Real-time scene analysis for 3D interaction

Adrien Kaiser

université PARIS-SACLAY

NNT : 2019SACLT025

Thèse de doctorat

Analyse de Scène Temps Réel pour l'Interaction 3D

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom Paris

Ecole doctorale n°580: Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Traitement du signal et des images

Thèse présentée et soutenue à Paris, le 1er juillet 2019, par

ADRIEN KAISER

Composition du Jury :

Florence Tupin Professeure des Universités Télécom Paris (LTCI / IDS / IMAGES)	Présidente
Raphaëlle Chaine Professeure des Universités Université Claude-Bernard Lyon 1 (Dpt. Informatique / LIRIS)	Rapportrice
Franck Hétroy-Wheeler Professeur des Universités Université de Strasbourg (ICube UMR7357 / D-IR / IGG)	Rapporteur
Mathieu Brédif Chargé de Recherche IGN (LaSTIG / MATIS / GEOVIS)	Examinateur
Jean-Emmanuel Deschaud Maître de Conférences MINES ParisTech (CAOR / NPM3D)	Examinateur
Tamy Boubekeur Professeur des Universités Télécom Paris (LTCI / IDS / IMAGES / CG)	Directeur de thèse
José Alonso Ybanez Zepeda Directeur Technique Ayotle / Fogale Paris	Co-encadrant de thèse

Real-Time Scene Analysis for 3D Interaction

PhD Thesis

Adrien Kaiser

October 2015 - May 2019

ABSTRACT

In 2019, ten years after the announcement of project *Natal* at E3[1], the destinies of visual scene analysis and commodity depth cameras are strongly tied. The performance and accuracy of the former are crucial for mass development of its applications such as augmented reality, interactive games, 3D scanning, security, autonomous vehicles or household robots, to name a few. On the other hand, the latter provides easy and affordable access to 3D information of the surroundings through structured light, stereo-vision or time-of-flight technologies. The problem of converting this accessible data into knowledge and understanding of the scene has sparked interest among both *computer vision* and *computer graphics* research communities in the last ten years. Processing this organized 3D data set requires specifically tailored algorithms that also need to be efficient. Recent advances in scene analysis provide fast and robust ways to build a meaningful geometric and structural representation of observed scenes, localize sensors within that representation or acquire semantic information on surrounding objects.

In that regard, this PhD thesis lies at the frontier between *computer graphics* and *computer vision* and focuses on the use of 3D geometry analysis tools for visual depth data in terms of enhancement, registration and consolidation. In particular, we aim at showing how shape abstraction can generate lightweight representations of the data for fast analysis with low hardware requirements. This last property is important as one of our goals is to design algorithms suitable for live embedded operation in e.g., wearable devices, smartphones or mobile robots. The context of this thesis is the live operation of 3D interaction on a mobile device, which raises numerous issues including placing 3D interaction zones with relation to real surrounding objects, tracking the interaction zones in space when the sensor moves and providing a meaningful and understandable experience to non-expert users. Towards solving these problems, we make contributions where scene abstraction leads to fast and robust sensor localization as well as efficient frame data representation, enhancement and consolidation. While simple geometric surface shapes are not as faithful as heavy point sets or volumes to represent observed scenes, we show that they are an acceptable approximation and their light weight as geometric substitute makes them well balanced between accuracy and performance. In addition, they provide a consistent spatial support to process and analyze raw data.

[1]Kinect, the first consumer grade depth sensor: http://en.wikipedia.org/wiki/Kinect

RÉSUMÉ

Cette thèse se concentre sur l'analyse visuelle de scènes intérieures capturées par des caméras de profondeur. La performance et la précision de cette analyse sont cruciales pour le développement d'applications telles que la réalité augmentée, les jeux interactifs, la numérisation 3D, la sécurité, les véhicules autonomes ou les robots domestiques. Les capteurs de profondeur modernes offrent un accès aisé et abordable à une information tridimensionnelle de l'environnement. Le traitement de cet ensemble organisé de données 3D doit permettre sa conversion en information de haut niveau et compréhension de la scène mais nécessite des algorithmes spécifiques, qui doivent également être efficaces. Les avancées récentes en matière d'analyse de scènes fournissent des moyens rapides et robustes pour construire une représentation géométrique et structurelle des scènes observées, localiser des capteurs dans cette représentation ou obtenir des informations sémantiques sur les objets environnants.

Dans ce cadre, cette thèse se situe à la frontière entre l'*informatique graphique* et la *vision par ordinateur* et porte sur l'application d'outils d'analyse géométrique 3D à des données visuelles de profondeur en termes d'amélioration de qualité, de recalage et de consolidation. En particulier, elle vise à montrer comment l'abstraction de formes permet de générer des représentations légères pour une analyse rapide avec des besoins matériels faibles. Cette propriété est liée à notre objectif de concevoir des algorithmes adaptés à un fonctionnement embarqué en temps réel dans le cadre d'appareils portables, téléphones ou robots mobiles.

Le contexte de cette thèse est l'exécution d'un procédé d'interaction 3D temps réel sur un appareil mobile. Cette exécution soulève plusieurs problématiques, entre autres, le placement de zones d'interaction 3D par rapport à des objets environnants réels, le suivi de ces zones dans l'espace lorsque le capteur est déplacé ainsi qu'une utilisation claire et compréhensible du système par des utilisateurs non experts. Nous apportons des contributions vers la résolution de ces problèmes pour montrer comment l'abstraction géométrique de la scène permet une localisation rapide et robuste du capteur et une représentation efficace des données fournies ainsi que l'amélioration de leur qualité et leur consolidation.

Bien que les formes géométriques simples ne contiennent pas autant d'information que les nuages de points denses ou les ensembles volumiques pour représenter les scènes observées, nous montrons qu'elles constituent une approximation acceptable et que leur légèreté en tant que substitut géométrique leur donne un bon équilibre entre précision et performance. De plus, elles fournissent un support spatial cohérent pour traiter et analyser les données brutes.

Dans un premier temps, nous décrivons une approche utilisant des plans qui se recoupent dans plusieurs vues, afin de déterminer le déplacement d'un capteur de profondeur. La mise en correspondance de ces plans grâce à plusieurs heuristiques puis l'utilisation de leurs paramètres géométriques permet de retrouver l'ensemble des degrés de liberté du capteur.

