesis on a restricted class of polyhedra (polyhedra that may be decomposed into a 2.5D polyhedral surface and a single arbitrarily low bottom facet). These polyhedra do not suffer from improper self-intersections. Checking improper self-intersections may be included as an additional geometric requirement. This would however not invalidate the topological search space as the set of abstract triangulations.

Second, the locality assumption yields multiple benefits. It simplifies the vertex trihedralization problems by having to consider only the local arrangements. The dual polygon is then readily available as it is given by the circular list of adjacent facets, whereas a non-local vertex trihedralization problem has to compute the intersection of the restricted polyhedral surface with the vertex zone (a set of 3D points, 3D segments and/or non-planar 3D polygons) to get multiple circular lists of adjacent facets. The topological search space has then to be extended to triangulate these sets of abstract polygons rather than a single abstract polygon. The simultaneous abstract triangulation of such a set of abstract polygons would have to be properly defined as the combinatorial part of the triangulation of a set of polygon with holes.

Third, reducing vertex trihedralizations to abstract triangulations considers that the polyhedral overconstrained vertices have a single circular list of adjacent planes (*i.e.* topological polyhedral surfaces are manifold). This may not be the case in our 2.5D application. It would require a similar treatment as a non-local vertex trihedralization, with multiple rings of adjacent planes.

Chapter 6 will provide a framework to cope with the lack of trihedralization locality and decomposability.

5.5 Ear-cutting-based Local Vertex Trihedralization

This section proposes a local vertex trihedralization algorithm, that takes advantage of the abstract triangulation formulation of the previous section. Section 5.5.1 exposes the abstract ear-cutting paradigm. Then, section 5.5.2 quickly reviews its initial application: triangulating simple polygons. Finally, we propose to apply this ear-cutting paradigm to our local vertex trihedralization problem in section 5.5.3.

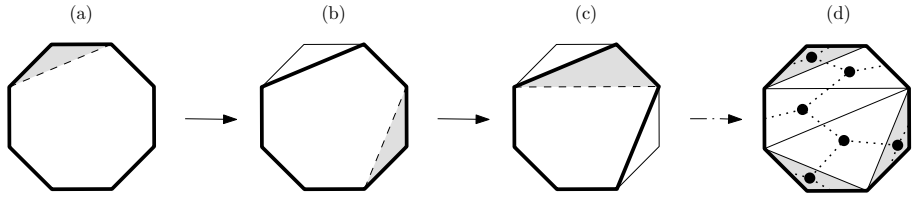


Figure 5.17: Abstract ear cutting. (a) The initial abstract polygon (in bold) has an abstract ear (shaded) cut, and then recursively with 2 (b), 3 (c) and finally 5 abstract ears (d). (d) shows a triangulation of (a) with 3 abstract ears (shaded) together with the dual of the triangulation.

5.5.1 Ear-cutting Abstract Triangulation

A triangulation of a planar convex polygon, or, equivalently, an abstract triangulation of an abstract polygon, may be computed by splitting the polygon recursively with diagonal edges between two of its vertices. If the original polygon has n vertices, the two half polygons caused by a diagonal split will have k and $n+2-k$ vertices (indices are taken modulo n to account for the circular ordering). A possibility is to take a vertex and to triangulate the polygon using a triangle fan around this vertex, that is using all the diagonals between the chosen vertex and all the $n-3$ other non-adjacent vertices. This process readily provides an abstract triangulation. However, by varying the choices of the splitting diagonals, one can generate all the $T(n)$ (see section 5.4.1) abstract triangulations of the polygon.

Instead of splitting using arbitrary diagonals, the choice may be limited to cut the polygon into a triangle ($k=3$) and a polygon of size $n-1$. Such an abstract triangle is called an abstract ear:

Definition 37 (abstract ear)**.** *An **abstract ear** is a triangle of an abstract triangulation formed by 3 consecutive abstract vertices* (v_{i-1}, v_i, v_{i+1}).

Recursively cutting ears rather than allowing splits from arbitrary diagonals simplifies the process by providing a constant size problem (the abstract ear) and a reduced size problem (treated recursively), rather than two problems of varying sizes (fig. 5.17). The following theorem [Mei75] can be proven by considering the abstract triangulation as the dual topology of an unrooted binary tree (fig. 5.17.d).

Theorem 5.7. *An abstract triangulation of more than 3 vertices has at least 2 abstract ears.*

Since at least one abstract ear is always available, exploring the $T(n)$ abstract triangulations may performed by recursively splitting abstract ears. Namely, limiting diagonal splits to abstract ear cuttings is not restrictive.

The goal of an abstract triangulations problem is to find an abstract triangulation that satisfies a given specific geometric requirements. finding an abstract triangulation that fulfills some geometric requirements may be performed by testing all the possible abstract ears and recursively computing the abstract triangulation on the ear-cut polygon (fig. 5.17). The ear-cutting approach is thus a generic way to enumerate all the abstract triangulations. Before applying it to our trihedralization problem, this approach is presented on the simple polygon triangulation problem.

5.5.2 Ear-cutting Triangulation of a Simple Polygon

A simple, yet suboptimal, simple polygon triangulation algorithm can be designed using the ear-cutting approach. Figure 5.18 illustrates the following definition.

Definition 38 (Primal ear)**.** *A **primal ear** is the geometric realization of an abstract ear that verifies the geometric property that its supporting triangle lies entirely inside the polygon.*

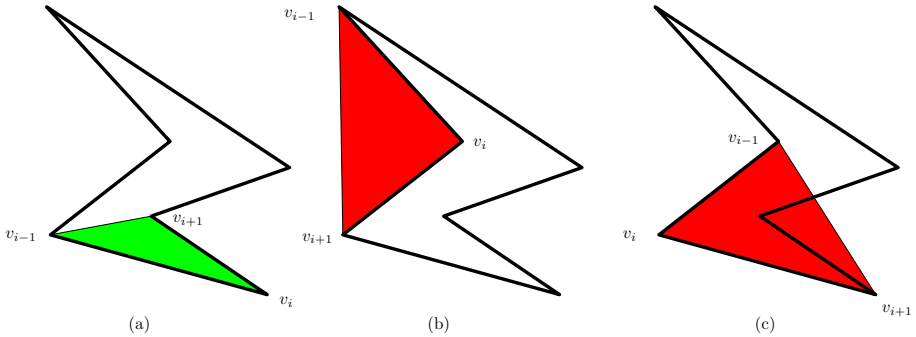


Figure 5.18: (Primal) ear geometric requirements. Among the 3 abstract ears (a), (b) and (c) of the pictured simple polygon, only (a) is a primal ear. The abstract ear in (b) falls outside the polygon and the one in (c) does not lie entirely inside it.

A primal ear is usually simply referred to as an ear in the literature. However the term *primal ear* is used here in contrast with the *abstract ear* and the forthcoming *dual ear* terms.

Splitting a simple polygon with n vertices along the diagonal edge $[v_{i-1}, v_{i+1}]$ of an ear (v_{i-1}, v_i, v_{i+1}) will yield a direct triangle and a simple polygon with $n-1$ vertices that does not contain the vertex v_i. Without considering the geometric requirement that the triangle is entirely contained inside the polygon, the n abstract ears, one for each vertex, are all equivalent. Theorem 5.7 states that any abstract triangulation, except for triangles, has at least two abstract ears. Since they cannot be consecutive, one may only test the $n-2$ consecutive abstract ears $1 \dots n-2$, when searching for a primal ear to cut out of the polygon.

Algorithm 4 *EarCuttingTriangulation*$(v_0 \dots v_{n-1}, \mathcal{T})$

Require: $(v_0 \dots v_{n-1})$ is a list of $n \geq 3$ vertices.
Require: $\mathcal{T}$ is the running set of triangles of the triangulation (initially empty).

```
for i = 1 to n − 2 do
  if (v_{i−1}, v_i, v_{i+1}) is a primal ear then
    T' ← T ∪ (v_{i−1}, v_i, v_{i+1})          // Add the abstract ear to the triangulation
    if n = 3 then
      return T'
    else
      return EarCuttingTriangulation(v_0 ... v_{i−1}, v_{i+1} ... v_{n−1}, T')
return ∅
```

Algorithm 4 returns an abstract triangulation $\mathcal{T}$ that is not empty if and only if the input polygon is simple. The *primal ear* test can be performed in $O(r)$, where r is the number of reflex vertices (*i.e.* of vertices which interior angle is greater than π), by checking that v_i is convex and that the triangle v_{i-1}, v_i, v_{i+1} contains no reflex vertex. A complexity analysis proves that, far from the optimal $O(n)$ algorithm [Cha91], this algorithm takes $O(n^2 r)$ time, when applied on a simple polygon. This is due to the algorithm being called recursively $n-2$ times, the $n-2$ *for loop* iterations, and the returned triangulation never being empty, since the polygon is simple. This algorithm is highly suboptimal but provides a framework to explore the abstract triangulations of an abstract polygon intuitively. For instance, [EET93] developed a $O(n^2)$ optimized ear-cutting triangulation algorithm.

5.5.3 Ear-cutting Local Vertex Trihedralization

Unluckily, the triangulation of a simple 3D polygon is not the dual of our trihedralization problem. Whereas they compute the same combinatorial object, namely an abstract triangulation, their geometric requirements are not dual of each other. A 3D polygon is supported by a plane and bounded by coplanar points. Its dual is then a point lying on meeting planes. Therefore, a trihedralization of such a vertex only adds null length edges. Thus any abstract triangulation fullfills the geometric requirement that the resulting facets are self-intersection free, as in the case of the triangulation of a planar convex polygon.

This section solves the trihedralization problem by a dual *ear-cutting* triangulation approach. Our dual problem of making a vertex trihedral is performed by iteratively intersecting 2 consecutive edges of the high valence vertex which produces a new ear-like trihedral vertex and decrements the valence of this vertex by 1. An ear cutting then amounts to splitting a vertex of valence n into a trihedral vertex and a vertex of valence $n-1$ by uncollapsing an edge.

5.5.3.1 Ear-cutting Trihedralization Algorithm

As described in section 5.4.2, the duality only applies to the topology of the problem, but not to its geometric requirements of generating simple facets. The geometry of a facet is known as soon as its dual vertex is only surrounded by triangles, which makes it the apex of a triangle fan. The simplicity property can thus only be checked for facet f_i, when the abstract ear (f_{i-1}, f_i, f_{i+1}) is considered. A dual ear is then defined as an abstract dual ear which middle dual vertex refers to a simple facet. Note that, contrary to the primal ear definition, when the polygon has only 3 vertices, this dual ear definition is not symmetric, since it might occur that (f_0, f_1, f_2) is a dual ear (*i.e.* f_1 is self-intersection free), but not (f_2, f_0, f_1) (*i.e.* f_0 self-intersects). The algorithm 5 is a translation of the primal algorithm 4. Apart from the translation from vertices to facets and primal ears to dual ears, there are two variations :

1. When only 3 facets are left, one has to check that (f_0, f_1, f_2), (f_2, f_0, f_1), (f_1, f_2, f_0) are all dual ears, because of the asymmetry of the dual ear geometric requirement.
2. After a recursive call, the result is only reported if it is not empty (*i.e.* if the subproblem is trihedralizable). Given one of its primal ears, a non-triangular polygon is triangulable if and only if the polygon obtained by cutting this primal ear is triangulable. A theorem of this kind is yet to be proved concerning trihedralizations. Thus, finding a dual ear is not sufficient, and the algorithm has to test all the dual ears until one dual ear yields a trihedralizable subproblem.

To handle the degeneracies discussed in section 5.4.3, the test whether (f_{i-1}, f_i, f_{i+1}) is a dual ear, is performed in 3 steps :

1. If f_i is supported by the same plane as f_{i-1} or f_{i+1}, then the simplicity check is postponed to the handling of this vertex, and the abstract ear is considered as a dual ear. If an adjacent facet is parallel but not equal (including its orientation), then the abstract ear is not considered a valid dual ear. If no adjacent facet is coplanar, then the facet is delimited by a single polygonal chain. Finally, if f_i is supported by the same plane as N of its other adjacent facets, then the abstract ear, if accepted, finalizes the representation of a facet in which the trihedralization created $N+1$ polygonal chains. This case occurred in figure 5.15.d, where the facet $B_1 = B_2$ is bounded by the polygonal chain delimited by A and the one delimited by C and D. For instance, given a dual edge between 2 equal facets, their lists of adjacent planes $(P_1 \dots P_{k-1}, P_k, P_{k+1} \dots P_m)$ and $(Q_1 \dots Q_{l-1}, Q_l, Q_{l+1} \dots Q_n)$ where $P_k = Q_l$ are split and merged to form the 2 new lists $(P_1 \dots P_{k-1}, Q_{l+1} \dots Q_n)$ and $(Q_1 \dots Q_{l-1}, P_{k+1} \dots P_m)$.
2. The geometry of the facet f_i is given by circular lists of facets adjacent to f_i, or only one such list if f_i has no holes. When no splitting and merging occurred, the trihedralization provides a sublist of this circular lists through the fan of edges outgoing from the facet in the trihedralization. When N coplanar facets are connected in the trihedralization, the first step

Algorithm 5 *EarCuttingDualTriangulation*$(f_0 \dots f_{n-1}, \mathcal{T})$

Require: $(f_0 \dots f_{n-1})$ is a list of $n \geq 3$ dual vertices (*i.e.* facets).
Require: $\mathcal{T}$ is the running set of dual triangles (*i.e.* trihedral vertices) of the trihedralization (initially empty).

```
for i = 1 to n − 2 do
  if (f_{i−1}, f_i, f_{i+1}) is a dual ear then
    T' ← T ∪ (f_{i−1}, f_i, f_{i+1})          // Add the abstract ear to the triangulation
    if n = 3 then
      if (f_2, f_0, f_1) and (f_1, f_2, f_0) are dual ears then
        return T'
      else
        return ∅
    T' ← EarCuttingDualTriangulation(f_0 ... f_{i−1}, f_{i+1} ... f_{n−1}, T')
    if T' ≠ ∅ then
      return T'
return ∅
```

splitted and merged their N edge fans to form N distinct sublists of the circular lists of the adjacent facets. Section 5.4.3.3 proposed a method to filter these lists so that they represent polygonal chains with well defined finite points and non-trivial edges. If this filtering aborts because it creates a vertex of the polygonal chain that is not well-defined and finite, then the ear (f_{i-1}, f_i, f_{i+1}) is not considered as a dual ear.

3. The last step is to consider the N filtered lists of adjacent planes of the facet f_i (supported by a plane P) due the trihedralization, denoted $(P_1^1 \dots P_{n_1}^1)$ to $(P_0^N \dots P_{n_N}^N)$. If we consider 4 integers i, j, k, l such that $1 \leq k, l \leq N$, $1 < i < n_k$, $1 < j < n_l$ and if $k = l$ then $|i - j| > 1$, then the 4 dual triangles (P, P_{i-1}^k, P_i^k) and (P, P_i^k, P_{i+1}^k) on one hand, and (P, P_{j-1}^l, P_j^l) and (P, P_j^l, P_{j+1}^l) on the other hand, define 2 edges of the boundary of f_i that are not consecutive (see figure 5.14). The simplicity of facet f_i is finally tested by checking that no such combination (i, j, k, l) refers to a pair of intersecting segments.

The test whether two segments intersect is described in section 5.4.2.2. Since only pairs of non-consecutive segments are considered, the segments are closed, so that if their intersection is one of their endpoints, the intersection is reported and the facet is not considered simple.

5.5.3.2 Discussion

To comment on algorithm 5, section 5.7.2 will prove that a trihedralization does not always exist. In such a case, algorithm 5 returns an empty triangulation ($\mathcal{T} = \emptyset$). Likewise, section 5.7.1 will show that a self-intersection free trihedralization is not necessarily unique. Thus, algorithm 5 only reports the first self-intersection free trihedralization it discovers. However, a simple modification would enable the reporting of all the trihedralizations instead. A preliminary analysis of the algorithmic complexity yields $O(n!)$ time and $O(n)$ space.

Proof. The dual ear predicate takes time $D(n)$ proportional to the number of non consecutive edge pairs in the filtered triangle fan of the middle vertex of the ear. This can be bounded by $\binom{n}{2} = \frac{n(n-1)}{2}$. A recursive call, when $n-k-2$ ears have already been cut, takes $3D(n)$ time when the subproblem has only 3 facets left ($k = 1$), or makes k dual ear checks and recursive calls. The time complexity $T(n, k)$ is then:

$$T(n,3) = 3D(n) = O(n^2)$$

$$T(n,k) = k\,(D(n) + T(n,k-1)) = D(n)\left(k + k(k-1) + k(k-1)(k-2) + \ldots + \frac{k!}{2}\right) = D(n)k!\overbrace{\left(\sum_{i=2}^{k-1}\frac{1}{i!}\right)}^{=O(1)}$$

Initially $k = n - 2$, and thus, the overall time complexity is $T(n, n-2) = O(n!)$. This analysis is pessimistic as it counts the complexity of all the recursive steps as if all the dual ears were fullfilling their geometric requirement, but yet the algorithm backtracked to explore all the recursive paths. □

The worst-case valence of a vertex is linear with the size of the polyhedron. However, within our applicative context, the valence of each vertex is rather a constant. Thus, the asymptotic complexity is not really relevant when fitting building polyhedra to some input DSM. The space complexity of this algorithm is optimal : this is the one of a single abstract triangulation, which is linear.

However, more global self-intersections may still occur even if the facets are self intersection free. To report a self-intersection free trihedralization, the dual ear geometric requirement must be extended to avoid the intersection of the facet it closes with all the already closed facets by previously cut dual ears. An other approach is to simply compute the list of trihedralization with self-intersection free facets, and then filter them for proper facet intersections.

5.6 Local Vertex Trihedralizations and Straight Skeletons

The trihedralization problem has a strong connection with the straight skeleton of a 2D polygon, and more precisely its weighted variant. Section 5.6.1 introduces the unweighted straight skeleton, while section 5.6.2 presents its weighted variant. Then, our contributions follow. Section 5.6.3 casts the weighted straight skeleton problem as a trihedralization problem. Conversely, section 5.6.4 reduces a specified subclass of trihedralization problems to weighted straight skeleton problems. Finally, section 5.6.5 discusses the results and provides a possible extension.

5.6.1 Unweighted Straight Skeleton

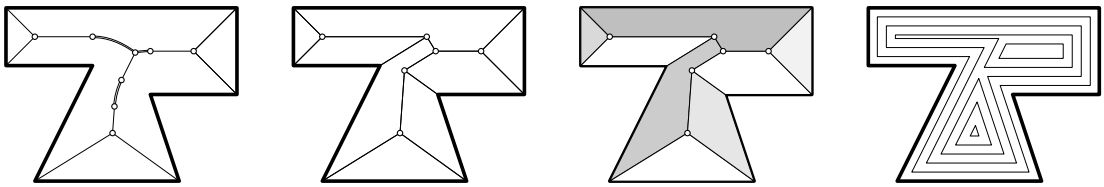
The straight skeleton, introduced by [AAAG95], is a kind of skeleton for a polygon, similar to the medial axis [CSW99].

The medial axis (fig. 5.19.a) is defined as a the set of points inside the polygon that have more than one closest point on the boundary of the polygon. This set of points forms a skeleton, which is a tree-like set of curves that partition the polygon. Those curves capture the local symmetries of the polygon but are not guaranteed to be linear. If the polygon presents a segment in front of a reflex vertex (a vertex with an interior angle greater than π), the set of disk centers describes a parabola.

The straight skeleton (fig. 5.19.b) is another type of skeleton that features only linear curves. It is defined as the trace of the polygon vertices as its edges move inward at the same speed. This describes the straight skeleton constructively by a shrinking process. It has the nice property that it is equivalent to the medial axis when the polygon has no reflex vertex (*i.e.* is convex). The set of polygons created by this shrinking process, called offset polygons (fig. 5.19.d), are by construction linear, whereas the offset curves of the medial axis may contain circular arcs.

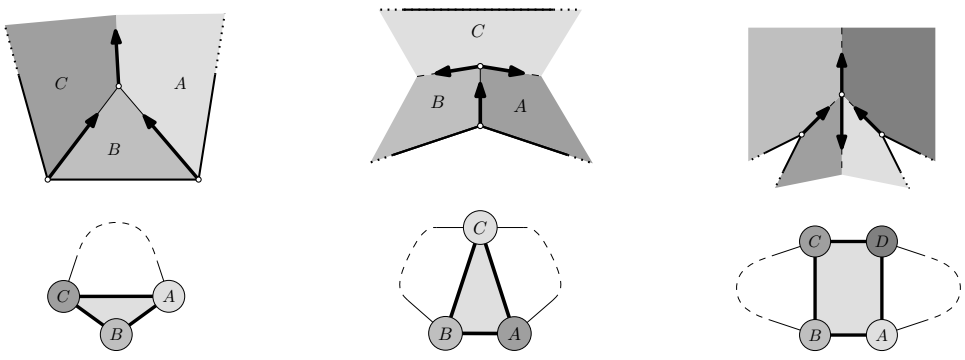
This 2D problem is well-known in the building reconstruction field [Bre00], since the 2D straight skeleton of a building footprint computes the horizontal projection of the building roof with one roof facet per wall and equal roof facet slopes. This 3D embedding has been used to compute the shadings of figures in this section (*e.g.* figure 5.19.c). The straight skeleton is however harder to compute than the medial axis since it has no Voronoï diagram based interpretation. The non local effect of the reflex vertices on the straight skeleton makes incremental construction or divide and conquer approaches fail.

A simple algorithm to compute the 2D straight skeleton is to simulate directly the polygon shrinking process, as proposed in [AAAG95]. This original algorithm takes $O(n^2 \log n)$ time and



(a) Medial axis (b) Straight skeleton (c) Shaded view of (b) (d) Offset polygon

Figure 5.19: The medial axis (a), the straight skeleton (b,c) and a set of offset polygons (d) of a T-shaped polygon, shown in bold. The double edges of (a) are parabolic, while other edges are supported by straight lines.



(a) Edge event (b) Split event (c) Vertex event

Figure 5.20: Straight skeleton topological events, primal and dual view. The edge (a), split (b) and vertex (c) events occurring during the shrinking of a polygon. The second row shows the mapping of these events in terms of the resulting abstract triangulation.

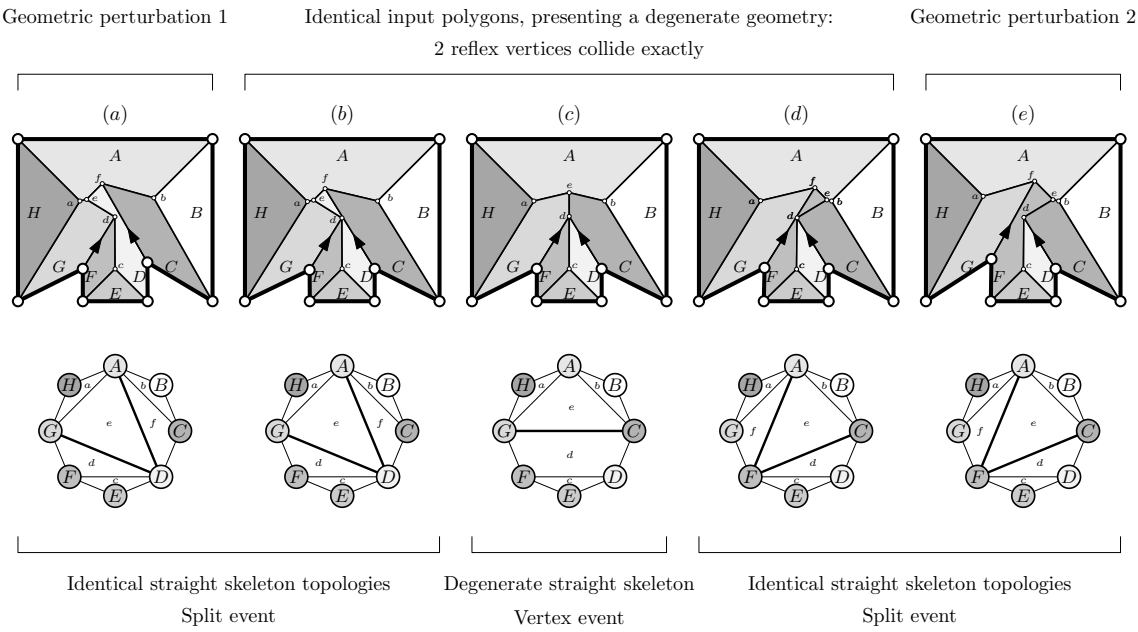


Figure 5.21: An unstable degenerate straight skeleton problem. Vertex events have been introduced as a consistent processing of the degenerate meeting of 2 reflex vertices, as in the straight skeleton of the polygon of (c). (a) and (e) are two straight skeletons of a small geometric perturbation of the polygon of (c). (b) and (d) are the straight skeletons of the polygons of (a) and (e) as their perturbation magnitudes tend to zero.

$O(n^2)$ space. The shrinking of the polygon is simulated by detecting and processing 3 types of shrinking events (see figure 5.20).

Edge event: The length of an edge shrinks to zero. The processing of this event simply requires collapsing the edge by making its neighboring edges adjacent.

Split event: A reflex vertex hits an edge. This splits the colliding edge and vertex, and, as a consequence, the whole shrinking polygon into two disjoint shrinking polygons. Both resulting splitted vertices, which may or may not be reflex, are adjacent to an edge of the previous reflex vertex and one of the two halves of the splitted colliding edge.

Vertex event: It is a degenerate split event, that occurs when two reflex vertices collide. It creates a vertex of valence 4 and splits the polygon, creating two new straight skeleton edges.

Figure 5.21 shows that the construction of the straight skeleton is not continuous as a function of the vertices of the input polygon, and thus that the straight skeleton of the degenerate polygon, common to (b), (c) and (d), cannot be defined as the limit of the straight skeleton of polygons that tend to the degenerate polygon. To define a consistent straight skeleton in these degenerate configurations, [EE99] introduced these vertex events.

A fourth type of event is the vanishing event where the shrinking polygon area becomes null. The shrunk polygon is then reduced to a connected set of edges and vertices that are added to the straight skeleton. The simplest case is the meeting of the three vertices of a shrinking triangle, which creates a single straight skeleton vertex at the collision location, linked to the 3 paths traced out by the 3 vertices of the shrinking triangle.

Subsequent algorithmic improvements have been made to handle more efficiently the non-locality of reflex vertices. [EE99] introduced the first subquadratic algorithm for computing the straight skeleton of a simple polygon of n vertices, r of which are reflex, taking $O(n^{1+\epsilon}+n^{\frac{8}{11}+\epsilon}r^{\frac{9}{11}+\epsilon})$ time and space for any $\epsilon > 0$. The r-insensitive bound of $O(n^{\frac{17}{11}+\epsilon})$ is still higher than the only known lower bound $\Omega(n \log n)$. A better expected time bound has been reached assuming that the

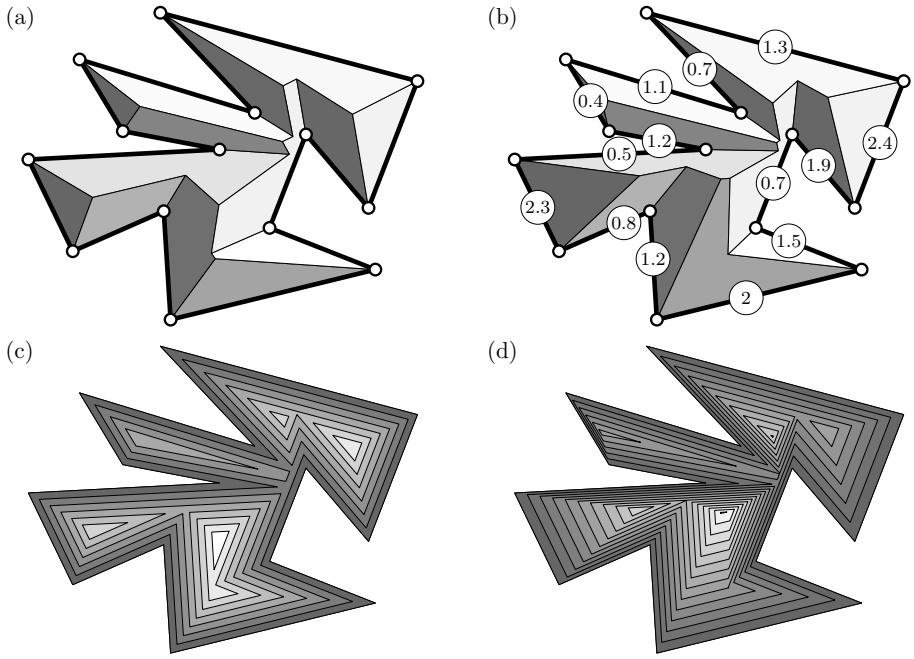


Figure 5.22: (a) and (b) are the weighted straight skeletons of the same non-convex polygon. The edge weights of (a) are all equal to 1, yielding the unweighted straight skeleton, whereas (b) has random real positive edge weights (circled). (c) and (d) are offset polygons generated respectively from (a) and (b).

polygon is non degenerate (no vertex events): [CV02] have provided a $O(n\sqrt{h+1}\log^2 n + r\sqrt{r}\log r)$ expected time algorithm to compute the straight skeleton of a non-degenerate simple polygon with h holes, n vertices, r of which are reflex. In particular, the straight skeleton of a non-degenerate polygon with n vertices can be computed in $O(n\sqrt{n}\log n)$ expected time.

5.6.2 Weighted Straight Skeleton

The straight skeleton problem has been extended in two ways. First, it can be defined on the whole plane for any set of segments, and not only in the bounded region enclosed by a set of segments that describe a simple polygon [AA96].

The second extension is the weighted straight skeleton, mentionned in [EE99] and illustrated in figure 5.22. This is a slight modification of the unweighted straight skeleton where edges are moving at different specified speeds. It maps to roofs that have specified slopes, which are not necessarily equal, for each wall. A weight $w = 0$ maps to a vertical wall, while $w = \infty$ corresponds to a flat horizontal roof plane. If all the weights are positive, then the area of the shrinking set of polygons decreases and then vanishes, proving the termination of the shrinking process, and thus the existence of the weighted straight skeleton.

Negative weights are also a possible extension, where roof planes are allowed to be bottom facing as in roofs with overheads. However special care has to be introduced in the scheduling of events since they may both occur inside and outside the input polygon. Lastly, whereas the unicity is given by its construction process (at least for non degenerate inputs), the existence of the 2D weighted straight skeleton is not guaranteed since the proof that the area of the offset polygons is decreasing no longer holds.

5.6.3 Reducing Weighted Straight Skeletons to Vertex Trihedralizations

The weighted straight skeleton of a polygon can be seen as a particular instance of a trihedralization problem, when considering its 3D embedding as a roof surface. The trihedralization problem is instantiated by specifying the initial polyhedron topology, the vertex to be made trihedral and the facet supporting plane geometries. The initial polyhedron has the topology of a pyramid: its base facet is the input polygon and its apex is the overconstrained vertex, linked to each base segment with a triangular facet. The plane geometry is given by the bottom facing plane $\vec{Z} = [0 : 0 : -1 : 0]$ for the base facet, and planes of the form $[a : b : w\sqrt{a^2 + b^2} : d]$ for the triangular facets. w is the weight of the base edge, and $[a : b : d]$ are the 2D homogeneous coordinates of the line supporting the base edge. This simulates a rotation of the plane around its defining edge from the plane that initially and abstractly meet at the vertex to be trihedralized to the plane that has the slope according to the edge speed w. Figure 5.23.a illustrates such a reduction. The top polygon is embedded in 3 space using planes passing through its supporting edges with a slope according to the edge weights. Then an over-constrained vertex $ABCDEFGH$ is introduced, along with the vertices of the polygon, as abstract triangles between a bottom facing face Z and two consecutive planes. To simplify the figures, the facet supported by the plane Z is only shown in the dual view of the first column.

Figure 5.20 shows how the edge events map to a dual ear cutting, and the split events to the decomposition of the abstract polygon in two using an abstract triangle that is not an ear. The vertex event is a degenerate split event where the polygon is split in two using an abstract quadrangle (the degeneracy implies that both triangulations of the quadrangle describe the same polyhedral surface).

5.6.4 Reducing Vertex Trihedralizations to Weighted Straight Skeletons

If the overconstrained vertex satisfies the following property, its trihedralization may be reduced to a weighted straight skeleton problem.

Definition 39 (Extremal vertex)**.** *A vertex is **extremal** if there exists a direction $\vec{n}$ such that all the outgoing edges of the vertex are directed by a vector $\vec{v}$ such that $\vec{n} \cdot \vec{v} < 0$.*

This may not seem well-defined since the location of the over-constrained vertex is not well-defined by definition, being at the intersection of non-meeting planes. However the directions of its edges that point to an over-constrained vertex are well defined: an edge is supported by the well-defined intersection line of its two neighboring planes, and oriented according to the position of the adjacent vertex along this line. If a vertex is extremal along the vector $\vec{n}$, then for each edge oriented by a vector $\vec{v}$ from an adjacent vertex to the extremal vertex, we have $\vec{n} \cdot \vec{v} < 0$.

Under the locality assumption, the converse reduction from a trihedralization to a weighted straight skeleton is true for extremal vertices, by treating such an extremal vector as the vertical vector, and cutting the extremal vertex by a new auxiliary plane as in figure 5.11:

Theorem 5.8. *A trihedralization of an extremal over-constrained vertex reduces to a weighted straight skeleton computation by sweeping a plane in an extremal direction $\vec{n}$, under the condition that an offset d exists such that the plane $[\vec{n} : d]$ intersects all the adjacent edges and that its intersection with the adjacent planes is a simple polygon.*

Any of of these simple polygonal sections can be used as the input polygon. The edge weights are given by the tangents of the dihedral angle between the vector $\vec{n}$ and the plane normal. If the locality assumption is verified, there is an offset d, such that the intersection of the section plane $[\vec{n} : d]$ and the polyhedron is a simple polygon and the section planes have the query vertex on one side and all its neighbors on the other side. Under this assumption, this construction proves the existence of a trihedralization for extremal vertices.

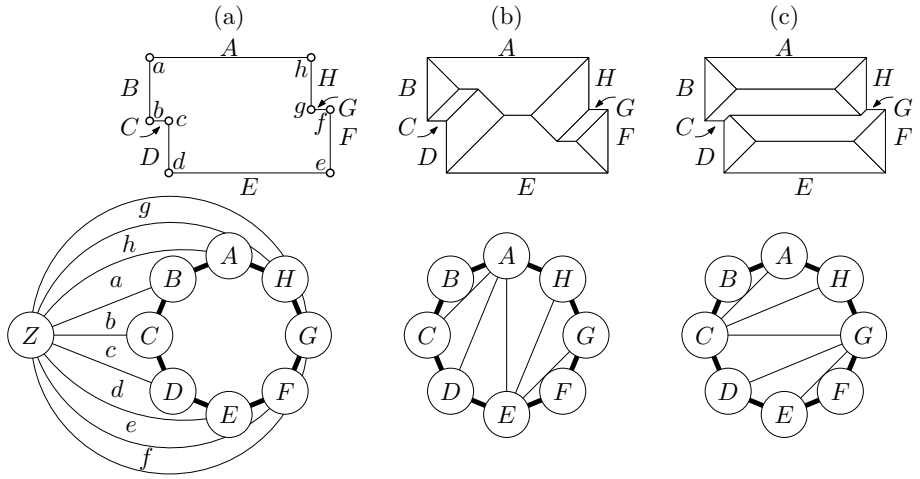


Figure 5.23: The straight skeleton, as a trihedralization. (a) A polygonal building footprint and its (dual) topology when reducing the straight skeleton problem to a trihedralization problem, (b) its straight skeleton and (c) an alternative trihedralization that may or may not be closer to the reality. The first row is extracted from [Bre00], while the second is the underlying abstract triangulations.

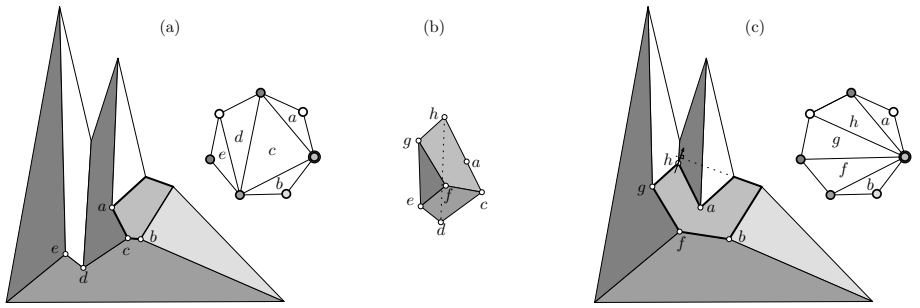


Figure 5.24: Two trihedralizations : (a) satisfies the property 5.1, while (c) does not, since the vertex h has a minimal z-coordinate but is not contained in the edge that generated it. (b) shows the difference of the straight skeletons (a) and (c) viewed as volumes.

The topology of the resulting weighted straight skeleton does not depend on the extremal direction. Varying the extremal direction modifies the section polygon but also the weights accordingly. This ensures the unicity of the weighted straight skeleton as a trihedralization (up to degeneracies).

5.6.4.1 Saddle Vertices

A **saddle** vertex is a vertex that is not extremal. [BEGV08] proposed to trihedralize saddle vertices using two complementary weighted straight skeletons, in a restricted context where the adjacent planes undergo a translation along their normal of the same infinitesimal distance, from the initial degenerate geometry where all adjacent planes meet at a single point. However, it is unproven whether the method proposed by [BEGV08] form a trihedralization that is not self intersecting in our more general case. Thus we fail to provide a reduction to straight skeletons for a trihedralization of a saddle vertex.

5.6.4.2 z-minima Property

As [Bre00] mentionned, the (weighted) straight skeleton is only a distinguished trihedralization among all the trihedralizations that have simple facets. Namely, it is the trihedralization that fullfills the following property, stated here in terms of the 3D embedding of the weighted straight skeleton:

Property 5.1. *The local z-minima of the 3D polygons supporting the facets of the 3D embedding of the weighted straight skeleton of a 2D polygon are contained in the edges of the input polygon.*

This property, proved in [AA96], characterizes the weighted straight skeleton among the possible abstract triangulations. The weighted straight skeleton reduction, when applicable, provides a trihedralization that fulfills the z-minima property, where the z coordinate measures the distance to the section plane used in the reduction. The reduction thus provides a trihedralization that fulfils a stronger geometric requirement than only self-intersection free facets.

5.6.5 Conclusion

By construction, the (weighted) straight skeleton facets are connected, yielding a trihedralization algorithm with the required connected facets when the trihedralization problem is reducible to a weighted straight skeleton problem. However this reduction is not always possible. When the over-constrained vertex is not extremal, the property 5.1 cannot be generalized easily. It might be possible to generalize this property by considering that z-minima are points that are locally furthest points from a point at infinity along the vertical direction, and then moving this infinite point to a specific finite location p. This may allow to define a generalization of the straight skeleton problem to spherical polygons, which are a generalization of planar polygons to polygons on the surface of the sphere, with a sphere centered at p.

Nevertheless, even if such a generalization existed, reducing to such a generalization would mean restricting the set of the possible abstract triangulations to the ones that verify this generalized property. If this generalized property implies that the 3D polyhedral surface is self-intersection free, as in the planar case, then the reduction would be valid, and may provide a simple algorithm to compute the trihedralization if it exists. It may further help to understand the trihedralizability condition. However, restricting the possible abstract triangulations using a sufficient, but not necessary geometric condition, would lead to cases where the reduction fails to find a trihedralization of a trihedralizable vertex.

5.7 Discussion

Using this intersection predicate formulation, the trihedralization problem may now be analyzed for unicity and existence of a solution.

5.7.1 Unicity

Similar to the triangulation problem, there are cases were multiple trihedralizations satisfy the simplicity requirement. Figure 5.25 shows that an ambiguity exists even in the simplest case of making a vertex of valence 4 trihedral after updating the planes. In such ambiguous cases, a trihedralization algorithm may either output the first valid trihedralization it finds, or list all the valid ones, or provide the best trihedralization according to some scoring function.

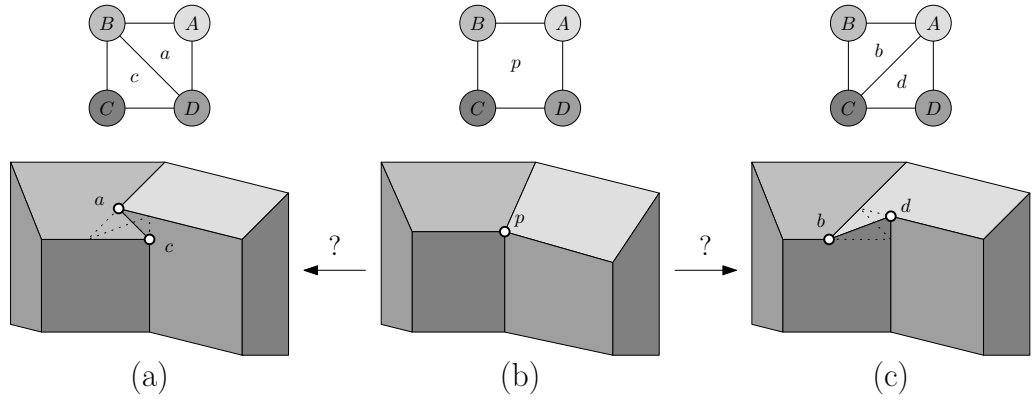


Figure 5.25: Trihedralization ambiguity. (b) A saddle vertex p of valence 4 and (a,c) its 2 trihedralizations. The symmetric difference of the polyhedra (a) and (c) is the tetrahedron defined by the 4 supporting planes $ABCD$, and is illustrated using dotted lines in (a) and (c).

5.7.2 Existence

Section 5.6 will show that a trihedralization is guaranteed to exist in a special case, where the trihedralization problem may be reduced to the so-called weighted straight skeleton problem.

Section 5.2 has introduced a trihedralization approach using winding numbers that is applicable as soon as the plane geometries are not highly degenerate. However it relaxes the search space by allowing interior facets, since it does not search the resulting topology as an abstract triangulation.

However, a valid self-intersection free trihedralization is not always guaranteed to exist. For instance, figure 5.26 illustrates such a case in the simplest case of a point adjacent to only 4 facets. One of the 2 possible abstract triangulations (b) has a self-intersecting facet B, and the other one (c,d) is invalid, having a point b at infinity.

We only provide a proof of existence in the following case :

Theorem 5.9. *A non-degenerate local vertex trihedralization problem of a vertex of valence 4 has a self-intersection free trihedralization.*

Proof. We can notice that the query points are defined by the intersection of 3 planes such that the above predicate is the sign of the ratio of 2 determinants:

$$Above(P_0 \cap P_1 \cap P_2, P_3) \quad = \quad sign\left(\frac{|\,\vec{P}_0\ \vec{P}_1\ \vec{P}_2\ \vec{P}_3\,|}{|\,\vec{n}_0\ \vec{n}_1\ \vec{n}_2\,|}\right)$$

The intersection predicate may then be rephrased as:

$$\begin{aligned} Intersect(P, P^k_{i-1}, P^k_i, P^k_{i+1}, P^l_{j-1}, P^l_j, P^l_{j+1}) \quad = \quad & \frac{|\,\vec{P}^l_j\ \vec{P}^k_i\ \vec{P}\ \vec{P}^k_{i-1}\,|\,|\,\vec{P}^k_{i+1}\ \vec{P}^k_i\ \vec{P}\ \vec{P}^k_{i-1}\,|}{|\,\vec{n}^l_j\ \vec{n}^k_i\ \vec{n}\,|\,|\,\vec{n}^k_{i+1}\ \vec{n}^k_i\ \vec{n}\,|} \geq 0 \\ \& \quad & \frac{|\,\vec{P}^l_j\ \vec{P}^k_i\ \vec{P}\ \vec{P}^k_{i+1}\,|\,|\,\vec{P}^k_{i-1}\ \vec{P}^k_i\ \vec{P}\ \vec{P}^k_{i+1}\,|}{|\,\vec{n}^l_j\ \vec{n}^k_i\ \vec{n}\,|\,|\,\vec{n}^k_{i-1}\ \vec{n}^k_i\ \vec{n}\,|} \geq 0 \\ \& \quad & \frac{|\,\vec{P}^k_i\ \vec{P}^l_j\ \vec{P}\ \vec{P}^l_{j-1}\,|\,|\,\vec{P}^l_{j+1}\ \vec{P}^l_j\ \vec{P}\ \vec{P}^l_{j-1}\,|}{|\,\vec{n}^k_i\ \vec{n}^l_j\ \vec{n}\,|\,|\,\vec{n}^l_{j+1}\ \vec{n}^l_j\ \vec{n}\,|} \geq 0 \\ \& \quad & \frac{|\,\vec{P}^k_i\ \vec{P}^l_j\ \vec{P}\ \vec{P}^l_{j+1}\,|\,|\,\vec{P}^l_{j-1}\ \vec{P}^l_j\ \vec{P}\ \vec{P}^l_{j+1}\,|}{|\,\vec{n}^k_i\ \vec{n}^l_j\ \vec{n}\,|\,|\,\vec{n}^l_{j-1}\ \vec{n}^l_j\ \vec{n}\,|} \geq 0 \end{aligned}$$

There are 2 abstract triangulations of an abstract quadrangle, depending on which diagonal is chosen. The new vertices that are not adjacent to the diagonal are only adjacent to 3 dual triangles of the extended abstract triangulation, that is including the 2 dual triangles corresponding to points at infinity. Thus their facets are

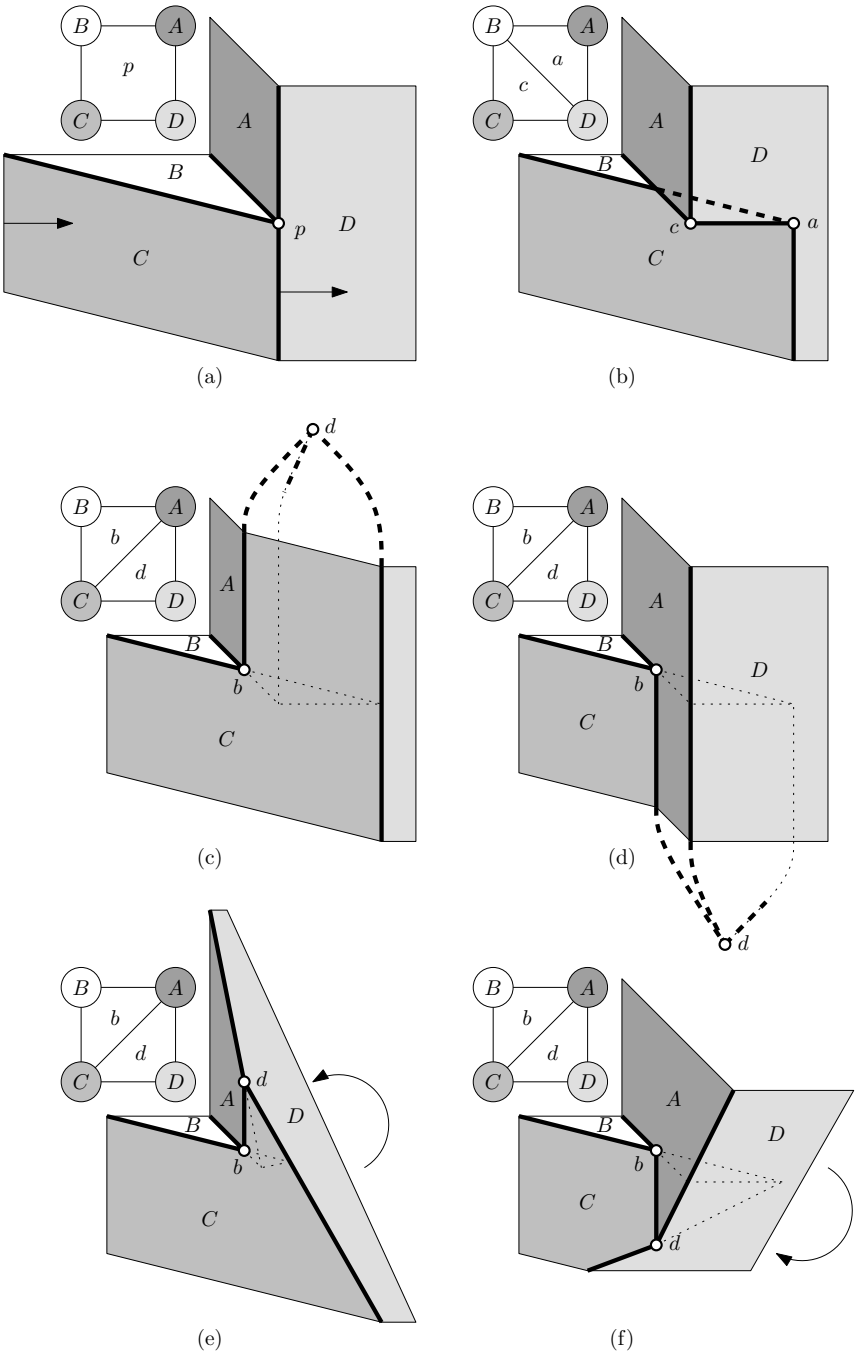
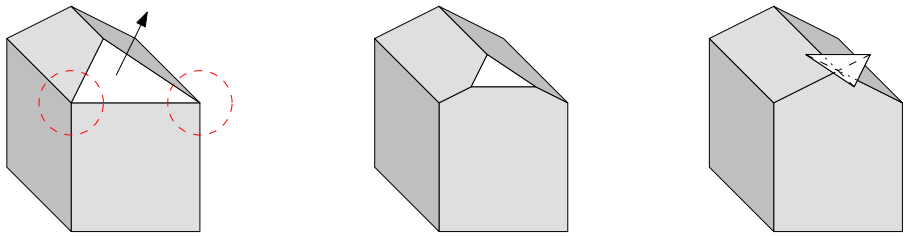


Figure 5.26: Ill-posed trihedralization problem with no valid solution. (a) A polyhedral surface with a vertex p adjacent to 4 facets supported by planes $ABCD$. In (b), (c) and (d), C is translated so that the lines $A \cap B$ and $A \cap D$ remain parallel. The abstract triangulation used in (b) makes the facet supported by B self-intersecting. (c) and (d) are the two possible geometric embeddings of the surface resulting from the other possible abstract triangulation. Their point $d = A \cap B \cap D$ is a point at infinity, making the trihedralization invalid. (c) considers it at infinity in the top direction, and (d) in the bottom one. By rotating the plane D so that the degeneracy is resolved, the trihedralization problem now has a solution. Its actual geometric embedding (e) or (f) depends on the rotation, so that (c) and (d) may be seen as a degenerate case of respectively (e) and (f).



(a) (b) (c)

Figure 5.27: Trihedralizations may not be local. (a) An initial polyhedron with 2 overconstrained vertices (circled). (b) A small upward translation of the triangular face requires solving 2 independent trihedralization problems. (c) Solving independently the trihedralization problems of a larger move (*i.e.* past the intersection of the front facet with the 2 other top facets) lead to self intersecting facets.

unconditionnaly simple. Let us assume that no self-intersection free trihedralization of such a dual quadrangle exists. This means that, for each diagonal, one of its dual vertices refers to a self-intersecting facet. Since the abstract polygon is a quadrangle, they have to be consecutive. Without lack of generality, let us consider that these dual vertices f_1 and f_2 are supported by the planes P_1 and P_2, where $(P_0, P_{\frac{1}{2}}, P_1, P_{\frac{3}{2}}, P_2, P_{\frac{5}{2}}, P_3, P_{\frac{7}{2}})$ is the circular list of supporting planes of the trihedralization problem. Now since, the abstract polygon is a quadrangle, a single pair of segment is tested for intersection, assuming the absence of degeneracies (sec. 5.4.3): the segments supported by $P_0 \cap P_1$ and $P_1 \cap P_2$ when f_1 is on the diagonal, and the ones supported by $P_1 \cap P_2$ and $P_2 \cap P_3$ when f_2 is. Since it fails for both trihedralization on respectively facets f_1 and f_2, each segment intersection test faces 4 non-negative expression, 3 out of the 8 inequalities are of particular interest:

$$\frac{|\,\vec{P}_0\ \vec{P}_1\ \vec{P}_2\ \vec{P}_3\,|\,\left|\,\vec{P}_1\ \vec{P}_{\frac{3}{2}}\ \vec{P}_2\ \vec{P}_3\,\right|}{|\,\vec{n}_0\ \vec{n}_1\ \vec{n}_2\,|\,|\,\vec{n}_1\ \vec{n}_{\frac{3}{2}}\ \vec{n}_2\,|} \geq 0 \quad , \quad \frac{|\,\vec{P}_1\ \vec{P}_2\ \vec{P}_3\ \vec{P}_0\,|\,\left|\,\vec{P}_1\ \vec{P}_{\frac{3}{2}}\ \vec{P}_2\ \vec{P}_0\,\right|}{|\,\vec{n}_1\ \vec{n}_2\ \vec{n}_3\,|\,|\,\vec{n}_1\ \vec{n}_{\frac{3}{2}}\ \vec{n}_2\,|} \geq 0 \quad \text{and} \quad \frac{\left|\,\vec{P}_0\ \vec{P}_1\ \vec{P}_2\ \vec{P}_{\frac{3}{2}}\,\right|\,\left|\,\vec{P}_1\ \vec{P}_2\ \vec{P}_3\ \vec{P}_{\frac{3}{2}}\,\right|}{|\,\vec{n}_0\ \vec{n}_1\ \vec{n}_2\,|\,|\,\vec{n}_1\ \vec{n}_2\ \vec{n}_3\,|} \geq 0$$

where the first comes from f_1, the second from f_2 and the third from both. By multiplying the first two inequalities,

$$\frac{-\,|\,\vec{P}_0\ \vec{P}_1\ \vec{P}_2\ \vec{P}_3\,|^2\,\left|\,\vec{P}_1\ \vec{P}_{\frac{3}{2}}\ \vec{P}_2\ \vec{P}_3\,\right|\,\left|\,\vec{P}_1\ \vec{P}_{\frac{3}{2}}\ \vec{P}_2\ \vec{P}_0\,\right|}{|\,\vec{n}_0\ \vec{n}_1\ \vec{n}_2\,|\,|\,\vec{n}_1\ \vec{n}_2\ \vec{n}_3\,|\,\left|\,\vec{n}_1\ \vec{n}_{\frac{3}{2}}\ \vec{n}_2\,\right|^2} \geq 0 \quad \text{, hence} \quad \begin{cases} |\,\vec{P}_0\ \vec{P}_1\ \vec{P}_2\ \vec{P}_3\,| = 0 \quad \text{or} \\ \dfrac{\left|\,\vec{P}_0\ \vec{P}_1\ \vec{P}_2\ \vec{P}_{\frac{3}{2}}\,\right|\,\left|\,\vec{P}_1\ \vec{P}_2\ \vec{P}_3\ \vec{P}_{\frac{3}{2}}\,\right|}{|\,\vec{n}_0\ \vec{n}_1\ \vec{n}_2\,|\,|\,\vec{n}_1\ \vec{n}_2\ \vec{n}_3\,|} \leq 0 \end{cases}$$

and thus, we prove that either the problem is degenerate or the third inequality is an equality, yielding:

$$|\,\vec{P}_0\ \vec{P}_1\ \vec{P}_2\ \vec{P}_3\,|\,\left|\,\vec{P}_0\ \vec{P}_1\ \vec{P}_2\ \vec{P}_{\frac{3}{2}}\,\right|\,\left|\,\vec{P}_1\ \vec{P}_2\ \vec{P}_3\ \vec{P}_{\frac{3}{2}}\,\right| = 0$$

This result invalidates the assumption that both abstract triangulations are non-degenerate and valid. Since the trihedralization problem is not degenerate, either a plane supports multiple facets or a trivial edge exists. If a plane supports multiple facets, then the over-constrained vertex is surrounded by 3 or less planes. The arrangement of these planes is simple, given that the problem is well-posed. Thus this vertex can be made trihedral with either abstract triangulations, or discarded if it is underconstrained. If a trivial edge exists, its 2 vertices are collocated, yielding a self-intersection free finite trihedralization, which is the final contradiction.

□

5.8 Conclusion

Within the targeted application of roof fitting, most over-constrained vertices have a valence of 4. A small fraction of the vertices is adjacent to 5 facets, mostly due to adjacent facets supported by the same plane, which is the case at the junction of a T-shaped roof. Higher valence vertices are typically extremely rare. Therefore the theoretical asymptotic complexity of the implemented approach was not a crucial concern. We chose to implement the ear-cutting approach due to its simplicity. And it proved to perform sufficiently well in practice. Section 6.4 will give some trihedralization timings in the context of topology-aware building model fitting.

Figure 5.27.b shows a simple example of trihedralizations resulting from a perturbation of the planes supporting the polyhedral facets. The trihedralization problem in figure 5.27.c is however

more global, due to a larger deviation from the input polyhedron with self-intersection free facets. In case (c), the 2 trihedralization problems may not be solved locally. As a sidenote, this need for a more global processing of the topology has already been illustrated in figure 1.3. Therefore, a more global approach has to be used to prevent facet self-intersections. Here a single facet support is updated, but in the general case, all facet supports are updated somewhat arbitrarily. Hence simple rule-based patching approaches are likely to fail, and a more global approach is required. One possibility would have been to use the winding number or the more flexible arrangement coloring approach. However, we discussed in sections 5.2 and 5.3.4 that these approaches may not be straightforwardly used to minimize the complexity of the resulting polyhedron.

We introduced this novel trihedralization problem, which has not been thoroughly investigated yet. We showed that the existing winding number based approach may be adapted to solve this problem but is however unable to guarantee the absence of "unnecessary" local features (sec. 5.2). Then, an extension of the winding number approach to a full plane arrangement coloring was discussed to meet the restricted output topology criterion (sec. 5.3). Given then vague definition of an "unnecessary" topological feature, we restrained ourself to trihedralize overconstrained vertices independently, minimizing the topological complexity of each splitted vertex by searching a topology dual to a triangulation (sec. 5.4). First, a simple implementation based on the ear-cutting paradigm was detailed (sec. 5.5). We then exhibited some reduction results with the weighted straight skeleton problem (sec. 5.6).

This chapter has shown how to relax the geometric constraints induced by the over-constrained vertices in order to fit a building model polyhedron to a DSM, without any topology-induced constraints. We saw, however, that if the building is not a topological simplification of the desired building, our simplifying locality assumption no longer holds. The trihedralization problem may then not be solved independently and locally. The next chapter introduces a kinetic framework that is able to lift this assumption, while guaranteeing the self-intersection free facets.

Chapter 6

A Kinetic Framework Guaranteeing Simple Facets

Contents

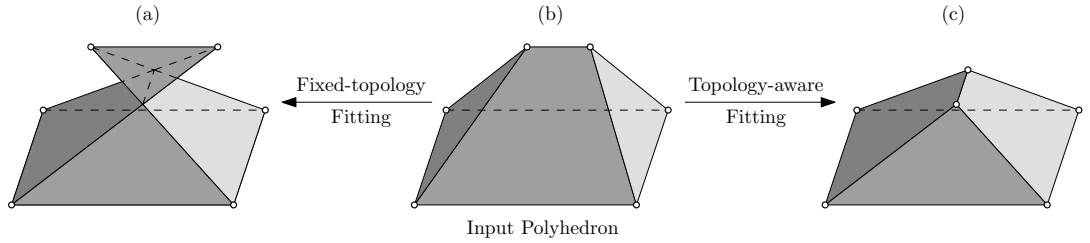


Figure 6.1: Updating supporting planes of a trihedral polyhedron may require topological updates to prevent self-intersecting facets. When the input polyhedron (b) is fitted to a DSM, with a fixed topology, the resulting polyhedron may self-intersect (a). A topological modification is required here to output the self-intersection free polyhedron (c).

6.1 Introduction

While fitting a building to a DSM, it occured that morphing the initial building polyhedral model into a new polyhedral model by updating its plane equations and recomputing the vertex positions from the plane equation of its adjacent facets, does not always yield a simple polyhedron. Even if the modified polyhedron is well defined (*i.e.* the intersection of the planes supporting the adjacent facets of a vertex is a single point), inversions may occur that make the polyhedron not simple, as illustrated in figure 6.1.

By solving the following problem, this chapter guarantees that a polyhedron undergoing a modification of its supporting plane equations, as in chapter 4, remains simple (*i.e.* with self-intersection free facets).

Problem Statement

Given a simple polyhedron, and a new plane equation for each of its facets, how to compute a simple polyhedron, the supporting plane geometry of which is the specified plane equations, and the topology of which is as close as possible to the initial topology?

Using the projective geometry notations introduced in section 4.2, the initial plane equation of the facet i is denoted $(\vec{N}_{i0})$ and is going to be updated to $(\vec{N}_{i1})$. The obvious application is to be able to deal with the artifacts occurring with the algorithm developed in chapter 4. Other interesting applications, like the computation of a 3D weighted straight skeleton, or automatic polyhedron LOD are presented in section 6.6.3.

6.1.1 Reduction to Plane Arrangement Coloring

This problem can be restated as a 0-1 coloring (see section 4.3.3) of the cells of the 3D plane arrangement (see section 4.3.1) of the target planes. The facets of the arrangement that are at the boundary of a 0-colored cell and a 1-colored cell of the 3D plane arrangement of the target planes, form the boundaries of a set of polyhedra volumes whose facets are supported by the target planes. A coloring maps to a single polyhedron if and only if the set of 1 colored cells is connected.

The number of finite cells in an arrangement of n planes is $O(n^3)$. This means that the answer to our problem is a polyhedron out of the $O(2^{n^3})$ sets of polyhedra produced by 0-1 colorings of the finite cells of the arrangement of the target planes $(\vec{N}_{i1})$.

A straightforward 0-1 coloring is obtained by volumetric thresholding: for each cell C, the ratio

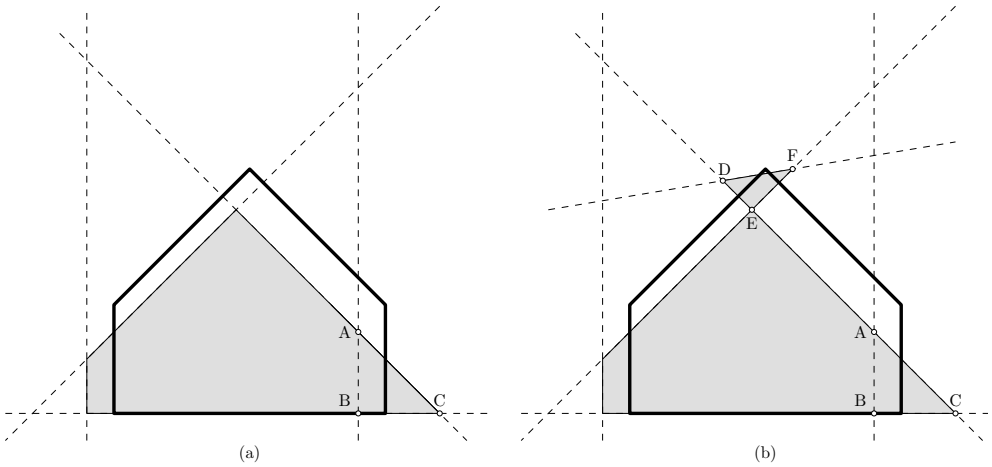


Figure 6.2: Problems with non desirable Volumetric thresholdings. 0-1 coloring by volumetric thresholding (cross-section): the initial polyhedron is in bold, dashed lines illustrate the estimated plane arrangement and the volumetric thresholding coloring shows 1 colored cells in gray (threshold=0.5). (a) shows a 5 plane arrangement where the right-handside façade has been lost (triangle ABC). (b) a 6^{th} plane creates an inverted triangle DEF that floats above the roof.

of volume of the intersection $P_0 \cap C$ of the initial polyhedron P_0 in the cell C to the volume of the cell is computed. Then a predefined threshold could assign a value of 0 or 1 to the cell. Figure 6.2 shows that the initial topology is not directly taken into account, leading to counter-intuitive results. This algorithm is likely to produce overly complex shapes due to the oversegmentation of the whole volume $\mathbb{R}^3$ given by the plane arrangement. Furthermore, the manifoldness can be enforced by splitting the tangent polyhedra at non-manifold vertices or edges (such as E in figure 6.2), but the result is then a set of polyhedra and not a single polyhedron.

Casting our problem as an arrangement coloring has the disadvantages of its advantages: the topology can vary arbitrarily, but is uneasy to control. To keep the control of the topology, we introduce a surface-based approach, that evolves the initial polyhedron to match the target dual geometry.

6.1.2 Proposed Approach

One idea is to try to tackle one by one, iteratively, the artifacts resulting from updating the geometry without modifying the topology. This typically just works when artifacts are isolated from each other and the fixed topology is obvious. However, this approach is error-prone. A topology with reasonably few modifications that will produce a simple polyhedron may not be obvious. Many correlated artifacts may have been introduced by the geometry update.

To further develop this approach, one may want to produce the polyhedron with the target dual geometry iteratively, by constructing polyhedra with intermediate dual geometries. The purpose would be to fix the artifacts while they are isolated, and thus easily detected and handled. However, choosing blindly the number of intermediate iterations is tricky in practice. Choosing too few iterations may lead to many interdependant artifacts, that would not thus be easily fixable, as in the single iteration approach. When choosing too many iterations, the algorithm will have poor computing times by repetitively computing polyhedra with the very same topologies and little geometric variations.

How to keep a polyhedron simple while the planes supporting its facets are evolving continuously? Maintaining geometric properties of a combinatorial data structure as the geometry is evolving is exactly the purpose of the kinetic data structures.

6.2 Kinetic Data Structures

6.2.1 Introduction

Kinetic data structures were first introduced in [BGH97]. The idea stems from the observation that most, if not all, computational geometry structures are built using predicates - functions on quantities defining the geometric input (*e.g.* point coordinates), which return a discrete set of values. Many predicates reduce to determining the sign of a polynomial on the defining parameters of the primitive objects. For example, to test whether a point lies above or below a plane we compute the dot product of the point with the normal of the plane and subtract the plane's offset along the normal. If the result is positive, the point is above the plane, zero on the plane, negative below. The validity of many combinatorial structures built on top of geometric primitives can be verified by checking a finite number of predicates of the geometric primitives. These predicates, which collectively certify the correctness of the structure, are called certificates. For a Delaunay triangulation in three dimensions, for example, the certificates are one InCircle test per facet of the triangulation, plus a point plane orientation test for each facet or edge of the convex hull.

The kinetic data structures approach is built on top of this view of computational geometry. Let the geometric primitives move by replacing each of their defining quantities with a function of time (generally a polynomial). As time advances, the primitives trace out paths in space called trajectories. The values of the polynomial functions used to evaluate the predicates now also become functions of time. We call these functions certificate functions. Typically, a geometric structure is valid when all predicates have a specific non-zero sign. In the kinetic setting, as long as the certificate functions maintain the correct sign as time varies, the corresponding predicates do not change values, and the original data structure remains correct. However, if one of the certificate functions changes sign, the original structure must be updated, as well as the set of certificate functions that verify it. We call such occurrences **events**.

Maintaining a kinetic data structure is then a matter of determining which certificate function changes sign next, i.e. determining which certificate function has the first real root that is greater than the current time, and then updating the structure and the set of certificate functions. In addition, the trajectories of primitives are allowed to change at any time, although C0-continuity of the trajectories must be maintained. When a trajectory update occurs for a geometric primitive, all certificates involving that primitive must be updated. We call the collection of kinetic data structures, primitives, event queue and other support structures a simulation.

Sweep line algorithms for computing arrangements in d dimensions easily map on to kinetic data structures by taking one of the coordinates of the ambient space as the time variable. The kinetic data structure then maintains the arrangement of a set of objects defined by the intersection of a hyperplane of dimension d-1 with the objects whose arrangement is being computed.

[CGAL] is a computational geometry library that provides a framework [Rus09] for implementing kinetic data structure algorithms. This introduction to kinetic data structures (section 6.2.1) has been adapted from the CGAL documentation of this framework [Rus09]. It is best illustrated by one of the simplest kinetic data structures: a sorted list of moving 1D points.

6.2.2 Kinetic Algorithm Examples

Maintaining a Sorted List of Objects One of the most simple example of kinetic algorithms is the maintenance of a sorted list of objects. Each object is associated with a real number key which defines a total order on the list of objects. The objective is to maintain the sorted list of objects as the keys are allowed to change continuously. This can be done, from scratch, for each query, by first computing the key function value at the queried time and then sorting the objects according to those key values. However, since the key variations are continuous, as long as no object keys become equal, the order is not modified and the sorted list has not to be updated. However,

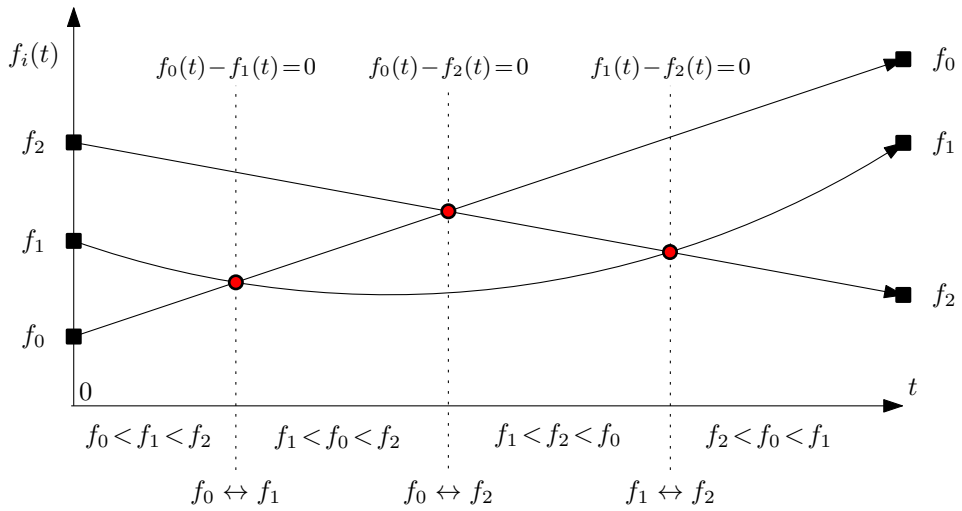


Figure 6.3: Kinetic sort. The maintenance of the ordering of 3 objects moving in 1D according to the functions $f_0(t)$, $f_1(t)$ and $f_2(t)$, as t increases.

for each inversion of the ordering of two objects, their positions in the list have to be swapped, to recover a correct ordering of the list. Even if the object keys are changing continuously, there is only a discrete set of times, called events, where an action is required to maintain the geometric property that the list is sorted. This action does not modify the geometry of the problem (the key values as a function of time) but only the topology of the maintained combinatorial structure (the ordering of the list).

This algorithm maintains a proof that the list is sorted. When it detects that the proof is no longer valid, the list is no longer sorted. The algorithm has to update the list order and update the proof so that it applies to the new ordering. An obvious way to proof that a list of n objects is sorted is to check whether the $n-1$ pairs of consecutive objects are in the right order. This check involves evaluating the sign of the difference of the object key functions. When all the differences are strictly positive, the list, viewed as a list of pairs of consecutive objects, constitutes a valid proof that it is itself sorted. When two objects are crossing, they must have been consecutive in the sorted list, since the trajectories are assumed to be continuous, and this corresponds to a root of the function that computes the difference of the key values, where a sign change occurs. At a time immediately after this root, the function will be strictly negative and the proof will no longer hold. To reconstruct a valid proof, only a minimal effort is required. After swapping the crossing objects, one has to inspect up to three key difference functions: one between the two crossing objects, one between the smaller crossing object and its new neighbor in the list if it is not the first, and one between the greater crossing object and its new neighbor in the list if it is not the last.

The straight skeleton, introduced in section 5.6, is a more complex combinatorial structure that can be computed using a kinetic approach. Even its definition as a shrinking process, is indeed of a kinetic nature. Its 3 event types detect the self-intersection times of the shrinking polygons. These events are processed by creating a new straight skeleton node and scheduling new future events if the shrinking polygon has not vanished.

6.2.3 Definitions

The key definitions of the kinetic algorithms may now be introduced. Each concept is illustrated by the corresponding Kinetic Sorted List object, using the notation [*KSL: corresponding object*] :

Primitive The basic geometric type of the objects handled in the problem. It can be 2D, 3D or higher dimension points, planes, circles... The geometry of a primitive is provided by a vector of coordinates. [*KSL: A 1D point of index i (single real number key).*]

Trajectory As primitives are moving, their coordinate vectors are a function of time. A trajectory is the continuous set of the successive positions of the coordinate vector of the primitive during the time evolution. [*KSL: the i^{th} 1D point moves according to a continuous function $f_i(t)$.*]

Kinetic/Static Something kinetic involves moving primitives. This is in contrast with **static** concepts that only handle non-moving primitives. Static algorithms are the vast majority of computational geometry algorithms: Delaunay triangulations, convex hulls... [*KSL: The initial list sorting is statically performed on the instantaneous initial object keys $f_i(0)$, whereas the subsequent maintenance of the ordering is said to be kinetic.*]

Instantaneous Having to do with the geometry of the problem at a given time only, without any knowledge of the primitive trajectories, except the current position. A kinetic algorithm may always be statically audited by verifying whether the maintained combinatorial data structure verifies the desired properties using only the instantaneous coordinates.

Combinatorial structure A topological structure that expresses relationships between the primitives. Being only a topological structure, it does not rely on the actual coordinates of the primitives. This is typically the structure that will be maintained by a kinetic algorithm, like a triangulation, the topological description of a polyhedron or an ordered list. [*KSL: A list of object indices $(i_1 \dots i_n)$*]

Kinetic Data Structure (KDS) The combinatorial data structure maintained by the kinetic algorithm so that it fulfills a given set of combinatorial and/or geometric properties: A Delaunay triangulation as points move, a convex hull. By extension, the algorithm that maintain a KDS is also referred to as a KDS. [*KSL: A list of object indices $(i_1 \dots i_n)$, sorted by increasing keys.*]

Predicate This well-known computational geometry term denotes a function which takes the instantaneous coordinates of several primitives as input and produces one of a discrete set of outputs (definition 14). This includes the orientation test of a triangle(clockwise, degenerate or counterclockwise?), the InCircle test of the Delaunay algorithm(in, on or out?)... [*KSL: the ordering test : $a < b$, $a = b$ or $a > b$?*]

Certificate function A certificate function is a function of time. It is the kinetic equivalent of the predicate that evaluates the instantaneous sign of the same function. The convention is to design the function so that they are strictly positive, (resp. null or strictly negative) when the tested property is as true (resp. degenerate or false). [*KSL: A difference $(f_{i_l}(t) - f_{i_k}(t))$ where $k < l$ and $(i_1 \dots i_n)$ is the current sorted list.*]

Certificate A set of certificate functions that constitutes a proof that the kinetic data structure verifies the desired properties, when all the certificate functions are non-negative. For instance, a certificate that a polygon is simple is given by the orientation certificate functions of each triangle of a triangulation of the polygon. [*KSL: the $(n-1)$ certificate functions $(f_{i_k+1}(t) - f_{i_k}(t))_{k=1\dots n-1}$, where $(i_1 \dots i_n)$ is the current sorted list.*]

Event A root of a certificate where a sign change occurs, the certificate is no longer valid and thus needs to be updated. However, the combinatorial structure may or may not have to be updated. If an event make the certificate fails to prove the maintained property, but that the maintained property still holds, this event is called **internal**. If the maintained property no longer holds, the maintained combinatorial data structure also has to be updated, and the event is called **external**. [*KSL: A root of a certificate function $f_i(t) - f_j(t)$. Events are times where two or more objects have the same key. All KSL events are external.*]

Event Queue To be able to know what is the next event, and process them in increasing order, all events are stored in a priority queue, ordered by the event time.

One may wonder, why the term kinetic has been chosen over dynamic, which is a more natural antonym of static. The reason is that dynamic data structures were designed before the development of kinetic algorithms. The purpose of a dynamic algorithm is to maintain a combinatorial

data structure as primitives are added or removed, rather than when their coordinates are evolving continuously.

6.2.4 Arbitrary Precision Arithmetics

Computational geometry algorithms are sensitive to the precision of the computations. For instance, if they are performed using floating point arithmetics, within a computational geometry algorithm that neglects the rounding errors, it is predictable that it will compute erroneous results in degenerate or quasi degenerate setups. This is caused by the degeneracies: a predicate that involves a degeneracy considers the sign of an expression that should evaluate to zero with arbitrary precision arithmetics. However, using floating point computations, such a predicate is only considering the sign of the rounding errors. If the sign of these rounding errors were consistent, it would end up modeling a small perturbation of the input that disambiguates the degeneracies. However, those rounding errors have no consistency guarantees. For instance, when testing whether three almost aligned points A, B and C are describing a clockwise or counterclockwise triangle, it may occur that the triangle ABC is clockwise but that the triangle BCA is counterclockwise. Thus, the control flow of the algorithm is faced with inconsistent predicate results. It is even likely that, without special care, the algorithm may crash or never terminate, if it expects only consistent results, such as the mutual exclusion of the left turn and right turn predicates, or a consistent orientation of the triangles ABC, BCA and CAB.

The most robust approach to properly overcome this situation is to use arbitrary precision arithmetics. This way, the algorithms will only face consistent predicate results, at the cost of increased computing times and space requirements. To speed up predicate evaluations, which are basically a determination of the sign of an expression, interval arithmetics is used. Every arbitrary precision quantity is bounded within an interval. Tests are first performed on the interval bounds, which is less expensive. If the test result cannot be determined based solely on the bounding interval, the interval is recursively refined, or the expensive arbitrary precision operation is performed.

Sturm sequences The precision problem is exaggerated when the considered quantity is no more a rational function of the input coordinates but the root of a polynom of the input coordinates. The proposed framework has to handle robustly roots of polynoms that have degrees 3, 4 and 8 (see section 6.3.7). The [CGAL] library uses the well established Sturm sequences and square-free (*i.e.* without square polynomial factors) factorizations of univariate polynomials to achieve robust comparisons of roots, in a way similar to [HM90]. The big picture is that Sturm sequences provide a way to count the number of real roots of a square-free polynomial, disregarding their multiplicities, enclosed in an interval, that is possibly unbounded. Roots are isolated so that they can be represented exactly as the only root of a given polynom p, within a given isolating interval $[a, b]$. Then, the comparison of two roots $(p_1, [a_1, b_1])$ and $(p_2, [a_2, b_2])$ is carried out by refining their isolating intervals $[a_i, b_i]$ until they are disjoint or by testing whether they are equal. Root equalities are tested by first computing a common isolating interval, and then the Sturm sequence of p_1 and $p'_1 p_2$, where p'_1 is the derative of p_1. This allows to evaluate the sign of p_2 at a root of p_1.

6.3 Kinetic Polyhedron with Simple Facets

The kinetic framework that has been developed in [BGH97, BCG+99] appears to be a good solution for our problem. Applied to the context of this chapter, the idea is to move continuously the geometry of the planes supporting the polyhedron facets from the initial geometry at time $t = 0$ to the target geometry at $t = 1$. In a kinetic framework, a global geometric property is

maintained by constructing and maintaining a kinetic data structure (KDS) throughout the time evolution. The maintained data structure is here the topology of the polyhedron.

The purpose of this KDS is to maintain the geometric property that the facets of a polyhedron remain simple, while providing a certificate (the proof of this property). To prove that all the facets are simple, the KDS provides for each facet of the polyhedron a triangulation of the convex hull of its vertices, constrained by its edges. A certificate is the list of the certificate functions that together prove the global property of simplicity. If all the triangles of the triangulation constrained by the facet have a consistent orientation and the triangulated domain remains convex, then this facet, which is, in the general case, a 2D polygon with holes, is simple. As the supporting planes are oriented, the consistency of the orientation of the triangles is checked by verifying the consistency of each triangle with the orientation of its supporting plane (Sec. 6.3.7). Likewise, the maintenance of the convex hull is performed by examining the orientation of the triangles formed by triplets of successive triplets on the maintained convex hull. Thus the KDS is based solely on the orientation predicate.

These orientation certificates rely on rational certificate functions in terms of the interpolation time t, and of the parameters of up to seven planes neighboring the triangle, rather than functions that are typically polynomial in other kinetic data structures. Roots and also poles of these certificate functions are called events. Since the trajectories of the planes are continuous, so are the certificate functions, and thus their sign remains constant between events, be it roots or poles. During the simulation, the time does not evolve continuously: the KDS computes the events of each of its certificate functions and orders them in a priority queue. The interpolation time t is then iteratively advanced to the closest event in the future. At that time, a certificate function fails - becomes negative - and the certificate has to be updated. The polyhedron itself may not have to be updated, but the certificate that certifies that the polyhedron is simple is no longer valid and should be updated. However, events are likely to invalidate the simplicity of the polyhedron. Then the topology of the polyhedron has to undergo minimal changes to reestablish the simplicity of the polyhedron. These updates reestablish a valid certificate, so that time can then be advanced either to the next event of one of the certificate functions or to the evolution ending time $t = 1$.

6.3.1 Polyhedron Interpolation

To handle the polyhedron kinetically, the intermediate geometry of a polyhedron between two polyhedra that share the same topology but have different geometries has to be defined. Given that the initial and final geometries are given by the dual geometries (the homogeneous coordinates $\vec{N} = [\vec{n} : d] = [a : b : c : d]$), we propose to linearly interpolate the dual geometry. Let us first get some intuition with the homogeneous interpolation of the primal geometry.

Point Interpolation When interpolating point coordinates (the primal geometry), the linear combination $(1-t)\vec{P_0} + t\vec{P_1}$ with $t \in [0,1]$ spans the segment from P_0 to P_1, provided that the homogeneous coordinates of the points, denoted w_0 and w_1, have the same sign. If their signs are different, the interpolation traces out the line passing through P_0 and P_1, except for the interior of the segment [P_0,P_1]. The Cartesian coordinates $\frac{\vec{p}}{w}$ are rational functions of time of degree 1, with a pole when the signs of w_0 and w_1 differ. The interpolated point goes from P_0, away from P_1, it reaches infinity at the pole. Then, it comes back from the point at infinity in the other direction along the line, to the point P_1.

Proof. The derivative of the Cartesian trajectory of the interpolated point is computed as follows:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\vec{p}_0 + t.(\vec{p}_1 - \vec{p}_0)}{w_0 + t.(w_1 - w_0)}\right) &= \frac{(\vec{p}_1 - \vec{p}_0).w_0 - \vec{p}_0.(w_1 - w_0)}{(w_0 + t.(w_1 - w_0))^2} \\ &= \frac{w_0.w_1}{(w_0 + t.(w_1 - w_0))^2} \cdot \left(\frac{\vec{p}_1}{w_1} - \frac{\vec{p}_0}{w_0}\right) \end{aligned}$$

Since $\left(\frac{\vec{p}_1}{w_1} - \frac{\vec{p}_0}{w_0}\right)$ is the Cartesian vector from P_0 to P_1, the direction of the trajectory is given by the sign of $w_0.w_1$. One can notice that the trajectory reaches infinity at the pole $t^* = \frac{w_0}{w_0 - w_1}$, and that this pole falls within $[0, 1]$ if and only if $w_0.w_1$ is negative. □

Plane Interpolation Grounding our intuition on the primal geometry interpolation, we propose to interpolate in the same way the dual geometry. Considering two planes N_0 and N_1, the linear combination of their homogeneous coordinates is a possible interpolant between the two planes. As before, the coordinates of plane N_i is denoted $\vec{N}_i = [\vec{n}_i : d_i]$. The linear combination $(1-t)\vec{N}_0 + t\vec{N}_1$ with $t \in [0, 1]$ has a geometric interpretation too: if the planes are not parallel, they intersect at a common line L and the linear combination lets the interpolated plane rotate around the line L from N_0 to N_1. If they are parallel ($\vec{n}_0$ and $\vec{n}_1$ are collinear), the linear combination spans the set of planes that translate from N_0 to N_1. A special case that will be avoided in our framework is to compute the linear combination of two planes that are parallel with opposite orientations, since this is the only case where there is a $t^* \in [0, 1]$ for which the interpolated normal $(1 - t^*)\vec{n}_0 + t^*\vec{n}_1$ is the null vector, which is the plane containing all the points at infinity.

The interpolation scheme is not required to be linear: any rational trajectory is supported by this framework. A possible extension would be to investigate higher order interpolations like Bezier curves or B-Splines. Furthermore, continuous but only piecewise rational trajectories, using, for instance, Bezier spline curves, could be handled by adding events between rational patches to update the certificate functions that depend on the change of the plane equation. For now on, the trajectories are assumed to be linear.