Ensuite, nous définissons une superstructure composée de plusieurs objets géométriques de formes simples qui permet de modéliser, traiter et consolider les données du flux RGB-D à la volée. Cette structure est basée sur la cohérence de formes simples suivies dans le temps et l'espace, ce qui permet l'agrégation d'échantillons du capteur au sein d'un même accumulateur. Notre approche fournit plusieurs primitives de traitement qui améliorent la qualité du flux RGB-D ou allègent les traitements ultérieurs. Elle consiste à générer et mettre à jour un ensemble de statistiques locales compactes paramétrées sur des *proxies* géométriques de formes simples détectés dans la scène. En tant que substitut géométrique de la scène observée, l'ensemble de *proxies* peut également être utilisé pour compresser ou reconstruire la scène. Nous montrons plusieurs résultats sur des ensembles de données publics synthétiques et réels où notre modèle géométrique se montre robuste et cohérent dans la consolidation des données du flux, pour une performance adaptée à une exécution embarquée.

Contents

List of Figures

List of Tables

I INTRODUCTION

I.1 Context and Motivation

I.1.1 Context

At the start of this PhD thesis, in late 2015, Paris-based computer vision software company *Ayotle* was working on a new project that would be unveiled a year later as *Hayo*[1]. While having developed computer vision based interaction tools on desktop platforms for multiple industries in the last five years, the company was willing to bring its technology to general consumers through a dedicated device. At the core of this device, the custom interaction algorithm would run seamlessly on an embedded platform to enable live control of connected devices, as detailed in section I.1.3. In 2015, the efficient implementation of the interaction algorithm fed by an RGB-D stream from commodity sensors was already running in real-time on a *Raspberry Pi* platform[2].

This transfer from desktop to embedded platform leads to multiple problems that are at the origin of this PhD thesis. The first issue that appeared is the spatial consistency needed for the interaction software to run. As explained in section I.1.3, the interaction is based on user-defined local zones, fixed with relation to the sensor, which is not an issue for desktop operation, where the setup has to be done only once. But an embedded device can be moved around, and in that case the positions of interaction zones with relation to the sensor have to be recomputed, in order to be at the same real physical locations as they were before sensor motion. Hence, tracking the sensor in the scene was the first goal of this PhD and had to be achieved with the embedded constraints detailed below in section I.1.4.

[1] https://twitter.com/meethayo/status/784068130205229056
[2] https://www.raspberrypi.org/

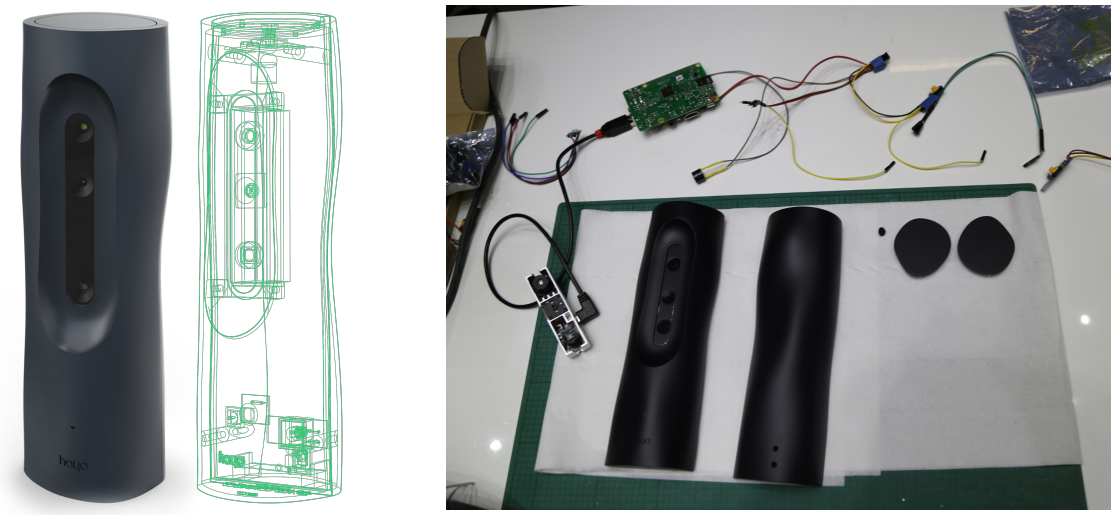


Figure I.1: Left: *Hayo* sensor photo and case wireframe (2016). Right: Breakdown of the sensor with case and RGB-D sensor connected to a *Raspberry Pi* computer (2016). More information on the *Hayo* project at www.hayo.io.

Second, the device is only an RGB-D sensor connected to an embedded computer, enclosed in a case, as shown in Figure I.1. As it has no graphical interface and one is needed for the user to manually setup the interaction zones as explained in section I.1.3, the team developed a mobile application that displays the live stream of the sensor to enable the user to place these zones. However, the sparse but heavy content of the sensor image is hard to stream and meaningfully display for non experienced users. We therefore imagined that abstracting the observed scene with simple geometric shapes would allow streaming and displaying both lighter and more understandable information. Eventually, the construction of a complete while compact representation of the user's environment based on this high level abstraction seemed appealing as it would enable development of new advanced features on the embedded device, such as automatic setup of interaction zones by analyzing the structure and semantic of surrounding objects.

In light of these different issues and opportunities, and after discussions between *Ayotle* and the *computer graphics* group at *Telecom Paris*, this PhD title was defined as **Real-Time Scene Analysis for 3D Interaction**. Its goal, further developed in section I.2, is broadly to *perform lightweight scene analysis on RGB-D data to enable real-time interaction on embedded devices*, through shape abstraction.

The next sections detail the different components of the problem:

- accurate visual scene analysis (section I.1.2);
- embedded 3D interaction (section I.1.3);
- requirements of live performance (section I.1.4).