6.3.2 Continuous Evolution

The proposed kinetic framework allows to evolve the (dual) geometry of the problem by continuously moving its supporting planes. The proposed kinetic algorithm guarantees that the volume inside the maintained polyhedron as a function of time is continuous. This seems to rule out all the possible topological changes. However, during the kinetic evolution of the supporting planes, special geometric events, called singularities, occur. At a singularity, the polyhedral representation is no longer regular. In between these events, topological changes are not required to guarantee self-intersection free facets. Furthermore, it would produce a discontinuous evolution of the described polyhedral surface and are thus prohibited. However, at a singularity, multiple representations exist that describe the same polyhedral surface, among others, the current representation that fails immediately to be regular, and the regular one. The proposed approach is then to regularize the polyhedral representation and then to choose, among the immediately irregular representations, a representation that becomes regular immediately after the singularity and yields a polyhedral surface without self-intersecting facets. This approach ensures that the evolution is continuous, that the facets remain self-intersection free, and that the topological changes induced by the representation changes are only performed when necessary.

The entire purpose of this chapter is two fold:

1. Detect the singularity events efficiently.
2. Update the representation so that it becomes self-intersection free and regular immediately after the singularity event.

Within our context of polyhedra described by their supporting planes and the topological relations between the facets supported by these planes, these representation changes involve discarding plane geometries that no longer support any plane (*e.g.* after a facet collapses), and updating the topology while representing the same surface. Consequently, the final topology output by a kinetic evolution is as close as possible to the initial topology in the sense that topology updates are performed lazily (*i.e.* only as required).

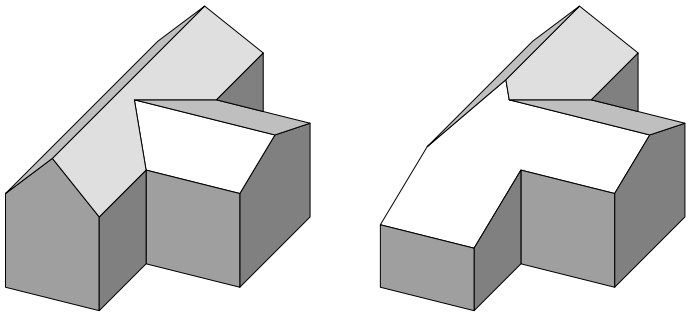


Figure 6.4: The facets of those two simple polyhedra are supported by the same oriented planes.

6.3.3 Non-canonical Data Structure

Many problems have been successfully kinetized: Delaunay triangulation, closest pair of objects, convex hull, minimum spanning tree, range trees... See [Gui98] or more recently [GKR04] for surveys on kinetic algorithms. Most of these kinetic algorithms maintain a canonical data structure. For instance, the 2D visibility polygon[HP02] is unique by definition. This means that the maintained data structure is canonically, and thus uniquely, defined by the instantaneous configuration of the input primitives. Thus those KDS are maintaining a combinatorial data structure that may be deleted and rebuild from scratch at any time.

Some of them however maintain a non-canonical data structure: they maintain a data structure that is not unique, given an instantaneous configuration, such as a planar triangulation of a 2D point set. However, it may be possible to distinguish a canonical data structure that fulfills the requirements of the algorithm: for instance, the Delaunay triangulation may be used to maintain a triangulation.

In the context of the maintenance of a simple polyhedron undergoing translations or rotations of the supporting planes of its facets, no simple polyhedron seems to be a reasonable canonical candidate. This problem is analogous to the kinetic point set triangulation problem in the sense that multiple simple polyhedra may exist, which are supported by the instantaneous interpolating oriented planes only. Figure 6.4 illustrates that multiple simple polyhedra may share the same dual topology and yet describe different surfaces. The property that the maintained data structure is not canonical provides an additional degree of freedom which is combinatorial by nature. Which of the multiple possible data structures is the desired data structure? This extra (combinatorial) degree of freedom is used to keep the polyhedron as topologically close as possible to the initial simple polyhedron. This is achieved by only performing topological modifications when required in order to keep the polyhedron simple. Thus, the kinetic framework is here used to design an algorithm that is sensitive to the initial topology, rather than the usual computing-cost driven motivation of taking advantage of the temporal locality.

6.3.4 Algorithm Overview

The naive and error prone approach of adding constant or even adaptative time steps, then verifying the simplicity lacks the knowledge of what exactly went wrong between the iterations, whereas this kinetic approach is able to handle well identified events one at a time. We propose the following algorithm:

The priority queue $\mathcal{Q}$ is the backbone of a KDS algorithm. It is able to efficiently output the next element scheduled, denoted $next(\mathcal{Q})$, be it in the future or immediate. It further allows the unregistration of previously scheduled events. Each event element stores a timestamp and the indices that identify a failing triangle that becomes clockwise oriented relative to its supporting oriented plane, immediately after the timestamp. If the queue is empty, meaning that no events

Algorithm 6 *KineticSimplePolyhedron*

Require: A regular input polyhedral representation and the trajectory of each supporting plane

$\mathcal{Q} \leftarrow \emptyset$ { Build an empty event queue }
$t \leftarrow 0$
repeat
{ Section 6.3.5: Ensure vertex locations are well-defined }
Trihedralize all non-trihedral vertices (chapter 5)

{ Section 6.3.6: Update the self-intersection free certificate }
Update the constrained triangulation of the convex hull of each facet

{ Section 6.3.7: Update the event queue to compute the maximal time step}
Schedule future failing events of the new triangles in the event queue $\mathcal{Q}$.

{ Advance the evolution time t to the next event. }
$t \leftarrow \min(1, time(next(\mathcal{Q})))$

{Section 6.3.8: Ensure the polyhedral representation is regular }
Pop all immediate events $\mathcal{Q}_t = \{q \in \mathcal{Q} / time(q) = t\}$ out of $\mathcal{Q}$.
Analyse all immediate events $\mathcal{Q}_t$.
Batch-process the events $\mathcal{Q}_t$.
until $t = 1$

are scheduled to occur between the current time and $t = 1$, then, by setting $next(\emptyset) = +\infty$, the algorithm will immediately terminate.

Let us now comment this algorithm. The regularity requirement on the initial polyhedron ensures that its topology provides a faithfull description of the polyhedral surface, without colocated vertices or other geometric singularities (sec. 5.4.3). Trihedralizing a regular representation of the polyhedron yields well-posed and independent trihedralization problems, since it involves an infinitesimal perturbation of a well-defined vertex of high valence. A vertex trihedralization, given its adjacent plane trajectories, guarantees that the split vertices are well-defined as time evolves, and that there is no immediate self-intersection (sec. 6.3.5). Then the retriangulations (sec. 6.3.6) of the convex hull of the recently modified facets update the self-intersection free facet certificates (sec. 6.3.7). Now that the polyhedral facets are triangulated and the vertices have a well-defined trajectory, time may be advanced directly to the timestamp of the next degeneracy event if one exists or to the end ($t = 1$). Finally, the loop invariant, stating that the maintained polyhedral representation is regular, is restored (sec. 6.3.8).

6.3.5 Vertex Trihedralization

While processing an event, the topological updates are likely to result in under-constrained or over-constrained vertices. Immediately after the current event, the plane equations will be interpolated to the time of the next event. The input polyhedral representation is required to be regular, so that vertices that have a valence of 2 or less have been discarded, as they do not convey any information on the shape of the closed polyhedron.

On the other hand, vertices that have a valence of 4 or more may be present and are very likely to be over-constrained, preventing any time evolution. Thus they have to be split into well-defined vertices, by *uncollapsing* edges. This is exactly the trihedralization problem presented in chapter 5. The only difference is that predicates are no longer evaluated using the target dual geometry but at the time immediately after the immediate events. The evaluation of the trihedralization predicates at $t+\epsilon$ is carried out by evaluating the sign of not only the certificate function but also its derivative

at t.

A key remark is that the vertex zones of these soon-to-be overconstrained vertices are reduced to a point at t and will be infinitesimal at $t+\epsilon$. Thus the vertex trihedralization problems are local and decomposable. The local vertex trihedralizations may then be performed independently. A caveat is that the polyhedral surface may not be manifold at the overconstrained vertex. Trihedralizing such a vertex is thus not possible without extending the ear-cutting approach but may be done using the arrangement coloring approach.

6.3.6 Facet Triangulations

The constrained triangulations of the convex hull of each facet introduce utilitary edges, denoted **soft** edges, that are not relevant to describe the polyhedral surface. These edges only encode a proof that the facets are not self-intersecting by exhibiting a triangulation of its convex hull with non degenerate triangles in between events. By contrast, the constrained edges of the triangulation refer to **hard** edges of the polyhedron and describe a 1D feature of the polyhedral surface. Whenever the topology of a vertex changes, due to a trihedralization (section 6.3.5) or an event processing (section 6.3.8), the possible events of its neighboring triangles are descheduled from $\mathcal{Q}$ and soft edges are discarded. All in all, at this step, the convex hulls of the facets are not (or only partially) triangulated.

The trihedralization has modified the polyhedral topology representation but not the described surface. Besides, the polyhedral representation is no longer guaranteed to be regular. Namely, the edges introduced by the vertex trihedralizations are instantanneously zero-length. An instantaneous triangulation is then required to accept degenerate triangles due to these co-located neighboring vertices. However, this instantanneous triangulation does not guarantee that none of the generated triangles that are instantaneously degenerate will have a positive orientation immediately after the current time. To be able to guarantee that facets will not self-intersect in the immediate future, we require the triangles of the constrained triangulation of the facet convex hull to be positively oriented immediately after the current trihedralization time. This is performed, similar to the trihedralization step, by evaluating both the orientation certificate function and its derivative at the current time.

6.3.7 Orientation Certificate Functions

The simplicity of the 3D polygons (possibly with holes) supporting facets of the polyhedron is proven by the constrained triangulation of their convex hull with triangles that are oriented consistently with the oriented supporting planes. To certify that the polyhedron keeps simple facets, the KDS maintains such a constrained triangulation for each of the facets. These triangulations introduced a second type of edges: the **soft** edges that delimit coplanar triangles, contrary to the **hard** edges of the polyhedron that constrain the triangulation. In figures 6.5 and 6.6, soft edges are represented with dotted segments, and hard edges are solid. For readability, note that only the triangulation of the facet polygons are illustrated rather than the triangulation of their convex hull.

Since the polyhedron undergoes a time evolution, the predicate turns into a certificate function. The triangulation step guarantees that the certificate function of each new triangle is strictly positive immediately after the event, until its first sign-changing root. This root is thus computed in order to schedule the corresponding event in the event queue $\mathcal{Q}$. Triangles that were unaffected by the recent topological changes are already scheduled in $\mathcal{Q}$ and do not require any computing, apart from the maintenance of $\mathcal{Q}$ as a priority queue.

This orientation certificate function is a rational function, and the maximum degrees of its numerator and denominator can be considered. The coordinates of $[\vec{n} : 0]$ are polynoms of degree at most 1. Using the construction of equation 4.2.1, each of the 4 homogeneous coordinates of the 3

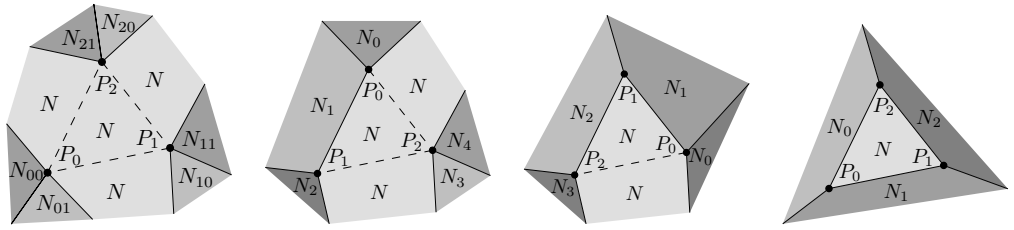


Figure 6.5: Soft and hard triangulation edges. 4 cases of a triangle $P_0P_1P_2$ after the triangulation of a polyhedron facet supported by a plane N. Triangles (a), (b), (c) and (d) are delimited by, respectively, 3, 2, 1 and 0 soft edges, and 0, 1, 2 and 3 hard edges. The triangle geometries involve respectively 7, 6, 5 and 4 planes, including N.

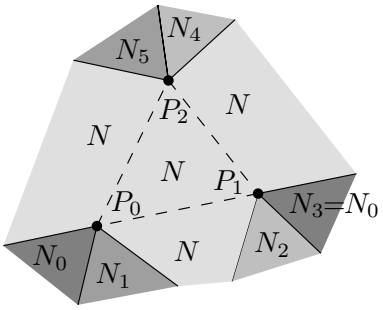


Figure 6.6: Special case without any hard edge. The triangle $P_0P_1P_2$ has 3 soft edges and no hard edges and yet only depends on only 6 distinct planes, since $N_3 = N_0$.

points are polynoms of degree at most 3, in terms of the homogeneous coordinates of their defining planes. Since the interpolation of the homogeneous plane coordinates is linear as a function of time, the homogeneous point coordinates are polynomial and their maximum degree is 3.

The reason why the vertices are assumed to remain bounded is to avoid the tricky situation where the Cartesian coordinates $\frac{\vec{P}}{w}$ of a point are evaluated at one of its poles. Keeping all the vertices bounded ensures that sign changes in the orientation certificates are only due to its roots and not its poles.

This yields a rational function that has a numerator of degree at most 10 and a factorization of the denominator into 3 polynoms of degree at most 3. The sign of the certificate can be determined by the independent inspection of the signs of the factors of its numerator and denominator and the evaluation of the root multiplicities if the current time is a root of the denominator. This speeds up the evaluation compared to computing blindly roots of the multiplication of the numerator by the denominator because this polynom can be of degree 10+3+3+3=19 !

For each of the triangles of the facet triangulations, a triangle orientation predicate, as introduced in equation 4.2.4, is computed. It tests whether the triangle $P_0P_1P_2$ is direct, degenerate or indirect with respect to the supporting oriented plane N:

$$Orientation(\vec{N},\vec{P}_0,\vec{P}_1,\vec{P}_2) = sign\left(\frac{1}{w_0w_1w_2}\begin{vmatrix}\vec{n} & \vec{p}_0 & \vec{p}_1 & \vec{p}_2\\ 0 & w_0 & w_1 & w_2\end{vmatrix}\right) \quad \text{with} \begin{cases}\vec{P}_i = [\vec{p}_i : w_i]\\ \vec{N} = [\vec{n} : d]\end{cases}$$

Orientation Certificate Factorization The numerator of the orientation certificate function can be further factorized: $|\vec{n}|^2$ always divides the numerator. It is a square that is explicitly enforced to remain strictly positive, otherwise the plane would become the plane at infinity, which

is avoided by disallowing interpolation between planes that are parallel with opposite orientations. Since the $|\vec{n}|^2$ can not produce any sign change, there is no need to consider it to compute the roots of the certificate function. If Q denotes the polynom that results from the division of $\begin{vmatrix} \vec{n} & \vec{p}_0 & \vec{p}_1 & \vec{p}_2 \\ 0 & w_0 & w_1 & w_2 \end{vmatrix}$ by $|\vec{n}|^2$, its maximum degree is 8, and the sign of the certificate function can be simplified as:

$$Orientation(\vec{N}, \vec{P}_0, \vec{P}_1, \vec{P}_2) = sign\left(\frac{|\vec{n}|^2\, Q}{w_0 w_1 w_2}\right) = sign\left(\frac{Q}{w_0 w_1 w_2}\right)$$

In the general case, the certificate function involves 7 distinct planes: the plane that supports the triangle and 2 extra planes $\vec{N}_j, \vec{N}_k$ to define the location of each vertex $\vec{P}_i$ of the triangle. Since all those planes are distinct, all the 3 edges of the triangle are soft (triangulation) edges rather than hard edges of the polyhedron (Fig. 6.5(a)). In this general case, the polynom Q_7 (indexed by the number of distinct planes involved) can not be further factorized. An efficient way to compute Q_7 in this general case is yet to be designed. Roots are searched for the unfactored polynom, of maximum degree 8, computed as the division of $\begin{vmatrix} \vec{n} & \vec{p}_0 & \vec{p}_1 & \vec{p}_2 \\ 0 & w_0 & w_1 & w_2 \end{vmatrix}$ by $(n \cdot n)$, using the symbolic computer algebra system [MAXIMA].

As soon as only 6 or less planes are involved, Q may be factored into 2 polynoms of maximum degree 4. It occurs when at least one edge is a hard edge (Fig. 6.5 (b), (c) and (d)). Even a triangle with 3 soft edges may be factorized, if the supporting line of one of its edges is defined by the intersection of two planes, as it is the case for the edge P_0P_1 in figure 6.6. The property that at least one of the 3D lines supporting an edge of the 3D triangle is defined as the intersection of two planes greatly reduces the root finding time costs, by allowing the factorization of Q into 2 polynoms of maximum degree 4. When only 4, 5, or 6 plane equation are involved, the factorizations of Q are:

$$Q_4 = -\left|\vec{N} \quad \vec{N_0} \quad \vec{N_1} \quad \vec{N_2}\right|^2 \tag{6.3.1}$$

$$Q_5 = -\left|\vec{N} \quad \vec{N_0} \quad \vec{N_1} \quad \vec{N_2}\right| . \left|\vec{N} \quad \vec{N_1} \quad \vec{N_2} \quad \vec{N_3}\right| \tag{6.3.2}$$

$$Q_6 = -\left|\vec{N} \quad \vec{N_0} \quad \vec{N_1} \quad \vec{N_2}\right| . \left|\vec{N} \quad \vec{N_1} \quad \vec{N_3} \quad \vec{N_4}\right| \tag{6.3.3}$$

When only 4 planes are involved in the certificate function (fig. 6.5 (d)), Q_4 is the opposite of a square. Thus, it will not provoke any sign change, so there is no point in computing its roots.

Proof. **Factorization:** When the orientation certificate involves only 6 planes or less, one of the edges of the triangle can be defined both as the line passing through two points and the intersection line of two planes. Using the equation 4.2.3, with the point and plane indices defined in figure 6.6, we get:

$$\begin{aligned} \vec{L}(\vec{N}, \vec{N_0}) &= [\vec{n} \wedge \vec{n}_0 : d\vec{n}_0 - d_0\vec{n}] \\ \vec{L}(\vec{P}_0, \vec{P}_1) &= [w_0\vec{p}_1 - w_1\vec{p}_0 : \vec{p}_0 \wedge \vec{p}_1] \\ \vec{L}(\vec{P}_0, \vec{P}_1) &= \left|\vec{N} \quad \vec{N_0} \quad \vec{N_1} \quad \vec{N_2}\right| \vec{L}(\vec{N}, \vec{N_0}) \end{aligned}$$

One may interpret the numerator of the triangle orientation predicate as the coplanarity predicate $\vec{L}(\vec{P}_0, \vec{P}_1) \odot \vec{L}(\vec{P}_2, \vec{n})$ of the line that goes from P_0 to P_1, and the line that passes through P_2 in the direction $\vec{n}$:

$$\begin{aligned} \begin{vmatrix} \vec{n} & \vec{p}_0 & \vec{p}_1 & \vec{p}_2 \\ 0 & w_0 & w_1 & w_2 \end{vmatrix} &= \frac{1}{w_0}\begin{vmatrix} \vec{n} & \vec{p}_0 & w_0\vec{p}_1 - w_1\vec{p}_0 & \vec{p}_2 \\ 0 & w_0 & 0 & w_2 \end{vmatrix} \\ &= \frac{1}{w_0}\left(w_0 \left|\vec{n} \quad w_0\vec{p}_1 - w_1\vec{p}_0 \quad \vec{p}_2\right| + w_2 \left|\vec{n} \quad \vec{p}_0 \quad w_0\vec{p}_1 - w_1\vec{p}_0\right|\right) \\ &= \left|w_0\vec{p}_1 - w_1\vec{p}_0 \quad \vec{p}_2 \quad \vec{n}\right| + w_2 \left|\vec{n} \quad \vec{p}_0 \quad \vec{p}_1\right| \\ &= (w_0\vec{p}_1 - w_1\vec{p}_0) \cdot (\vec{p}_2 \wedge \vec{n}) + w_2\vec{n} \cdot (\vec{p}_0 \wedge \vec{p}_1) \\ &= [w_0\vec{p}_1 - w_1\vec{p}_0 : \vec{p}_0 \wedge \vec{p}_1] \odot [w_2\vec{n} : \vec{p}_2 \wedge \vec{n}] = \vec{L}(\vec{P}_0, \vec{P}_1) \odot \vec{L}(\vec{P}_2, \vec{n}) \end{aligned}$$

Since $\vec{L}(\vec{P}_0, \vec{P}_1)$ is proportional to $\vec{L}(\vec{N}, \vec{N}_0)$, we are interested in simplifying $\vec{L}(\vec{N}, \vec{N}_0) \odot \vec{L}(\vec{P}_2, \vec{n})$, using the

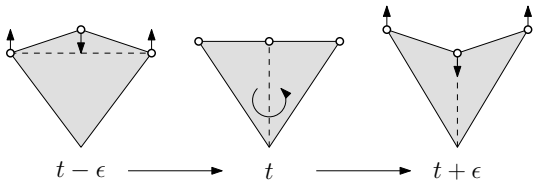


Figure 6.7: Example evolution of a vertex colliding with the opposite soft edge

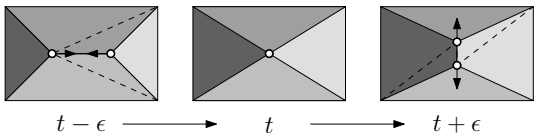


Figure 6.8: Example evolution of a collapsing hard edge

relations $\forall \vec{x},\vec{y},\vec{z} \in \mathbb{R}^3$, $\vec{x} \wedge (\vec{y} \wedge \vec{z}) = \vec{y}(\vec{x}\cdot\vec{z}) - \vec{z}(\vec{x}\cdot\vec{y})$ and $\vec{x}\cdot(\vec{y}\wedge\vec{z}) = |\vec{x}\ \vec{y}\ \vec{z}| = \vec{y}\cdot(\vec{z}\wedge\vec{x})$:

$$\begin{aligned}
\vec{L}(\vec{N},\vec{N}_0) \odot \vec{L}(\vec{P}_2,\vec{n}) &= [\vec{n}\wedge\vec{n}_0 : d\vec{n}_0 - d_0\vec{n}] \odot [w_2\vec{n} : \vec{p}_2\wedge\vec{n}] \\
&= (\vec{n}\wedge\vec{n}_0)\cdot(\vec{p}_2\wedge\vec{n}) + (d\vec{n}_0 - d_0\vec{n})\cdot w_2\vec{n} \\
&= \vec{p}_2\cdot(\vec{n}\wedge(\vec{n}\wedge\vec{n}_0)) + dw_2\vec{n}_0\cdot\vec{n} - d_0w_2\vec{n}\cdot\vec{n} \\
&= \vec{p}_2\cdot(\vec{n}(\vec{n}\cdot\vec{n}_0) - \vec{n}_0(\vec{n}\cdot\vec{n})) + w_2(d\vec{n}_0\cdot\vec{n} - d_0\vec{n}\cdot\vec{n})) \\
&= \vec{P}_2\cdot(\vec{N}(\vec{n}\cdot\vec{n}_0) - \vec{N}_0(\vec{n}\cdot\vec{n})) \\
&= -\vec{P}_2\cdot\vec{N}_0(\vec{n}\cdot\vec{n}) \qquad \text{(since } \vec{P}_2 \text{ is in the plane } \vec{N},\quad \vec{P}_2\cdot\vec{N} = 0)
\end{aligned}$$

Using $\vec{P}_2\cdot\vec{N}_0 = \left|\vec{N}_0\quad\vec{N}\quad\vec{N}_4\quad\vec{N}_5\right|$, this finally proves the factorization of Q_6 with the plane indices of figure 6.6:

$$\begin{vmatrix}\vec{n} & \vec{p}_0 & \vec{p}_1 & \vec{p}_2 \\ 0 & w_0 & w_1 & w_2\end{vmatrix} = -(\vec{n}\cdot\vec{n})\left|\vec{N}\quad\vec{N}_0\quad\vec{N}_1\quad\vec{N}_2\right|\left|\vec{N}_0\quad\vec{N}\quad\vec{N}_4\quad\vec{N}_5\right|$$

The same derivation, up to the labeling of the planes, yields the derivation of equation 6.3.3. The derivation of Q_4 or Q_5 in equations 6.3.1 and 6.3.2 follows easily by adjusting the plane indices accoding to figure 6.5. □

6.3.8 Orientation Event Processing

6.3.8.1 Orientation Event Analysis

Since the polyhedron remains bounded, the 3 vertices of the failing triangle have finite coordinates. The failing of the triangle certificate means that those 3 points are aligned. The topological update is simply a topological translation of the geometric singularity.

In general, the 3 points are distinct, meaning that one of the 3 vertices is colliding with its opposite triangle edge. Two cases may occur:

Soft edge collide: The vertex is colliding with a soft edge, which does not delimit the boundary of a polyhedral facet. This event is auxiliary: the maintained polyhedron is not ceasing to be

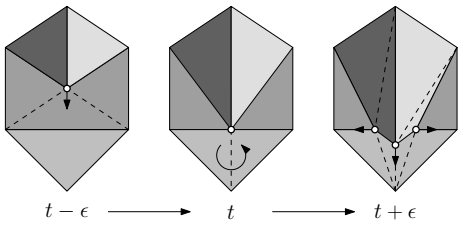


Figure 6.9: Example evolution of a vertex colliding with the opposite hard edge

simple, but the proof that it is simple is no longer valid. This type of event is qualified as an **internal event** in contrast with the following **external events** that require a modification of the maintained polyhedron to maintain its simplicity. To update the certificate that the polyhedron is valid, a simple flip of the colliding edge suffices. This topological change is continuous because it only involves 2 triangles in a common supporting plane, and thus does not change the plane adjacencies of the polyhedral vertices (Fig. 6.7). The flip is required to prove that all the instantaneous singularities of the colliding vertex have been translated into the polyhedron topology. If the null area triangle was kept, it would be harder to detect when the colliding vertex is also simultaneously colliding an other edge beyond the first edge.

Hard edge collide: When the colliding edge is the intersection of two distinct polyhedral facets, the colliding vertex now faces a singularity of the polyhedron itself rather than a singularity of its simplicity proof. The vertex is instantly adjacent to one more plane. To translate this into the polyhedron topology, the polyhedron is updated by affecting the failing triangle to the plane supported by the facet that is opposite to the colliding vertex. This topological modification is continuous because the area of a failing triangle is null. Thus, the colliding hard edge is now a soft edge. In order to verify that the singular vertex is distinct from its adjacent vertices, the colliding soft edge is flipped as in the previous case. An example evolution is sketched with a hard edge collision in figure 6.9.

In singular cases, either 2 or all the 3 points of the triangle are equal, yielding a collapsing edge or triangle:

Edge collapse: The polyhedron is updated by topologically collapsing one of the collapsing edges. An example result at $t + \epsilon$ is sketched in Fig. 6.8. This occurs when the rank of the adjacent plane coordinates is 3. The topological edge collapse is geometrically continuous, since it removes only two triangles of null area.

Triangle collapse: The triangle collapsing is performed by two successive edge collapses.

The analysis has been implemented by considering the signs at the exact time of the event of the following quantities for a triangle $P_0P_1P_2$, for $i = 0, 1, 2$:

$$(\vec{P}_{i+1} - \vec{P}_i) \cdot (\vec{P}_{i+2} - \vec{P}_i) \quad , \quad \text{where } P_i = P_{(i \mod 3)} \tag{6.3.4}$$

When expressed in terms of the plane supporting the P_i, this formula straightforwardly turns into a rational function factored into a numerator of degree 12 and 4 denominators of degree 3, assuming homogeneous plane evolutions of degree 1.

There is however a caveat that prevents a purely static evaluation of the predicate above. Namely, the current time may be a root of the 4 homogeneous coordinates of a point P_i, leaving this point statically undefined $[0 : 0 : 0 : 0]$. This happens, for instance, when the convex angle formed by two hard edges adjacent to a vertex in a polyhedron facet becomes concave, similar to the example of figure 6.7. In this case, the root multiplicities of the rational function must be compared to assess the sign of the rational function as time tend to the current time.

6.3.8.2 Processing Analyzed Events

The events come are of two natures, they either encode a degenerate triangle of the constrained triangulation of the facet convex hull, or a degenerescence of a convex hull itself. The processing of convex hull events are trivial and we focus on the processing of the degenerate triangles of the constrained triangulation.

Processing these analyzed events modifies the polyhedral topology so that it gives a regular representation of the polyhedral surface. In other words, its purpose is to translate the instantaneous geometric singularities into the topology of the polyhedron.

This is performed by first topologically collapsing all the triangles undergoing a geometric edge collapse. A special care is needed to order the collapses to prevent collapsing edges that do not satisfy the link condition [DEGN98]. This condition, well-known to edge collapse-based mesh

simplifiers, prevent topological singularities, such as antennae. More precisely it is a combinatorial condition that guarantees that the collapse will not change the topology of the desribed surface, such as the genus. We assume that there is always an ordering of collapses that satisfies the link condition. This may however not be always the case : a simple example is a vanishing tetrahedron. It is composed of 4 collapsing triangles, and none of the 6 edges satisfy the link condition.

Now only triangles with a colliding edge are left. Their support is a non-trivial segment. A colliding edge may face multiple cases, depending on the facet neighboring the degenerate triangle through the colliding edge :

1. The neighboring facet is not a degenerate triangle. Then it may be merged with the degenerate triangle, discarding the colliding edge and adding the colliding vertex to the neighboring facet.
2. The neighboring facet is a triangle with a colliding edge which is not the same edge. The processing of this triangle is postponed after the processing of its neighboring triangle.
3. The neighboring facet shares the same colliding edge. Then a predicate similar to equation 6.3.4 is used to order the two opposite colliding vertices along the colliding edge.

To conclude, this step successfully processes the set of instantanneously degenerate triangles and modifies the topology of the polyhedral representation to make it regular.

6.3.9 Discussion

It would have been tempting to process events one at a time without grouping simultaneous events. However, it would have been unclear how to assert the validity of this alternate approach. More precisely, avoiding infinite loops where topological changes recursively create immediately failing configurations would have been difficult.

By contrast, the proposed approach guarantees that, at the end of the simultaneous event processing, the polyhedral topology encodes the regular representation of the polyhedral surface, which is the loop invariant. This loop invariant provides the independance of the trihedralization problems and the existence of the non-degenerate triangulations in-between events. Efficiency-wise, it uses directly the readily available set of immediately degenerate triangles to performed the required topological changes locally without exploring the polyhedron.

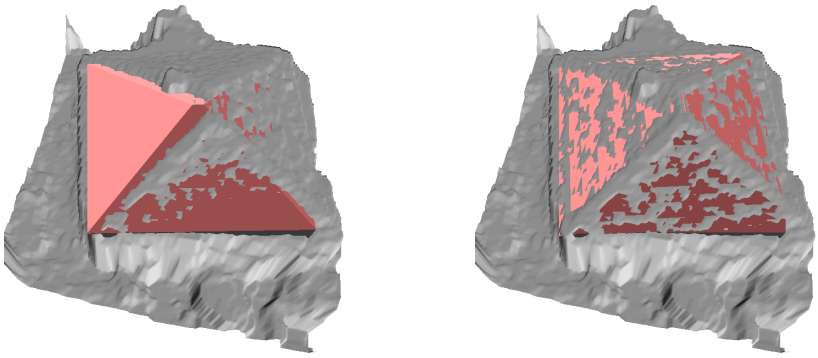
6.4 Topology-Aware Fitting of a 3D Building Model

The proposed approach has been implemented on top of the kinetic data structure package of the CGAL library [CGAL], and heavily relies on CGAL exact arithmetic to prevent numerical inconsistencies, at the expense of high computing times for nearly degenerate predicate evaluations. These sample evolutions of figures 6.10, 6.11, 6.12, 6.13, 6.14 and 6.15 are driven by successive plane estimations according to the DSM, as presented in chapter 4.

Figure	6.10	6.11	6.12	6.13	6.14	6.15
Plane estimations	0.18s	11.31s	0.23s	3.21s	1.57s	1.86s
Trihedralizations	0.09s	0.1s	0.54s	5.53s	0.19s	3.04s
	7	11	14	9	14	30
Kinetic evolution	0.46s	19.18s	1.47s	0.97s	0.51s	79.06s
Event analysis	0.14s	16.05s	0.25s	0.05s	0.08s	75.79s
Events processed	3	23	8	5	12	65
Total (**P**+**T**+**K**)	0.73s	30.59s	2.24s	9.71s	2.27s	83.96s

Table 6.1: Computing times and number of events processed and trihedralizations, measured on a single Intel Xeon 1.60GHz CPU core.

Table 6.1 illustrates how the computing times of the roof fittings involved in figures break down among the various tasks. The *plane estimation* time (chapter 4) measures the reestimation times of all the planes for all iterations. The *trihedralization* time (chapter 5) accounts for all trihedralizations needed at initialization or during the evolution. The *kinetic evolution* time (this chapter) refers to the scheduling, analysis and processing of the events.



(a) DSM and input building (b) DSM and output building

Figure 6.10: Simple building fitting (identical to figure 1.3).

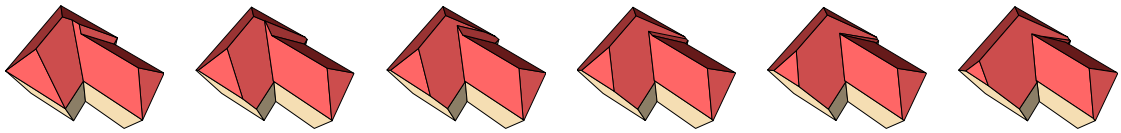


Figure 6.11: T-shaped building fitting (identical to figure 1.4).

The dominating task in the kinetic evolution appears to be the analysis of a failing event (section 6.3.8.1). During the kinetic evolution, this event analysis step involves determining the sign of high degree polynomials, using exact arithmetics, at the current time, which is the root of another polynomial, in order to discover which edge collapses or which vertex collides with its opposite edge. Figure 6.15 is an example where this particular step takes more than 90% of the total computing time. We believe that this event analysis step may be optimized, by factorizing the polynomials involved, as with the orientation predicate function, or by a global analysis of all the simultaneously failing events.

Since our algorithm performs minimal topological changes, it never creates any facet, so a case like in figure 6.13, the missing facet at the right is not reconstructed. A simple extension would be to check if the facets with high residual errors can be advantageously split into some small number of facets, with robustly estimated target plane coordinates, and carry on the evolution with the split facet.

In figure 6.14, the top, bottom and left facets are competing to fit the same data (the points of the DSM of the left roof plane). At $2 < t < 3$ and later at $4 < t < 5$, regularizations occured that discarded of a collapsing facet. Another possible outcome could have been that redundant facets settle to be distinct but nearly coplanar, as in figure 6.15. Some possible extension could be to detect these nearly coplanar facets to merge them together, when no regularization simplifies the topology automatically.

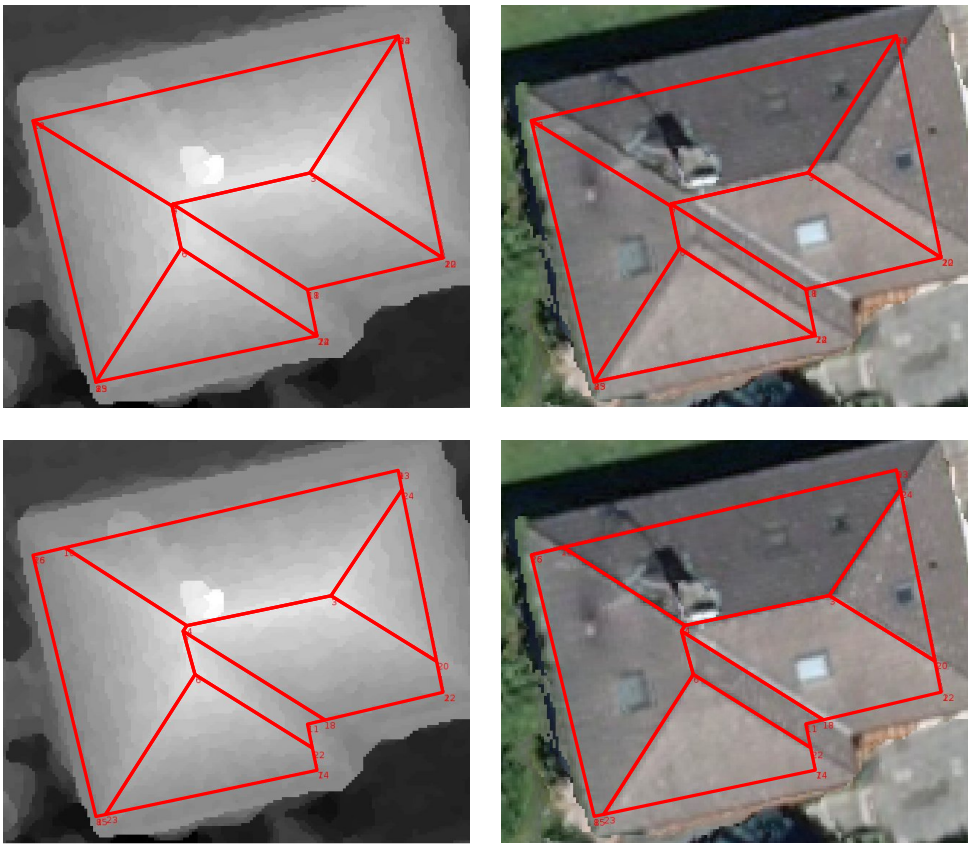


Figure 6.12: (left) the DSM, (right) an aerial image, (top) the initial building wireframe, (bottom) the fitted building.

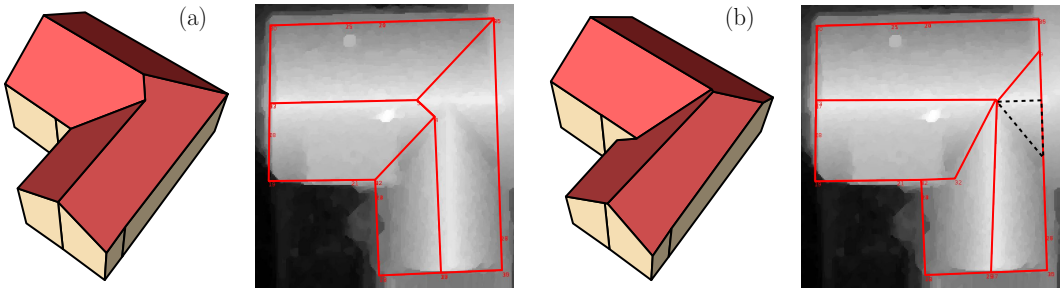


Figure 6.13: (a) Initial building, (b) fitted building : the missing triangular facet (dashed) may not be created.

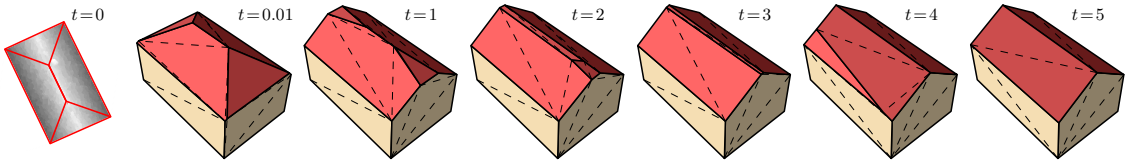
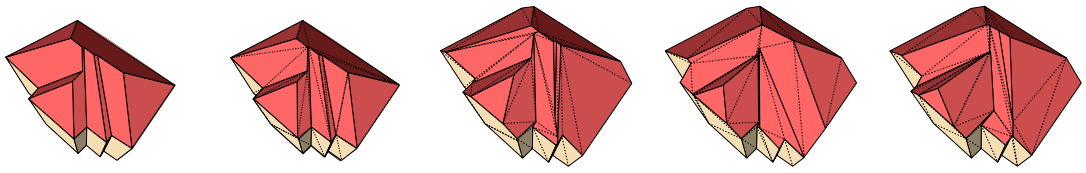


Figure 6.14: Fitting this building requires 5 iterations. 2 facets are removed, due to the intermediate regularizations.

6.5 Discussion

6.5.1 Complexity

It is not straightforward to analyze a kinetic data structure with the usual worst-case or expected time complexity, since it is higly dependent on the primitive trajectories and the resulting number



$t = 0$ Initial Model | $t = 0.01$ | $t = 1$ | $t = 2$ | $t = 7$ Convergence

Figure 6.15: A more complex building fitting example, requiring 7 iterations of plane estimations. Note that 3 almost coplanar facets remain.

of external events. Therefore, a terminology was introduced in [BGH97, BCG+99] to qualify the complexity of a kinetic data structure:

Compactness: The space complexity of the KDS. A triangulation is linear in the size of its vertices, and so is its convex hull. The maximum number of events in the queue is one orientation event per triangle of the constrained facet triangulations and one per convex hull vertex. Since events have a constant size, this makes the proposed KDS **compact** (linear in size).

Efficiency: The efficiency measures the ratio of the number of external events to the number of all events (internal and external). In the proposed KDS, the internal events are the soft edge collision and the convex hull events, while the hard edge collisions and the edge and triangle collapses are external events.

Responsiveness: This qualifies the time complexity of processing an event. Processing the orientation events takes constant time. The trihedralization step may be much more complex for highly degenerate configurations where more than 4 planes simultaneously meet. However, except for such degenerate input initial and target plane geometries, trihedralizations are performed on vertices adjacent to 4 planes only, and can thus be considered to have a constant time complexity.

Locality: This measures the cost of recomputing events created by a topological modification or after a plane trajectory has changed. The KDS is as **local** as it could be: the number of events that depend on a single plane is linear relative to the complexity (the number of edges and vertices) of the facets of the polyhedron supported by this plane, and the triangles sharing a vertex with them but supported by other planes.

6.5.2 Method Invariance by an Invertible Affine Transform

The following proposition is proven in appendix D:

Proposition 6.1. *The approach proposed in chapters 5 and 6 is unconditionally invariant under a projective transform of the point coordinates (the primal geometry), if and only if this transform is invertible and affine.*

We define the meaning of the unconditional invariance such that the combinatorial data structures computed by the proposed method for the transformed and original problems are isomorphic, up to ambiguities of the trihedralization subproblems, without any condition on the geometry or topology of the problem.

This insensitivity to an affine transform proves that the proposed approach is not biased by a preferred direction such as the vertical direction (rotation invariance), and that it is scale and translation invariant. The assymetric treatment of the vertical planes in the roof fitting problem is thus only caused by the plane estimation step of chapter 4.