I.1.2 Visual Scene Analysis

As we saw in the last section, the efficient operation of a 3D interaction framework within an embedded mobile device could greatly benefit from live scene analysis. Here, we present the problem of 3D scene analysis, at the crossing between *computer vision* and *computer graphics*, whose goal is to provide a meaningful 3D representation of a scene captured by a visual sensor. To be more specific, a 3D scene analysis framework would analyze a set of images and attempt to recover any of:

- geometric relations between sensor positions, if multiple, such as position and orientation;
- geometric information on the components of the real scene observed by the sensor, such as distance, size or motion;
- structural information and geometric relations between static or dynamic components;
- semantic information on observed scene parts.

Modern visual methods for 3D modeling of scenes make use of all recent acquisition modalities, leaning towards embedded operation for interactive mobile applications. Among them, commodity RGB-D cameras, as presented in section II.1, become more and more accessible and provide partial geometry information in addition to a regular color image. The data provided by the sensor or generated by the system can either be composed of color images, depth maps, dense and unstructured 3D point clouds or connected polygonal meshes, as detailed in section II.4.1.

The problem of relative localization of multiple sensor acquisitions is detailed in section II.2. The scene reconstruction process, aiming at aggregating multiple visual observations into a global geometric model, is then detailed in section II.3. In particular, *simultaneous localization and mapping (SLAM)* systems attempt to solve both problems simultaneously. From a stream of images, geometric relations between nearby frames are estimated to initialize the scene representation. Their samples are then consolidated into a global map model, which contains more complete features from the scene that can be tracked in subsequent frames [KM07].

In this thesis, we focus on a certain type of scene representations, in order to satisfy our initial performance requirements as presented in section I.1.4. We aim at building a high level abstraction of the scene, composed of simple geometric elements that have a structural meaning. While most modern reconstruction methods use volumetric representations, we make the observation that visual sensors provide samples of object surfaces and not of their volume. In consequence, we think that representing non-deformable real scene elements with compact geometric surface shapes allows building a model of acceptable quality with less impact on computing resources. In other words, we want to apply advanced analysis and reasoning to build a virtual structural and semantic understanding of the surroundings which would be similar to human visual perception of the world, where surface elements of the representation are intuitive and make more sense from an operational point of view.

I.1.3 3D Interaction with RGB-D Cameras

In the context of this thesis, we refer to 3D interaction[3] as human-machine interaction in 3D space. Analyzing the output of commodity 3D sensors, described in section II.1.1, allows defining a 3D user interface, where locations and motions of objects and people in 3D space have impact and control on real or virtual entities. The most famous instance of 3D interaction as we mean it is probably the learning-based skeleton tracking embedded in the *Kinect* camera [SFC+11], unveiled in 2011. Existing video games based on 3D human skeletons became more robust and stable when this new technology was integrated into the *XBox* video game console. The original paper is based on the structured light depth sensing technology initially patented by *Primesense* in 2007 [SZ07].

As any user interface, a 3D interaction system must provide real time feedback to users for them to intuitively understand the actions triggered by their motion. In practice, we observe that even short delays in interaction feedback can confuse users about their communication with the interface. This requirement was tackled by the *Kinect* through custom hardware components, however real time skeleton tracking is a challenging problem whose solution is hard to generalize to more complex setups, such as small gestures, multiple people tracking or fast motion [EAG17]. In addition, most robust recent methods are based on neural networks [AII+18, XWW18] and are not adapted to embedded platforms.

[3] https://en.wikipedia.org/wiki/3D_interaction

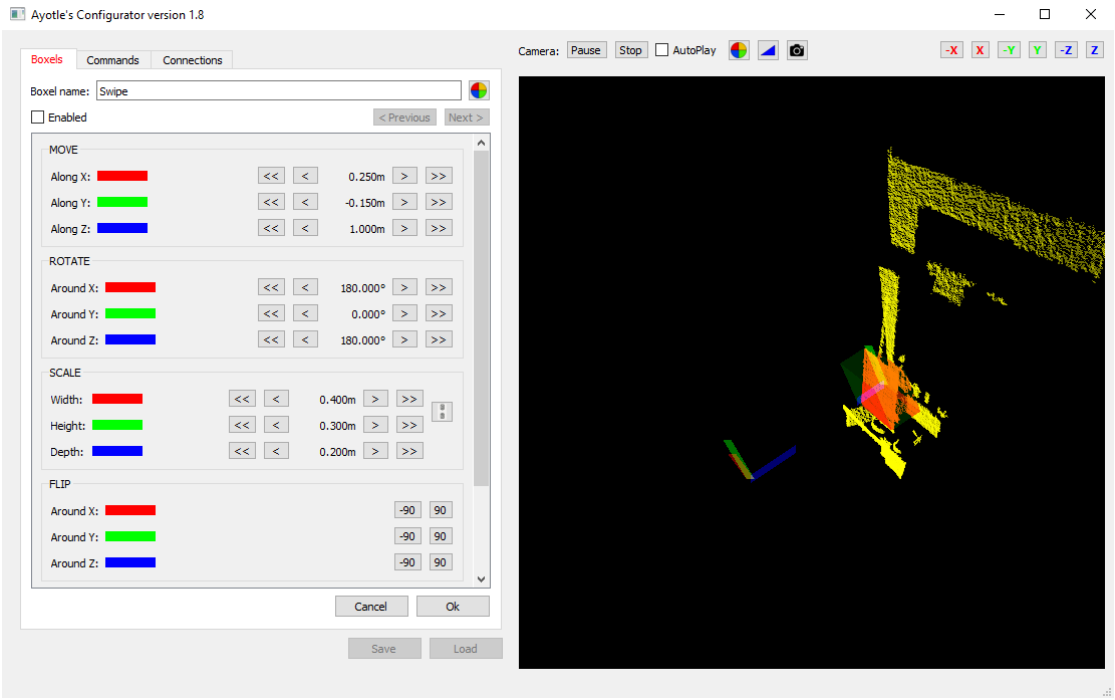


Figure I.2: Ayotle's *Configurator* software interface (2015) allows an operator to manually place virtual interaction zones and configure the actions triggered with the activation of such zones. This desktop application is however hard to manipulate for inexperienced users and requires tedious setup time.

Ayotle's Skeleton-free 3D Interaction

In order to overcome issues raised by the real time construction and operation of a virtual human skeleton, a new method was developed by *Ayotle*, based on geometric analysis of the depth sensor output [YZ13b, YZ13a]. The system is based on local 3D activation zones represented by geometric objects such as boxes, spheres or cylinders having different functionalities. However, in contrast to skeleton based approaches which rely only on the relative positions and orientations of body parts, the interaction zones are defined based on surrounding objects with a fixed absolute position with relation to the sensor. Hence, setting up the interaction zones for a given application requires tedious manipulation of custom 3D software allowing the user to manually define the type and shape of the interaction as well as the position and orientation of the geometric activation zone. In practice, interaction zones are setup by an operator using *Ayotle's Configurator* software, as shown in Figure I.2.

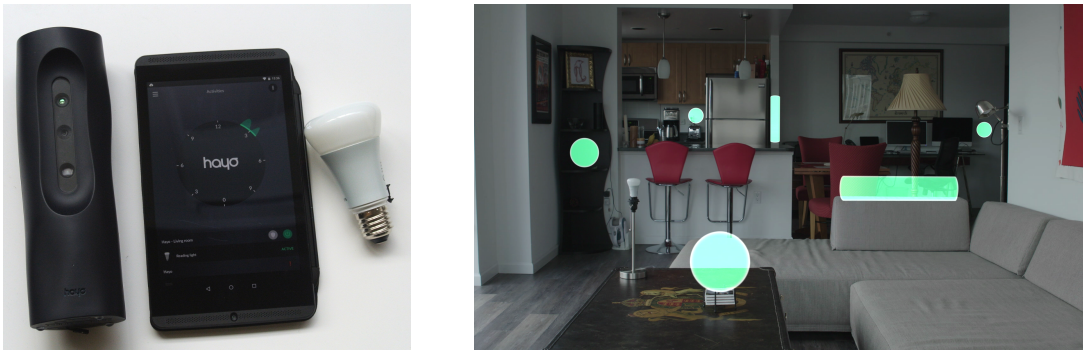


Figure I.3: *Hayo* camera, application and interaction zones for the connected house (2016). The *Hayo* sensor, paired with its mobile application (left), allows users to place virtual interaction zones represented geometrically by spheres or cylinders anywhere in their house (right), to control connected devices such as e.g., *Philips Hue* smart bulbs.

3D Interaction in the Connected Home

With the rise of connected objects in the 2010s, applying *Ayotle*'s efficient 3D interaction technology to the control of these objects appeared as a good way to broaden the audience of the tool. Hence, in 2015, prototypes of embedded interaction for project *Hayo*[4] were built and the algorithm was optimized to run in real time on a *Raspberry Pi 2* platform. In addition, embedding the interaction technology in a mobile device enables many more applications than the fixed desktop computer setup. Combined with a mobile application designed to place and configure virtual interaction zones in the same spirit as with the *Configurator*, the sensor provides inexperienced users with an intuitive way to enable 3D interaction with connected objects anywhere in their house, as shown in Figure I.3.

I.1.4 Performance Requirements

As we saw in the last section, intuitive 3D interaction with RGB-D cameras requires high performance processing to achieve real time operation. The embedded platform also adds a constraint on the available computer memory that the algorithm can use, despite the recent advances of the hardware in smartphones as well as single-board computers, such as the *Raspberry Pi 3*[5] or the *ASUS Tinker Board*[6]. In addition, providing 3D interaction technology to general consumers raises privacy issues, where visual data from the sensor should be kept within the device and not being transfered on an external server where the user looses control over it. This enforces complete onboard processing of the incoming data stream.

[4] www.hayo.io
[5] https://www.raspberrypi.org/
[6] https://www.asus.com/Single-Board-Computer/Tinker-Board/

From a computing point of view, the scene analysis processing should not disturb the 3D interaction computations in order to keep real time operation and user feedback. In that sense, dividing the processing between available computing cores provides some independence between tasks. However, it is important to keep in mind that the lower the requirements to analyze the scene, the larger the available resources to enable more complex reasoning and functionality.

Hence, this PhD thesis enforces several design goals to satisfy its performance requirements:

- reason on the scene representation to find the most efficient model for both construction and operation;
- moderate the algorithm's impact on resources in terms of memory, bandwidth and CPU time;
- allow extension of functionality by providing meaningful high level representation of the 3D space.

More generally, our work was oriented from the start towards the design of a light framework, which in our vision comes down to the following paradigms:

- defining light data structures, taking into account the weight of each member;
- enforcing efficient memory management and freeing;
- avoiding data copy and transfer as much as possible;
- reducing dependencies to external tools;
- developing only the needed functionality.

I.2 Objectives

Our main goal is to create a scene analysis and abstraction methodology based on RGB-D data and geometric shapes which is robust within the context expressed in the last section, in order to enable and enhance live 3D interaction on mobile devices. Such a high level interpretation would provide the system and user with a simple perception of the surroundings, hence easy to update and operate. In particular, we want to be able, at any time in the processing, to:

- localize an RGB-D sensor with relation to a given representation of the surroundings, including data from another RGB-D sensor;
- display a meaningful and as complete as possible 3D model of the observed scene understandable by non-expert users;
- stream the model to a remote client with as few delay as possible.

I.2.1 Research Axes

In order to come closer to that goal, we wish to study three research axes. For each of them, the proposed solution will have to stay robust to illumination changes as well as dynamic elements and static change of position of objects. The axes are:

- fast relative sensor localization;
- lightweight while faithful representation of indoor scenes, in particular with knowledge of physical holes and adjacent elements;
- live processing and analysis of the stream from RGB-D sensors;

These axes are related through the need for spatial and structural knowledge of the environment. Knowing the pose of the sensor at different locations is needed to enforce temporal and spatial consistency of geometrical objects detected in the data. This consistency is the key to the accumulation of samples from multiple observations to build an accurate representation of the scene. Such a geometric accumulator will allow improving the quality and completeness of the stream, which itself will improve the localization from frame to frame.

I.2.2 Industrial Expectations

In the context of this thesis, the performance and robustness of the algorithm are key for a seamless integration into a commercial product. Hence, throughout the document, the design of the different tools is thought as to handle ”real life” constraints such as lifelong[7] operation and presence of dynamic elements, computing power and bandwidth limitations as well as an experience tailored for non-expert users.