6.5.3 Normalization Dependance

An homogeneous scaling by α of the point coordinates, using the identity matrix I times any non-null scalar α is a particular case of an affine invertible transformation. The polyhedron computed using the proposed approach is thus invariant by such a transform on the primal geometry. Since $\text{com}(\alpha I) = \alpha^3 I$, any uniform scaling by α of the dual geometry may be viewed as the uniform scaling by α^{-3} of the primal geometry. Thus the invariance is also valid for uniform scalings of the dual geometry. Thus, the proposed approach is invariant under a **uniform** scaling of the primal and/or of the dual geometry.

However, the approach is sensitive to the relative normalization of the homogeneous coordinates of the planes. It is caused by the sensitivity of the interpolation scheme itself to those normalizations. The greater the target plane normalization is relative to the initial plane normalization, the faster the interpolated plane will come close to the target plane. Therefore, a **non-uniform** scaling of the primal or the dual geometry will modify the relative interpolating speed between primitives, although the initial and target Cartesian formulation of the problem is unchanged. The modification of the relative speeds of the rotating or translating planes will modify the point trajectories at the intersection of those planes. Thus the order of the events may change which could imply differing resulting topologies.

The next paragraphs describe how the trajectory of the interpolation of two points, two intersecting planes or two parallel planes is insensitive to their normalizations but how the speed of the interpolating primitive along its trajectory is affected by the relative normalization of the interpolated primitives.

Finite points: The interpolated point $(1-t).\vec{P}_0 + t.\vec{P}_1$ between two finite points $\vec{P}_0$ and $\vec{P}_1$ only has $(1-t, t)$ as its barycentric coordinates relative to $\vec{P}_0$ and $\vec{P}_1$ if they have the same point normalization ($\alpha = \frac{w_1}{w_0} = 1$). In the general case ($w_1 \neq w_0$), its barycentric coordinates are $\left(\frac{1-t}{1-t+\alpha t}, \frac{t\alpha}{1-t+\alpha t}\right)$:

$$\frac{(1-t)\vec{p}_0 + t\vec{p}_1}{(1-t)w_0 + tw_1} = \left(\frac{1-t}{1-t+\alpha t}\right)\frac{\vec{p}_0}{w_0} + \left(\frac{t\alpha}{1-t+\alpha t}\right)\frac{\vec{p}_1}{w_1} \quad \text{with } \alpha = \frac{w_1}{w_0}$$

For instance, the midpoint of the segment $[\vec{P}_0\vec{P}_1]$ is reached when the two barycentric coordinates are equal at time $t = \frac{1}{1+\alpha} = \frac{w_0}{w_0+w_1}$ (which is only possible with $t \in [0,1]$ if $w_0 w_1 \geq 0$).

Parallel planes: Using now the relative plane normalization $\alpha = \frac{|\vec{n}_1|}{|\vec{n}_0|}$, the same relation holds with the translation of an interpolating plane between two similarly oriented parallel planes ($\vec{n}_0 \cdot \vec{n}_1 > 0$ and $\vec{n}_0 \wedge \vec{n}_1 = \vec{0}$). The signed distances of the planes to the origin are $\frac{d_0}{|\vec{n}_0|}$ and $\frac{d_1}{|\vec{n}_1|}$, and the one of the interpolating plane at time t is:

$$\begin{aligned}
\frac{(1-t)d_0 + td_1}{|(1-t)\vec{n}_0 + t\vec{n}_1|} &= \frac{(1-t)d_0 + td_1}{(1-t)|\vec{n}_0| + t|\vec{n}_1|} \quad \text{since } \vec{n}_0 \cdot \vec{n}_1 > 0 \text{ and } \vec{n}_0 \wedge \vec{n}_1 = \vec{0} \\
&= \left(\frac{1-t}{1-t+\alpha t}\right)\frac{d_0}{|\vec{n}_0|} + \left(\frac{t\alpha}{1-t+\alpha t}\right)\frac{d_1}{|\vec{n}_1|} \quad \text{with } \alpha = \frac{|\vec{n}_1|}{|\vec{n}_0|}
\end{aligned}$$

Intersecting planes: The interpolation of two intersecting plane is the rotation of the initial plane to the target plane around their intersecting line. The barycentric coordinates of the unnormalized plane normal $(1-t)\vec{n}_0 + t\vec{n}_1$ relative to the interpolated plane normalized normals $\frac{\vec{n}_0}{|\vec{n}_0|}$ and $\frac{\vec{n}_1}{|\vec{n}_1|}$ are $\left(\frac{(1-t)|\vec{n}_0|}{(1-t)|\vec{n}_0|+t|\vec{n}_1|}, \frac{t|\vec{n}_1|}{(1-t)|\vec{n}_0|+t|\vec{n}_1|}\right)$. The barycentric coordinates of the plane bisector is $(\frac{1}{2}, \frac{1}{2})$. Thus, the plane bisector of two intersecting planes is reached at time $t = \frac{|\vec{n}_0|}{|\vec{n}_0|+|\vec{n}_1|}$.

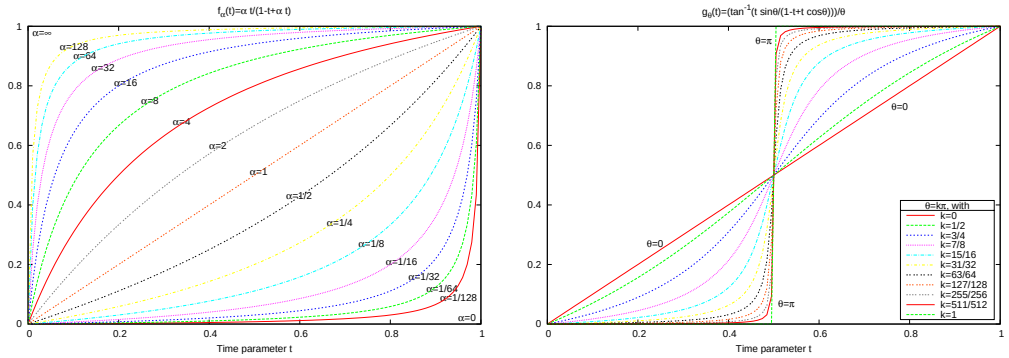


Figure 6.16: Dependance on plane normalization. The interpolation parameterization warping function f_α, depending on the relative plane normalization $\alpha = \frac{|\vec{n}_1|}{|\vec{n}_0|}$ and the angle ratio function g_θ, giving the rotation rate of the interpolation of two intersecting planes, depending on the dihedral angle θ.

Those barycentric coordinates may also be written as $(1-t', t') = \left(\frac{1-t}{1-t+\alpha t}, \frac{\alpha t}{1-t+\alpha t}\right)$ using the relative plane normalization $\alpha = \frac{|\vec{n}_1|}{|\vec{n}_0|}$. The derivation of the rotation angle of the interpolating plane around the line of intersection of the two interpolated planes yields, using the barycentric coordinates $(1-t', t')$ of the normal relative to the normalized initial and target normal:

$$\begin{aligned} Angle\left(\vec{n}_0,\ (1-t')\frac{\vec{n}_0}{|\vec{n}_0|} + t'\frac{\vec{n}_1}{|\vec{n}_1|}\right) &= \tan^{-1}\left(\frac{t'|\vec{n}_0 \wedge \vec{n}_1|}{(1-t')|\vec{n}_0||\vec{n}_1| + t'\vec{n}_0 \cdot \vec{n}_1}\right) \\ &= \tan^{-1}\left(\frac{t'\sin\theta}{1-t'+t'\cos\theta}\right) \quad \text{with } \theta = Angle(\vec{n}_0, \vec{n}_1) \\ &= \tan^{-1}\left(\tan\theta\left(f_{(\cos\theta)}(t')\right)\right) \quad \text{if } \cos\theta \neq 0 \end{aligned}$$

Function analysis: The function $f_\alpha(t) = \frac{\alpha t}{1-t+\alpha t}$ (fig. 6.16, left) expresses how the time is warped by the ratio α of the target by the initial normalization (w or $|\vec{n}|$, concerning respectively points or planes). $f_1(t) = t$ is the identity function and this family of functions is closed under compositions: $f_{\alpha_1} \circ f_{\alpha_2} = f_{(\alpha_1\alpha_2)}$.

The function $g_\theta(t') = \frac{\tan^{-1}\left(\frac{t'\sin\theta}{1-t'+t'\cos\theta}\right)}{\theta}$ (fig. 6.16, right) evaluated at $t' = f_\alpha(t)$ measures the percentage of the rotation achieved as a function of time for interpolated intersecting planes, with a dihedral angle of θ. If $\cos\theta \neq 0$, then $\tan(\theta g_\theta(f_\alpha(t))) = \tan\theta\left(f_{(\cos\theta)}(f_\alpha(t))\right) = \tan\theta\left(f_{(\cos\theta)\alpha}(t)\right)$: tangents have their interpolation time t warped by $f_{(\cos\theta)\alpha}$.

Assuming $\alpha > 0$ and $0 < \theta < \pi$, the functions $f_\alpha(t)$ and $g_\theta(t')$ are both strictly increasing from $f_\alpha(0) = g_\theta(0) = 0$ to $f_\alpha(1) = g_\theta(1) = 1$. Thus, f_α, g_θ and $(g_\theta \circ f_\alpha)$ are increasing bijections from $[0,1]$ to itself: $f_\alpha([0,1]) = g_\theta([0,1]) = g_\theta(f_\alpha([0,1])) = [0,1]$. Thus, varying the angle between interpolated planes and/or the normalization of the plane coordinates only changes the speed of the interpolated plane along the trajectory, but the trajectory itself remains unchanged. Finally even if the construction of the points is insensitive to the normalization of the interpolating planes, the trajectory of a trihedral vertex is very likely to change because its three defining planes are moving at speeds that depend on the initial and target plane normalization.

Discussion: The relative normalization $\frac{|\vec{n}_1|}{|\vec{n}_0|}$ of the homogeneous coordinates of each plane adds a degree of freedom to the problem. The current implementation does not deeply take advantage of these degrees of freedom. One obvious possibility is to normalize all the initial and target plane

coefficients. All exact arithmetics algorithm libraries are not able to represent square roots, and those who can handle square roots undergo the computing costs of the augmented complexity of this feature. Thus, the normalization ratios are only kept close to 1, rather than exactly 1. Alternatively, within our building fitting context, planes are classified into three groups : the infinitely low bottom plane ($c = -1$), the facades ($c = 0$) and the roofs ($c = 1$, no overhead). Since planes are not allowed to move from one of these groups to another, their geometric evolution may be performed with a constant c coordinate. The huge benefit of this normalization is that it decreases the degrees of the geometric predicates and consequently speeds up the computations. For instance, for the general-case orientation predicate which has factors of degree 10, 3, 3 and 3, dealing with planes with constant c coordinates brings factor degrees down to 8, 2, 2 and 2.

These degrees of freedom may allow the exploration of different geometric paths from the initial to the target dual geometry. For instance, these degrees of freedom may be used to avoid intermediate degeneracies by slightly pertubating the relative normalizations of the planes involved. These degrees of freedom, may also be combined with higher order interpolations than the basic linear interpolation: Bezier curves or any other Spline curve. The evolution of each plane would no longer be restricted to a simple translation or rotation.

Using these extra possibilities would however be tricky. For instance, it would be easy to always end up with a polyhedron that collapses to a single point if the plane trajectories are arbitrary. However, if a specific application provides a motivation for complex plane trajectories, the only downsides would be the handling of higher degree rational functions in the certificates, and the internal events between the rational patches for piecewise rational interpolations.

6.6 Perspectives

6.6.1 Diverging Vertices

The plane evolution of the 3D building model fitting application described in section 6.4, driven by the plane estimates of chapter 4, is, by construction, unable to make a vertex of the polyhedron go to infinity.

Proof. This due to the way planes are fitted to the DSM: the vertical planes are not updated and the estimated coordinates of the top-facing planes are of the form $[a : b : 1 : d]$. The bottom facing plane and its neighboring planes, supporting the walls of the building footprint, remain fixed. Furthermore, the bottom-facing plane is treated as the plane at infinity: or equivalently, it is set sufficiently low, so that the vertices of the top facing planes will never collide with the bottom facing plane. This ensures that the polyhedron footprint (its vertical projection) remains fixed. Thus the only direction in which a polyhedron vertex may diverge is along the z axis: $(0, 0, \pm\infty)$, which is written $[0 : 0 : z : 0]$, with $z \neq 0$ using homogeneous coordinates. A point adjacent to at least one top facing facet may not contain such an infinite point: $[a : b : 1 : d] \cdot [0 : 0 : z : 0] = z \neq 0$. Vertices adjacent to the bottom facet are fixed, which reduces the candidate infinite vertices to the vertices adjacent to vertical facets only. Since the trihedralization step ensures that infinite cells are outside the polyhedron, an infinite vertex may not be created. □

However, when moving arbitrarily the plane equations, a vertex may pass through the plane at infinity during the evolution. This is reflected in a pole in its Cartesian trajectory. If the multiplicity of the pole of at least one of its Cartesian coordinates is odd, then the Cartesian trajectory of this point is not continuous. Hence, the polyhedral surface, which passes through this point, will not be continuous. Detecting these poles and thus the diverging vertices by scheduling poles of the Cartesian coordinates is easy. However, processing these event while keeping a continuous evolution is impossible, since there is a discontinuity in the described surface: a point is going to infinity in one direction and coming back from the opposite direction.

As an extension, we propose to avoid the infinite vertex events by forcing the evoluting polyhedron to stay inside an input convex polyhedral volume V. The property of staying inside V would be maintained using the kinetic framework by introducing a complementary certificate proving that the polyhedron remains inside V and the processing of events where the polyhedron would cease to be inside V. For instance, the volume V could be a bounding box $[-M, M]^3$ for any sufficiently large real value M.

6.6.1.1 Boundedness Certificate

Because a polyhedron can be triangulated and each triangle is a convex combination of 3 points of the polyhedron, a polyhedron is inside a convex volume V if and only if all of its vertices are inside this convex volume. The convex bounding volume V may be given by a set of oriented planes N_{iV} that point towards the exterior of the bounded volume. A vertex P remains inside V if and only if the *above* certificate is negative for all faces of the bounding volume: $\forall i, \frac{\vec{P}}{w_P} \cdot \vec{N}_{iV} \leq 0$. Thus for each vertex $\vec{P}$ of the polyhedron and each plane of the convex bounding volume V, we can define its boundedness certificate function $-\frac{\vec{P}}{w_P} \cdot \vec{N}_{iV}$. These certificate functions are not negative for each vertex and each bounding plane, if and only if the polyhedron is bounded by V. Since the plane equations are the result of a linear interpolation, their coefficients are polynoms of maximum degree 1. If the bounding volume V is kept fixed, that means that the certificate function is a rational function and the maximum degrees of its numerator and denominator are both 3. These certificate functions are handled exactly as the triangle orientation certificate functions.

6.6.1.2 Boundedness Event Processing

The boundedness event processing amounts to intersecting the polyhedron with V. For instance, if the event is not degenerate, meaning that a single vertex of the polyhedron collides with a single facet of V, the topological modification is to cut the failing vertex by the bounding plane. Since the vertex is trihedral, this produces a triangle supported by a plane of the bounding volume. Second, when an edge is colliding with a single plane, the edge would have to be cut out to form a quadrilateron. Third, a colliding facet does not require any topological change, but only a trajectory update of its supporting plane, which is set fixed to the supporting plane of the bounding volume. Arbitrary types of collisions should be processed likewise.

6.6.1.3 Complexity Analysis

The proposed handling of diverging vertices has the following KDS properties (cf. section 6.5.1):

Compacity: Handling diverging vertices requires only one Boundedness event per vertex and per bounding plane. This is linear in the size of the maintained polyhedron if the number of supporting planes in V is treated as a constant.

Efficiency: The inside events are all external. Any failing event requires a topological modification or at least a trajectory update of a plane.

Responsiveness: Processing an inside event takes time proportional to the degree of the vertex in the triangulated polyhedron.

Locality: A modification of the plane equation of a single facet of V requires the computation of an event per vertex. Modifying a plane trajectory only invalidates the inside certificates of its incident vertices.

6.6.2 Dealing with Global Self Intersections

The proposed approach only guarantees that the polyhedron facets are not auto intersecting, but it does not guarantee that the polyhedron as a whole is not auto intersecting, and thus defines

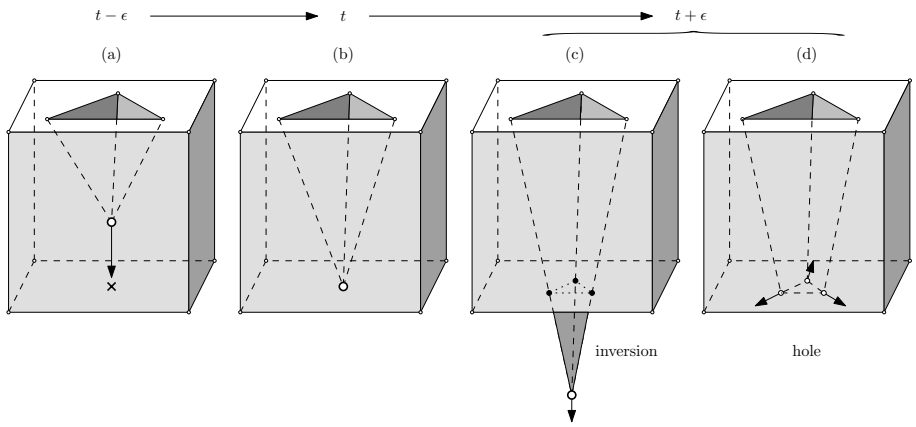


Figure 6.17: Genus increasing event. (a) A moving vertex of a simple polyhedron collides (b) at time t with a non-adjacent facet. (c) Without any topological change, an inversion occurs. (d) The topological modification required to maintain the simplicity of the polyhedron would be to allow the colliding vertex to cut a hole on the colliding facet, making the polyhedron homeomorph to a torus.

an interior and an exterior volume. Such intersections that have a more global nature are illustrated in figures 6.17 and 6.18. Discussing these non-local intersections requires the definition of the genus of a polyhedron and its shells.

Definition 40 (Genus)**.** *The genus may loosely be defined as the number of holes that penetrate the solid.*

Definition 41 (Shell)**.** *A shell is a maximal connected surface of the polyhedron.*

We previously assumed (sec. 3.2.1.2) that polyhedra were connected, meaning that they had a single shell. Lifting this assumption, a polyhedron can be decomposed into a disconnected set of connected polyhedral surfaces that model a set of volumes with piecewise planar boundaries, including solid holes inside them. This makes shells the 3D equivalent of the outer and inner loops of edges of a 2D polygon with holes.

Definition 42 (Euler-Poincaré Formula)**.**

$$V - E + F - H = 2(S - G)$$

where V, E, F, H, S and G are respectively the number of vertices, edges, facets, facet holes, shells and the genus.

The Euler-Poincaré formula [Seq] relates the number of combinatorial elements (V, E, F and H) of a polyhedron to the topology of its surface (S and G).

In the general case, maintaining a simple polyhedron would require to not only make local topological modifications (which leave S and G unchanged) but also to change its genus and to create or destroy shells (see figures 6.17 and 6.18). The topological modifications used in the proposed approach (edge collapses and edge flips during the event processing steps and vertex splits during the trihedralization steps) are local modifications that are genus-preserving, and are not creating or destroying any shell.

As the proposed kinetic data structure only relies on triangulations of each polyhedron facet, it assumes that the polyhedron surface remains connected and that it only requires genus-preserving topological modifications: there will be no need to create or remove handles, or to split a single shell polyhedron into a set of polyhedral shells or connect a set of polyhedral shells.

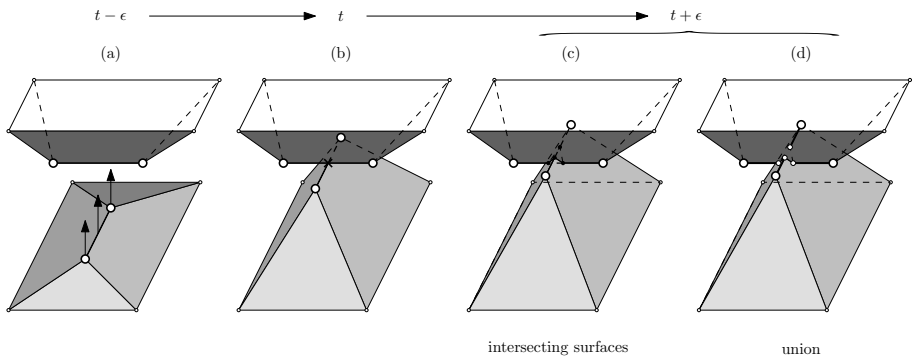


Figure 6.18: Connected component merging event. (a) 2 simple moving polyhedra are not intersecting at time $t - \epsilon$. (b) At time t, an edge of the top and one of the bottom polyhedron are colliding within the interior of their supporting segments. (c) Without any topological modification, the polyhedra are intersecting after the collision. (d) To keep the described surface manifold, the two polyhedra have to be connected: the resulting polyhedron is delimiting the union of the interior volume of the 2 input polyhedra.

Undetected Events The following geometrical events invalidate the polyhedron simplicity and yet are not detected by the proposed framework, because those events are not local on the surface, and thus cannot be detected by the facet triangulations:

- The collision between a vertex and the interior of a polyhedron facet, as illustrated in figure 6.17, creates a handle, if the vertex crosses the plane after the collision. If the facet and the vertex are parts of two distinct polyhedra, it merges them in a single polyhedron instead of creating the handle. At the collision time, the surface is no longer manifold at the colliding vertex, as it lies within two tangent surfaces: its surface prior to the collision and the colliding facet.
- The collision of two edges also breaks the polyhedron simplicity (figure 6.18). If the collision is not only a tangency and the surfaces neighboring the two edges intersect after the collision, then a genus-modifying topological operation is required to keep the polyhedron simple.
- Degenerate cases: The collision of a vertex with an edge or a point is a degenerate case of the collision with one adjacent facet. The collision of an edge with a facet is a degenerate case of the collision of either the edge with one edge adjacent to the facet, or one vertex of the edge with the facet (or both). The collision of 2 facets may even occur if two facets intersect within their instantly common supporting plane.

Non-manifold Trihedralization If a surface is not manifold at a vertex, the neighborhood of this vertex is locally equivalent to multiple disks that only intersect at the singular vertex, without considering the degenerate cases where the collision locus is larger than a single point. For instance, in figures 6.17 and 6.18, the neighborhoods of the singular vertices are locally equivalent to the union of two disks that intersect at the singular vertex.

Furthermore, the events that split polyhedral shells or decrease the genus are detected but are not currently handled by the proposed algorithm. These events could be illustrated using figures 6.17 and 6.18 by reversing the time evolution. The issue is that the surface is instantaneously not manifold at the vertex involved in the event.

The trihedralization step would have to be extended to handle non-manifold neighborhoods. The ear-cutting (sec. 5.5) and straight skeleton approaches (sec. 5.6) rely on the neighborhood being equivalent to a single disk and are thus not directly applicable to this more complex case. The winding number thresholding approach (sec. 5.2) and its extension as an arrangement coloring (sec. 5.3) should however work as is.

Detecting Global Intersection There are prospective alternatives to the proposed approach that are able to tackle the problem of preventing self-intersections on a moving polyhedron by detecting global self intersections rather than only local intersections between edges adjacent to a common face:

- A kinetic 3D plane arrangement could keep track of the 0-1 coloring directly. However the number of its internal events may slow down drastically the algorithm.
- A kinetic tetrahedralization constrained by the polyhedron would be compact and relatively efficient but there is no guarantee that a polyhedron can be partitioned into tetrahedra without introducing utility vertices [TVWZ93]. Dealing with this utility vertices may be tricky. The certificate functions would measure that no tetrahedron undergoes an inversion. The events will be more complicated and the degree of the certificate will be greater than the current planar orientation certificates.
- An extension of our KDS to deal with intersections between coplanar facets is trivial. Instead of maintaining separate constrained triangulations for each facet, there would be for each plane a constrained triangulation of the convex hull of all the vertices lying within this supporting plane. This treats all the coplanar facets together, rather than handling them individually. This would allow the detection of the global collision of vertices with any other vertex or edge defined by a common supporting plane.
- Another possibility is to schedule all the possible events (vertex-facet and edge-edge collisions are sufficient and allow to capture degenerate vertex-vertex, vertex-edge, edge-facet and facet-facet collisions) between all possible pairs without the help of a triangulation or a tetrahedralization. The algorithm would no longer be compact (linear in size) : the number of events would be quadratic. To keep the algorithm local and the certificates easily computable, the certificate would only test if the vertex collides with the supporting plane or if two supporting lines intersect. The point in polygon test or segment intersection test would thus be deferred to the processing of the event. This would yield a local kinetic data structure with relatively inexpensive certificate functions (the 4 by 4 determinant of the matrix of homogeneous coefficients of the 4 adjacent planes). However, many events would be internal (*i.e.* would not require any topological change).
- Finally, one may be able to detect global intersections more efficiently than by scheduling only vertex-facet and edge-edge events. The maintenance of a constrained triangulation of the convex hull of the facets within each supporting plane, allows an efficient detection of external events where the two colliding primitives share a common plane. These constrained triangulations may be used to answer efficiently the point in polygons query of the global intersection events.

6.6.3 Alternative Applications

This framework can be seen as a variant of [CSAD04] to perform variational shape approximation, that provides simplicity guarantees. Apart from solving the problem that motivated this thesis, we believe that this framework can have multiple alternative applications.

6.6.3.1 Planar Primitive-based Editing

This framework could also be applied to design an intuitive geometry editing tool. Geometry editing is typically performed by letting the user move the point location of the mesh vertices. The modification of facets, such as in Google Sketchup [Sketchup], is performed by an extrusion along the plane normal. The hypothesized tool would let a user split, merge and move freely planes as easily as moving and snapping points. The complexity of keeping the facets self-intersection free would completely be hidden to the user.

6.6.3.2 Weighted 3D Straight Skeleton and Offset Polyhedron

We saw in section 5.6.3 that the weighted straight skeleton of a polygon, defined in [EE99, AAAG95], is a special case of trihedralization. Likewise, a special application of this new framework is to compute straight skeletons of convex polyhedra. This 3D extension of the 2D straight skeleton of a polygon is similarly defined by construction as a shrinking process, by translating the facets inward at a constant speed. The plane equation mapping involved is then only a translation of the planes at equal velocity without rotations. This is simply the shrinking mapping : $[\vec{n} : d] \rightarrow [\vec{n} : d + t]$ if $\vec{n}$ is normalized and point outward. When the evolution has reached the time where the polyhedron is reduced to a single vertex, the loci of the edges spanned during the evolution describe the faces of a straight skeleton of the polyhedron. The offset polyhedra are the instantaneous polyhedra during the shrinking process. Similar to the generation of offset polygons from the straight skeleton of a 2D polygon, the straight skeleton of a 3D polyhedron can generate the offset polyhedra at various offsets.

The recent work of [BEGV08] has developed a similar kinetic approach specialized to this particular application that is also able to handle non-convex polyhedra. Contrary to their approach, the proposed algorithm presented here is only able to detect the local events, and is thus limited to convex polyhedra. Section 6.6.2 discusses the multiple possible extensions of the proposed algorithm to handle global events that are required for the computation of straight skeletons of non-convex polyhedra.

Nevertheless, the proposed approach leads the way to extend the Weighted Straight Skeleton in 3D. The simplest extension is to move the planes along their normals at heterogeneous speeds w_i. This only involves modifying the plane evolutions to incorporate the weights w_i: $[\vec{n} : d] \rightarrow [\vec{n} : d + w_i t]$. However these plane evolutions are fairly restricted, and the proposed approach would allow to explore 3D Straight Skeletons where the plane coordinates are arbitrary rational functions of time.

6.6.3.3 Polyhedron Generalization

One possible use of the offset polyhedra is to automatically generate polyhedra with lower Levels of Detail. Using the concepts of mathematical morphology, a dilation of the polyhedron can be computed using an outside polyhedron offset, and an erosion refers to an inside polyhedron offset. To remove small features, the standard approach is to apply a closing(*i.e.* a successive dilation and an erosion of the same magnitude) or an opening(*i.e.* the erosion is performed before the dilation).

The generalization of rectilinear buildings has already been studied in [May99, Kad02, Kad06]. These works handle buildings with plane evolutions of the type $\vec{N}(t) = [\vec{n} : d_0 - t]$ with $\vec{n} = (\pm 1, 0, 0)$, $(0, \pm 1, 0)$ or $(0, 0, \pm 1)$. The direct application of our KDS is able to generalize convex polyhedra, and the extensions discussed in section 6.6.2 would make the generalization of arbitrary sets of polyhedra possible.

6.6.3.4 Polyhedron Simplification

Defining the polyhedron geometry using its plane rather than its points make certain operations more intuitive. A state of the art approach to simplify a triangular mesh is through edge collapses [GH97]. Note that this approach does not intrinsically prevent self-intersection.

A dual approach would be to merge adjacent almost coplanar facets. In this case, the resulting movement of the vertex locations may be arbitrarily large, which makes the usually unsatisfactory self-intersections ubiquituous. The proposed KDS approach may be used to maintain a polyhedron with self-intersection free facets as quasi-coplanar facets are merged.

There would then be an alternative between merging facets 2 at a time using the kinetic

evolution until the polyhedron is sufficiently simplified, or clustering all the facets to perform a single evolution where each facet evolves from its initial plane support to the centroid plane of its cluster.

6.7 Conclusion

This chapter presented a kinetic method to evolve continuously a polyhedron which facet supporting planes undergo a specified evolution. This method ensures that the facets of the polyhedron do not self-intersect throughout the evolution.

Given the specific polyhedral building models that motivated this chapter, self-intersection free facets are sufficient to prove that the polyhedron does not self-intersect. However, extensions have been discussed that would be less efficient but would handle genus changes and further guarantee that the polyhedron itself is not self-intersecting, rather than only its facets.

Part IV

Evaluation

Chapter 7

Results of the 3D Building Model Refinement System

Contents

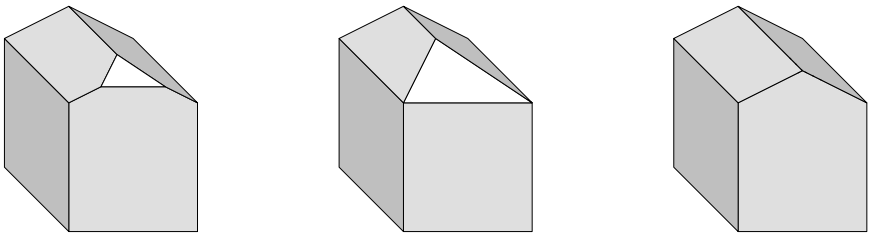
Our 3D building model refinement system has been tested on various contexts, from suburban residential areas to downtown centers. We provide here an evaluation of our approach on a challenging downtown area in Annecy, France which presents densely-packed, irregular and complex rooftops. Results on less challenging areas are not providing enough material to thoroughly discuss the limitations of our proposed approach. For instance, on typical suburban areas, all buildings are well separated, and the majority of buildings have only two roof planes and a small number of roof superstructures that are easily distinguishable based on the DSM only.

7.1 Input Data, Test Area

This area presents complex buildings in an extremely challenging context. On this area, a correlation-based DSM and an orthophotography (figure 7.1) have been generated from aerial images (GSD: 10cm).

Turning to the input 3D city model, we decided to depart from the scope of this thesis by not inputing a semi-automatic reconstruction but a fully automatic reconstruction from a manual 2D building footprint database. This choice has been driven by two reasons. First, it shows how the proposed approach may be used to fully automatically reconstruct 3D building models from their 2D footprint. Second, the semi-automatic building modeling process available at IGN, named Bati3D, is based on [DT06] and tends to favor simplified buildings, as this process is tuned for

Figure 7.1: Downtown area: Orthophotography and shaded DSM (Annecy, France)



(a) (b) Hipped roof (c) Gable roof

Figure 7.2: Small triangular facets and reconstructions from [DT06]: (a) Real-world building with a triangular facet cutting a gable roof. (b-c) Possible reconstructions from [DT06], with eaves constrained to same height. (c) is more likely to be output than (b) as the choice between models is based on the DSM, which tends to be closer to (c) than to (b) for DSM of buildings of type (a).

robustness. For instance, all their eaves have the same height, preventing the reconstruction of small triangular facets cutting the top of a gable roof (fig. 7.2). The lack of such facets may not be addressed by the fitting process as it never proposes complexifications of the topology. A straight skeleton-based reconstruction has been chosen as it provides one roof plane per eave edge, giving more degrees of freedom to the fitting process (at the cost of the reconstruction robustness).

Available 2D databases, as one extracted from a cadastral map, are not precise enough for our application and give poor results when used with the proposed system, as the proposed system does not fit façade planes (see section 4.6.1 for a discussion on lifting this limitation). Thus, the input 2D building footprint database has been input manually by an operator on an orthophotography, so that all the height discontinuities of the façades are input precisely (the input error is of the order of a few orthophotography pixels, *i.e.* ∼20cm). Note that, contrary to [LDZPD10], the 2D database of superstructures is not input manually and will be reconstructed by the proposed system.

The input 3D database has been constructed using the manual 2D input of the building footprints and the DSM. A straight skeleton-based reconstruction has been performed (*i.e.* the 3D model has a roof plane for each 2D segment, each roof plane have the same slope and each gutters have the same height - see section 5.6.1) and fitted to the DSM (to estimate the height and slope parameters).

Tests with a Bati3D-based input showed poor results as the densely-packed building blocks were overly generalized. Consequently, the model was robustly but poorly fitting the DSM relative to the height of a superstructure, yielding many erroneous superstructures. Furthermore, as the proposed roof fitting approach does not explicitly add or remove facets, the relatively few available roof facets were unable to correctly model the base roof planes. The manual 2D delineation of the height discontinuities (which are not addressed by our approach) and the straight skeleton reconstruction gives an input 3D dataset that precisely models the height discontinuities but provides a large number of superfluous planes that poorly fit the DSM.

This chapter evaluates on this area the results of the proposed system (section 7.2.2) relative to a manually input set of 3D line segments (section 7.2.1). These two datasets are only two approximating representations of the real world scene and do not model exactly the complexity of real-world buildings. Building reconstruction evaluation is still a complex and unresolved field of research [Bou07]. Comparing these two incomplete and approximating representations of the same scene will allow us to emphasize the limitations of our system as well as the tasks that are difficult even for a trained human operator.

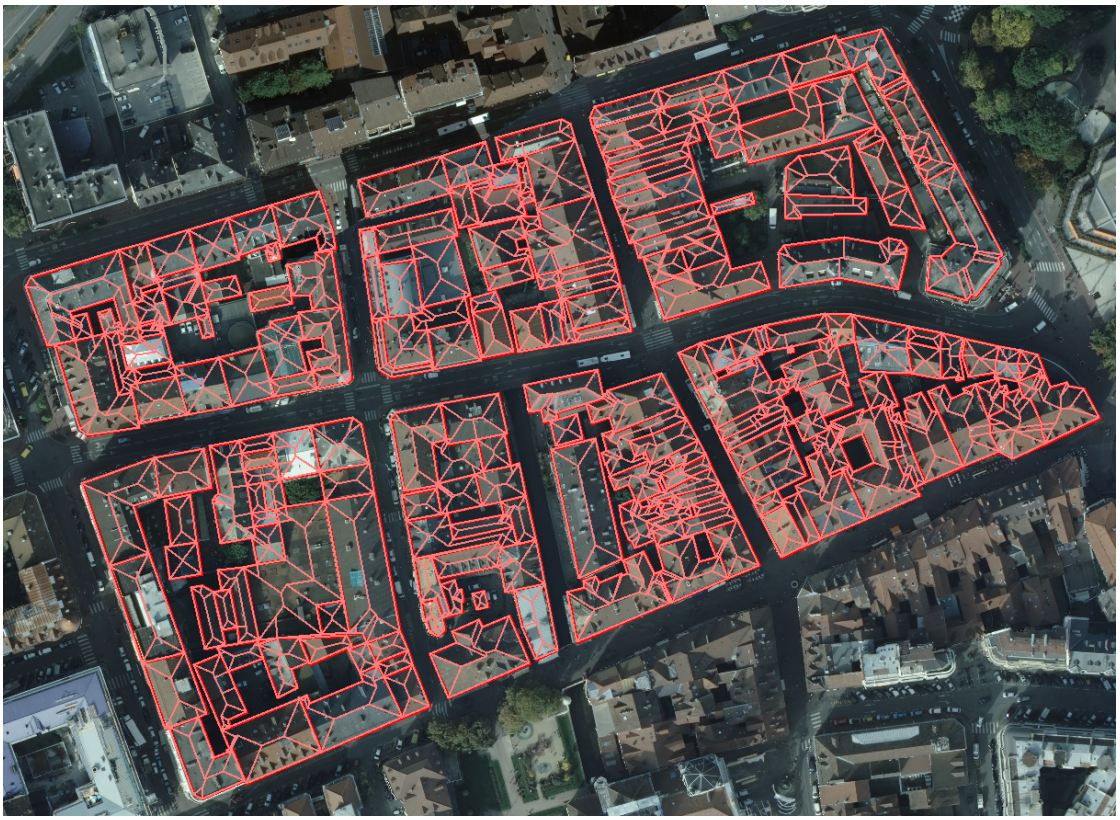


Figure 7.3: Straight Skeleton-based Reconstruction

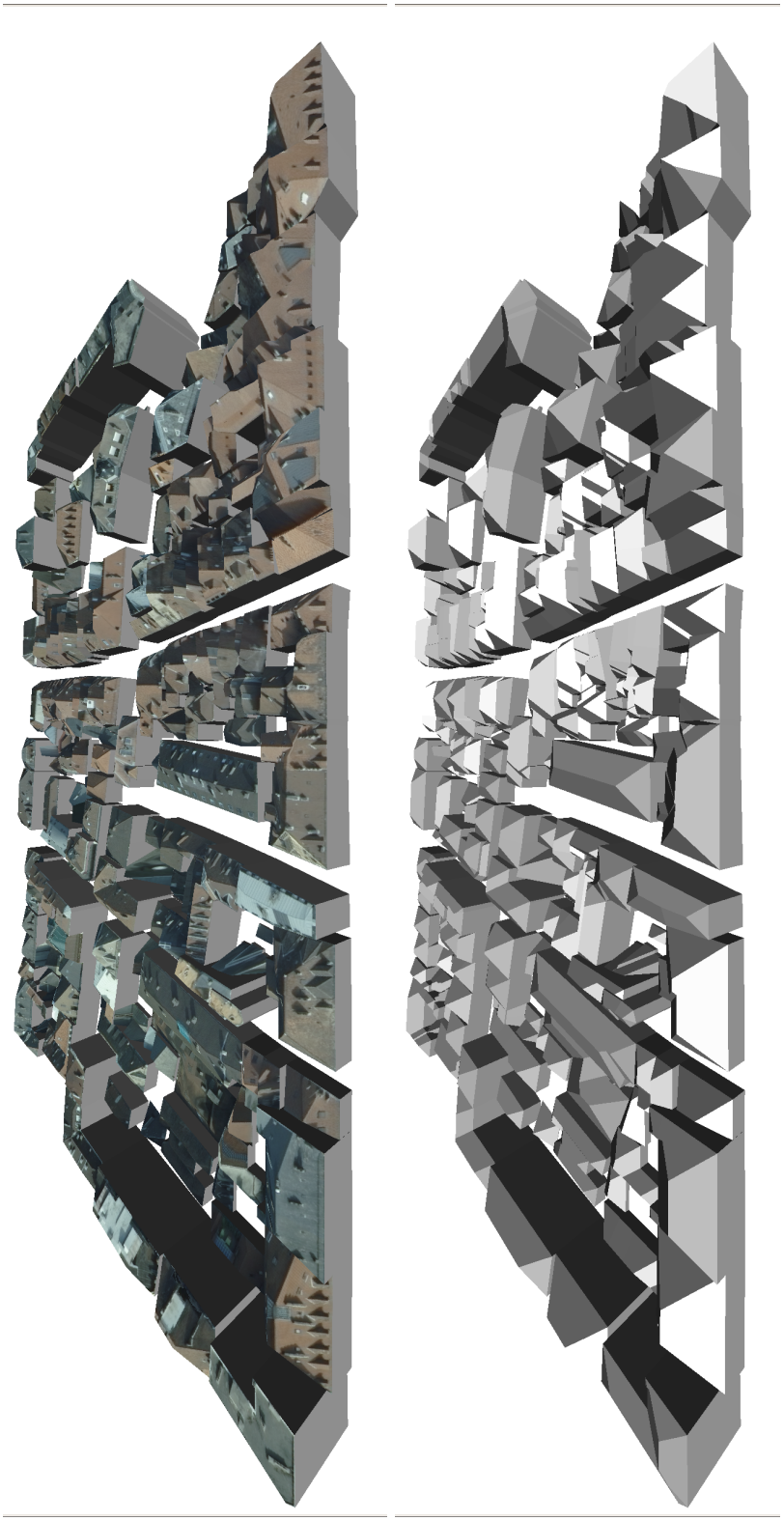


Figure 7.4: Straight Skeleton-based initial 3D building models (shaded and textured 3D view)

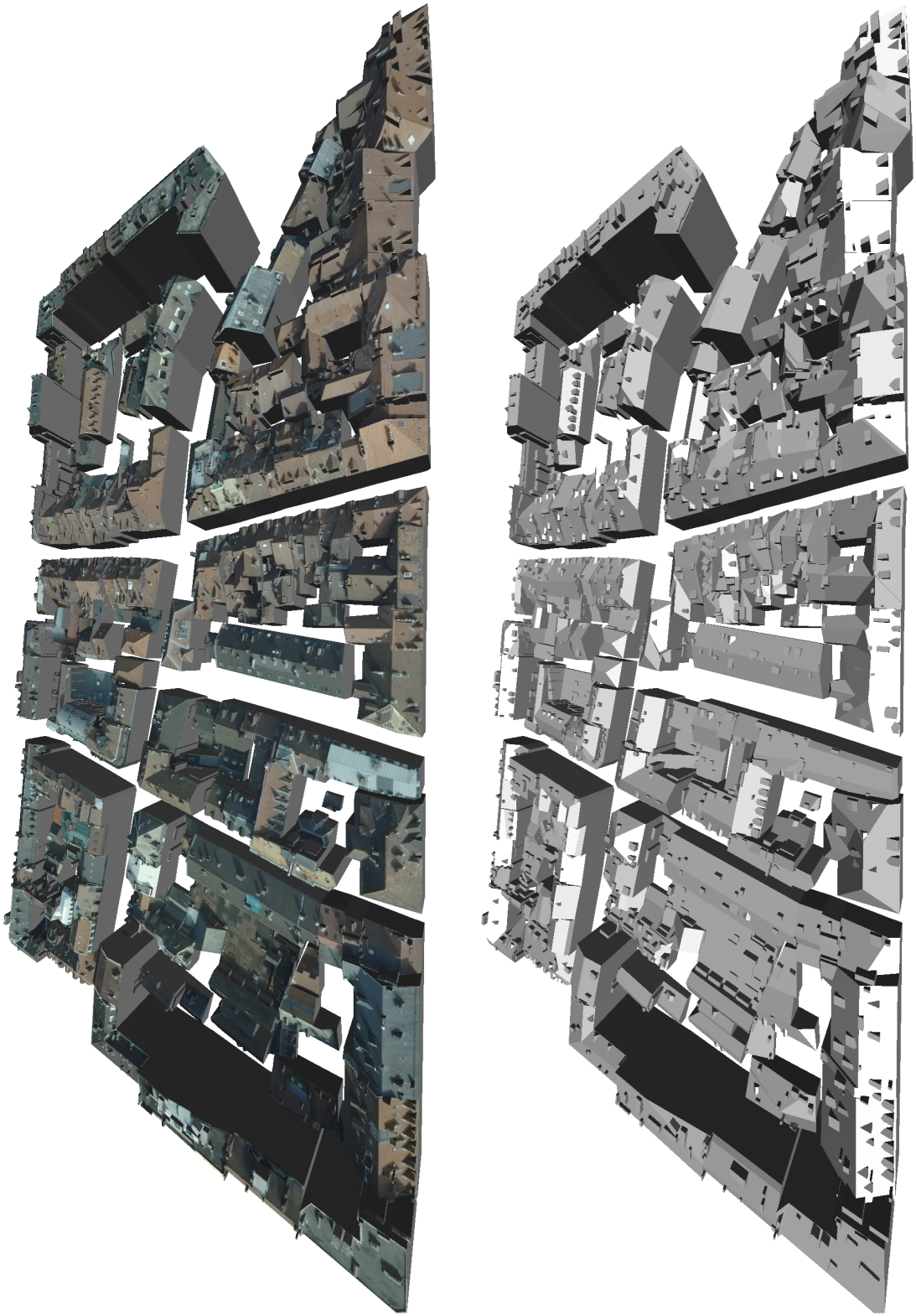


Figure 7.5: Fitted 3D building models with superstructures (shaded and textured 3D view)

7.2 Datasets

7.2.1 Reference Dataset

To evaluate the proposed approach, an experimented operator manually produced 3D line segments on stereoscopic views of aerial image pairs. This operator is highly trained for 3D stereoscopic

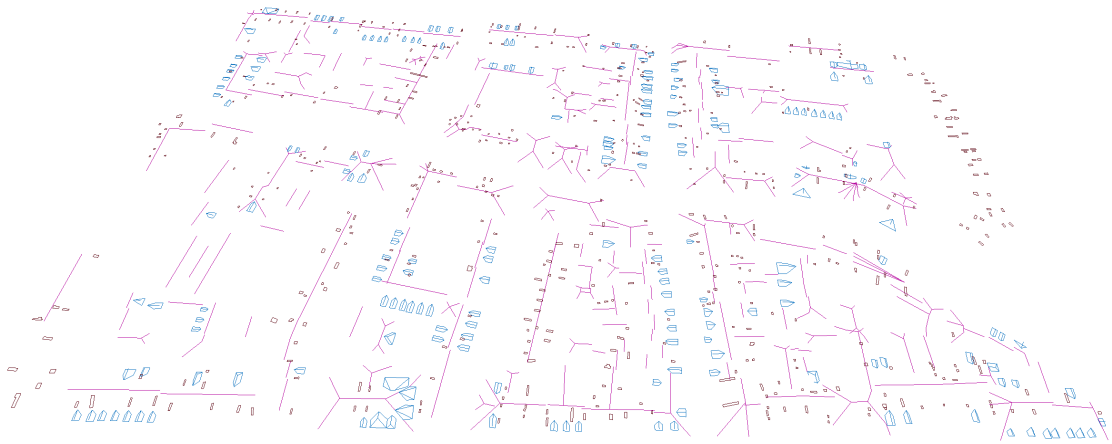


Figure 7.6: Ground truth of 3D line segments of interior roof edges and superstructure edges acquired by a trained operator (3D view).

data collection and produces this type of 3D datasets on a daily basis. Figure 7.6 illustrates this manually acquired dataset. The operator was asked to input the following roof and superstructure edges:

Roof All interior roof edges: ridges, hips and valleys but no eaves. More geometrically this corresponds to roof edges at the boundary of two non-vertical surfaces.