In addition, the privacy issues raised in section I.1.4 where users should keep control over their personal data follow the *General Data Protection Regulation*[8] from the European Union implemented in May 2018, in which any use and export of personal data should be justified. In that sense, we will enforce ”privacy by design” by fully processing all visual data on the fly onto the embedded platform, which avoids the need to store or send any private data beyond its processing.

[7] By lifelong, we mean that the acquisition of the data is not constrainted in the time frame of a handheld capture, but rather spans an unknown number of hours or days, where we do not control the modification of content in the scene.

[8] https://eugdpr.org/

I.3 Organization and Contributions

I.3.1 Contributions

As presented in the last section, the objective of this thesis is to generate a 3D representation of a scene, live from an RGB-D stream, whose operation and analysis would enable the definition and tracking of 3D interaction zones on a mobile device. In this document, we present four contributions towards that goal:

- a study and classification of state-of-the-art methods for geometrical approximation of 3D data by simple shapes (section II.4);
- a novel method that leverages planar representation and consistence of scene structure to register overlapping RGB-D frames (chapter III);
- a lightweight and consistent geometric superstructure to model structural elements of captured indoor scenes (chapter IV);
- an efficient framework for live enhancement and consolidation of RGB-D streams (chapter V).

The first contribution was published as a state-of-the-art report in *Computer Graphics Forum* [KYZB19]. The second contribution was unveiled as a patent submitted in February 2019 [KYZBC19]. The last two contributions were presented in 2018 at the *European Conference on Computer Vision (ECCV)* [KYZB18]. A complete list of scientific communications produced during this PhD is available in section A.2.

I.3.2 Organization

This document is organized in four parts, followed by a general conclusion in chapter VI.

First, chapter II presents the state-of-the-art in domains linked to our objectives. Fundamentals of RGB-D data, its limitations and existing methods to enhance it are shown in section II.1. Methods to locally and globally register RGB-D frames are presented in section II.2. Methods to build complete 3D reconstructions modeling acquired scenes are presented in section II.3. Eventually, section II.4 summarizes our study [KYZB19] on recent methods to detect simple geometric primitives in captured 3D data.

The second contribution of this thesis is then presented in chapter III, namely a new method to register together pairs of 3D views, stored as RGB-D frames or point clouds. Our plane matching strategy is presented in section III.2, while our plane-based motion computation is detailed in section III.3. Experiments with each method are shown in section III.4, based on a toy example and a public online benchmark dataset.

Our geometric superstructure to model incoming RGB-D streams is detailed in chapter IV, with information on their live construction and update (section IV.2), stored statistics (section IV.3) as well as performance results and examples (section IV.4).

Applications of this superstructure are shown in chapter V, with both live frame-wise improvements (section V.2) and a method to consolidate the stream into a global regular model (section V.3).

II BACKGROUND

As presented in the introduction, this thesis aims at introducing geometry analysis into some essential *3D computer vision* tasks presented in section I.1.2. In this chapter, we present the different problems arising from the following tasks and list state-of-the-art methods to solve them:

- processing raw RGB-D frames from the sensor;
- registering multiple 3D images into a global model;
- building a complete 3D model of the observed scene;
- creating a high level representation of the scene using primitive shapes.

II.1 RGB-D Data

This thesis focuses on the processing of raw RGB-D data into meaningful scene information through the application of geometrical tools. This section broadly presents the specifics of RGB-D data, its limitations and the different existing methods to solve them, as well as its applications.

II.1.1 Fundamentals of Depth Data

Brief History of Depth Sensing

3D capture first appeared during the 1960s with tedious techniques based on lights, cameras and projectors [Ebr15]. At this time, most 3D scanning devices made use of physical contact probes. With the evolution of electronics and computer science in the mid-1980s, efficient optical methods for 3D scanning were developed. Devices based on point, area or stripe lighting allowed faster generation of 3D images. One of the first commercial successes of the stripe 3D scanners was the human body modeling in the animation industry in early 1990s.

In the following years, several expensive professional laser range scanners were released, such as the *REPLICA* and *ModelMaker* sensors and eventually the *Faro Focus*. In the late 2000s, multiple small time-of-flight sensors were available such as the *Mesa Imaging SwissRanger* and the first *pmd* sensors, although their price range was around 10 000 $.

2011 saw the release of the first *Kinect* consumer grade RGB-D camera, based on structured light technology fom *Primesense* [SZ07], allowing broader availability of 3D scanning. While discontinued since 2017, the *Kinect* project sparked interest in the commodity RGB-D sensor market, with the development of many applications and the release of affordable sensors from e.g., *Intel*[1] or *Texas Instruments*[2]. Although research was conducted using depth sensors before 2010, the availability of such sensors to general consumers produced a growing interest among the *3D computer vision* community.

More recently, the *Google Tango* platform, available from 2014 to 2018, was the first globally available embedded device with integrated RGB-D sensing based on time-of-flight technology from German manufacturer *pmd*[3]. This opened the way to RGB-D sensor integration in modern smartphones such as recent devices from e.g., *Lenovo*[4] or *Honor*[5].

Data Structure

RGB-D cameras provide two images. The first one is a regular 3-channel color image containing texture information about the observed scene. The second one is a depth image, also called depth map, where each single channel pixel encodes the distance from the camera to the viewed object on 8 to 16 bits. This part of the data set allows extracting structural information from the scene. In most consumer depth cameras, color and depth sensors have different intrinsic parameters, hence the need for a registration step between the two modalities before any processing can be done. The quality of this registration plays an important role in the shift between depth and color and can lead to misalignment. Given the intrinsic parameters of both sensors, it is possible to unproject the data and generate colored 3D points for all depth points to create a 2.5D point cloud. In that case, the single view capture only provides parts of the observed 3D scene and the data is incomplete. Figure II.1 shows an example of data contained in an RGB-D image.

[1] https://www.intelrealsense.com/
[2] http://www.ti.com/sensors/specialty-sensors/time-of-flight/overview.html
[3] https://www.pmdtec.com/news_media/press_release/google_tango_tablet.php
[4] https://www.lenovo.com/gb/en/tango/
[5] https://www.hihonor.com/global/products/smartphone/honorview20/

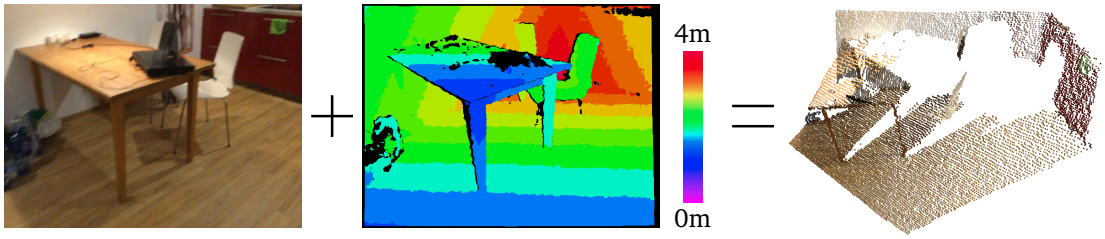


Figure II.1: Data structure of an RGB-D image. An RGB-D image is composed of a regular 3-channel color image (left) and a 1-channel depth map (middle) encoding pixel-wise distances from the scene to the sensor. They allow building a partial 3D point cloud of a single view, also called 2.5D point cloud (right).

Depth Acquisition

The acquisition of this data is possible through different technologies [AFT14] such as *structured light*, *time-of-flight*, active and passive *stereo vision* or *LIDAR*. We present here the three main technologies used in commodity depth sensors. A more complete comparison is provided by *Texas Instruments* [Li14] (Table 1).

Structured light (also known as *light coding*) depth sensing is composed of an infrared (IR) projector and an IR sensor. The projector sends a known pattern onto the scene, which is captured by the sensor. Patches of the returned warped pattern are then matched to reference patterns to estimate the depth at each pixel. Even though the use of IR allows capturing information with no visible light, such technology is very sensitive to strong sunlight.

Time-of-flight depth sensing is made possible by computing distance based on the time needed for a light signal to hit a surface and come back to the sensor. As there is no need for computations to get the depth values from the signal, the capture is faster than with computational methods. In addition, this technology is less sensitive to sunlight than a structured light sensor, however its sensing range is usually lower for the same similar input power, as the light source has high power requirements.

Stereo vision emulates human vision by matching patterns in two calibrated images. Prior calibration allows more robust matching of features for which the disparity is estimated. However, finding correspondences in images can be challenging, in particular with repeated patterns or low textured areas. *Active* stereo vision makes use of infrared signal to improve the observed features by projecting a highly discriminative pattern read with an IR sensor. This allows finding more correspondences, however adding a requirement for power, which then controls the range of the IR projection and capture.

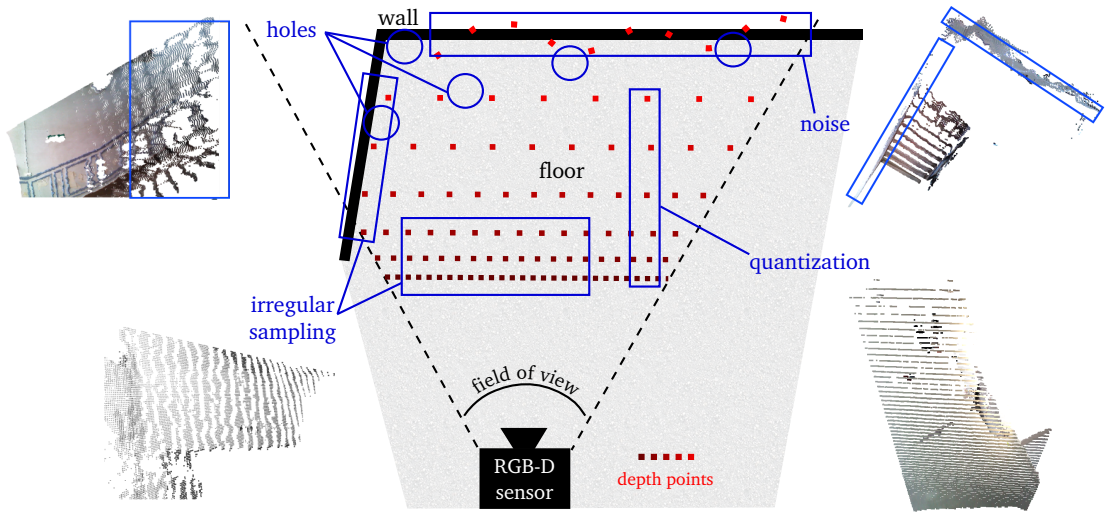


Figure II.2: *Geometrical limitations of depth sensing.* We show a synthetic indoor scene with floor and wall corner, seen from the top and sampled by a depth sensor. Due to quantization of depth information, irregular sampling appears over the surfaces. Instability of the sensor signal creates noise, especially farther from the sensor where the signal is weaker. Holes in the data can be due to sensor topology and also specular elements or light sources which confuse the signal. In addition, depth data is highly temporally inconsistent.

II.1.2 RGB-D Data Limitations

In spite of their affordability, and although they are constantly improving, the impact spectrum of consumer depth cameras is limited by the low quality of their RGB-D stream. This mostly originates in the low resolution of the frames and the inherent noise, incompleteness and temporal inconsistency stemming from single view capture. In particular, we list the following issues, summarized in Figure II.2.

- noise and temporal flickering;
- holes and missing data;
- irregular sampling;
- heavy memory footprint.

Both axial and lateral noise appear when sensing depth and their amplitude increases further from the axis or the origin of the camera [NIL12]. This also generates temporal inconsistencies, especially on the edge on objects.

Holes in RGB-D data are created by the range limits and high noise levels of depth sensing. Given the material of observed objects and the type of technology used, some surfaces are harder to detect. The orientation of the surface with regards to the sensor and the perturbations due to light sources can also lower the quality in some areas.

The inherent data structure of RGB-D data can also be an issue, as it represents many pixels to store with a potentially important size, given the accuracy used by the sensor. This makes the memory footprint of real-time RGB-D data acquisition a challenge that has to be dealt with, possibly by abstracting the dataset into a lighter data structure.