Chimney The top surface of the chimney.

Dormers The boundary of the 2 or 3 roof facets of the dormer superstructure.

This manual dataset is considered as a reference dataset, to which our approach will be compared. It represents one day of work and provides no further information than line segments being grouped as 3D polylines or non-necessarily planar 3D polygons tagged as *roof*, *chimney* or *dormer*. We stress the fact that this dataset is topologically very poor. Only edges are collected and they are not grouped into facets. Even if they were, they would not form planar surfaces, as their vertices have been input without any planarity constraint.

The only other type of superstructures present in this area is glass roofs (roof terraces are not present in this area). They are however indistinguishable in the DSM due either to a poor correlation in the images or to their negligible geometric offset relative to the roof plane. This shows a limitation of our system based only on a DSM: these glass roofs are clearly visible in the original aerial images.

7.2.2 Results of the Proposed System

Figure 7.5 shows the building blocks with fitted roof planes and superstructures reconstructed. We remind that the only manual input is the 2D building footprints and that all the following steps are fully automatic (straight skeletons and alternating roof fittings and superstructure reconstructions). Even with this lack of human intervention, the resulting 3D model is visually pleasing and seems to give a good approximation of these city blocks.

10 fitting iterations have been performed followed by a first superstructure reconstruction step, a second batch of 10 fittings iteration steps and then a final reconstruction step. Concerning parameters, the fitting step is parameter-less and the superstructure steps have been performed using the same set of parameters, a $\mathcal{L}_2$ error metric, and restricting the search to chimneys and dormer parametric objects.

7.3 Roof Fitting Evaluation

To evaluate quantitatively and qualitatively the roof fitting process, the set of interior roof line segments has been extracted from the reconstructed roofs. Comparing two sets of line segments is however not a trivial task. After a cautious inspection, a manual registration has been performed between these two sets. This registration has been performed by manually specifying 1-to-n or n-to-1 relationships between reconstructed and reference line segments. In particular, this is not a simpler 1-to-1 registration as segments of either dataset may be broken into subsegments.

As none of the two sets are a complete representation of the real world and both contain errors, both of the compared datasets contain unregistered line segments. 285 out of the 462 reference line segments have been manually registered to 298 out of the 862 line segments extracted from our reconstruction. This registration is stored as a set of 314 pairs of line segments, one from each dataset. The following section evaluates the geometric accuracy of these roof line segment pairs. In sections 7.3.2 and 7.3.3, the unregistered line segments will be classified according to a typology of errors.

7.3.1 Quantitative Evaluation of Registered Roof Line Segments

To assess the geometric accuracy of our reconstruction relative to the reference dataset, four error-measures have been applied to the set of pairs of registered line segment. For each of the four measures, figure 7.7 provides two histograms. The first histogram weighs equally all line segment pairs, whereas the second histogram takes the lengths of the segments into account such that the histogram is insensitive to the possible oversegmentation of one of the two line segments. This weighting is achieved using the product of the line segment lengths. This way, breaking one of the two line segments into subsegments does not change its contribution to the histogram. Generally speaking, the performance illustrated by unweighted histograms is worse than their weighted counterpart as they emphasize small line segments, which are more difficult to reconstruct accurately than longer line segments.

Translation errors are based on the vector $\vec{v}_{PS}$ which is the projection of the point P on the segment S. Thus $\|\vec{v}_{PS}\|$ is the distance between P and S, denoted $d(P,S)$. Furthermore, this distance $d(P,S)$ may be divided into a signed vertical component $d_V(P,S) = z_{\vec{v}_{PS}}$ and an unsigned horizontal component $d_H(P,S) = \sqrt{x^2_{\vec{v}_{PS}} + y^2_{\vec{v}_{PS}}}$.

The horizontal error measures how the two segments differ horizontally. It is built on the asymmetric error distance $d_H(S_0, center(S_1))$ that measures the 2D distance between a line segment S_0 and the center point of the line segment S_1 (discarding its z-coordinate). This asymmetric measure is symmetrized by averaging and exchanging the roles of the two segments:

$$\frac{d_H(S_0, center(S_1)) + d_H(S_1, center(S_0))}{2}$$

The vertical error is similarly computed by symmetrizing the signed distance $d_V(S_0, center(S_1))$:

$$\frac{d_V(S_0, center(S_1)) - d_V(S_1, center(S_0))}{2}$$

There appears to be a systematic vertical bias of 10cm between the two datasets. This bias may come from a error in the handling of the various conventions to geolocalize the images, the DSM and the datasets, as they were used different softwares. However, Figure 7.8 illustrates a more significant error. The reconstructed segment (red) seems correct when projected on the aerial images, whereas the reference segment is approximatively 45cm below. This long segment thus

100%	515	false positives
20.58%	106	Superstructures
19.03%	98	Planar
17.67%	91	Omitted
14.76%	76	Overfit
12.04%	62	DSM
09.32%	48	Multiple detection
06.60%	34	Missing plane

Table 7.1: Classification of overdetected roof line segments.

produces a spike in the weighted histogram which is not noticeable in the unweighted histogram. This kind of operator bias is not surprising and vary from operator to operator. In particular, it was foreseeable to find a vertical accuracy below the horizontal accuracy as the operators collected segments by first inspecting the stereoscopic image pair with the most vertical image axes, and checking other image views only in ambiguous cases.

The azimuth and slope errors measure the angular disparity between two segments. If the line segment direction is parameterized using spherical coordinates (θ, ϕ), the azimuth error coresponds to the difference $\Delta\theta$ and the slope error to the difference $\Delta\phi$. The histograms (c-d) of figure 7.7 clearly show the sensitivity of the direction estimation with respect to the segment length, as the weighted histograms are more tightly packed around the optimal value of 0 degrees than the unweighted histograms.

7.3.2 Typology of False Positive Errors

Among the 564 line segments that may be extracted from the reconstructed buildings but that may not be paired to a reference line segment, we classified the 515 line segments that are longer than 50cm. This filtering allows to discard small line segments that are not semantically meaningful and for which the (admittedly subjective) explanation of the error is not straightforward. This introduces a third understanding of the scene beyond the one provided by the 2 compared datasets, which is our understanding of the scene (based on its DSM, its orthophotography and its aerial images), and the algorithms involved. Object of this third understanding of the scene are hereafter denoted as *real* or *real-world* objects.

Table 7.1 provides the classification of the false positives (*i.e.* overdetected line segments) according to the following typology of errors, provided here by decreasing order of occurrence:

Superstructures: These line segments are due to superstructures biasing the roof fitting step. This may be a due, for instance, by a row of close dormer windows, which is approximated by a single roof plane.

Planar: These line segments are at the interface of almost coplanar roof facets. In section 4.6.2, we mentioned the merging of these almost coplanar facets.

Omitted line segments correspond to real roof edges according to the aerial images. They however show up as false positives as the operator omitted to input these line segments (the reference dataset is not complete).

Overfit Initializing the 3D building models with a straight skeleton-based reconstruction provides many planes that may used by the fitting process. Without the extension discussed in section 4.6.2, no explicit topological modification is performed. Thus, if the input DSM is noisy or does not convey clearly the piecewise-linear nature of the base roof surface, the many input planes may be kept and thus overfit the DSM.

DSM In this case, the input DSM shows an artifact that is reconstructed as a part of the polyhedral roof surface. This mainly occurs near vertical discontinuities that tend to be smoothed out

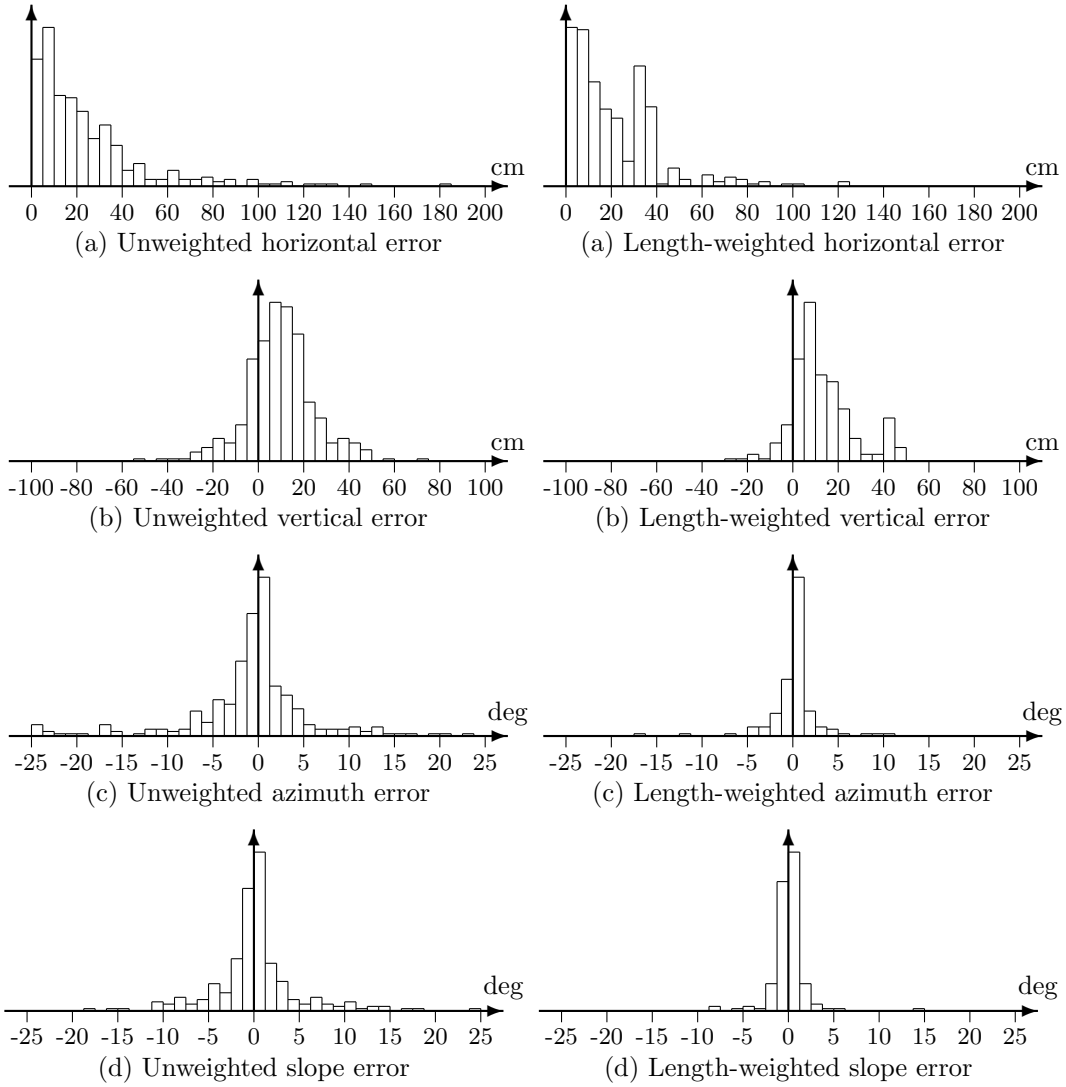


Figure 7.7: The accuracy of roof lines is evaluated through histograms of errors in position (a, b) and direction (c, d). In the first column, each pair of registered line segments contribute equally, whereas in the second column, each pair is weighted by the product of the line segment lengths, which makes this column of histograms invariant to line segments broken into subsegments.

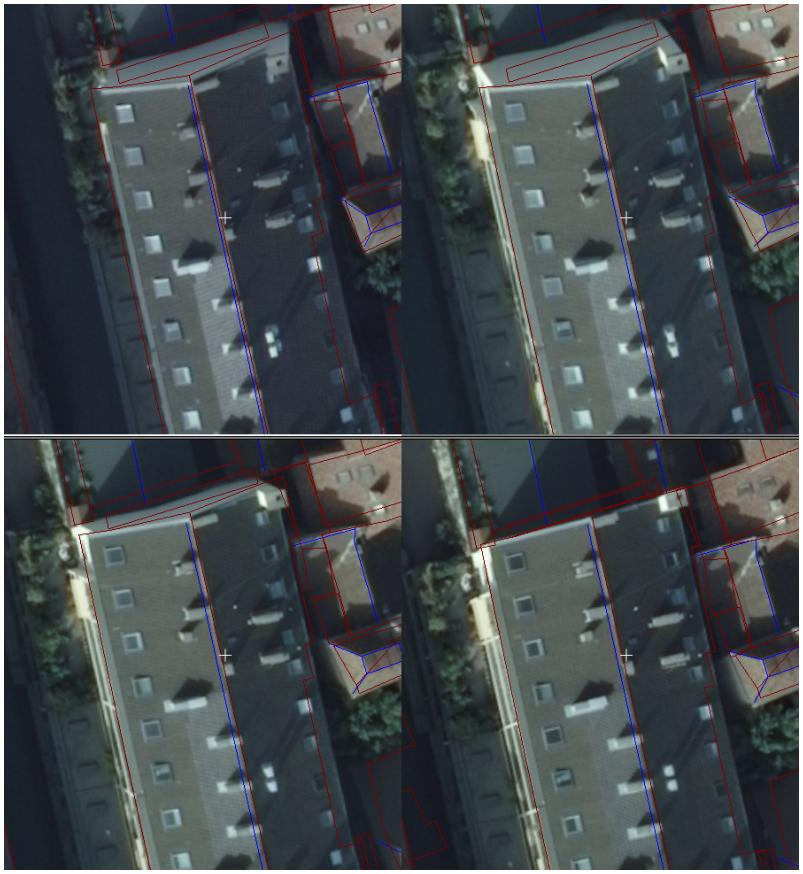


Figure 7.8: Operator bias: the result of the roof fitting (red) appears more accurate than the 3D line segment captured by the operator (blue), when superimposed on the aerial images.

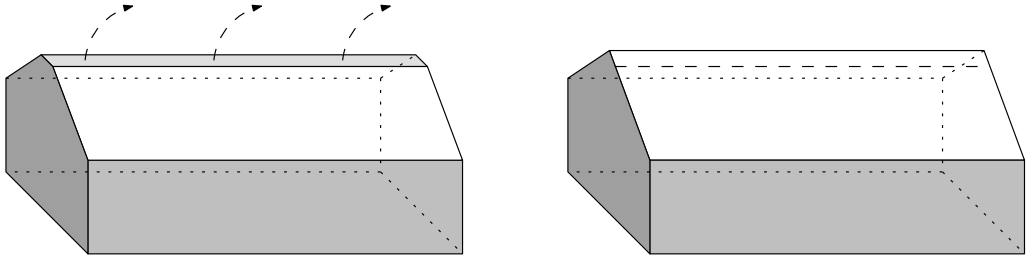


Figure 7.9: Multiple edge detection due to an unsimplified elongated facet. Rotating this facet around one of its long edges to merge it with the neighboring facet is a possibility to handle this overdetection.

by the DSM generation process.

Multiple detection This is a special case of overfit that appears to be easier to handle than the general case. In this class of erroneous segments, two nearly parallel reconstructed segments model a single reference segment. The best of the two segments has been paired with the reference segment and included in the evaluation of the previous section. This pair of parallel reconstructed edges is due to an elongated facet cutting a roof edge (figure 7.9-left). Similar to the **Planar** case, an explicit topological modification may be introduced to merge the elongated facet to one of its adjacent facet, without significantly modifying the described polyhedral surface (figure 7.9).

Missing plane These line segments are present to guarantee the topological correctness of the polyhedral roof surface. They are however only accessory as the reference roof surface uses an extra plane that is missing in the reconstructed building.

7.3.3 Typology of True Negative Errors

Turning to underdetections, the 176 reference line segments longer than 50cm that may not be registered have been classified (table 7.2) according to a similar typology of errors:

Missing plane: If a plane is missing in the reconstructed building relative to the reference building, its edges will be missing too.

Superstructures: This class groups all underdetected segments due to the presence of superstructures that prevent a relevant understanding of the real roof surface.

Out of Specification: These line segments have been input by the operator but are out of the specification. They mainly correspond to edges adjacent to facades such as eaves or gables.

Overfit: Buildings featuring **Overfit** overdetected line segments may also be subject to underdetections.

Planar: These line segments are neighboring almost coplanar facets in the real world and are thus very hard to detect and reconstruct based on the DSM only. The reference line segment is however input by the operator based on the inspection of aerial images, and thus their small dihedral is not preventing their reconstruction. This is a example of an ambiguous building reconstruction problem. Depending on the application, these **Planar** segments may or may not have to be modeled, whether the application needs simpler buildings or needs the semantic knowledge added by these **Planar** segments.

DSM: Similar to the overdetected roof line segments, underdetected roof line segments may be due to defects in the DSM.

Concerning both over- and underdetections, using Bati3D-produced building models rather than straight skeleton based building models would have resulted in more **Missing plane** and **Roof** errors and less **Multiple detection**, **Superstructures** and **Overfit** errors, as these model are topologically simpler, more robust but more generalized (not modeling exactly the DSM surface).

100%	176	True negatives
39.20%	69	Missing plane
34.09%	60	Superstructures
15.91%	28	Out of Specification
05.11%	9	Overfit
03.41%	6	Planar
02.27%	4	DSM

Table 7.2: Classification of underdetected roof line segments.

	reference chimney	reference dormer	false positives
reconstructed chimney	122	27	829
reconstructed dormer	120	160	167
true negatives	272	13	

Table 7.3: Confusion matrix for superstructures.

7.4 Superstructure Reconstruction Evaluation

This section evaluates qualitatively and quantitatively the detection and reconstruction accuracy of superstructures. Since two types of superstructures are present in the scene and in both datasets, the object-based correspondence between reconstructed and reference superstructures may be analyzed using a confusion matrix (table 7.3).

This table shows that reference dormer windows are clearly more robustly detected than the reference chimneys, which is understandable since dormer windows are larger structures that are easier to detect and reconstruct than small chimneys.

There also appears to be a noticeable confusion with 120 chimneys being detected as dormer windows. Inspecting the images, these chimneys are rather long, are large enough on the DSM to be a dormer window, have a relatively low height above the roof and their largest dimension is oriented along the steepest direction of the roof. Thus, this confusion is due to a limitation of the proposed system that considers only the DSM and not the original images.

7.4.1 Dormer Evaluation

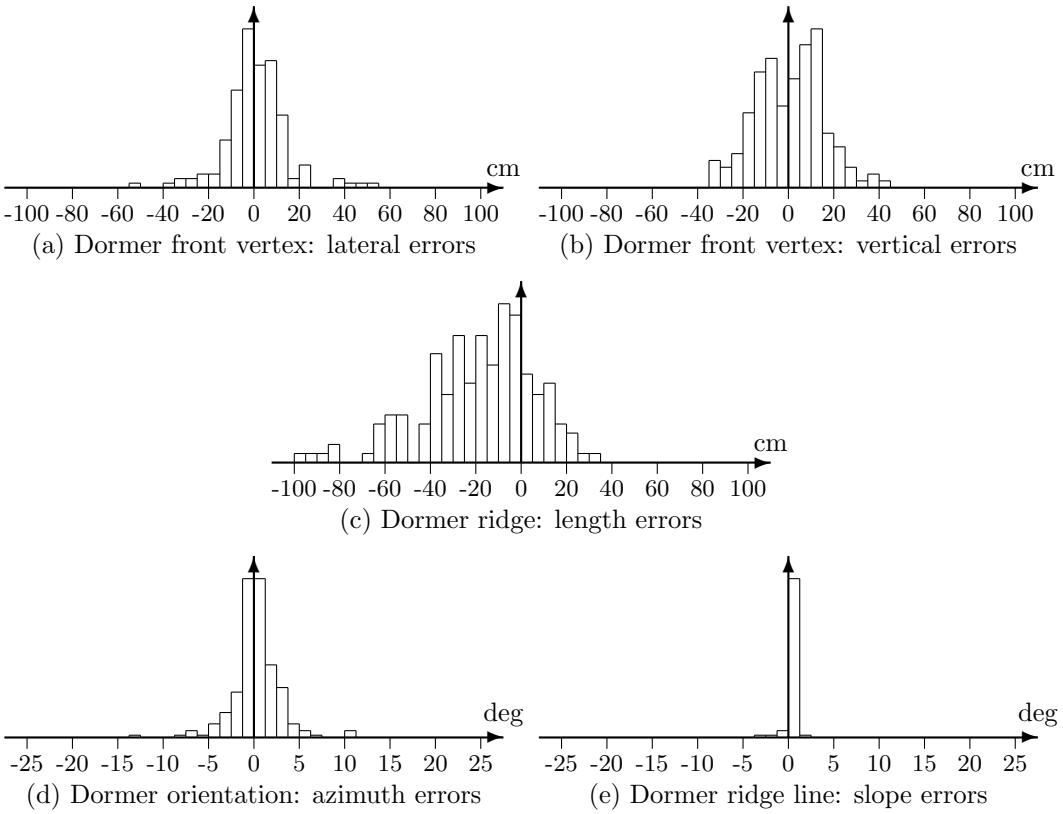
7.4.1.1 Quantitative Evaluation of Registered Dormers

The reconstruction accuracy of the 160 successfully reconstructed dormer windows is evaluated using five error measures that, for simplicity, only consider the top edge of the dormer window (denoted as its ridge).

The translation error is evaluated using the position of the front vertex of the reconstructed dormer window relative to the front vertex of the reference dormer window. More precisely, the lateral error (across the reference dormer orientation) and the vertical error are illustrated in table 7.10.ab.

The dimension estimation error is evaluated using the length of the dormer ridge. The histogram 7.10.c shows a more important negative tail. This is due to a non-negligible number of dormer windows that are more longer than the maximal dimension of the brute-force dormer candidate enumeration. It is however arguable, considering their sizes, whether these dormer windows should be considered as roof superstructures or regular roof facets.

The orientation errors are evaluated using the same $\Delta\theta$ and $\Delta\phi$ angular differences as the roof edge orientation errors. Orientation errors are at most of the order of a few degrees, which proves, considering the size of the dormer window objects, validates the discretization of the dormer



(a) Dormer front vertex: lateral errors

(b) Dormer front vertex: vertical errors

(c) Dormer ridge: length errors

(d) Dormer orientation: azimuth errors

(e) Dormer ridge line: slope errors

Figure 7.10: Accuracy evaluation of dormers.

100%	167	False positives
55.09%	92	Roof
23.95%	40	DSM
20.36%	34	Omitted
00.60%	1	Multiple detection

Table 7.4: Classification of overdetected dormers.

orientations in the proposed approach.

The slope error is particularly low: almost all dormers of the considered area have horizontal ridges that are well estimated. This histogram is asymmetric because the reconstructed slope was constrained to be negative, which explains the bias.

7.4.1.2 Typology of False Positive Errors

The 167 overdetected dormer windows are due to (table 7.4):

Roof: A poor estimation of the main roof planes. For instance, 32 dormers have been reconstructed to correct the modeling error of a mansard roof represented by a single roof plane instead of two roof planes with differing slopes.

DSM: As with roof edge errors, a defect of the DSM might be interpreted as a dormer window.

Omitted: 34 dormer windows have been overlooked by the human operator or have been input as part of the base roof surface rather than roof superstructures.

100%	36	True negatives
72.22%	26	Roof
11.11%	4	Overlap
11.11%	4	DSM
05.56%	2	Other

Table 7.5: Classification of underdetected dormer windows.

Multiple detection A single dormer window is detected as two separate dormer windows due to a poor orientation prior. These two detected dormer windows are evaluated as a multiply overdetected dormer in this false positive error class, and a well reconstructed dormer window which has been included in the quantitative evaluation.

7.4.1.3 Typology of True Negative Errors

The 36 underdetected dormer windows are classified according to table 7.5. Noticeably, the 4 **Overlap**-tagged dormer windows have not been reconstructed due to one or more chimneys overlapping the dormer window, contradicting our non-overlap assumption in these few cases. The non-overlap constraint does not appear to be to too restrictive as it only prevents the reconstruction of these 4 dormer windows.

7.4.2 Chimney Evaluation

The reconstruction accuracy of chimneys is evaluated based on the 122 registered chimney pairs, using the following measures: **The chimney center** is the center of gravity of the top facet. Considering the chimney center location errors allows a simple evaluation of the translation error between reconstructed chimneys that feature a rectangular and horizontal top facet and the reference chimneys which are given by a non-planar 3D polygon representing its top facet. **The chimney area** is the area of the 2D support of the chimney superstructure

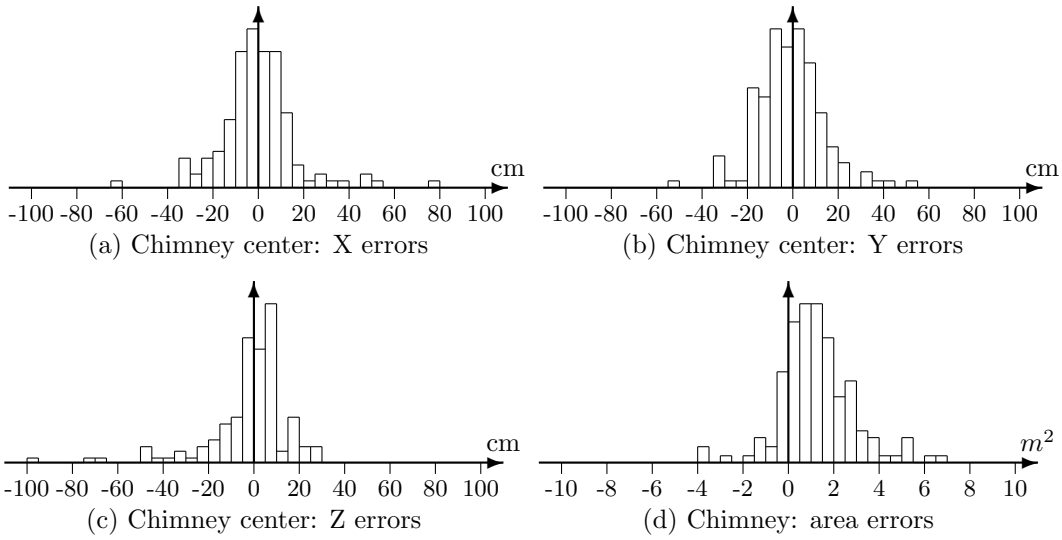
Histograms 7.11.abc show a good accuracy for the chimney center location, which is of the order of the DSM sample distance (10cm). However, histograms 7.11.d show that the reconstructed chimneys tend to be overestimated by $1m^2$ on average. This is due to the difficulty for the DSM to model the vertical discontinuities of the chimney facades. Thus the DSM tends to fatten chimneys and thus mislead their area estimation in the DSM-only proposed approach. Reconstructed chimneys tend to be larger but also slightly lower than the real chimney due to the overestimated support. This however does not show up in histogram 7.11.c. A possible explanation is that this effect is balanced by the vertical operator bias discussed in section 7.3.1.

Chimney detection errors are more numerous (829 overdetections and 272 underdetections) and less easy to classify than roof edge or dormer detection errors. However there are three unambiguous major causes of chimney detection errors:

DSM: The overwhelming case of overdetections is due to DSM artifacts at the boundary of the roof of a building neighboring a higher building. The lack of sharpness of the DSM is then interpreted as a row of chimneys along the boundary. Turning to underdetections, small chimneys where the correlation between images is not coherent due for instance to distracting objects like antennae may not be modeled in the DSM due to its necessary regularization. These small chimneys do not appear in the DSM and are thus not reconstructible using the proposed approach.

Roof-geometry: A poor fit of the roof triggers the reconstruction of chimneys to reconstruct a surface close to the DSM if the poorly fit roof is below the DSM.

Roof-topology: Chimney overdetections may also be caused by a reconstructed roof edge between almost coplanar roof facets. These **Planar** roof edges may prevent the correct reconstruction



(a) Chimney center: X errors (b) Chimney center: Y errors

(c) Chimney center: Z errors (d) Chimney: area errors

Figure 7.11: Accuracy evaluation of chimneys.

of a dormer window are they are not allowed to span multiple roof facets in the proposed implementation. Therefore, their backs are correctly reconstructed, but truncated by at the **Planar** roof edges. The front part of the truncated dormer windows is then generally erroneously reconstructed as a chimney.

7.5 Conclusion

This chapter presented and evaluated the fully automatic reconstruction of building with superstructures from a manual 2D database and an input DSM using the proposed building refinement system on a challenging downtown area in Annecy, France. The reconstruction accuracy of successfully reconstructed roof edges, chimneys and dormer windows have been evaluated quantitatively and their various classes of under- and overdetection have been identified and commented. The reference dataset input by a human operator illustrated the difficulty of our problem as it is itself not complete nor unbiased.

We discussed how the explicit topological modification extension mentioned may help the roof fitting step by reducing **Overfit**, **Planar** and **Missing plane**-typed roof errors. The proposed approach is based only on a DSM and does not inspect directly the input aerial images. To reduce the **DSM**-classified errors, future work will have to handle directly these aerial images. The high rate of **Roof**-caused superstructure reconstruction errors and **Superstructure**-caused roof fitting errors show the fragility of the proposed approach based on alternating roof optimizations and superstructure reconstructions independently. Together with the ambiguous classification of superstructures like large dormer windows as superstructures or roof parts, this leads to the need to consider these two problems simultaneously to achieve better results.

Chapter 8

Conclusion

Before concluding this manuscript, we review the contributions of the proposed approach and highlight its main limitations, which leads to mentioning some of its possible extensions.

8.1 Main Contributions

Superstructure Detection and Reconstruction This work is the first to focus on the detection and reconstruction of roof superstructures. Our most thematic contribution is a parametric approach that is efficient when the underlying roof is perfectly modeled. The idea of using the understanding of reconstructed superstructures to filter out the bias they caused in the main roof plane coordinate estimations is also novel. Moreover, although only a few superstructure types have been introduced here, the library is easily extensible.

Full automation This system is fully automatic. Since it does not rely on any user interaction, its operating cost is extremely low. On the flip side, this lack of user control impacts directly its robustness, since it does not rely on any human understanding beyond the library of parametric superstructures.

Hybrid 3D Building Model - Semantics The proposed hybrid building representation labels roof surfaces as main roof planes, chimneys, dormers... This provides useful semantics to a number of higher level thematic applications, and readily provides two levels of detail : the building representations with and without superstructures.

Hybrid 3D Building Model - Dual geometry-parameterized Base Polyhedron We claim that parameterizing the polyhedral base building using its plane coordinates rather than its point coordinates is well-suited to the building modeling application as it takes the best of both worlds : the genericity and direct access to the represented surface of the point-parameterized boundary representation, and the structural expressivity of Constructive Solid Geometry representation.

Trihedralization Problem Formalization We formalized the new problem of trihedralizing a plane-parameterized polyhedron with over-constrained vertices. To this end, relevant definitions were introduced (vertex zones, local arrangements...) and various simplifying assumptions were discussed : the decomposability assumptions that breaks the polyhedron trihedralization problem into independent vertex trihedralization subproblems, and the locality assumption that a vertex tri-

hedralization subproblem only depends on the planes adjacent to the over-constrained vertex. Four trihedralization algorithms, applicable under increasingly restrictive assumptions, were presented :

1. The first trihedralization variant is always applicable and provides a unique trihedralization solution by reduction to an existing problem based on winding numbers.
2. The arrangement coloring based variant is able to enumerate all the valid and invalid topologies so that the filtered list of all valid topologies may be reported. Simplifications of this algorithm due to the decomposability assumption were discussed.
3. We expressed a topologically minimal local vertex trihedralization solution as an abstract triangulation. This led to an implementation of the new ear-cutting based algorithm applicable to local and decomposable trihedralization problems.
4. Finally, if the vertex of a local vertex trihedralization problem is extremal, its trihedralization may be reduced to computing a weighted straight skeleton.

Kinetic Data Structure Maintaining a Polyhedron with Simple Facets This constitutes the first kinetic data structure that may not be rebuild instantaneously. This work thus explored how kinetic data structures may be used in contexts where multiple topologies are valid and the application needs a certain form of hysteresis to choose the topology that requires the least topological modifications (in an application-dependant sense that needs to be specified).

The proposed kinetic data structure maintains a polyhedron with self-intersection-free facets by modifying parsimoniously its topology, while its planes undergo a specified continuous evolution (translation or rotation). Using the projective geometry formalism has been primordial to the design of this algorithm, which relies on local vertex trihedralizations to perform the topological updates required to prevent self-intersections of facets.

Topology-aware Variational Optimization By combining superstructure-aware plane estimations and the proposed kinetic data structure (which is built on top of the trihedralization routine), we proposed a variational polyhedron optimization that maintains a polyhedron at all times while guaranteeing facets that are planar by construction and self-intersection free thanks to the kinetic data structure. Previous approaches failed to provide a polyhedron with such guarantees : they either keep a fixed topology or export their internal pseudo-topology as a post-processing step (pseudo-facets may have to be split into multiple facets to ensure planarity, self-intersections are rarely not accounted for).

8.2 Main Limitations

Representation Limitations The proposed representation is not able to model complex superstructure agglomerates or even simple overlapping superstructures such as chimneys partially or totally overlapping a dormer window. Furthermore, the classification between main roofs planes and superstructures is somewhat arbitrary and ambiguous for instance for large dormer windows that may arguably be considered as main roof planes.

Superstructures: Computing times The computing time of the superstructure candidate generation step grows (neglecting boundary influences) in $O(r^{-4})$ where r is the horizontal discretization distance. This computing requirements are reasonable for resolutions down to 10cm for typical buildings but explode at finer resolutions. For higher accuracy, solely decreasing the discretization distance will not be tractable. However, accounting for the size of the superstructures themselves, such a brute-force exploration is likely not to be needed. At a 10cm sampling, superstructures are already individualized and increased accuracy will be attained using extensions such as the ones discussed in section 8.3 or appendix B.

Another major pitfall is that when the base polyhedral roof is not a good enough approximation of the DSM, candidate superstructures are numerous and the computation time of the selection step is no longer negligible relative to the candidate generation step.

Kinetic Data Structure : Implementation Complexity The implementation of the proposed kinetic data structure is complex and error prone. Our implementation heavily relies on the [CGAL] library for its exact arithmetics capabilities and its generic kinetic data structure framework. At first glance, the addressed roof fitting problem does not seem to require such a complex non-interactive process. This heavy machinery seems however needed to design such a parameterless approach, with guarantees that may not be easily addressed by rule-based approaches such as [EAH08].

Robustness The validity of the simplification introduced by alternatively optimizing the roof superstructures and the roof plane geometry is our main simplifying assumption. Chapter 7 showed that this assumption is relevant in simple cases and in a number of more complex cases. However, it also proved that our approach is often not robust enough for more complex cases where the two problems are too interrelated.

8.3 Possible Extensions

Alternate Input Datasets Future work should definitely look into using directly the raw data (the images or the lidar point clouds) rather than only the preprocessed DSMs. Using DSMs was a convenient approach because of their regular sampling and their simple handling as a well-defined 2.5D quad-mesh surface. However, they may contain artefacts that are not present in the raw data. Moreover, using DSMs rather than any other data type is not a key part of this thesis, provided that an other input dataset provides a description of the world at a similar accuracy. Namely, chapter 5 and 6 are independent of the data used to estimate the roof planes, and chapter 3 and 4 discuss the opportunities to handle alternative input data.

Theoretical Study of the Trihedralization Problem The proposed trihedralization problem appears to be new and deserves a more thorough theoretical study beyond our results on the existence and lack of unicity of a solution. What is its asymptotic complexity? What is its relation with the non-planar 3D polygon triangulation problem?

Dealing with Improper Intersections A polyhedron may self-intersect without any self-intersecting facets. These improper intersections are not addressed in this work and, for instance, would be required in our context to correctly deal with building with overheads. Modifications will have to be made at the kinetic detection level, and during the topological trihedralization update, which will no longer be local. We discussed in section 6.6.2 a number of possible ways to detect such events. Then, being topologically equivalent to the problem of triangulating a polygon with holes, a non-local vertex trihedralization problem has to deal with multiple rings of adjacent planes. To find an abstract triangulation of the corresponding abstract polygon with holes, a possibility would be to generalize the weighted straight skeleton problem to spherical polygons with holes, which would represent the local cone of all the planes instantaneously meeting at the singular vertex.

Dealing with Diverging Vertices Besides improper intersections, an unconstrained plane estimation (compared to our estimation of top-facing planes only, with a $c > 0$ estimated plane) yields the problem of diverging vertices. We discussed in section 6.6.1 an extension to deal with such vertices using a bounding volume.

Improving the Numerical Scheme of the Variational Optimization The proposed approach uses a simple first order numerical scheme. It proved to not always stable (oscillating plane estimations instead of a steady convergence, or a poor plane estimate that breaks well-fitted areas). A higher order scheme is likely to lessen these problems.

Introducing Regularization, Joint Superstructure/Base-Roof Optimization The only forms of regularizations of the proposed approach are the verticality of façades, the regularization induced by the parametric superstructures and the quantification of the horizontal position, dimensions and orientation of the superstructures. Some applications may benefit from increasing the degree of regularization of the output model : aligned identical superstructures, symmetric roof slopes... This would require a modification of the minimized energy to value this regularity. Consequently the optimization algorithms would have to be adapted and would possibly have to jointly optimize both superstructures and the roof geometry. Minimizing a complex energy over a variable dimensional space with complex topological relations is an active subject of research. The theory of Reversible Jump Monte Carlo Markov Chain (RJ-MCMC), possibly combined with Jump-Diffusion techniques appear to be a good match. We are however concerned by the resulting computing times of such an approach.

Explicit Roof-Topology Updates The proposed variational optimization approach only modifies implicitly the roof topology. A simple extension to optimize the topology of the main roof planes is to check the estimation errors for poorly fitted planes for tentatively splitting them, such as with a RANSAC technique [FB81]. Conversely, almost coplanar facets may tentatively be merged with a single supporting plane. The proposed kinetic framework is readily able to handle these splits and merges. For instance, a merge may be implemented by a joint estimation of a single plane equation over the data corresponding to the facets to be merged. Then all these facets evolve to this common estimated geometry using the kinetic framework. Finally, the now-coplanar facets are topologically merged.

Alternative Applications The proposed kinetic data structure appears to be very versatile and may likely be used beyond the scope of this manuscript. Foreseen applications include mesh editing (the user moves or translates facets rather than vertices), an weighted extension of the 3D polyhedral straight skeletons and offset polyhedra[BEGV08], and mesh simplification. The main caveat here is computing time and improper intersections, as the computing times of the proposed implementation are not interactive and improper intersections are not handled. However, in the mesh editing application for instance, if one is only concerned about facet self-intersections and moves only a few planes at a time, interactive times may be achieved.

8.4 Conclusion

This thesis presented a successful system to refine an approximate building model according to a DSM. The refinement takes place both at the geometric level, minimizing the overall distance between the refined building and the DSM, and at the topological level, reconstructing the missing roof superstructures and correcting erroneous roof topologies through the topology-aware roof optimization of the fitting step.

Operationally, the superstructure reconstruction process is not robust enough because of the geometric and topological quality of the underlying base roof planes. The required precision of the base roof is the major bottleneck of the proposed approach. We argue that a more robust process will have to reconstruct superstructures from the raw image or lidar data, without relying on the knowledge of an approximate base roof, along the lines of [Nan06].

The topology-aware roof fitting process has been successfully integrated in IGN's Bati3D production suite. It is admittedly not yet suitable for interactive user interaction for all but simplest buildings. However, batch-fitting the roof off-line is a seriously considered option.

Part V

Appendices

Appendix A

$\mathcal{L}_2$ Estimation of $\vec{\phi}_{\max}$

Contents

This appendix provides the implementation details of the $\mathcal{L}_2$ estimation of $\vec{\phi}_{\max}$ that is sketched in section 3.4.1.2. The 3D building model used in this appendix and its notations are presented in section 3.2.

The need to optimize the altimetric parameters $\vec{\phi}$ occurs when a superstructure is hypothesized and everything apart from the vector $\vec{\phi}$ is determined: its planimetric position and dimension given by a 2D rectangle θ, its supporting roof $\mathcal{R}$, and its type τ. The hypothesized superstructure is fitted to the DSM by optimizing a quantity that measures how modifying the roof $\mathcal{R}$ with this superstructure make it closer to the DSM (section 3.4.1.1). The definition of $\vec{\phi}_{\max}$ for the $\mathcal{L}_2$-metric is:

$$\vec{\phi}_{\max} = \underset{\substack{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}} \\ s=(\vec{\phi},\theta,\tau,\mathcal{R})}}{\arg\min} \left(\sum_{\pi_s \cap \vec{p}_{DSM}(\mathbb{Z}^2)} (z_{DSM} - z_s)^2 - (z_{DSM} - z_{\mathcal{R}})^2 \right)$$

As in section 3.4.1.2, we only consider superstructure types for which the heightfield z_s that determines the geometry of the superstructure s can be written as an affine combination of $\vec{\phi}$-independent heightfields where the coefficients are scalars that do not depend on the position $\vec{p}$. This allowed section 3.4.1.2 to introduce an error vector field $\vec{e}_{\theta,\tau,\mathcal{R},DSM}(\vec{p}) : \mathbb{R}^2 \to \mathbb{R}^{d_\tau+1}$ to express the pointwise error between a superstructure $s = (\vec{\phi}, \theta, \tau, \mathcal{R})$ and a DSM as a dot product. Concerning the superstructure types proposed in chapter 3, the coefficients of the affine combination are simply the coordinates of the vector $\vec{\phi}$. Thus the error can be expressed as the dot product $(z_{DSM} - z_{\vec{\phi},\theta,\tau,\mathcal{R}})(\vec{p}) = \vec{e}_{\theta,\tau,\mathcal{R},DSM}(\vec{p}) \cdot \binom{1}{\vec{\phi}}$. The error vector fields of the proposed superstructure types are :

$$\begin{aligned}
\vec{e}_{\theta,Chimney,\mathcal{R},DSM}(\vec{p}) &= (z_{DSM}(\vec{p}), -1) \\
\vec{e}_{\theta,Terrace,\mathcal{R},DSM}(\vec{p}) &= (z_{DSM}(\vec{p}), -1) \\
\vec{e}_{\theta,GlassRoof,\mathcal{R},DSM}(\vec{p}) &= ((z_{DSM} - z_{\mathcal{R}})(\vec{p}), -1) \\
\vec{e}_{\theta,Dormer,\mathcal{R},DSM}(\vec{p}) &= \left((z_{DSM} - z_{\mathcal{R}})(\vec{p}), |\lambda_\theta(\vec{p})|, \frac{\mu_\theta(\vec{p}) - 1}{2}\right)
\end{aligned}$$

This finally allows the following reformulation of $\vec{\phi}_{\max}$:

$$\vec{\phi}_{\max} = \underset{\substack{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}} \\ s=(\vec{\phi},\theta,\tau,\mathcal{R})}}{\arg\min} \left(\sum_{\pi_s \cap \vec{p}_{DSM}(\mathbb{Z}^2)} \left(\vec{e}_{\theta,\tau,\mathcal{R},DSM} \cdot \begin{pmatrix} 1 \\ \vec{\phi} \end{pmatrix} \right)^2 - (z_{DSM} - z_{\mathcal{R}})^2 \right) \tag{A.0.1}$$

Disregarding the $\vec{\phi}$-varying term $D_{\pi_s}(\mathcal{R}) = \sum_{\pi_s \cap \vec{p}_{DSM}(\mathbb{Z}^2)} (z_{DSM} - z_{\mathcal{R}})^2$, the optimization of $\vec{\phi}_{\max}$ translates into a $\mathcal{L}_2$ minimization over a possibly $\vec{\phi}$-varying support $\pi_s \cap \vec{p}_{DSM}(\mathbb{Z}^2)$, which is constrained ($\vec{\phi} \in \Phi_{\theta,\tau,\mathcal{R}}$).

A.1 Constant Time Case

Exact $\mathcal{L}_2$ minimization can be run in constant time for a piecewise-linear superstructure (without curved facets) if its facet supports are rectangular and does not depend on $\vec{\phi}$, as it is the case for the *Chimney*, *GlassRoof* and *Terrace* superstructure types in the extensible library of superstructures proposed in section 3.2.2.

If the support π_s does not depend on $\vec{\phi}$ either, as it is the case with the proposed *Chimney*, *Terrace* and *GlassRoof* types of superstructures where $\pi_{(\vec{\phi},\theta,\tau,\mathcal{R})} = \theta$, the $D_{\pi_s}(\mathcal{R})$ term becomes constant with respect to $\vec{\phi}$ and the expression defining $\vec{\phi}_{\max}$ can be further simplified as:

$$\text{If } \pi_{(\vec{\phi},\theta,\tau,\mathcal{R})} = \theta, \qquad \vec{\phi}_{\max} = \underset{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}}}{\arg\min} \left(\sum_{\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)} \left(\vec{e}_{\theta,\tau,\mathcal{R},DSM} \cdot \begin{pmatrix} 1 \\ \vec{\phi} \end{pmatrix} \right)^2 \right)$$

Using the shorthand notation $e_k(\vec{p})$ for the coordinate $(k+1)$ of $\vec{e}_{\theta,\tau,\mathcal{R},DSM}(\vec{p})$, $\vec{\phi}_{\max}$ can be expressed directly:

$$\begin{aligned}
\vec{e}_{\theta,\tau,\mathcal{R},DSM}(\vec{p}) &= (e_k(\vec{p}))_{0\le k\le d_\tau} \\
\vec{\phi}_{\max} &= \underset{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}}}{\arg\min} \left(\sum_{\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)} \left(e_0 + \sum_{k=1}^{d_\tau} \phi_k . e_k \right)^2 \right) \\
\Rightarrow \vec{\phi}_{\max} \in Boundary(\Phi_{\theta,\tau,\mathcal{R}}) \quad &\text{or} \quad A \cdot \vec{\phi}_{\max} = B \\
\text{with a matrix } A &= \left(+ \sum_{\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)} e_k . e_l \right)_{1\le k,l\le d_\tau} \\
\text{and a vector } B &= \left(- \sum_{\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)} e_k . e_0 \right)_{1\le k\le d_\tau}
\end{aligned}$$

For the *Chimney*, *GlassRoof* and *Terrace* superstructure types, $d_\tau = 1$, so A and B are scalar values, and $\Phi_{\theta,\tau,\mathcal{R}}$ is either empty, a single point or a closed interval. For those types, $e_1(\vec{p}) = 1$, thus, $A = \sum_{\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)} 1^2 = \left|\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)\right|$. If $A = 0$ or $\Phi_{\theta,\tau,\mathcal{R}} = \emptyset$, no superstructure hypothesis is generated. Otherwise, the optimized function being convex, $\phi_{\max}$ is set to the projection of the scalar value $\frac{B}{A}$ onto $\Phi_{\theta,\tau,\mathcal{R}}$:

$$\text{If } d_\tau = 1,\ A \neq 0 \text{ and } \Phi_{\theta,\tau,\mathcal{R}} \neq \emptyset, \qquad \vec{\phi}_{\max} = \begin{cases} \frac{B}{A} & \text{if } \frac{B}{A} \in \Phi_{\theta,\tau,\mathcal{R}} \\ \min(\Phi_{\theta,\tau,\mathcal{R}}) & \text{if } \frac{B}{A} < \min(\Phi_{\theta,\tau,\mathcal{R}}) \\ \max(\Phi_{\theta,\tau,\mathcal{R}}) & \text{if } \frac{B}{A} > \max(\Phi_{\theta,\tau,\mathcal{R}}) \end{cases}$$

If $d_\tau > 1$, the optimal value $\vec{\phi}_{\max}$ may no longer be the projection of the unconstrained least square optimal value projected to the constrained set $\Phi_{\theta,\tau,\mathcal{R}}$.

$\frac{B}{A}$ corresponds to the mean DSM height $\frac{\sum_{\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)}(z_{DSM})}{|\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)|}$ for the *Chimney* and *Terrace* and to the mean DSM offset from the roof $\frac{\sum_{\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)}(z_{DSM} - z_{\mathcal{R}})}{|\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)|}$ for the *GlassRoof*.

Preprocessing Without preprocessing, the evaluation of A and B takes a computing time proportional to the discrete area of the support, defined as the number of enclosed DSM pixel centers $\left|\pi_s \cap \vec{p}_{DSM}(\mathbb{Z}^2)\right|$. This section presents a preprocessing that allows the computation of the elements of A and B in constant time for superstructures that are piecewise linear and have rectangular facet supports that do not depend on $\vec{\phi}$.

As described in section 3.2.2.3, each building orientation yields a grid of points $\vec{p}_{\theta_0}(\mathbb{Z}^2)$ that is a scaled, rotated and translated version of $\mathbb{Z}^2$, by the means of a reference rectangle θ_0. Any function $f : \mathbb{R}^2 \to \mathbb{R}$ can thus be resampled along such a grid into an image $\left(f^{\theta_0}(i,j)\right)_{ij}$, with a pixel (i,j) corresponding to the rectangle θ_{ij} of indices $(i, j, i+1, j+1)$ relative to θ_0:

$$f^{\theta_0}(i,j) = \sum_{\theta_{ij} \cap \vec{p}_{DSM}(\mathbb{Z}^2)} f$$

Then, for any rectangle θ that has indices (i_0, j_0, i_1, j_1) relative to θ_0, the computation of the sum of a function f over the finite set $\left(\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)\right)$ can be written as a sum of the resampled function f^{θ_0} over the indices $[i_0, i_1[\times[j_0, j_1[$:

$$\sum_{\theta \cap \vec{p}_{DSM}(\mathbb{Z}^2)} f = \sum_{\substack{i_0 \leq i < i_1 \\ j_0 \leq j < j_1}} f^{\theta_0}(i,j)$$

To compute this sum in constant time, a preprocessing step computes the cumulative function $C(f^{\theta_0})$, where i_{min} and j_{min} are the minimum indices of the subset of the set of candidate rectangles $\Theta_{\tau,\mathcal{R}}$ that are generated by θ_0.

$$C(f^{\theta_0})(i,j) = \sum_{\substack{i_{min} \leq k < i \\ j_{min} \leq l < j}} f^{\theta_0}(k,l)$$

Using this preprocessing, the sum of the values $f^{\theta_0}(i,j)$ over a rectangular set of indices $[i_0, i_1[\times[j_0, j_1[$ can be computed in constant time:

$$\sum_{\substack{i_0 \leq k < i_1 \\ j_0 \leq l < j_1}} f^{\theta_0}(k,l) = C(f^{\theta_0})(i_1, j_1) + C(f^{\theta_0})(i_0, j_0) - C(f^{\theta_0})(i_0, j_1) - C(f^{\theta_0})(i_1, j_0)$$

Concerning the *Chimney*, *GlassRoof* and *Terrace* superstructure types, to compute $\vec{\phi}_{\max}$ in constant time, we finally have to preprocess by resampling and accumulating two functions: $e_0 = z_{DSM}$ or $(z_{DSM} - z_{\mathcal{R}})$ on the one hand, and the constant function equal to $e_1 = -1$ on the other hand, to respectively be able to construct the scalars B and A in constant time. To also be able to compute the benefit $\Delta E(s)$ in constant time for *Chimney* and *Terrace* superstructures, one will have to preprocess the functions $(e_0)^2 = (z_{DSM} - z_{\mathcal{R}})^2$. This is not required for the *GlassRoof*s, because the term $(e_0)^2 - (z_{DSM} - z_{\mathcal{R}})^2 = 0$ can be canceled out of the benefit $\Delta E(s)$.

More generally, the superstructure is not restricted to have a single facet and a rectangular support, for this preprocessing to be used. If the support of each of the possibly many facets may be rewritten as a disjoint union of a bounded number of rectangles, the computation of A and B can still be carried out in constant time. All it requires is computing independently A and B on each rectangular elements of each facet and then adding all the contributions together.

A.2 θ-varying Error Fields

When an error field $e_i(\vec{p})$ of a superstructure depends on θ, the computation of A and B requires the computation of the sum over its facet supports of quantities that depend on θ. Thus the application of the preprocessing using a 2D accumulation is not direct. However if a θ-varying error field can be expressed as an affine combination of θ-independent heightfields with $\vec{p}$-independent coefficients, then those θ-independent heightfields may be preprocessed to build the A and B efficiently.

The λ_θ and μ_θ functions, that express the local coordinates of a 2D point $\vec{p}$ in the local frame defined by the rectangle θ, are linear combinations of the functions λ_{θ_0} and μ_{θ_0}, where the rectangle θ_0 has the same orientation as θ. Thus, within each facet of a superstructure, z_s is linear and the $(e_k.e_l)_{1\leq k,l\leq d_\tau}$ values can be expressed as a linear combination of functions $\lambda_{\theta_0}(\vec{p})$ and $\mu_{\theta_0}(\vec{p})$ that do not depend on the index (i_0, j_0, i_1, j_1) of θ relative to θ_0, with multiplying factors that are rational functions in (i_0, j_0, i_1, j_1) and do not depend on the point $\vec{p}$. The rational functions can easily be derived and are not included here to avoid their verbosity.

The *Dormer* superstructures are piecewise-linear but not linear, the absolute values around λ_θ are caused by the 2 non vertical facets of the superstructure (the one where $\lambda_\theta \leq 0$ and the other where $\lambda_\theta \geq 0$). This requires A and B to be computed separately for each facet support. For each of the two facet supports, the partial values of A and B can be computed as a linear combination of the summation of the following functions:

$$1, \quad \lambda_{\theta_0}, \quad \mu_{\theta_0}, \quad z, \quad z.\mu_{\theta_0}, \quad z.\lambda_{\theta_0}, \quad \lambda_{\theta_0}.\mu_{\theta_0}, \quad \lambda_{\theta_0}^2, \quad \mu_{\theta_0}^2, \text{ with } z = (z_{\mathcal{R}} - z_{DSM})$$

Similar to the case with the simpler superstructures, those functions will thus be preprocessed by resampling them according to θ_0 and then computing their accumulation.

A.3 $\vec{\phi}$-varying Supports

*Dormer*s have supports that depend on the vector $\vec{\phi}$. However, by the definition of $\Phi_{\theta,\tau,\mathcal{R}}$ for these superstructure types, a non-empty polygonal area, included in the rectangle θ, is guaranteed to be included in the support of the superstructure hypothesis if $\vec{\phi}$ is inside $\Phi_{\theta,\tau,\mathcal{R}}$. This minimal support is the intersection of all the plausible supports:

$$\bigcap_{\vec{\phi}\in\Phi_{\theta,\tau,\mathcal{R}}} \left(\pi_{(\vec{\phi},\theta,\tau,\mathcal{R})}\right)$$

Concerning the *Dormer* superstructure type, this polygon is the triangle $\{\vec{p} / -1 \leq \mu_\theta(\vec{p}) \leq (1 - 2\,|\lambda_\theta(\vec{p})|)\}$, as illustrated in figure A.1.

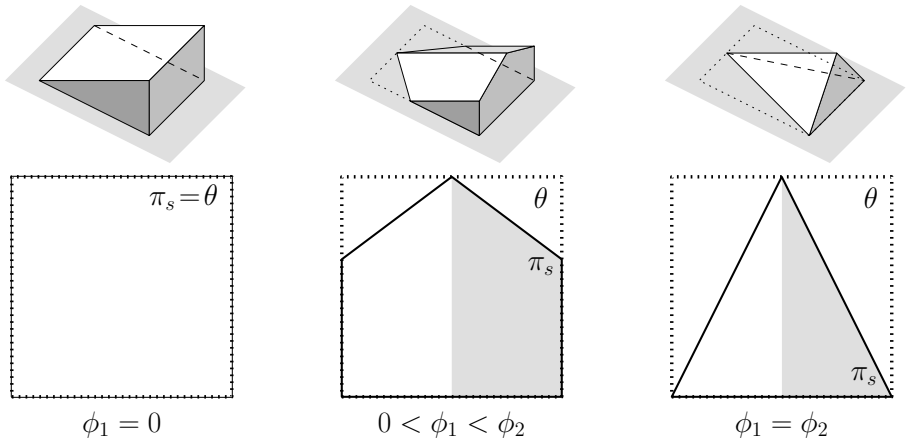


Figure A.1: The minimal support $\bigcap_{\vec{\phi} \in \Phi_{\theta,\tau,\mathcal{R}}} \left(\pi_{(\vec{\phi},\theta,\tau,\mathcal{R})}\right)$ of the *Dormer* superstructure type always contains the triangular support pictured occuring when $\phi_1 = \phi_2$.

To limit the computing costs, the proposed approach is to optimize $\vec{\phi}$ only over the minimal support, instead of the entire support π_s (that depends on $\vec{\phi}$ itself!) and to project the result on $\Phi_{\theta,\tau,\mathcal{R}}$, like in the simpler case of *Chimney*, *GlassRoof* and *Terrace* superstructures. The derivation of A and B is still valid, apart from the support being the minimal support instead of the rectangular support θ.

An alternative possiblity would be to estimate iteratively the vector $\vec{\phi}$, starting with an estimation over the minimal support, the rectangle θ or any other support and reestimating iteratively $\vec{\phi}$ using the support of the superstructure reconstructed by the previous $\vec{\phi}$ estimation.

A.4 Non-rectangular Supports

The computation of the $(d_\tau)^2$ values $(\sum e_k.e_l)$ of A and the d_τ values $(-\sum e_k.e_0)$ of B is no longer done in constant time for supports that are not the disjoint union of rectangles. However, some approximations allow an estimation that is linear with respect to the perimeter of the non-vertical superstructure facets. This section takes the *Dormer* superstructure type as an example of a piecewise planar superstructure with non rectangular facet supports

The facets of both types of dormer superstructures have non-rectangular supports. Thus it is no longer possible to use the 2 dimensional preprocessed accumulations to compute sums over the facet supports in constant time. Instead we propose to use one-dimensional accumulations to compute sums over a non-rectangular support in a time that is linear with the perimeter of the superstructure facets, instead of the direct implementation that has a computing time proportional to the support area. The idea is to partition the support using slabs $S_i = \{\vec{p}/\lambda_{\theta_0}(\vec{p}) \in [i, i+1[\}$ or $T_j = \{\vec{p}/\mu_{\theta_0}(\vec{p}) \in [j, j+1[\}$, depending on the smallest dimension. Since $S_i \cap T_j$ is the rectangle $\theta_{ij} = \vec{p}_{\theta_0}([i, i+1[\times[j, j+1[)$, we can derive, given a face support π:

$$\begin{aligned}
\sum_{\pi \cap \vec{p}_{DSM}(\mathbb{Z}^2)} f &= \sum_{i,j \in \mathbb{Z}} \sum_{(\theta_{ij} \cap \pi \cap \vec{p}_{DSM}(\mathbb{Z}^2))} f \\
&= \sum_{\substack{i,j \in \mathbb{Z} \\ \theta_{ij} \subseteq \pi}} f^{\theta_0}(i,j) + \sum_{\substack{i,j \in \mathbb{Z} \\ \theta_{ij} \not\subseteq \pi \\ \theta_{ij} \cap \pi \neq \emptyset}} \sum_{(\theta_{ij} \cap \pi \cap \vec{p}_{DSM}(\mathbb{Z}^2))} f
\end{aligned}$$

There are various ways to deal with rectangles θ_{ij} that overlap only partially with a facet support.

Exact computation: There is no obvious way to express more efficiently $\sum_{(\theta_{ij}\cap\pi\cap\vec{p}_{DSM}(\mathbb{Z}^2))} f$ when θ_{ij} and π overlap partially. However, only a number proportional to the perimeter of the support is needed rather than proportional to its area.

Rasterized computation: To get rid of those partial overlaps, the support π may be approximated by a union of rectangles $\pi \simeq \bigcup_{area(\theta_{ij}\cap\pi)\geq area(\theta_{ij})} \theta_{ij}$

Proposed approximation: As a trade-off between the exact computation versus the approximation introduced by the rasterization and the computing costs, we propose to give the partially overlapping rectangles a weight according to the relative area of their intersection with the support. This avoids expensive exact computations with the input data, by only considering the preprocessed values $f^{\theta_0}(i,j)$:

$$\begin{aligned}\sum_{\pi\cap\vec{p}_{DSM}(\mathbb{Z}^2)} f &\simeq \sum_{i,j\in\mathbb{Z}} \frac{area\,(\theta_{ij}\cap\pi)}{area\,(\theta_{ij})} f^{\theta_0}(i,j) \\ &= \sum_{\substack{i,j\in\mathbb{Z}\\ \theta_{ij}\subseteq\pi}} f^{\theta_0}(i,j) + \sum_{\substack{i,j\in\mathbb{Z}\\ \theta_{ij}\not\subseteq\pi\\ \theta_{ij}\cap\pi\neq\emptyset}} \frac{area\,(\theta_{ij}\cap\pi)}{area\,(\theta_{ij})} f^{\theta_0}(i,j)\end{aligned}$$

While there is a number of terms in the second sum proportional to the perimeter of the support, the first term is rather proportional to the support area. To remain proportional to the perimeter, the sum is decomposed by using the identity linking the orthogonal slabs S_i and T_j and a rectangle $\theta_{ij} = S_i \cap T_j$.

$$\sum_{\substack{i,j\in\mathbb{Z}\\ \theta_{ij}\subseteq\pi}} f^{\theta_0}(i,j) = \sum_{\substack{i\in\mathbb{Z}\\ S_i\cap\pi\neq\emptyset}} \left(\sum_{\substack{j\in\mathbb{Z}\\ S_i\cap T_j\subseteq\pi}} f^{\theta_0}(i,j) \right)$$

The inner summation partitions the intersection of the slab S_i and the support π approximated by a union of rectangles, into a set of rectangles $\{\theta_{ij}$ with j such that $\theta_{ij} \subseteq \pi\}$. By merging neighboring rectangles within each slice S_i, a set of rectangles is constructed that contains a number of rectangles proportional to the perimeter of the support π. As a sum over a rectangle can be done in constant time using a cumulative preprocessing of f, the proposed algorithm to estimate $\vec{\phi}_{\max}$ and to compute the maximum benefit ΔE is indeed proportional to the sum of the perimeters of the superstructure facets.

Dormer windows have their rooftop along the line $\lambda_{\theta_0}(\vec{p}) = \frac{i_0+i_1}{2}$. When this coordinate is an integer, no approximation is made along this edge because no rectangle θ_{ij} is only partially overlapping the support along this edge. However, if $\frac{i_0+i_1}{2} \in \mathbb{Z} + \frac{1}{2}$, it is possible to improve the approximation at the cost of an increased space complexity, by also resampling and accumulating the functions on rectangles $\vec{p}_{\theta_0}([i, i+0.5[\times[j, j+1[)$:

$$f^{\theta_0}(i+0.5, j) = \sum_{\vec{p}_{\theta_0}([i,i+0.5[\times[j,j+1[)\cap\vec{p}_{DSM}(\mathbb{Z}^2)} f$$

A.5 Conclusion

This appendix has presented various methods to speed up the $\mathcal{L}_2$ estimation of the vector of altimetric parameters $\vec{\phi}_{\max}$ of a superstructure candidate given its type and its planimetric parameters, detailing section 3.4.1.2.

Appendix B

Superstructure Detection and Reconstruction preventing an Exhaustive Search

Contents

As a trade-off between computing time and robustness of the detection, we designed an alternative detection algorithm that was published in [DB08]. This is an alternative to the exhaustive search method exposed in chapter 3 that is an extension of [BBPDM07] that allows superstructures to overlap multiple roof planes. The main advantage of the technique proposed in this appendix is that it does not require the exhaustive search followed by a local minima filtering described in sections 3.4.1 and 3.4.3. Furthermore, its computation time is proportional to the number of the real superstructures.

The key idea is to coarsely detect the location of possible superstructures. Then, the parametric models of such superstructures are refined and validated in latter stages. Thus, the proposed approach is split into three main stages. The first stage consists in detecting and locating 3D objects able to evolve to feasible superstructures (Section B.1). In the second stage, the set of 3D objects is then refined to a set of superstructure candidates via parameter fitting (Section B.2). This refinement stage can use either an iterative improvement scheme or a stochastic diffusion scheme. The last stage is not modified: it provides the final solution as a set of non-overlapping superstructures by solving the maximum weighted clique problem described in section 3.4.2.

B.1 Coarse detection

As in the rest of this chapter, we assume that the planes containing the 3D polygons of the roofs $\mathcal{R}$ were obtained robustly, that is, their estimation was accurate enough despite the presence of superstructures. Typically a superstructure corresponds to a set of outlier 3D points in the

DSM - those 3D points that are far from the plane containing the 3D polygon. Therefore, a simple search for clusters of outliers can provide a coarse localization of possible superstructures. Every pixel belonging to the support of a 3D polygon stores the error between the DSM and the parametric model of the 3D polygon at this pixel. The obtained difference map is then thresholded using two thresholds in order to get two kinds of outliers. The lower threshold is set to the DSM noise. The upper threshold separates the pixels into low and high superstructure clusters. Because of the nature of the DSMs used, the map of outliers will be populated with many isolated pixels and undesirable components. To eliminate them, we apply a morphological opening with a circular structuring element whose diameter is proportional to the planimetric DSM resolution. After filtering, we obtain smoothed homogeneous areas of connected pixels. Connected component labelling is then performed, which gives the number of regions used in the next stages of the algorithm. Each of these 2D regions (for example, see Figure B.1.b) represents an approximation of the support of a candidate superstructure. The obtained 2D homogeneous regions are upgraded to a list of 2D rectangles θ_k representing the support of possible superstructures. This is performed by first choosing one of the plausible orientations described in section 3.2.2.3.

B.2 Model refinement

Algorithm 7 Model refinement

for all detected 2D rectangle θ_k **do**
  for all superstructure type $\tau \in T$, such that $\theta_k \in \Theta_{\tau,\mathcal{R}}$ **do**
    repeat
      $(\vec{\phi}_{max}, \theta_k) \leftarrow \arg\max_{\vec{\phi},\theta} \Delta E(\vec{\phi}, \theta, \tau, \mathcal{R})$, with $\theta \in Neighbor(\theta_k) \cap \Theta_{\tau,\mathcal{R}}$ and $\vec{\phi} \in \Phi_{\theta,\tau,\mathcal{R}}$
    until convergence
    $\mathcal{H} \leftarrow (\vec{\phi}_{max}, \theta_k, \tau, \mathcal{R})$
return $\mathcal{H}$

We propose two algorithms for model refinement that share the canvas of Algorithm 7. At this step, due to the DSM noise and imprecision, these 2D rectangles do not necessarily correspond to the support of the real superstructures. Examples are given by i) a detected rectangle may represent a part of a glass roof (for example, see rectangles 1 and 2 in Figure B.1.c-upper left), and ii) a detected rectangle may be larger than the real support of a chimney (for example, see rectangle 3 in Figure B.1.c).

In order to overcome this and to get a set of plausible superstructure candidates we adopt a fine modeling scheme by which the support θ_k and the altimetric parameters $\vec{\phi}$ are locally improved. For each rectangle estimate θ_k and each superstructure type, a given set $Neighbor(\theta_k)$ of rectangles that are small variations of θ_k are considered. For a given type τ and each neighboring rectangle $\theta \in Neighbor(\theta_k) \cap \Theta_{\tau,\mathcal{R}}$, the estimation of the best $\vec{\phi}$ is carried out by the parameter fitting that maximizes the benefit ΔE as exposed in section 3.4.1.1. The successive improvement and stochastic diffusion variations of this refinement stage differ by their definitions of convergence and neighboring sets $Neighbor(\theta_k)$.

B.2.1 Successive improvements

The proposed algorithm is very similar to a hill climbing maximization where the parameter space (explored by the indices of the rectangle θ_k) is a subset of $\mathbb{R}^4$ and the displacement vector is given by $\delta\theta = (\delta i, \delta j, \delta k, \delta l)$, where the $\delta.$ belong to the set $\{-1, 0, 1\}$. One can notice that any superstructure can grow and shrink at any given iteration.

The successive improvement is carried out by iteratively performing, given a fixed type τ, a local maximization of the score $\max_{\vec{\phi} \in \Phi_{\theta,\tau,\mathcal{R}}} \Delta E(\vec{\phi}, \theta, \tau, \mathcal{R})$ over $\theta \in Neighbor(\theta_k) \cap \Theta_{\tau,\mathcal{R}}$. In

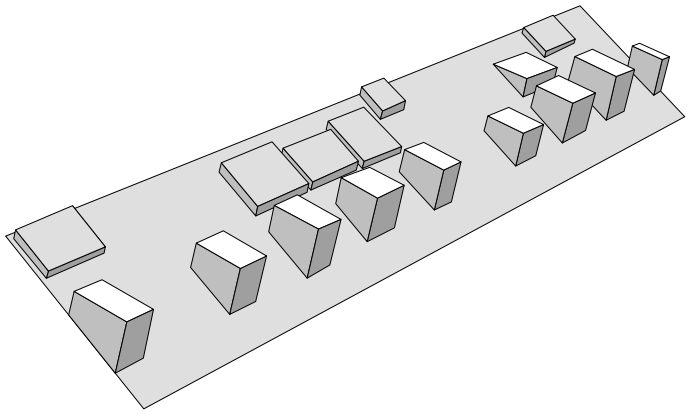
(a)

(b)

(c)

(d)

(e)



(f)

Figure B.1: Approach overview. **(a)** The orthophotography. **(b)** The connected regions obtained by clustering outlier points. **(c)** The initial and coarse 2D rectangles/supports. **(d)** The superstructure candidates obtained by iterative improvements. **(e)** The selected set of superstructures. **(f)** The final roof 3D model.

practice, due the iterative nature of the process, only $N = 10$ neighboring rectangles out of the 81 rectangles $Neighbor(\theta_k) = \{\theta_k + \delta\theta \quad / \quad \delta\theta \in \{-1, 0, +1\}^4\}$ are used. We have chosen the identity - $\delta\theta = (0, 0, 0, 0)$ - 5 expansion directions - $\delta\theta = (-1, 0, 0, 0)$, $(0, -1, 0, 0)$, $(0, 0, 1, 0)$, $(0, 0, 0, 1)$ or $(-1, -1, 1, 1)$ - and 5 shrinking directions - $\delta\theta = (1, 0, 0, 0)$, $(0, 1, 0, 0)$, $(0, 0, -1, 0)$, $(0, 0, 0, -1)$ or $(1, 1, -1, -1)$.

The convergence is met when the rectangle θ_k remains unchanged after one iteration, meaning that it is a local maximum according to the neighboring relation defined by the 11 displacement vectors $\delta\theta$. It is worth noting that, in practice, not all coarse supports require iterations since many of them are already corresponding to a local maximum.

B.2.2 Stochastic diffusion

The refinement algorithm presented above upgrades the coarse supports to superstructure 3D models by maximizing iteratively over a deterministic set of neighboring rectangles. In this paragraph, we propose an alternative refinement algorithm that is based on a stochastic diffusion. In other words, for each coarse support and each type, the coarse support is diffused using a given number of drawn rectangles without any iteration - the convergence is assumed to be reached at once.

The non-deterministic set of neighbors $Neighbor(\theta_k)$ is determined by drawing a fixed number of rectangles. Since the 2D orientation is already known, every drawn rectangle requires drawing four continuous random variables (width, length, and 2D position). We have used uniform distributions to get such rectangles. The range of widths and lengths is set to the type range that defines $\Theta_{\tau,\mathcal{R}}$. The 2D position is limited to keep the center of the detected coarse rectangle θ_k in the drawn rectangle θ.

Although this algorithm is very similar to the previous algorithm, the iterative step is replaced with a stochastic diffusion step (one single pass). The number of drawn rectangles was experimentally determined. It was set to 60. The main advantage of the stochastic diffusion scheme is its ability to escape non-desired local maxima since the search for 2D supports is performed at random.

B.3 Results

The alternative method developed within this appendix has been evaluated in the context of the chapter 3. As a result the input data is given by the DSM only, and the images are not used directly. We have implemented the $\mathcal{L}_1$ and $\mathcal{L}_2$ metrics. The $\mathcal{L}_1$ metric is more adapted to non-Gaussian noise typically affecting correlationbased DSMs, but the computation time of the latter is much faster.

Figure B.1 illustrates the application of the proposed approach to a typical building roof facet. (a) shows the orthophotography of the corresponding 3D polygon. (b) shows the results of the coarse detection scheme—a set of connected regions, (c) shows the initial and coarse supports, (d) shows the result of applying the iterative fine modeling scheme (the set of refined superstructures Ci), (e) shows the selected non-overlapping superstructures, (f) shows the roof model with the reconstructed superstructures . As in chapter 3, we stress the fact that the orthophotography was not used by the proposed algorithms. It is only used for visualization and validation purposes.

Figure B.2 shows the score evolution associated with two coarse supports of Figure B.1.c (rectangles 1 and 2). These two coarse supports represent a fragment of the same upper left superstructure. Iteration 0 corresponds to the score of the coarse support and the final iteration corresponds to the obtained local maximum. One can notice that both supports have converged to roughly the same superstructure (see Figure B.1.d-upper left). Figure B.3 shows the score evolution associated with another coarse support (rectangle 3). The lower part of this figure shows the coarse support

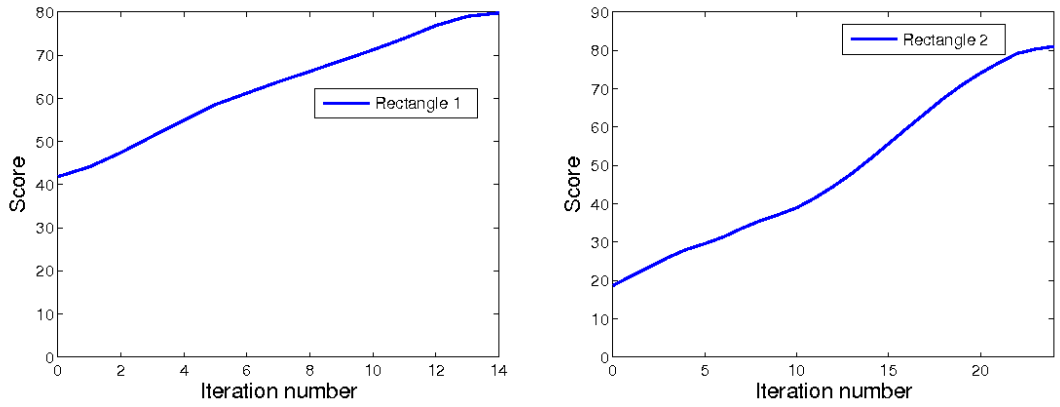


Figure B.2: The score evolution as a function of the iteration number associated with the two rectangles shown in Figure B.1.c(the upper left corner). Iteration 0 corresponds to the coarse support and the final iteration corresponds to a local maximum. One can notice that both supports have converged to the same superstructure (see Figure B.1.d).

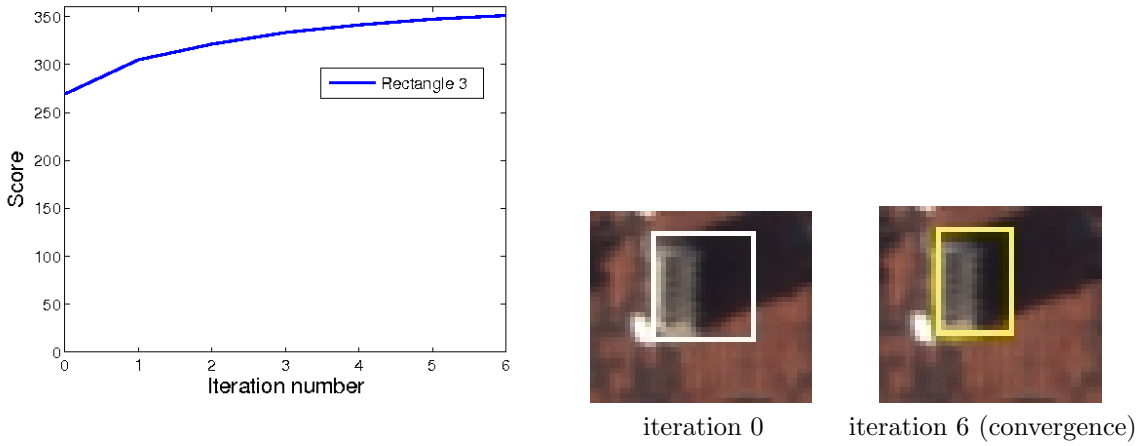
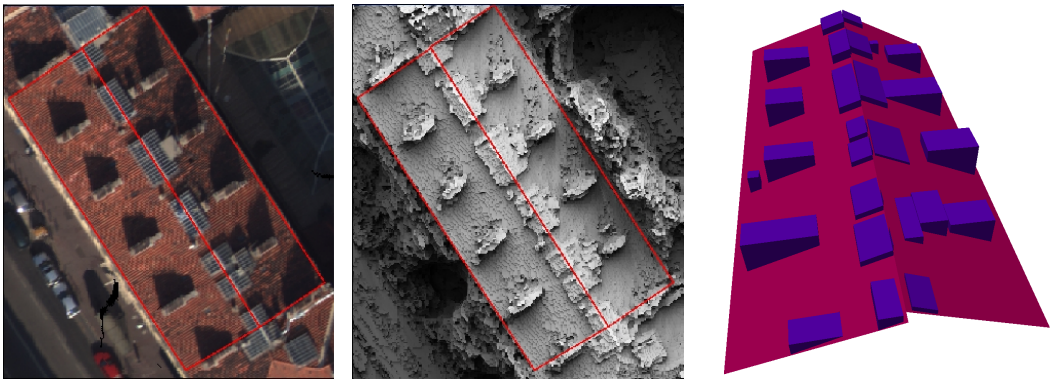


Figure B.3: The score evolution as a function of the iteration number associated with the coarse rectangle 3 (figure B.1.c, bottom-left).

Figure B.4: The detected and reconstructed superstructures using the exhaustive search-based method (Chapter 3).



(a) **(b)** **(c)**

Figure B.5: Reconstructed chimneys and glass roofs. **(a)** The orthophotography. **(b)** A shaded view of the DSM. **(c)** The detected and reconstructed superstructures.

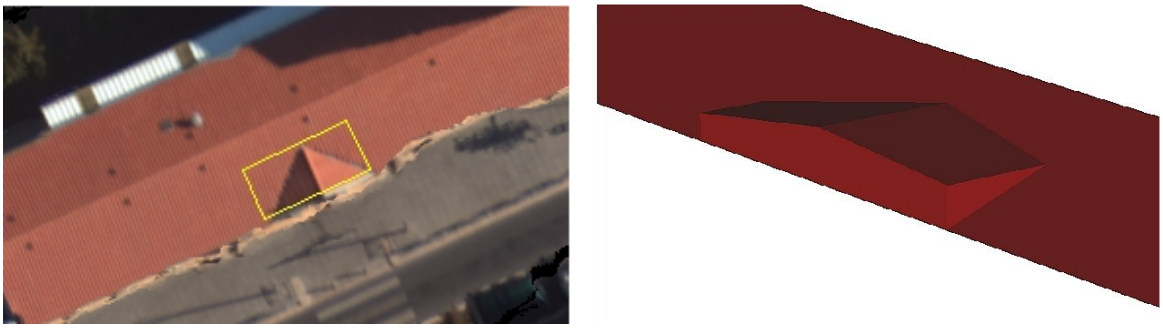
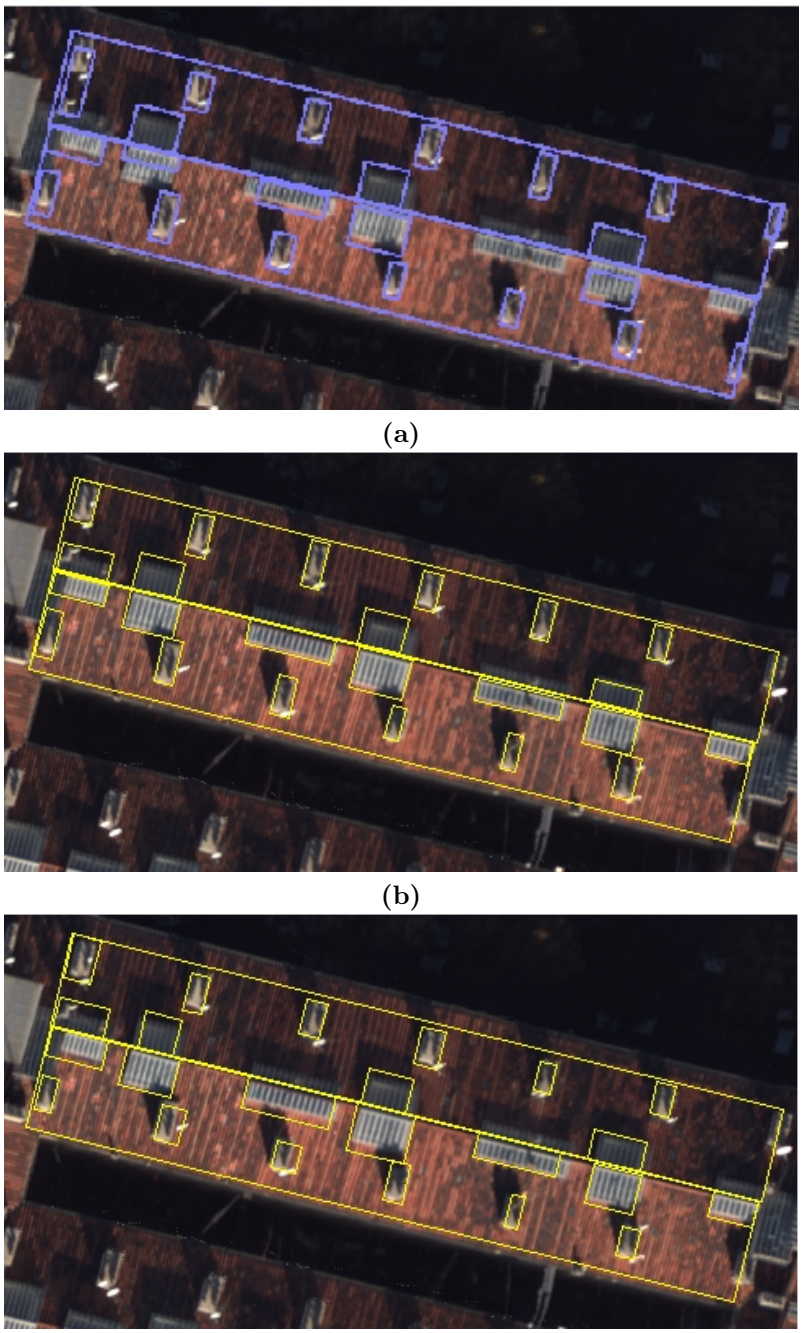


Figure B.6: Reconstructed dormer window. **(a)** The orthophotography. **(b)** The detected and reconstructed superstructure.