II.1.3 RGB-D Data Processing

Since the appearance of commodity depth sensors, a number of methods have been developed in order to improve the low quality of their generated images. In particular, we present methods developed to:

- remove noise and temporal flickering;
- fill holes and complete missing data;
- resample depth points and control point density;
- compress the heavy amount of incoming data;
- use planar abstraction for data enhancement.

Depth maps can be denoised using spatial filters [Li16] e.g., Gaussian, median, bilateral [TM98, SHKZ14, EGD+12], adaptive or anisotropic [LCK16, LJW14] filters, often refined through time. Other methods include non-local means [BRR15], bilateral filters with time [EGD+12], Kalman filters [CS12], over-segmentation [SJ13] and region-growing [CLL12]. Depth sensors manufacturers such as *pmd* also develop custom filtering methods allowing to increase signal noise ratio at far distance, such as adaptive *high dynamic range* [CHL15].

They can be upsampled using cross bilateral filters such as *joint bilateral upsampling* [KCLU07] or *weighted mode filtering* [MLD12]. Such methods are particularly useful to recover sharp depth regions boundaries and enforce depth-based segmentation. In particular, Wu et al. [WZN+14] present a shape-from-shading method using the color component to improve the geometry, which allows adding details to the low quality input depth. They show applications of their method to improve volumetric reconstruction on multiple small scale and close range scenes.

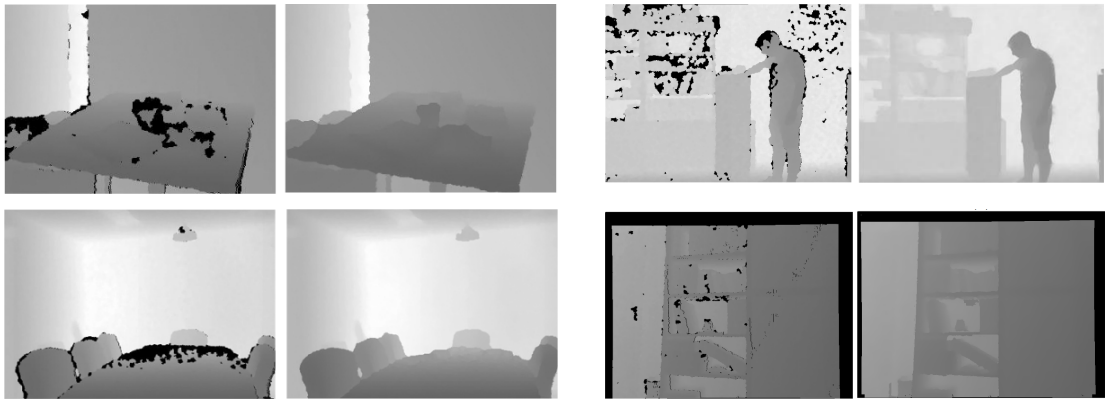


Figure II.3: Depth improvement methods. Spatial filters [BRR15] (left) allow denoising and filling holes in depth maps. Bilateral filters [LJW14] (top right) and over-segmentation [SJ13] (bottom right) allow filling holes and upsampling the data. Images taken from papers.

In order to fill holes in the depth component, one can use the same spatial filters as those used for denoising [BRR15], or morphological filters [LCK16, LJW14]. Inpainting methods [LDY+16], over-segmentation [BDTL15] or multiscale [SA12] processing are also used to fill holes for e.g., *depth image-based rendering* (DIBR) under close viewing conditions. Figure II.3 shows examples of depth map improvement methods.

A set of 3D planes offers a faithful yet lightweight approximation for many indoor environments. Only a few methods have used planar proxies as priors to process 2.5D data, with in particular Schnabel et al. [SDK09] who detect limits of planes to fill holes in static 3D point clouds. *Fast sampling plane filtering* [BV12] detects and merges planar patches in static indoor scenes. The detected planes allow filtering the planar surfaces of the input point cloud, however the primitives are quite sensitive to the depth sensor noise and lack spatial consistency.

II.1.4 Applications of RGB-D Data

Direct applications of RGB-D data streams provided by both high end and commodity depth sensors are at the core of many research efforts in *3D computer vision*. In particular, we focus in this thesis on three main topics:

- pairwise and global sensor registration to recover the relative position and orientation of multiple sensor acquisitions with relation to each other;
- full reconstruction to build a consistent and accurate geometrical 3D model of the observed scene;
- scene abstraction and generation of a high level model with geometric, spatial and semantic meaning.

As it provides fast visual knowledge of the surroundings, the real time RGB-D stream output of modern commodity consumer depth cameras is widely used for indoor 3D capture. It can feed a growing set of end applications in domains ranging from modeling to robotics, through gaming, such as:

- augmented reality;
- human computer interaction;
- 3D reconstruction;
- industrial design;
- motion tracking;
- autonomous robots.

II.2 Indoor Scene Registration

This section surveys the existing methods to register multiple overlapping 3D images of an indoor scene. We will focus on four categories of methods:

- pairwise registration of overlapping depth and RGB-D images;
- pairwise registration of overlapping point sets;
- global registration of multiple 3D images;
- registration based on planar constraints.

II.2.1 Pairwise RGB-D Frames Registration

First, we focus on pairwise registration of overlapping views of a scene acquired by an RGB-D sensor. The input is composed of two single RGB-D images of an indoor scene sampling overlapping objects. The goal is to estimate the rigid transformation matrix that transforms the first scan into the second scan. It also corresponds to the motion of the sensor between the two acquisitions. In a recent survey, Morell-Gimenez et al. [MGSCVM+18] divide these methods into two categories. First, the construction of sparse feature descriptors at 2D or 3D point locations and their matching using RANSAC [FB81]. Second, the dense registration methods pioneered by *Iterative Closest Point (ICP)* [BM92] and applied to depth data through the *KinectFusion* [NIH+11] line of work.