	Exhaustive search - Chapter 3	*Successive improvements*	*Stochastic diffusion*
$\mathcal{L}_2$ metric	49 s	3.5 s	10.5 s
$\mathcal{L}_1$ metric	67 s	7.1 s	14.3 s

Table B.1: Computational time associated with the building shown in Figure B.7.



(a) **(b)** **(c)**

Figure B.7: Detecting and reconstructing superstructures associated with a typical building roof. **(a)** results obtained with the exhaustive search-based method. **(b)** results obtained with the proposed approach based on successive refinements. **(c)** results obtained with the proposed approach based on stochastic diffusion.

(left) and the support obtained at convergence (right).

For comparison purposes, the DSM data associated with the roof shown in Figure B.1 were processed by the exhaustive search-based method described in chapter 3. The detection results are shown in Figure B.4. As can be seen, the detection of chimneys was almost the same for both approaches. However, the detected low superstructures were not the same although their overall detection rates were almost the same (see Figure B.1.e).

Figure B.5 shows the results of roof superstructure reconstruction using our proposed method based on stochastic diffusion. Figure B.6 shows another result of roof superstructure reconstruction.

B.3.1 Method Comparison

In addition to the comparison provided by Figures B.1.e and B.4, we have compared the performance of three approaches. Figure B.7 shows the superstructure detection results associated with another typical building roof: **(a)** corresponds to the exhaustive search-based method (Chapter 3), **(b)** to the proposed approach based on successive improvements, and **(c)** to the proposed approach based on stochastic diffusion. As can be seen, the detection results obtained with the proposed approaches (Section B.2) are very similar to those obtained with the exhaustive search-based method. However, the proposed approach is much faster (see Table B.1). It should be noticed that some glass roofs were not detected by any method. This under-detection is due to the difficulty to separate the imperfections of the DSM from the glass roof models which have very small height. The rightmost chimneys were not detected by our proposed method since their size (within the building footprint) was smaller than the size of the opening operator. Furthermore, by comparing **(b)** and **(c)** one can notice that the method based on the successive improvements has provided very good delineation of the 2D supports. This is due to the iterative process by which the 2D support is estimated by locally maximizing a score in the parameter space $\mathbb{R}^4$. On the other hand, the stochastic diffusion scheme explores the parameter space at random. However, the stochastic diffusion scheme can be very useful in cases when the DSM noise is significant.

B.3.2 Computation time

Table B.1 summarizes the CPU time associated with the typical building shown in Figure B.7. The first column corresponds to the exhaustive search-based method (Chapter 3). The second column corresponds to the method based on successive improvements. The third column corresponds to the method based on stochastic diffusion. The first row corresponds to the $\mathcal{L}_2$ metric-based parameter fitting, and the second row to the $\mathcal{L}_1$ metric-based parameter fitting. An Intel Xeon 1.6 GHz PC with a standard C++ code has been used.

The computation time is dominated by the reconstruction of the superstructures, which is the evaluation of its altimetric parameters. Thus, the CPU time of the exhaustive search-based method is proportional to the building size, whereas the CPU time of the approaches developed in this appendix is proportional to the number of the real superstructures.

B.4 Conclusion

This variant is an efficient and modular approach for reconstructing building superstructures using only a DSM, an initial building model without superstructures and an easily extensible collection of parametric models defining the available superstructure types. The approach consists of three main stages. In the first stage, possible superstructures are roughly detected using the DSM and a polyhedral building model. In the second stage, the superstructure parametric models are estimated using either successive improvements or a stochastic diffusion. The final stage selects the most consistent set of superstructures.

The computational time of the proposed approach is proportional to the number of real superstructures. Several comparisons with the exhaustive search based approach described in chapter 3 show that the approach give similar results.

Appendix C

Maximum Weighted Clique

Contents

In section 3.4.2, the proposed approach needs to solve a Maximum Weighted Clique problem with real positive weights. This appendix gives more details about the problem and its implementation. After the necessary definitions, we will present the Maximum Weighted Clique problem and then discuss its implementation within our context.

C.1 Graph and Clique Definitions

A node-weighted graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, w)$ is described by:
– a finite set of nodes $\mathcal{N}$
– a set of edges without loops $\mathcal{E} \subseteq \{(n_1, n_2) \in \mathcal{N} \times \mathcal{N} / n_1 \neq n_2\}$.
The graph is undirected if and only if $\mathcal{E}$ is symmetric: $\forall (n_1, n_2) \in \mathcal{E}, (n_2, n_1) \in \mathcal{E}$.
– a positive node weight function $w : \mathcal{N} \rightarrow \mathbb{R}^+$
By extension, the weight of a subset $\mathcal{X}$ of $\mathcal{N}$ denotes the sum of the weights of its nodes:

$$w(\mathcal{X}) = \sum_{n \in \mathcal{X}} w(n)$$

A **Clique** $\mathcal{C}$ is a subset of $\mathcal{N}$ that satisfies the property that there is an edge of $\mathcal{E}$ between every two distinct nodes $n_1, n_2 \in \mathcal{C}$. For example, cliques of cardinal 1 are simply the singletons for each node of $\mathcal{N}$, and $\{n_1, n_2\}$ is a clique if and only if $(n_1, n_2) \in \mathcal{E}$.

A graph is said to be **complete** when all its edges are present ($\mathcal{E} = \{(n_1, n_2) \in \mathcal{N} \times \mathcal{N} / n_1 \neq n_2\}$). Another important definition is the definition of a subgraph : a **subgraph** of the node-weighted graph $\mathcal{G}$ with nodes $\mathcal{N}' \subseteq \mathcal{N}$ (denoted $\mathcal{G}[\mathcal{N}']$) is the graph $\mathcal{G}[\mathcal{N}'] = (\mathcal{N}', \mathcal{E} \cap (\mathcal{N}' \times \mathcal{N}'), w)$.

	node #	edge #	maximum clique size
unfiltered	1000 - 4000	200 000 - 1 000 000	1 - 20
filtered	10 - 40	30 - 100	1 - 20

Table C.1: Typical graph sizes encountered in the superstructure selection problem of section 3.4.2, when all candidate superstructures are considered (top row) and when only local maxima have been kept (bottom row).

Using these definitions, a clique may be redefined as the set of nodes of a complete subgraph. More concisely, the set of cliques of a graph $\mathcal{G}$ can be defined as:

$$Cliques(\mathcal{G}) = \{\mathcal{C} \subseteq \mathcal{N} / \mathcal{C} \times \mathcal{C} \subseteq \mathcal{E} \cup \{(n,n)/n \in \mathcal{C}\}\}$$

C.2 Maximum Weighted Clique Problem

The Maximum Weighted Clique problem is the following graph optimization problem:

Maximum Weighted Clique (MWC)

Given an undirected node-weighted graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, w)$, the Maximum Weighted Clique problem may be stated as computing:

$$MWC(\mathcal{G}) = \underset{\mathcal{C} \in Cliques(\mathcal{G})}{\arg\max} \left(w\left(\mathcal{C}\right)\right) = \underset{\mathcal{C} \in Cliques(\mathcal{G})}{\arg\max} \left(\sum_{n \in \mathcal{C}} w(n)\right)$$

There is no ambiguity when computing the maximum clique weight, but distinct cliques may share the same maximum weight. Slight variations of the algorithms may allow to report all the maximum weighted cliques or only one of the maximum ones.

The Maximum Clique problem is a special case of the Maximum Weighted Clique problem, where the weight function is constant equal to 1 : it maximizes the size of the clique instead of its weight.

Since the Maximum Clique problem is, by itself, well known to be NP-hard [GJ79], no polynomial time algorithms are known to exist. For an extensive survey of this problem and its applications, see [BBPP99]. However, using state of the art algorithms, the problem is tractable for the sizes of graphs that typically result from the superstructure selection problem of section 3.4.2, when the local maximum filter is applied (table C.1).

C.3 Maximum Weighted Clique Algorithms

C.3.1 Exhaustive Clique Enumeration

To get a better understanding of the state of the art algorithms, we begin by describing an algorithm that computes the maximum weighted clique by enumerating all the cliques and keeping track of the running maximum weighted clique. The basic idea is to order the nodes $\mathcal{N}$ as a sequence $n_1 \dots n_{|\mathcal{N}|}$. Using this ordering, the set of cliques can be partitioned according to the greatest index i of the nodes n_i in the cliques.

$$Cliques(\mathcal{G}) = \emptyset \cup \bigcup_{i=1}^{|\mathcal{N}|} \left\{ \mathcal{C} \in Cliques(\mathcal{G}) \quad / \quad i = \max_{n_j \in \mathcal{C}}(j) \right\}$$

Using the notations $Neighbor(n_i, \mathcal{N}) = \{n \in \mathcal{N} / (n_i, n) \in \mathcal{E}\}$ for the neighbors of n_i in $\mathcal{N}$ and $Next(n_i, \mathcal{N}) = \{n_j \in \mathcal{N} / j > i\}$ for the subset of nodes of $\mathcal{N}$ that have a greater index, the set $\mathcal{N}_i = Neighbor(n_i, \mathcal{N}) \setminus Next(n_i, \mathcal{N})$ can be introduced to rewrite the partition subsets:

$$\left\{ \mathcal{C} \in Cliques(\mathcal{G}) \quad / \quad i = \max_{n_j \in \mathcal{C}}(j) \right\} = \{\{n_i\} \cup \mathcal{C} \quad / \quad \mathcal{C} \in Cliques(\mathcal{G}[\mathcal{N}_i])\}$$

The enumeration of all the cliques $Cliques(\mathcal{G})$ can thus be solved by considering the $|\mathcal{N}|$ subproblems $Cliques(\mathcal{G}[\mathcal{N}_i])$ according to the greatest node index i included in the clique. Each subproblem is handled recursively: each recursive step adds a node n_i to a set of included nodes $\mathcal{I}$ and restrict the graph nodes $\mathcal{N}$ to the neighbors of the last included node n_i that have a smaller index, denoted $\mathcal{N}_i$.

By construction the set of nodes $\mathcal{I}$ is thus a clique. The enumeration is performed recursively until the subgraph $\mathcal{G}[\mathcal{N}]$ has no nodes left ($\mathcal{N} = \emptyset$). Thus, it can be proven by induction that the successive values of $\mathcal{I}$ enumerate all the cliques of the input graph.

Algorithm 8 $MaximumWeightedClique(\mathcal{G}, \mathcal{N}, \mathcal{I}, \mathcal{C})$ // Exhaustive Clique Enumeration

$\mathcal{G} = (\mathcal{N}^0, \mathcal{E}, w)$: an undirected weighted graph,
$\mathcal{N} \subseteq \mathcal{N}^0$: a subset of the nodes of $\mathcal{G}$ that defines the current subgraph $\mathcal{G}[\mathcal{N}]$,
$\mathcal{I} \subseteq \mathcal{N}^0$: a subset of the nodes of $\mathcal{G}$ included in the clique currently being built,
$\mathcal{C} \subseteq \mathcal{N}^0$: the heaviest clique of the whole graph $\mathcal{G}$ found so far.
Ensure: $MaximumWeightedClique(\mathcal{G}, \mathcal{N}^0, \emptyset, \emptyset) = MWC(\mathcal{G})$,
Ensure: $MaximumWeightedClique(\mathcal{G}, \mathcal{N}, \mathcal{I}, \mathcal{C}) = \begin{cases} \mathcal{C} & \text{if } w(\mathcal{C}) > w(MWC(\mathcal{G}[\mathcal{N} \cup \mathcal{I}])) \\ MWC(\mathcal{G}[\mathcal{N} \cup \mathcal{I}]) & \text{otherwise} \end{cases}$
Require: $\mathcal{N} \cap \mathcal{I} = \emptyset$ and $\forall n \in \mathcal{N}, \mathcal{I} \cup \{n\} \in Cliques(\mathcal{G})$.

```
if w(I) > w(C) then
  C ← I
if N = ∅ then
  return C
for i = 1 to |N| do
  Neighbor(n_i, N) ← {n ∈ N / (n_i, n) ∈ E}
  Next(n_i, N) ← {n_j ∈ N / j > i}
  N_i ← Neighbor(n_i, N) \ Next(n_i, N)
  C ← MaximumWeightedClique(G, N_i, I ∪ {n_i}, C)
return C
```

Algorithm 8 provides the implementation of this algorithm, which returns $MWC(\mathcal{G})$ when $MaximumWeightedClique(\mathcal{G}, \mathcal{N}^0, \emptyset, \emptyset)$ is called. As a side note, the requirement that states that $\forall n \in \mathcal{N}, \mathcal{I} \cup \{n\}$ is a clique, ensures that $\mathcal{I} \subseteq MWC(\mathcal{G}[\mathcal{N} \cup \mathcal{I}])$: $\mathcal{I}$ is included in the maximum weighted clique of the subgraph $\mathcal{G}[\mathcal{N} \cup \mathcal{I}]$. Thus, if $w(MWC(\mathcal{G}[\mathcal{N} \cup \mathcal{I}])) > w(\mathcal{C})$, $MaximumWeightedClique(\mathcal{G}, \mathcal{N}, \mathcal{I}, \mathcal{C})$ has to include $\mathcal{I}$.

C.3.2 Branch and Bound

Most of the state of the art algorithms to solve the Maximum Weighted Clique problem are based on a Branch and Bound approach. The general idea is to modify the exhaustive clique enumeration algorithm to introduce laziness, skipping the evaluation of subproblems that do not

provably contain the maximum weighted clique. As its name implies, a MWC algorithm that follows the branch and bound scheme, sketched in Algorithm 9, has to provide two strategies that modify the exhaustive clique enumeration:

Branch: Order the vertices of $\mathcal{N}$ as $n_1, \ldots, n_{|\mathcal{N}|}$ **for all** $n_i \in \mathcal{N}$ in a specified order **do**

This strategy defines the ordering in which all the subproblems will be considered. A specific ordering may be required to efficiently - or even correctly! - evaluate the upperbound provided by the bounding strategy. It may use a heuristic in order to try to consider first subgraphs that have better chances to contain the heaviest clique early, to prune the exploration of more subgraphs with the upperbound test later.

Bound: **if** $w(\mathcal{I}) + w(n_i) + Upperbound(\mathcal{G}[\mathcal{N}_i]) \leq w(\mathcal{C})$ **then continue**

The bounding strategy is responsible for the laziness of the algorithm. It keeps track of the heaviest clique found so far and certifies whether there might be a heavier clique in the current subproblem $\mathcal{G}[\mathcal{N}_i]$ or not. That is, it provides a backtracking test to prune the exploration of a subproblem - a recursive call - that does not provably include the maximum weighted clique. This test involves $Upperbound(\mathcal{G}[\mathcal{N}_i])$ which gives an upperbound on the weight of the maximum weighted clique of the subgraph $\mathcal{G}[\mathcal{N}_i]$. This pruning test enables a lazy exploration of all the possible cliques.

Algorithm 9 $MaximumWeightedClique(\mathcal{G}, \mathcal{N}, \mathcal{I}, \mathcal{C})$ // Branch and Bound

$\mathcal{G} = (\mathcal{N}^0, \mathcal{E}, w)$: an undirected weighted graph,
$\mathcal{N} \subseteq \mathcal{N}^0$: a subset of the nodes of $\mathcal{G}$ that defines the current subgraph $\mathcal{G}[\mathcal{N}]$,
$\mathcal{I} \subseteq \mathcal{N}^0$: a subset of the nodes of $\mathcal{G}$ included in the clique currently being built,
$\mathcal{C} \subseteq \mathcal{N}^0$: the heaviest clique of the whole graph $\mathcal{G}$ found so far.
Ensure: $MaximumWeightedClique(\mathcal{G}, \mathcal{N}^0, \emptyset, \emptyset) = MWC(\mathcal{G})$,
Ensure: $MaximumWeightedClique(\mathcal{G}, \mathcal{N}, \mathcal{I}, \mathcal{C}) = \begin{cases} \mathcal{C} & \text{if } w(\mathcal{C}) > w(MWC(\mathcal{G}[\mathcal{N} \cup \mathcal{I}])) \\ MWC(\mathcal{G}[\mathcal{N} \cup \mathcal{I}]) & \text{otherwise} \end{cases}$
Require: $\mathcal{N} \cap \mathcal{I} = \emptyset$ and $\forall n \in \mathcal{N}, \mathcal{I} \cup \{n\} \in Cliques(\mathcal{G})$.

```
if w(I) > w(C) then
  C ← I
if N = ∅ then
  return C
Order the vertices of N as n_1, ..., n_|N|.
for all n_i ∈ N in a specified order do
  Neighbor(n_i, N) ← {n ∈ N / (n_i, n) ∈ E}
  Next(n_i, N) ← {n_j ∈ (N) / j > i}
  N_i ← Neighbor(n_i, N) \ Next(n_i, N)
  if w(I) + w(n_i) + Upperbound(G[N_i]) ≤ w(C) then
    continue
  C ← MaximumWeightedClique(G, N_i, I ∪ {n_i}, C)
return C
```

Interestingly, the brute force algorithm, that just enumerates all the cliques of the graph, follows this scheme with the following strategies. It has no pruning whatsoever ($Upperbound = \infty$) and its branching strategy is irrelevant because all the cliques will be considered during the exploration.

A more efficient best-in approach can be defined by the following strategies. A clique weight is smaller than the cumulative weight of all the nodes of the current subgraph $Upperbound(\mathcal{G}[\mathcal{N}_i]) = w(\mathcal{N}_i)$. This upperbound can be computed in constant time by maintaining the weight of the sets $\mathcal{N}_i$, $\mathcal{I}$ and $\mathcal{C}$, adding or substracting the weight of a node that is being inserted or removed from a subset, to the cached subset weight. The best-in strategy comes into play by first ordering the nodes by decreasing weight w and solving the subproblems by increasing values of i to start with subgraphs with the heaviest nodes, hoping that this will generate the heaviest cliques early. With

this heuristic, the first maximal clique (not included in any other clique) explored is computed by iteratively selecting the heaviest node and discarding all its non neighbors. This clique corresponds in practice to a good, but sub-optimal solution.

C.3.3 Branch and Bound with Exclusion

More advanced Branch and Bound algorithms [WH, BX91, Bab94] introduce a third strategy that allows the consideration of less than $|\mathcal{N}|$ subproblems. The intuition is that, given a subset $\mathcal{X}$, if a subgraph $\mathcal{G}[\mathcal{X}]$ does not contain a heavier clique than the current heaviest clique $\mathcal{C}$, a heavier clique in $\mathcal{G}[\mathcal{N}]$ must contain at least one node in $\mathcal{N} \setminus \mathcal{X}$. Thus the branching strategy only has to consider the nodes in $\mathcal{N} \setminus \mathcal{X}$ rather than all the nodes in $\mathcal{N}$.

Exclusion : $\boxed{\text{Find } \mathcal{X} \subseteq \mathcal{N} \text{ such that } MaximumWeightedClique\,(\mathcal{G}, \mathcal{X}, \mathcal{I}, \mathcal{C}) = \mathcal{C}}$

$\mathcal{X}$ is a subset of the current subgraph nodes $\mathcal{N}$ such that $\mathcal{X} \cup \mathcal{I}$ has provably no heavier cliques than $\mathcal{C}$ in the graph $\mathcal{G}$. It means that a clique in $\mathcal{N}$ heavier than $\mathcal{C}$ must contain at least one vertex $n_i \in \mathcal{N} \setminus \mathcal{X}$.

The widely used branch and bound scheme with exclusion, presented in [WH], may be written as:

Algorithm 10 $MaximumWeightedClique(\mathcal{G}, \mathcal{N}, \mathcal{I}, \mathcal{C})$ // Branch and Bound with Exclusion

$\mathcal{G} = (\mathcal{N}^0, \mathcal{E}, w)$: an undirected weighted graph,
$\mathcal{N} \subseteq \mathcal{N}^0$: a subset of the nodes of $\mathcal{G}$ that defines the current subgraph $\mathcal{G}[\mathcal{N}]$,
$\mathcal{I} \subseteq \mathcal{N}^0$: a subset of the nodes of $\mathcal{G}$ included in the clique currently being built,
$\mathcal{C} \subseteq \mathcal{N}^0$: the heaviest clique of the whole graph $\mathcal{G}$ found so far.
Ensure: $MaximumWeightedClique(\mathcal{G}, \mathcal{N}^0, \emptyset, \emptyset) = MWC(\mathcal{G})$,
Ensure: $MaximumWeightedClique(\mathcal{G}, \mathcal{N}, \mathcal{I}, \mathcal{C}) = \begin{cases} \mathcal{C} & \text{if } w(\mathcal{C}) > w(MWC(\mathcal{G}[\mathcal{N} \cup \mathcal{I}])) \\ MWC(\mathcal{G}[\mathcal{N} \cup \mathcal{I}]) & \text{otherwise} \end{cases}$
Require: $\mathcal{N} \cap \mathcal{I} = \emptyset$ and $\forall n \in \mathcal{N}, \mathcal{I} \cup \{n\} \in Cliques(\mathcal{G})$.

```
if w(I) > w(C) then
  C ← I
if N = ∅ then
  return C
Find X ⊆ N such that MaximumWeightedClique (G, X, I, C) = C.
Order the vertices of (N \ X) as n_1, ..., n_{|N\X|}.
for all n_i ∈ (N \ X) in a specified order do
  Neighbor(n_i, N) ← {n ∈ N / (n_i, n) ∈ E}
  Next(n_i, N \ X) ← {n_j ∈ (N \ X) / j > i}
  N_i ← Neighbor(n_i, N) \ Next(n_i, N \ X)
  if w(I) + w(n_i) + Upperbound(G[N_i]) ≤ w(C) then
    continue
  C ← MaximumWeightedClique (G, N_i, I ∪ {n_i}, C)
return C
```

Branch and bound algorithms with exclusion are however more involved and no implementation is currently publicly available. Some of these algorithms [Bab94] are only able to handle integer and not real weighted graphs. While it could have been possible to multiply the weights by some high value, say 10^6, and round them to the nearest integer, it would have introduced an unnecessary approximation in the process. Moreover, it is not clear whether the speed up gained by skipping some subproblems will overall counterbalance the increased complexity of finding a good excluding set $\mathcal{X}$ for the type of graphs considered in our applicative context.

C.3.4 Cliquer

The *cliquer* [Öst02] algorithm follows the branch and bound approach without exclusion (Algorithm 9), using the following strategies:

Branch :

Order the vertices of $\mathcal{N}$ as $n_1, \ldots, n_{|\mathcal{N}|}$

The initial ordering $n_1, \ldots, n_{|\mathcal{N}^0|}$ of the nodes $\mathcal{N}^0$ is given by the user, and the algorithm keeps this ordering intact throughout the optimization. That means that the bijective function $\sigma : \{1, \ldots, k\} \to \{i \quad / \quad n_i \in \mathcal{N}\}$, that defines the ordering $n_{\sigma(1)}, \ldots, n_{\sigma(k)}$ of a subset $\mathcal{N}$ is increasing.

for all $n_i \in \mathcal{N}$ in a specified order **do**

The first level of recursion, when $\mathcal{N} = \mathcal{N}^0$, considers the subproblems n_i with increasing i, whereas the smaller subproblems are considered with a decreasing order on i.

Bound :

$Upperbound(\mathcal{G}[\mathcal{N}_i]) \geq w(MWC(\mathcal{G}[\mathcal{N}_i]))$

Cliquer uses two complementary upperbound tests: the cumulative weight $w(\mathcal{N}_i)$ and an upperbound $W_{cliquer}[i]$, based on a dynamic programing approach, that might be tighter.

This upperbound is computed by maintaining an array $W_{cliquer}$ of size $|\mathcal{N}^0|$. $W_{cliquer}[i]$ stores the difference between the maximum weight of a clique in the subgraph that has nodes $\{n_1, \ldots, n_i\} = \mathcal{N}^0 \setminus Next(i+1, \mathcal{N}^0)$, and the weight of n_i.

$$W_{cliquer}[i] = w(\mathcal{C}_i) - w(n_i) \qquad \text{with} \qquad \mathcal{C}_i = MWC(\mathcal{G}[\{n_1, \ldots, n_i\}])$$

The fixed ordering and the increasing order of processing of the first level of recursion, that computes the subproblems $\mathcal{C}_i = MWC(\mathcal{G}[\{n_1, \ldots, n_i\}])$, guarantees that when computing the upperbound for a given value of i, $W_{cliquer}[1], \ldots, W_{cliquer}[i-1]$ are already computed.

Another property is that the series $w(\mathcal{C}_i) = w(n_i) + W_{cliquer}[i]$ is increasing, being the maximum clique weight of increasing node subsets $\{n_1, \ldots, n_i\}$. Thus, if $w(\mathcal{I}) + w(n_i) + W_{cliquer}[i] \leq w(\mathcal{C})$, then it will also be true for all $j < i$. This is why, deeper than the first level of recursion, where the subproblems are considered by decreasing indices, if the cliquer upperbound test fails, it is possible to not only avoid the exploration of the subproblem $\mathcal{N}_i$ but also of all the suproblems $\mathcal{N}_j$ with $j < i$. This speeds up the exploration by replacing, in the algorithm 9, the **continue** statement by a **break** statement when the cliquer test upperbound fails.

If the cliquer upperbound test does not fail, the cumulative weight test $w(\mathcal{N}_i)$ is performed and issues a **continue** statement.

Proof. $W_{cliquer}[i] \geq w(MWC(\mathcal{G}[\mathcal{N}_i]))$:

$w(\mathcal{C}_i) = w(n_i) + W_{cliquer}[i]$ may be defined by recurrence.

$$\begin{aligned} w(\mathcal{C}_0) &= w(MWC(\mathcal{G}[\emptyset])) = 0 \\ w(\mathcal{C}_i) &= w(MWC(\mathcal{G}[\mathcal{N}^0 \setminus Next(i+1, \mathcal{N}^0)])) \\ &= \max(w(\mathcal{C}_{i-1}), w(n_i) + w(MWC(\mathcal{G}[Neighbor(n_i, \mathcal{N}^0) \setminus Next(n_i, \mathcal{N}^0)]))) \end{aligned}$$

The fixed order $n_1, \ldots, n_{|\mathcal{N}^0|}$ facilitates the definition of the set $Next(n_i, \mathcal{N})$:

$$\begin{aligned} Next(n_i, \mathcal{N}) &= Next(n_i, \mathcal{N}^0) \cap \mathcal{N} \\ \text{which yields} \quad \mathcal{N}_i &= Neighbor(n_i, \mathcal{N}) \setminus Next(n_i, \mathcal{N}) \\ &= \big(Neighbor(n_i, \mathcal{N}^0) \cap \mathcal{N}\big) \setminus \big(Next(n_i, \mathcal{N}^0) \cap \mathcal{N}\big) \\ &= \big(Neighbor(n_i, \mathcal{N}^0) \setminus Next(n_i, \mathcal{N}^0)\big) \cap \mathcal{N} \\ &\subseteq \big(Neighbor(n_i, \mathcal{N}^0) \setminus Next(n_i, \mathcal{N}^0)\big) \end{aligned}$$

This set inclusion proves that $W_{cliquer}[i]$ is an upperbound of $w(MWC(\mathcal{G}[\mathcal{N}_i]))$:

$$\begin{aligned} w(MWC(\mathcal{G}[\mathcal{N}_i])) &\leq w(MWC(\mathcal{G}\left[Neighbor(n_i,\mathcal{N}^0) \setminus Next(n_i,\mathcal{N}^0)\right])) \\ &\leq \max(w(\mathcal{C}_{i-1}) - w(n_i), w(MWC(\mathcal{G}[Neighbor(n_i,\mathcal{N}^0) \setminus Next(n_i,\mathcal{N}^0)]))) \\ &= w(\mathcal{C}_i) - w(n_i) = W_{cliquer}[i] \end{aligned}$$

□

It must be noted that the algorithm is very sensitive to the node ordering $n_1, \ldots, n_{|\mathcal{N}^0|}$. There is no clear best ordering: some weighted graphs are faster processed with nodes sorted by increasing weights, others by decreasing weights, a random ordering may also be performant...

C.3.5 Efficiency upperbound

This upperbound is an attempt at designing a variant of *cliquer* that is more specifically tailored to the particular type of graphs that results from our superstructure selection problem. Within our context, the graph nodes, corresponding to superstructure candidates, have another attribute than the weight $w(n) = \Delta E(n)$ measuring the benefit of selecting the superstructure in the final reconstruction: the 2D polygon of its support π_n, which is, in short, the vertical projection of the superstructure 3D facets onto a horizontal plane. Furthermore, the edge set $\mathcal{E}$ is induced by the polygons $(\pi_n)_{n\in\mathcal{N}}$ and the non-overlap relationship $\not\cap$, defined as $\pi_{n_1} \not\cap \pi_{n_2} \Leftrightarrow area\,(\pi_{n_1} \cap \pi_{n_2}) = 0$:

$$\mathcal{E} = \{(n_1, n_2) \in \mathcal{N} \times \mathcal{N} \quad / \quad \pi_{n_1} \not\cap \pi_{n_2}\}$$

From the polygons and the node weights, two other attributes may be computed for each node $n \in \mathcal{N}$: its area $a(n) = area\,(\pi_n)$ and its efficiency $\frac{w(n)}{a(n)}$. Using those extra attributes, it is possible to design an upperbound on the weight of the maximum weighted clique of a given set of nodes that is reasonably efficient to compute and tighter than just the sum of all the node weights $w(\mathcal{N}_i)$.

If $a(n) = 0$, the efficiency is not well defined, but those nodes can be handled easily. Since they have no interior surface, they do not intersect any other nodes, and are guaranteed to be contained in the maximum weighted clique. Thus we can compute the maximum weighted clique of nodes with a strictly positive area and add those with a null area to form the maximum weighted clique of all the nodes.

By extension we define the area $a(\mathcal{N})$ of a set of nodes $\mathcal{N}$ by the area of the union of its node polygons $area\left(\bigcup_{n\in\mathcal{N}} \pi_n\right)$. We also note the following inequality:

$$area\left(\bigcup_{n\in\mathcal{N}} \pi_n\right) \leq \sum_{n\in\mathcal{N}} area\,(\pi_n) \qquad \underset{notation}{\Leftrightarrow} \qquad a(\mathcal{N}) \leq \sum_{n\in\mathcal{N}} a(n)$$

Because a set of disjoint nodes is the definition of a clique when the edges are determined by the non-overlap relationship $\not\cap$, the area of the union of the node polygons $a(\mathcal{C}) = area\left(\bigcup_{n\in\mathcal{C}} \pi_n\right)$ is exactly the sum of the node polygon area. The inequality becomes an equality:

$$\mathcal{C} \in Cliques(\mathcal{G}) \Rightarrow area\left(\bigcup_{n\in\mathcal{C}} \pi_n\right) = \sum_{n\in\mathcal{C}} area\,(\pi_n) \underset{notation}{\Leftrightarrow} a(\mathcal{C}) = \sum_{n\in\mathcal{C}} a(n)$$

It follows that, given a set of nodes $\mathcal{N}$, one can compute the union of the polygons of all the nodes in the set and that the area of this union has to be greater than the sum of the node areas of any clique of the subgraph $\mathcal{G}[\mathcal{N}]$.

$$\forall \mathcal{C} \in Cliques(\mathcal{G}[\mathcal{N}]), \sum_{n\in\mathcal{C}} area\,(\pi_n) \leq area\left(\bigcup_{n\in\mathcal{N}} \pi_n\right) \underset{notation}{\Leftrightarrow} \sum_{n\in\mathcal{C}} a(n) \leq a(\mathcal{N}) \tag{C.3.1}$$

Now, an upperbound on the cumulative weight of the maximum weighted clique can be derived from this bound on the cumulative areas. If the clique constraint of the Maximum Weighted Clique problem is replaced by the cumulative area bounding constraint C.3.1, the modified problem is known as the Knapsack problem KS. Following the constraint C.3.1, $c(n)$ is set to $a(n) = area(\pi_n)$ and c_{max} to $a(\mathcal{N}) = area\left(\bigcup_{n\in\mathcal{N}} \pi_n\right)$.

Knapsack (KS)

Given a set of elements $\mathcal{N}$, a weight function $w : \mathcal{N} \to \mathbb{R}^+$, a cost function $c : \mathcal{N} \to \mathbb{R}^+$ and a maximum cost c_{max}, the Knapsack problem may be stated as computing:

$$KS(\mathcal{N}, w, c, c_{max}) = \underset{\mathcal{C}\subseteq\mathcal{N}/\sum_{n\in\mathcal{C}} c(n)\leq c_{max}}{\arg\max} \left(\sum_{n\in\mathcal{C}} w(n)\right)$$

or, equivalently, x being the indicator function of $\mathcal{C}$:

$$KS(\mathcal{N}, w, c, c_{max}) = \underset{x:\mathcal{N}\to\{0,1\}/\sum_{n\in\mathcal{N}} x(n)c(n)\leq c_{max}}{\arg\max} \left(\sum_{n\in\mathcal{N}} x(n)w(n)\right)$$

Note that in this problem the edge set $\mathcal{E}$ has no influence. Because being a clique implies the cumulative area bounding constraint, the Knapsack problem yields an upperbound on the maximum weight of the Maximum Weighted Clique:

$$w\left(MWC\left(\mathcal{N}, \mathcal{E}, w\right)\right) \leq w\left(KS\left(\mathcal{N}, w, a, a(\mathcal{N})\right)\right)$$

If the problem is further relaxed by allowing the selection x to be a percentage instead of only one of the two values 0 and 1, the NP-Complete Knapsack problem becomes a Single Constraint Linear Programming with an efficient implementation available.

Single Constraint Linear Programming (SCLP)

Given a set of elements $\mathcal{N}$, a weight function $w : \mathcal{N} \to \mathbb{R}^+$, a cost function $c : \mathcal{N} \to \mathbb{R}^+$ and a maximum cost c_{max}, the Single Constraint Linear Programming problem may be stated as computing:

$$SCLP(\mathcal{N}, w, c, c_{max}) = \underset{x:\mathcal{N}\to[0,1]/\sum_{n\in\mathcal{N}} x(n)c(n)\leq c_{max}}{\arg\max} \left(\sum_{n\in\mathcal{N}} x(n)w(n)\right)$$

This problem is finally not NP-hard anymore : linear programming in general may be solved using a worst-case polynomial-time algorithm. The particular instance SCLP of linear programming used here only has a single constraint. This allows the design of an output sensitive, linear time (after preprocessing) Algorithm 11 to solve the Single Constraint Linear Programming problem.

Algorithm 11 uses a best-in approach. Nodes are ordered by decreasing efficiency $\frac{w}{c}$ and are selected until the cost bound c_{max} is reached. If the cost of the last node to be selected is not exactly the remaining allowable cost, its maximum allowable fraction is selected instead. We are here only interested in an upperbound on the weight and not on the actual subset of $\mathcal{N}$ that achieves this upperbound, therefore the algorithm does not keep track of the best selection but only of its weight.

Proof. To prove that the efficiency ordering makes this best-in algorithm correct, let us consider a valuation of $x : \mathcal{N} \to [0, 1]$ and try to improve the cumulative weight by modifying x. As the objective function is linear, it

Algorithm 11 *SingleConstraintLinearProgrammingWeight*$(\mathcal{N}, w, c, c_{max})$

$\mathcal{N}$: a set of nodes, sorted by efficiency $\frac{w}{c}$,
$w(n)$: the weight of node $n \in \mathcal{N}$,
$c(n)$: the cost of node $n \in \mathcal{N}$.
c_{max}: the maximum cost allowed.

```
weight ← 0
for n ∈ N by decreasing efficiency do
  if c_max ≤ c(n) then
    return weight + (c_max / c(n)) w(n)
  weight ← weight + w(n)
  c_max ← c_max − c(n)
return weight
```

is convex and the maximum is at the boundary of the admissible domain and the constraint is an equality at the maximum point x. If we want to modify the x value for nodes n_1 and n_2 to $(x(n_1) + \delta x_1)$ and $(x(n_2) + \delta x_2)$, while staying at the boundary of the domain, the relation $c(n_1)\delta x_1 + c(n_2)\delta x_2 = 0$ must hold. The cumulative weight modification will then be $w(n_1)\delta x_1 + w(n_2)\delta x_2 = \left(\frac{w(n_2)}{c(n_2)} - \frac{w(n_1)}{c(n_1)}\right) c(n_2)\delta x_2$. Because c is positive, if, say, the efficiency $\frac{w}{c}$ of n_2 is higher than the efficiency of n_1 and all the x values but $x(n_1)$ and $x(n_2)$ are fixed, the only way to increase the objective function is to increase $x(n_2)$ and decrease $x(n_1)$ accordingly. □

The Knapsack being a constrained subproblem of the Single Constraint Linear Programming, it is upperbounded by the Single Constraint Linear Programming weight:

$$w\left(KS\left(\mathcal{N}, w, a, a(\mathcal{N})\right)\right) \leq w(SCLP(\mathcal{N}, w, a, a(\mathcal{N})))$$

Finally, this designs an upperbound W_{eff} for the maximum weighted clique problem when the graph edges are induced by the overlap of polygons. We call this upperbound the efficiency upperbound:

$$W_{eff}(\mathcal{N}, w, a) = w(SCLP\left(\mathcal{N}, w, a, a(\mathcal{N})\right)) \geq w\left(MWC(\mathcal{G})\right)$$

To avoid the recomputation of $a(\mathcal{N})$ for every subgraph considered by the algorithm, an upperbound $A(\mathcal{N})$ of this area is used instead of the exact area $a(\mathcal{N}) = area\left(\bigcup_{n\in\mathcal{N}} \pi_n\right)$. The area is computed only once for the whole set of nodes : $A(\mathcal{N}^0) = a(\mathcal{N}^0)$. Then, when a node n_i is selected in the included set $\mathcal{I}$, and that the nodes of the current subgraph are reduced from $\mathcal{N}$ to $\mathcal{N}_i = Neighbor(n_i, \mathcal{N}) \setminus Next(n_i, \mathcal{N})$, the area upperbound is updated by subtracting the area of n_i:

$$A(\mathcal{N}_i) = A(\mathcal{N}) - a(n_i)$$

Proof.

$$\begin{aligned}
a(\mathcal{N}_i) &= area\left(\bigcup_{n\in\mathcal{N}_i} \pi_n\right) \\
&= area\left(\left(\bigcup_{n\in\mathcal{N}_i\cup\{n_i\}} \pi_n\right) \setminus \pi_{n_i}\right) \\
&= area\left(\bigcup_{n\in\mathcal{N}_i\cup\{n_i\}} \pi_n\right) - area\left(\pi_{n_i}\right) \\
&= a(\mathcal{N}_i \cup \{n_i\}) - a(n_i) \\
&\leq a(\mathcal{N}) - a(n_i) \\
&\leq A(\mathcal{N}) - a(n_i) = A(\mathcal{N}_i)
\end{aligned}$$

The second equality is due to n_i being disjoint from all the nodes in $\mathcal{N}_i \subseteq Neighbor(n_i, \mathcal{N})$, by the definition of the set $Neighbor(n_i, \mathcal{N})$, and that if $\pi_1 \not\emptyset \pi_2$, then $area(\pi_1 \cap \pi_2) = 0$ and thus $area(\pi_1) = area((\pi_1 \cup \pi_2) \setminus \pi_2)$. The third equality comes from the trivial inclusion $\pi_{n_i} \subseteq \bigcup_{n \in \mathcal{N}_i \cup \{n_i\}} \pi_n$, whereas the first inequality comes from the inclusion $\mathcal{N}_i \cup \{n_i\} \subseteq \mathcal{N}$. □

This upperbound can be included inside the *cliquer* algorithm, by using the minimum of the *cliquer* upperbound and the efficiency upperbound as the modified upperbound. By construction, $W_{eff}(\mathcal{N}_i, w, a) \leq w(\mathcal{N}_i)$, so there is no advantage in computing $w(\mathcal{N}_i)$:

$$Upperbound(\mathcal{G}[\mathcal{N}_i]) = \min(W_{eff}(\mathcal{N}_i, w, a), W_{cliquer}[i])$$

Its drawback is that its evaluation is generally not constant time, but if the nodes are sorted by efficiency, the time cost becomes output sensitive. It is far less than its linear worse case if the minimum area of a node is not too small relative to the area of the union of the node polygons. For instance, within our applicative context, let us assume that the superstructure candidates have a minimum area of $\frac{area(\pi_{\mathcal{B}})}{\alpha}$ where α is a constant factor, and $area(\pi_{\mathcal{B}})$ is the area of the footprint of the whole building which is necessarily greater than the area of the union of the superstructure candidate polygons. Under this assumption, it is trivial to show that the upperbound takes $O(\alpha)$ time to compute, which is constant with respect to the number of superstructure candidates.

C.4 Conclusion

While it is possible to include the efficiency upperbound within *cliquer*, this test requires, to minimize its computation cost, that the nodes are sorted by efficiency. While the sorting itself is not expensive with a $O(|\mathcal{N}|log|\mathcal{N}|)$ computing cost, it corresponds to an unfavorable order for the unmodified *cliquer* algorithm when applied to graphs that solve the superstructure selection problem. This is why there is no substantial speed up resulting from the inclusion of the efficiency upperbound within the *cliquer* algorithm. The performance gain due to this tighter upperbound roughly counterbalances the performance loss of the unfavorable sorting and the additionnal cost of its computation.

As an extension, two orderings could be maintained for each node subset, as nodes are removed or added back. The first ordering would be an ordering which makes the *cliquer* algorithm efficient, while the second ordering would be able to provide the nodes of the subset by decreasing efficiency to compute the upperbound W_eff efficiently at the same time.

The C library *cliquer*[NÖ03], an implemention of [Öst02] by its authors, is available under the GPL license at http://users.tkk.fi/ pat/cliquer.html. The actual Maximum Weighted Clique implementation used in this work is a complete C++ rewrite of the *cliquer* library using the Boost Graph Library (BGL) framework that can handle real valued weights (the *cliquer* C library only supports integer weights). It also provides a convenient and flexible access to the inner parts of the algorithm, using function object templates, to be able to optionally use the efficiency upperbound in the bounding test. Finally, the efficiency upperbound presenting no real advantage over the original *cliquer* upperbound, it is disabled by default in our implementation of the superstructure reconstruction.

Appendix D

Method Invariance by an Invertible Affine Transform

Contents

D.1 Introduction

This appendix proves the following proposition, mentionned in section 6.5.2 :

Proposition D.1. *The approach proposed in chapters 5 and 6 is unconditionally invariant under a projective transform of the point coordinates (the primal geometry), if and only if this transform is invertible and affine.*

We define the meaning of the unconditional invariance such that, up to ambiguities of the trihedralization subproblems, the combinatorial data structures computed by the proposed method for the transformed and original problems are isomorphic, without any condition on the geometry or topology of the problem.

Throughout chapters 5 and 6, the geometric coordinates are only used through *Above* and *Orientation* predicates, given that the *Intersect* predicate of section 5.4.2.2 is build upon the *Above* predicate. Thus, the unconditional invariance is equivalent to a constant sign of these certificate functions undergoing a given geometric transformation.

Section D.2 details how a projective transform modifies the problem geometry. Then, sections D.3 and D.4 provide the respective proofs for the both predicates.

D.2 Problem Transformation

Using the projective geometry formalism, a projective transform is modeled by a 4 by 4 matrix. Such a transform is said to be invertible if the matrix A is invertible ($\det A \neq 0$). The transform is affine if the first 3 elements of its last row are null ($A_{41} = A_{42} = A_{43} = 0$).

To prove the proposition D.1, the expression of the transformation of a polyhedral geometry (either the plane coordinates or the point coordinates) implied by a transformation of its dual geometry (respectively the point and the plane coordinates) is required.

Theorem D.2. *If the dual geometry is transformed by a projective matrix A, then the primal geometry is transformed by* $\operatorname{com} A$

Proof. Let us denote by A^T the transposed of the matrix A and $\operatorname{com} A$ the matrix of the cofactors (*i.e.* signed minors) of A. The cofactor matrix verifies the identity that $\operatorname{com}(AB) = (\operatorname{com} A)(\operatorname{com} B)$, for any square matrices of equal order A and B. The cofactor matrix allows a simple expression of the homogeneous coordinates of a vertex of valence 3 in function of the 3 adjacent supporting planes $(\vec{N}_i)$:

$$\vec{P} = \left(\operatorname{com}\begin{bmatrix}\vec{0} & \vec{N}_0 & \vec{N}_1 & \vec{N}_2\end{bmatrix}\right)\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

This allows the expression of the point $\vec{P}_{A\vec{N}_i}$ computed by the intersection of the 3 transformed planes $A\vec{N}_i$ relative to the point $\vec{P}_{A\vec{N}_i}$ computed by the intersection of the 3 original planes $\vec{N}_i$.

$$\begin{aligned}\vec{P}_{A\vec{N}_i} &= \left(\operatorname{com}\begin{bmatrix}\vec{0} & A\vec{N}_0 & A\vec{N}_1 & A\vec{N}_2\end{bmatrix}\right)\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\\ &= \operatorname{com}\left(A\begin{bmatrix}\vec{0} & \vec{N}_0 & \vec{N}_1 & \vec{N}_2\end{bmatrix}\right)\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\\ &= (\operatorname{com} A)\left(\operatorname{com}\begin{bmatrix}\vec{0} & \vec{N}_0 & \vec{N}_1 & \vec{N}_2\end{bmatrix}\right)\begin{bmatrix}1\\0\\0\\0\end{bmatrix} = (\operatorname{com} A)\,\vec{P}_{\vec{N}_i}\end{aligned}$$

□

By the symmetry of the point-plane duality, the relation $\vec{N} = \operatorname{com}\begin{bmatrix}\vec{0} & \vec{P}_0 & \vec{P}_1 & \vec{P}_2\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$ also holds and the dual theorem is true:

Theorem D.3. *If the primal geometry is transformed by a projective matrix A, then the dual geometry is transformed by* $\operatorname{com} A$.

Moreover, one can easily verify that, after a transformation A, a transformed point $(A\vec{P})$ lies on its transformed supporting planes $((\operatorname{com} A)\vec{N})$:

$$\begin{aligned}(A\vec{P})\cdot((\operatorname{com} A)\vec{N}_i) = (A\vec{P})^T((\operatorname{com} A)\vec{N}_i) &= \vec{P}^T(A^T(\operatorname{com} A))\vec{N}_i\\ &= \vec{P}^T((\det A)I)\vec{N}_i = (\det A)(\vec{P}\cdot\vec{N}_i) = 0\end{aligned}$$

If the initial and target dual geometries $(\vec{N}_{i0})$ and $(\vec{N}_{i1})$ are transformed by a projective matrix A, then the interpolated homogeneous plane coordinates are also multiplied by A: $(1-t)A\vec{N}_{i0} + tA\vec{N}_{i1} = A\left((1-t)\vec{N}_{i0} + t\vec{N}_{i1}\right)$. Thus, not only the initial and target primal geometry will be transformed by $\operatorname{com} A$, but also the points of the transformed interpolated polyhedron. The linearity of the transformation allows to prove the proposition D.1 without taking into account the interpolation time t.

D.3 *Above* Certificate Function

The *Above* certificate function suffices to compute a plane arrangement. If the transformation A multiplies the *Above* certificate function by a constant factor, then the original and transformed arrangements will be isomorphic. Since the following of the trihedralization approach using a 0-1 coloring of the plane arrangement (section 5.3) only relies on the topology of this arrangement, it is then insensitive to A, up to the possible ambiguities of the trihedralization. The *Above* certificate function is also used by the proposed extension to handle diverging vertices (section 6.6.1).

If the primal geometry is transformed by A, (and the dual geometry by $\operatorname{com} A$), then the certificate function determining the position of a point relative to an oriented plane becomes:

$$Above(A\vec{P}, (\operatorname{com} A)\vec{N}) = \frac{A\vec{P}}{w_{A\vec{P}}} \cdot (\operatorname{com} A)\vec{N} = \det(A)\frac{\vec{P}\cdot\vec{N}}{w_{A\vec{P}}} = \left(\det(A)\frac{w_{\vec{P}}}{w_{A\vec{P}}}\right) Above(\vec{P}, \vec{N})$$

Lemma D.4. *$Above(\vec{P}, \vec{N}_A) = \frac{w_{A\vec{P}}}{w_{\vec{P}}}$ with $\vec{N}_A$ being te plane defined by the last column of A^T.*

Proof. $w_{A\vec{P}} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \cdot \left(A\vec{P}\right) = \left(A^T \begin{bmatrix}0\\0\\0\\1\end{bmatrix}\right) \cdot \vec{P} = \vec{N}_A \cdot \vec{P}$ □

Lemma D.5. *A is affine and invertible $\Leftrightarrow$ $\det(A)\frac{w_{\vec{P}}}{w_{A\vec{P}}}$ is non-null, well defined and its sign is independent of $\vec{P}$.*

Proof. If the projective transform A is affine, its last line may be written as $(0\ 0\ 0\ A_{44})$, and then $w_{A\vec{P}} = A_{44}w_{\vec{P}}$, for all $\vec{P}$. Moreover, if it is affine and invertible, then $\det A \neq 0$ and $A_{44} \neq 0$. Thus the point-plane certificate functions of the original and the transformed problem differ only by a constant non-null multiplicative factor $\left(\det(A)\frac{w_{\vec{P}}}{w_{A\vec{P}}}\right) = \frac{\det(A)}{A_{44}}$, which is the determinant of the upperleft 3 by 3 submatrix of the projective matrix A.

Conversely, if the last line of A only contains zeros, then the factor is not well defined. Thus the last line is not null and may represent the transposed homogeneous coordinates of an oriented plane $\vec{N}_A^T$ which is possibly the plane containing points at infinity. Thus the $\det(A)\frac{w_{\vec{P}}}{w_{A\vec{P}}}$ factor may be rewritten as $\frac{\det(A)}{Above(\vec{P},\vec{N}_A)}$. The sign of this quantity is constant as long as the Cartesian points represented by $\vec{P}$ lie on a single side of $\vec{N}_A$. Thus the plane $\vec{N}_A$ is the plane of points at infinity: the last line of A may be written as $(0\ 0\ 0\ A_{44})$ with $A_{44} \neq 0$, meaning that A is affine and that the factor is constantly equal to $\frac{\det(A)}{A_{44}}$. Finally, a non-null factor proves that $\det(A) \neq 0$: A is invertible. □

The previous two lemmas together prove the following theorem, with $\epsilon = sign\left(\frac{\det(A)}{A_{44}}\right)$:

Theorem D.6.

$$\begin{aligned} & A \text{ is affine and invertible} \\ \Leftrightarrow\quad & \exists \epsilon = \pm 1 \text{ s.t. } \forall \vec{P}, \vec{N}, \quad sign(Above(A\vec{P}, (\operatorname{com} A)\vec{N})) = \epsilon . sign(Above(\vec{P}, \vec{N})) \end{aligned}$$

If $\epsilon = 1$, then the sign of the certificate is unchanged, and thus the transformation will have no impact on the algorithm. When $\epsilon = -1$ however, the sign is reversed. This situation may be dealt with by applying $-A$ instead of A, which will yield $\epsilon = 1$, leave the points unchanged, and reverse the orientation of each plane.

Furthermore, the *Intersect* predicate and the plane arrangement do not directly consider the sign of a single *Above* certificate functions, but are comparing signs of 2 *Above* certificate functions. They are thus insensitive to the value of $\epsilon \in \{-1, 1\}$.

D.4 *Orientation* Certificate Function

This section gives an expression of the *Orientation* certificate function of the transformed problem in terms of the original problem, using the following lemmas D.7 and D.8. The lemmas D.9 and D.10 use this relation to prove the invariance of the algorithm under an affine invertible transform.

Lemma D.7. $\forall \vec{X}_i, \forall A, \left|\vec{X}_0 \quad A\vec{X}_1 \quad A\vec{X}_2 \quad A\vec{X}_3\right| = \left|(\operatorname{com} A)^T\vec{X}_0 \quad \vec{X}_1 \quad \vec{X}_2 \quad \vec{X}_3\right|$

Proof. Using $\operatorname{com}(AB) = (\operatorname{com} A)(\operatorname{com} B)$,

$$\begin{aligned}
|\vec{X}_0 \quad A\vec{X}_1 \quad A\vec{X}_2 \quad A\vec{X}_3| &= \vec{X}_0^T \left(\operatorname{com}\begin{bmatrix}0 & A\vec{X}_1 & A\vec{X}_2 & A\vec{X}_3\end{bmatrix}\right)\begin{bmatrix}1\\0\\0\\0\end{bmatrix} \\
&= \vec{X}_0^T (\operatorname{com} A)\left(\operatorname{com}\begin{bmatrix}0 & \vec{X}_1 & \vec{X}_2 & \vec{X}_3\end{bmatrix}\right)\begin{bmatrix}1\\0\\0\\0\end{bmatrix} \\
&= \left((\operatorname{com} A)^T\vec{X}_0\right)^T \left(\operatorname{com}\begin{bmatrix}0 & \vec{X}_1 & \vec{X}_2 & \vec{X}_3\end{bmatrix}\right)\begin{bmatrix}1\\0\\0\\0\end{bmatrix} \\
&= |(\operatorname{com} A)^T\vec{X}_0 \quad \vec{X}_1 \quad \vec{X}_2 \quad \vec{X}_3|
\end{aligned}$$

□

Lemma D.8. $\left(\operatorname{com}\begin{bmatrix}\vec{0} & \vec{P}_0 & \vec{P}_1 & \vec{P}_2\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\right) = Q_{\vec{P}_0\vec{P}_1\vec{P}_2}\vec{N}.$

Proof. By construction, the 4D column vector $\left(\operatorname{com}\begin{bmatrix}\vec{0} & \vec{P}_0 & \vec{P}_1 & \vec{P}_2\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\right)$ is colinear with $\vec{N}$, since the points $\vec{P}_i$ lie in the plane $\vec{N}$, which translates into the orthogonality of their 4D vectors. Thus, there is a function $\lambda_{\vec{P}_0\vec{P}_1\vec{P}_2}$ such that $\left(\operatorname{com}\begin{bmatrix}\vec{0} & \vec{P}_0 & \vec{P}_1 & \vec{P}_2\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\right) = \lambda_{\vec{P}_0\vec{P}_1\vec{P}_2}\vec{N}$. Using the polynom $Q_{\vec{P}_0\vec{P}_1\vec{P}_2}$:

$$\begin{aligned}
Q_{\vec{P}_0\vec{P}_1\vec{P}_2}|\vec{n}|^2 = \begin{bmatrix}\vec{n} & \vec{p}_0 & \vec{p}_1 & \vec{p}_2\\ 0 & w_0 & w_1 & w_2\end{bmatrix} &= \begin{bmatrix}\vec{n}\\0\end{bmatrix}\cdot\left(\operatorname{com}\begin{bmatrix}\vec{0} & \vec{P}_0 & \vec{P}_1 & \vec{P}_2\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\right) \\
&= \begin{bmatrix}\vec{n}\\0\end{bmatrix}\cdot\left(\lambda_{\vec{P}_0\vec{P}_1\vec{P}_2}\vec{N}\right) \\
&= \lambda_{\vec{P}_0\vec{P}_1\vec{P}_2}|\vec{n}|^2
\end{aligned}$$

which proves $\lambda_{\vec{P}_0\vec{P}_1\vec{P}_2} = Q_{\vec{P}_0\vec{P}_1\vec{P}_2}$ and the lemma. □

Using $B = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{bmatrix}$, the lemmas D.4, D.7, D.8 above, the *Orientation* certificate function of the transformed problem may also be rewritten in terms of the orientation certificate function of the original problem:

$$\begin{aligned}
Orientation((\operatorname{com} A)\vec{N}, A\vec{P}_0, A\vec{P}_1, A\vec{P}_2) &= \left|B(\operatorname{com} A)\vec{N} \quad \tfrac{A\vec{P}_0}{w_{A\vec{P}_0}} \quad \tfrac{A\vec{P}_1}{w_{A\vec{P}_1}} \quad \tfrac{A\vec{P}_2}{w_{A\vec{P}_2}}\right| \\
&= \frac{\left|(\operatorname{com} A)^T B(\operatorname{com} A)\vec{N} \quad \vec{P}_0 \quad \vec{P}_1 \quad \vec{P}_2\right|}{w_{A\vec{P}_0}w_{A\vec{P}_1}w_{A\vec{P}_2}} \\
&= \frac{\left((\operatorname{com} A)^T B(\operatorname{com} A)\vec{N}\right)\cdot\left(Q_{\vec{P}_0\vec{P}_1\vec{P}_2}\vec{N}\right)}{w_{A\vec{P}_0}w_{A\vec{P}_1}w_{A\vec{P}_2}} \\
&= \frac{\vec{N}^T(\operatorname{com} A)^T B(\operatorname{com} A)\vec{N}}{w_{\operatorname{com} A\vec{P}_0}w_{\operatorname{com} A\vec{P}_1}w_{\operatorname{com} A\vec{P}_2}}\left(Q_{\vec{P}_0\vec{P}_1\vec{P}_2}\right)
\end{aligned}$$

$$= \frac{\left\|B(\operatorname{com} A)\vec{N}\right\|^2}{\left\|B\vec{N}\right\|^2}\left(\frac{w_{\vec{P}_0}}{w_{A\vec{P}_0}}\frac{w_{\vec{P}_1}}{w_{A\vec{P}_1}}\frac{w_{\vec{P}_2}}{w_{A\vec{P}_2}}\right)\left(Orientation(\vec{N}, \vec{P}_0, \vec{P}_1, \vec{P}_2)\right)$$

If we denote by a the 3 by 3 upper left submatrix of $\operatorname{com} A$ and $\vec{b}$ the vector of the first three elements of its last column, then we can prove two lemmas that help analyzing $B(\operatorname{com} A)\vec{N}$:

Lemma D.9. $\left((i)\ \forall\vec{N},\quad B\vec{N} \neq \vec{0} \Rightarrow B(\operatorname{com} A)\vec{N} \neq \vec{0}\right) \quad \Leftrightarrow \quad \left((ii)\ \det(a) \neq 0\ \textit{and}\ (iii)\ \vec{b} = \vec{0}\right)$

Proof. The matrix $B(\operatorname{com} A)$ may be rewritten as $\begin{bmatrix} a & \vec{b} \\ 0 & 0 \end{bmatrix}$ and thus $B(\operatorname{com} A)\vec{N}$ may also be reformulated as:

$$B(\operatorname{com} A)\vec{N} = [a\vec{n} + \vec{b}d : 0] \qquad \text{with } \vec{N} = [\vec{n} : d]$$

$[(i) \Rightarrow (ii)]$: If a is not invertible, then there exists $\vec{n}^* \neq \vec{0}$ in the kernel of a. Thus there exists $\vec{N}^* = [\vec{n}^* : 0]$, such that $B\vec{N}^* = \vec{N}^* \neq \vec{0}$ and $B(\operatorname{com} A)\vec{N}^* = [a\vec{n}^* + b.0 : 0] = \vec{0}$. This proves $\neg(ii) \Rightarrow \neg(i)$.

$[(i) \wedge (ii) \Rightarrow (iii)]$: If a is invertible, the plane $\vec{N}^* = [-a^{-1}\vec{b} : 1]$ has a degenerate transformed normal $B(\operatorname{com} A)\vec{N}^* = [a(-a^{-1}\vec{b}) + \vec{b}(1) : 0] = \vec{0}$. Its normal $B\vec{N}^* = [-a^{-1}\vec{b} : 0]$ is degenerate if and only if $\vec{b} = \vec{0}$. This proves that $(ii) \Rightarrow (\neg(iii) \Rightarrow \neg(i))$.

$[(ii) \wedge (iii) \Rightarrow (i)]$: If $\det(a) \neq 0$ and $\vec{b} = \vec{0}$, then $\forall \vec{N},\ B\vec{N} = \vec{0} \Leftrightarrow \vec{n} = \vec{0} \Leftrightarrow a\vec{n} = \vec{0} \Leftrightarrow B(\operatorname{com} A)\vec{N} = \vec{0}$, which implies (i).

□

Lemma D.10. *A is affine and invertible* $\Leftrightarrow \Big(\det(a) \neq 0$ *and* $\vec{b} = \vec{0}\Big)$

Proof. As coordinates of $\vec{b}$ are the first 3 cofactors of the last column of A, they can be expressed as determinants of a 3 by 3 submatrix of A that contain the first 3 elements of the last line of A. Since A is affine, those determinants are necessarily null, yielding $\vec{b} = \vec{0}$. Now that $\vec{b} = \vec{0}$ is proven, $\det(\operatorname{com} A)$ may be written as $(\det a)\,((\operatorname{com} A)_{44})$. Using the property $\det(\operatorname{com} A) = (\det A)^3$ for any 4 by 4 matrices, if A is invertible, then $(\det A)^3 = \det(\operatorname{com} A) = (\det a)\,((\operatorname{com} A)_{44})$ is not null and neither is $\det a$.

Conversely, if $\vec{b} = \vec{0}$ then the last line of $\operatorname{com}(\operatorname{com} A)$ can be computed as $\begin{bmatrix} 0 & 0 & 0 & \det a \end{bmatrix}$, since the first 3 elements of this line are defined as plus or minus some 3 by 3 determinants that involve the $\vec{b}$ vector. Furthermore the bottom right element of $\operatorname{com}(\operatorname{com} A)$ is, by construction, $\det a$. Since $\operatorname{com}(\operatorname{com} A) = (\det A)^2 A$, $(\det a)$ is equal to $(\det A)^2 A_{44}$. This proves that if $\det a \neq 0$ then $\det A \neq 0$: A is invertible. Finally the last line of $A = \frac{\operatorname{com}(\operatorname{com} A)}{(\det A)^2}$ can be written as $\begin{bmatrix} 0 & 0 & 0 & \frac{\det a}{(\det A)^2} \end{bmatrix}$: A is affine. □

Using the lemmas D.9 and D.10, this proves that the transformation A is affine and invertible if and only if the factor $\left\| B(\operatorname{com} A)\vec{N} \right\|^2$ is non-null for any finite plane $\vec{N}$ ($B\vec{N} \neq \vec{0}$). Using again the lemma D.4 concerning the *Above* certificate function on the $\frac{w_{\vec{P}_i}}{w_{A\vec{P}_i}}$ factors, this finally proves that if the transformation A is affine and invertible, then the factor $\left(\frac{\left\| B(\operatorname{com} A)\vec{N} \right\|^2}{\left\| B\vec{N} \right\|^2} \right) \left(\frac{w_{\vec{P}_0}}{w_{A\vec{P}_0}} \frac{w_{\vec{P}_1}}{w_{A\vec{P}_1}} \frac{w_{\vec{P}_2}}{w_{A\vec{P}_2}} \right)$ linking the transformed and original certificate functions is non-null for finite planes and that its sign is constantly equal to the sign of A_{44}.

D.5 Conclusion

This finally proves that the proposed approach is unconditionally invariant, up to the ambiguities of the trihedralizations, by a projective transform A if and only if A is affine and invertible. The discussion on the implications of this result are discussed in section 6.5.2.

Bibliography

[AA96] O. Aichholzer and F. Aurenhammer. Straight skeletons for general polygonal figures. In *Proc. of the 2nd International Computing and Combinatorics Conference (COCOON)*, volume 1090, pages 117–126, Hong Kong, 1996. Springer-Verlag.

[AAAG95] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gärtner. A novel type of skeleton for polygons. *Journal of Universal Computer Science (J.UCS)*, 1(12):752–761, 1995.

[Bab94] L. Babel. A fast algorithm for the maximum weight clique problem. *Computing*, 52:31–38, 1994.

[BBPDM07] M. Brédif, D. Boldo, M. Pierrot-Deseilligny, and H. Maître. 3D building reconstruction with parametric roof superstructures. In *Proc. of the IEEE International Conference on Image Processing (ICIP)*, San Antonio, U.S., sep 2007.

[BBPP99] I.M. Bomze, M. Budinich, P.M. Pardalos, and M. Pelillo. The maximum clique problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 1–74. Kluwer Academic, Boston, Massachusetts, U.S.A., 1999.

[BCG+99] J. Basch, J. Comba, L.J. Guibas, J. Hershberger, C. Silverstein, and L. Zhang. Kinetic data structures: Animating proofs through time. In *Proc. of the Symposium on Computational Geometry (SoCG)*, pages 427–428, 1999.

[BDE96] G. Barequet, M. Dickerson, and D. Eppstein. On triangulating three-dimensional polygons. In *Proc. of the 12th Symposium on Computational Geometry (SoCG)*, pages 38–47, New York, NY, USA, 1996. ACM.

[BDH96] C.B. Barber, D.P. Dobkin, and H. Huhdanpaa. The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, dec 1996. http://www.qhull.org.

[BEGV08] G. Barequet, D. Eppstein, M.T. Goodrich, and A. Vaxman. Straight skeletons of three-dimensional polyhedra. In *Proc. of the 6th European Symposium on Algorithms (ESA)*, Karlsruhe, Germany, sep 2008.

[BGH97] J. Basch, L.J. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. of the 8th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 747–756, 1997.

[Bou07] L. Boudet. *Auto-qualification de données géographiques 3D par appariement multi-image et classification supervisée. Application au bâti en milieu urbain dense.* PhD thesis, Université Paris-Est, Marne-la-Vallée, sep 2007. Interne.

[BR06a] C. Brenner and N. Ripperda. Extraction of facades using RJ-MCMC and constraint equations. In *Photogrammetric Computer Vision*, pages 155–160, 2006.

[BR06b] F. Bretar and M. Roux. Recognition of building roof facets by merging aerial images and 3d lidar data in a hierarchical segmentation framework. In *Proc. of the International Conference on Pattern Recognition (ICPR)*, Hong-Kong, China, 2006.

[Bre00] C. Brenner. Towards fully automatic generation of city models. In Remote Sensing International Archives of the Photogrammetry and Spatial Information Sciences (IAPRS), editors, *Proc. of the XIXth ISPRS Congress*, volume 33, pages 85–92, Amsterdam, July 2000.

[BX91] E. Balas and J. Xue. Minimum weighted coloring of triangulated graphs, with the application to maximum weight vertex packing and clique finding in arbitrary graphs. *SIAM J. Comput.*, 20:209–221, 1991.

[BZ00] C. Baillard and A. Zisserman. A plane-sweep strategy for the 3D reconstruction of buildings from multiple images. In *Proc. of the XIXth ISPRS Congress*, Amsterdam, The Netherlands, 2000.

[CGAL] CGAL. Computational Geometry Algorithms Library. http://www.cgal.org.

[Cha91] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6:485–524, 1991.

[CSAD04] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions on Graphics, SIGGRAPH Proceedings*, pages 905–914, 2004.

[CSW99] F. Chin, J. Snoeyink, and C.A. Wang. Finding the medial axis of a simple polygon in linear time. *Discrete and Computational Geometry*, pages 382–391, 1999.

[CV02] S.W. Cheng and A. Vigneron. Motorcycle graphs and straight skeletons. In *Proc. of the 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 156–165, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[dB07] M. de Bovis. Etude de la typologie des superstructures de toit. Master's thesis, Ecole des Ingénieurs de la Ville de Paris, 2007.

[DB08] F. Dornaika and M. Brédif. An efficient approach to building superstructure reconstruction using digital elevation maps. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (IAPRS)*, volume 37 (Part 3A), Beijing, China, 2008.

[dBCvKO08] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3^{rd} edition, 2008.

[DEGN98] T.K. Dey, H. Edelsbrunner, G. Guha, and D.V. Nekhayev. Topology preserving edge contraction. *Publ. Inst. Math. (Beograd) (N.S*, 66:23–45, 1998.

[DLM05] M. Desbrun, M. Leok, and J.E. Marsden. Discrete Poincaré lemma. *Applied Numerical Mathematics*, 53(2):231–248, 2005.

[DT06] M. Durupt and F. Taillandier. Automatic building reconstruction from a digital elevation model and cadastral data: an operational approach. In *Proc. of the ISPRS Symposium on Photogrammetric Computer Vision (PCV)*, Bonn, Germany, 2006. ISPRS.

[DTC04] A.R. Dick, P.H.S. Torr, and R. Cipolla. Modelling and interpretation of architecture from several images. *International Journal of Computer Vision (IJCV)*, 60(2):111–134, Nov. 2004.

[EAH08] J. Engels, H. Arefi, and M. Hahn. Generation of roof topologies using plane fitting with RANSAC. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (IAPRS)*, volume 37 (Part 3A), Beijing, China, 2008.

[EE99] D. Eppstein and J.G. Erickson. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete and Computational Geometry*, 22(4):569–592, 1999. (Special issue for SoCG'98).

[EET93] H.A. ElGindy, H. Everett, and G.T. Toussaint. Slicing an ear using prune-and-search. *Pattern Recognition Letters (PRL)*, 14(9):719–722, 1993.

[EOS86] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15(2):341–363, 1986.

[ES86] H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1:25–44, 1986.

[FB81] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[FKL+98] A. Fischer, T.H. Kolbe, F. Lang, A.B. Cremers, W. Förstner, L. Plümer, and V. Steinhage. Extracting buildings from aerial images using hierarchical aggregation in 2D and 3D. *Computer Vision and Image Understanding (CVIU)*, 72(2):185–203, 1998.

[FM05] D. Flamanc and G. Maillet. Evaluation of 3D city model production from PLEIADES HR satellite images and 2D ground maps. In *Proc of the ISPRS International Symposium Remote Sensing and Data Fusion Over Urban Areas (URBAN)*, Tempe, USA, 2005.

[For97] S. Fortune. Polyhedral modeling with multiprecision integer arithmetic. *Computer-Aided Design*, 29(2):123–133, 1997.

[FZ03] C. Früh and A. Zakhor. Constructing 3D city models by merging aerial and ground views. *IEEE Transactions on Computer Graphics and Applications*, 23(6):52–61, 2003.

[GH97] M. Garland and P.S. Heckbert. Surface simplification using quadric error metrics. *ACM Transactions on Graphics, SIGGRAPH Proceedings*, 97.

[GHH+03] M. Granados, P. Hachenberger, S. Hert, L. Kettner, K. Mehlhorn, and M. Seel. Boolean operations on 3d selective nef complexes: Data structure, algorithms, and implementation. In Giuseppe Di Battista and Uri Zwick, editors, *Proc. of the 11th European Symposium on Algorithms (ESA)*, volume 2832 of Lecture Notes in Computer Science, pages 174–186, Budapest, Hungary, September 2003. Springer.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Company, New York, 1979.

[GKR04] L.J. Guibas, M. Karaveles, and D. Russel. A computational framework for handling motion. In *Proc. of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 129–141, 2004.

[Grü71] B. Grünbaum. Arrangements of hyperplanes. In *Congressum Numerantium III, Louisiana Conference on Combinatorics Graph Theory and Computing*, pages 41–106, 1971.

[Gui98] L.J. Guibas. Kinetic data structures: A state of the art report. In P. K. Agarwal, L. Kavraki, and M. Mason, editors, *Proc. of the 3rd Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998.

[HDD+93] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, and W. Stuetzle. Mesh optimization. *ACM Transactions on Graphics, SIGGRAPH Proceedings*, pages 19–26, 1993.

[Hir08] H. Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(2):328–341, 2008.

[HM90] D.G. Hook and P.R. McAree. Using sturm sequences to bracket real roots of polynomial equations. In *Graphics Gems I*, pages 416–422. Academic Press, 1990.

[HP02] S. Hornus and C. Puech. A simple kinetic visibility polygon. In *Proc of the 18th European Workshop on Computational Geometry (EuroCG)*, pages 27–30, 2002. Warsaw Uni.

[JPDPM00] H. Jibrini, M. Pierrot-Deseilligny, N. Paparoditis, and H. Maître. Automatic building reconstruction from very high resolution aerial stereopairs using cadastral ground plans. In *Proc. of the XIXth ISPRS Congress*, Amsterdam, The Netherlands, 2000.

[Kad02] M. Kada. Automatic generalisation of 3D building models. In *Joint International Symposium on Geospatial Theory, Processing and Applications*, 2002.

[Kad06] M. Kada. 3D building generalization based on half-space modeling. In *Proc. of the ISPRS Commission II Workshop*, Hannover, Germany, 2006.

[KE02] M. Kasser and Y. Egels. *Digital Photogrammetry.* Taylor & Francis, London, 2002.

[Ket99] L. Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry - Theory and Applications (CGTA)*, 13:65–90, 1999.

[KGP05] T.H. Kolbe, G. Groeger, and L. Pluemer. CityGML interoperable access to 3D city models. In Springer Verlag, editor, *Proc. of the 1st International Symposium on Geo-information for Disaster Management*, Delft, 2005.

[KP09] K. Karantzalos and N. Paragios. Recognition-driven 2D competing priors towards automatic and accurate building detection. *IEEE Transactions on Geoscience and Remote Sensing (TGRS)*, 47(1):133–144, 2009.

[KTS+09] P. Koutsourakis, O. Teboul, L. Simon, G. Tziritas, and N. Paragios. Single view reconstruction using shape grammars for urban environments. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2009.

[KZG06] S. Kocaman, L. Zhang, and A. Gruen. 3D city modelling from high resolution satellite images. In *Proc. of the ISPRS Conference Topographic Mapping From Space (With Special Emphasis on Small Satellites)*, pages 14–16, Ankara, Turkey, 2006.

[LDZPD10] F. Lafarge, X. Descombes, J. Zerubia, and M. Pierrot-Deseilligny. Structural approach for building reconstruction from a single DSM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(1):135–147, 2010.

[LN98] C. Lin and R. Nevatia. Building detection and description from a single intensity image. *Computer Vision and Image Understanding (CVIU)*, 72(2):101–121, 1998.

[Low04] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[LPK07] P. Labatut, J.-P. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, Rio de Janeiro, Brazil, Oct 2007.

[MAXIMA] MAXIMA. a Computer Algebra System. http://maxima.sourceforge.net.

[May99] H. Mayer. Scale-space events for the generalization of 3D-building data adjustment. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (IAPRS)*, pages 639–646, 1999.

[Mei75] G. Meisters. Polygons have ears. *American Mathematical Monthly*, 82:648–651, 1975.

[MICMAC] MICMAC, un logiciel pour la mise en correspondance automatique dans le contexte géographique. http://www.micmac.ign.fr.

[MV99] H.G. Maas and G. Vosselman. Two algorithms for extracting building models from raw laser altimetry data. *ISPRS Journal of Photogrammetry and Remote Sensing (IJPRS)*, 54(2-3):153–163, 1999.

[MWH+06] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. *ACM Transactions on Graphics, SIGGRAPH Proceedings*, 25(3):614–623, 2006.

[MZB+08] J. Milde, Y. Zhang, C. Brenner, L. Pluemer, and M. Sester. Building reconstruction using a structural description based on a formal grammar. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (IAPRS)*, volume 37 (Part 3B), Beijing, China, 2008.

[MZWVG07] P. Müller, G. Zeng, P. Wonka, and L. Van Gool. Image-based procedural modeling of facades. *ACM Transactions on Graphics, SIGGRAPH Proceedings*, 26(3):85, 2007.

[Nan06] L. Nanot. Détection de cheminées et de chiens assis à partir d'images aériennes à très haute résolution. Master's thesis, Université Paris-Est, 2006.

[NÖ03] S. Niskanen and P.R.J. Östergård. Cliquer user's guide, version 1.0. Technical Report T48, Communications Laboratory, Helsinki University of Technology, Espoo, Finland, 2003.

[ODZ07] M. Ortner, X. Descombes, and J. Zerubia. Building outline extraction from digital elevation models using marked point processes. *International Journal of Computer Vision (IJCV)*, 72(2):107–132, avril 2007.

[Öst02] P.R.J. Östergård. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120:197–207, 2002.

[PDP06] M. Pierrot-Deseilligny and N. Paparoditis. A multiresolution and optimization-based image matching approach: An application to surface reconstruction from SPOT5-HRS stereo imagery. In *Proc. of the ISPRS Conference Topographic Mapping From Space (With Special Emphasis on Small Satellites)*, Ankara, Turkey, feb 2006. ISPRS.

[RC98] R. Roy and I. J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, Bombay, 1998.

[Ris78] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[Rus09] D. Russel. Kinetic data structures framework. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.4 edition, 2009.

[SB03] K. Schindler and J. Bauer. A model-based method for building reconstruction. In *Proc. of the ICCV workshop on Higher-Level Knowledge in 3D Modeling and Motion (HLK)*, Nice, France, 2003.

[Seq] C. Sequin. Generalized Euler-Poincare theorem. http://www.cs.berkeley.edu/s̃equin/PAPERS/EulerRel.pdf.

[Sketchup] Google Sketchup. http://sketchup.google.com.

[Slo09] N.J.A. Sloane. The encyclopedia of integer sequences, 2009. http://www.research.att.com/ njas/sequences/A000108.

[SMG02] S. Scholze, T. Moons, and L. Van Gool. A generic 3d model for automated building roof reconstruction. In *Proc. of the ISPRS Commission V Symposium*, volume 34, pages 204–209, September 2002.

[Soh08] B. Soheilian. *Roadmark reconstruction from stereo-images of a mobile mapping system*. PhD thesis, Université Paris-Est, apr 2008.

[Sto91] J. Stolfi. *Oriented Projective Geometry: A Framework for Geometric Computations*. Academic Press, New York, 1991.

[SV04] I. Suveg and G. Vosselman. Reconstruction of 3D building models from aerial images and maps. *ISPRS Journal of Photogrammetry and Remote Sensing (IJPRS)*, 58(3-4):202–224, 2004.

[Tai05] F. Taillandier. Automatic building reconstruction from cadastral maps and aerial images. In U. Stilla, F. Rottensteiner, and S. Hinz, editors, *Proc. of the ISPRS Workshop on Object Extraction for 3D City Models, Road Databases and Traffic Monitoring - Concepts, Algorithms and Evaluation (CMRT)*, pages 105–110, Vienna, Austria, aug 2005.

[TD02] F. Taillandier and R. Deriche. 3D reconstruction of linear primitives from multiple images for urban area modelisation. In *Proc. of the ISPRS Symposium on Photogrammetric Computer Vision (PCV)*, Graz, Austria, 2002. ISPRS.

[TD04] F. Taillandier and R. Deriche. Automatic Builings Reconstruction from Aerial Images : a Generic Bayesian Framework. In *Proc. of the XXth ISPRS Congress*, Istanbul, Turkey, 2004.

[TMHF00] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–375. Springer Verlag, 2000.

[TP09] O. Tournaire and N. Paparoditis. A geometric stochastic approach based on marked point processes for road mark detection from high resolution aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing (IJPRS)*, 2009.

[TSP06] O. Tournaire, B. Soheilian, and N. Paparoditis. Towards a sub-decimetric georeferencing of ground-based mobile mapping systems in urban areas: matching ground-based and aerial-based imagery using roadmarks. In *Proc. of the ISPRS Commission I Symposium*, volume Part A, Marne-la-Vallée, France, jul 2006.

[TVWZ93] G.T. Toussaint, C. Verbrugge, C. Wang, and B. Zhu. Tetrahedralization of simple and non-simple polyhedra. In *Proc. of the 5th Canadian Conference on Computational Geometry (CCCG)*, pages 24–29, 1993.

[VT05] B. Vallet and F. Taillandier. Fitting constrained 3D models in multiple aerial images. In *Proc. of the British Machine Vision Conference (BMVC)*, Oxford, U.K., aug 2005.

[Wei85] K. Weiler. Edge-based data structures for solid modeling in curved-surface environments. *IEEE Transactions on Computer Graphics and Applications*, 5(1):21–40, jan 1985.

[WH] J.S. Warren and I.V. Hicks. Combinatorial branch-and-bound for the maximum weight independent set problem. (working paper).

[YL89] S.Y.K. Yuen and N.K.D. Leung. A shape-from-contour method for solid perception. In *Proc. of the 6th Scandinavian Conference on Image Analysis (SCIA)*, Oulu, Finland, June 1989.

Modélisation 3D de bâtiments

Recalage cinétique à topologie variable de toits polyédriques et
Reconstruction automatique de superstructures de toits

Il existe aujourd'hui une demande croissante pour des modèles numériques de ville de plus en plus précis. Alors que les travaux récents ont permis la production robuste de modèles polyédriques de bâtiments, les superstructures de toits telles que les cheminées et les chiens assis ne sont pas modélisées, et les erreurs géométriques et topologiques peuvent être importantes. L'approche itérative proposée affine géométriquement et sémantiquement un modèle de bâtiment approché sans superstructures, à l'aide d'un Modèle Numérique de Surface (MNS). Elle alterne la reconstruction de superstructures et le recalage des pans de toit principaux.

La détection et la reconstruction de superstructures sont basées sur une bibliothèque de modèles paramétriques de superstructures. Un ensemble de superstructures disjointes est recherché, en se réduisant au problème de recherche d'une clique pondérée maximale.

La phase de recalage tire parti des superstructures précédemment détectées afin de mieux estimer les pans de toit principaux. Elle corrige des simplifications tant géométriques telles qu'une symétrie erronée des toits, que topologiques telles que la fusion de sommets proches. Nous utilisons une représentation géométrique des bâtiments par les plans porteurs de chaque facette polyédrique, plus intuitive dans ce contexte que la représentation habituelle par la position de ses sommets. Nous introduisons le problème de triédralisation qui scinde les sommets surcontraints en sommets bien définis à l'intersection de 3 plans seulement. Nous proposons une structure de donnée cinétique garantissant des facettes non auto-intersectantes au cours de la réestimation itérative de leurs plans porteurs.

3D Building Modeling

Topology-Aware Kinetic Fitting of Polyhedral Roofs and
Automatic Roof Superstructure Reconstruction

There is nowadays a growing demand for increasingly more accurate 3D city models. Whereas recent works have lead to the robust generation of polyhedral building models, they do not model roof superstructures such as chimneys or dormer windows, and may feature large geometric and topological errors. We propose an approach to refine geometrically and semantically a superstructure-less approximate building model, using a Digital Surface Model (DSM). This iterative approach alternates between superstructure reconstructions and geometric fitting of the main roof planes.

Superstructure detection and reconstruction are based on a library of parametric superstructure models. A set of disjoint superstructures is searched to explain the height differences between the DSM and the building model, reducing the search to a maximum weighted clique problem.

The fitting step uses the previously detected superstructures to refine the main roof plane estimations. It corrects both geometric simplifications such as an erroneous roof symmetry, and topological simplifications such as the merging of close vertices of the polyhedral building model. The proposed representation of the building geometry uses the planes supporting each polyhedral facet, which is more intuitive in this context than the usual representation using the vertex locations. We introduce the trihedralization problem of splitting vertices that become over-constrained after updating their adjacent facet supports into well-defined vertices at the intersection of 3 planes. We propose a novel kinetic data structure that prevents facet self-intersections throughout the iterative reestimation of their supporting planes.