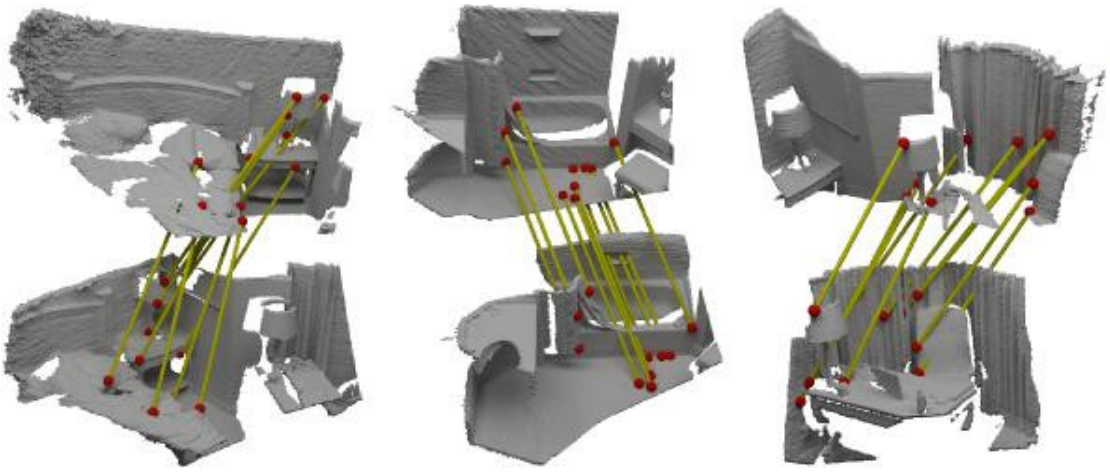


Figure II.4: *3DMatch* descriptors matched between different RGB-D views. We notice that most detected locations correspond to highly descriptive parts of the scene. Image taken from Zeng et al. [ZSN+17].

Here, we only detail methods dedicated to RGB-D frames, but one may also use point cloud methods, as described in section II.2.2, to register depth maps. Most methods aim at matching 3D locations between the two views and then compute the motion matrix by minimizing their Euclidean distance in the *least squares* sense [Ume91]. In order to acquire corresponding 3D locations, a first range of methods uses the color component and known 2D keypoint detectors and descriptors. After matching 2D descriptors in the color component such as SIFT [Low99], SURF [BTVG06] or ORB [RRKB11], the corresponding 3D locations can be directly read from the depth component. While the use of known 2D descriptors opens the range of available tools, the color component is sensitive to illumination changes, which can confuse the matching of such features.

Recent methods leverage the global availability of RGB-D reconstruction datasets to learn local 3D feature descriptors. In particular, *3DMatch* [ZSN+17] descriptors are learned through a self-supervised volumetric convolutional neural network. Ground truth correspondences are acquired using existing reconstructed models of RGB-D data and the definition of volumetric patches allows computing strong local descriptors that can be matched using RANSAC, as shown in Figure II.4. While the *3DMatch* descriptors show robustness in a variety of scenarios, it is important to note the high requirements of the method in terms of hardware and execution time.

II.2.2 Pairwise Point Cloud Registration

In this section, we present more general registration methods applied to 3D point sets. On one hand, efficient 3D descriptors can be matched between point sets and the transformation matrix can be estimated as for RGB-D frames. On the other hand, the family of *Iterative Closest Point (ICP)* variants allows accurate registration at the cost of multiple iterations over the full data.

3D Descriptors

Visual descriptors aim at describing features in visual data in terms of shape, color or texture. By analyzing the data at specific locations and their neighborhoods, they build a list of local characteristics to uniquely identify parts or elements. Here, we present the most used 3D descriptors used in the process of pairwise point cloud registration. For a comprehensive study, we point the reader to the 2016 survey from Guo et al. [GBS+16].

In 1999, Johnson and Hebert present the *spin images* [JH99] with the goal of recognizing 3D objects. At selected 3D oriented point locations, a local image plane is defined using cylindrical coordinates, as shown on Figure II.5. Projecting nearby points onto the virtual image plane allows aggregating local geometry information into a 2D, simple to compare image grid.

In 2008, Rusu et al. [RBMB08] define the *point feature histogram (PFH)* descriptor to locally encode the geometrical properties of a 3D point's neighborhood. By computing local deviations in surface orientation, they build a high dimensional histogram that describes well the local curvature and its variations. Although, as the base information used is the surface normals, the robustness of the method highly depends on their quality.

In 2008, Aiger et al. present a method to match congruent groups of 4 points in *4PCS* [AMCO08]. By designing local metrics for a group of 4 points and analyzing their geometry, they are able to reach a high level of description even in the presence of noise and outliers, as shown in Figure II.6. This descriptor was then optimized by Mellado et al. in *Super4PCS* [MAM14] to reach linear complexity.

In 2010, Drost et al. present a new local geometric descriptor called *point pair features (PPF)* [DUNI10]. A *PPF* is defined for two oriented points to describe their relative position and orientation and an accumulator space allows fast comparison between point pairs. A recent paper presents *PPFNet* [DBI18], a learning approach to the detection and estimation of *PPF* descriptors.

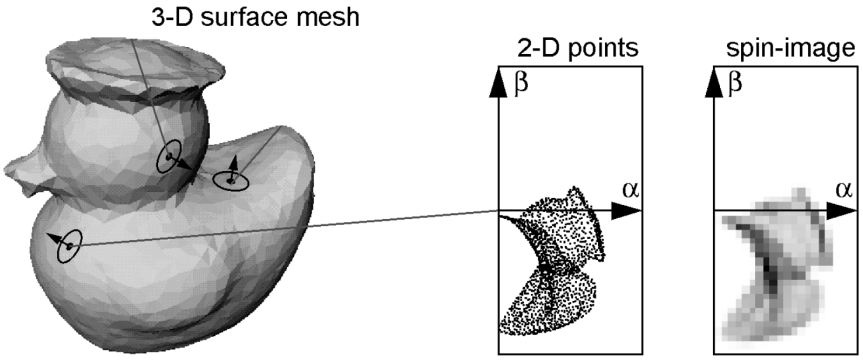


Figure II.5: *Spin image* defined at an oriented point location on a 3D surface. The projection of nearby geometry onto a local 2D patch allows easy comparison of transformation invariant descriptors. Image taken from Johnson et Hebert [JH99].

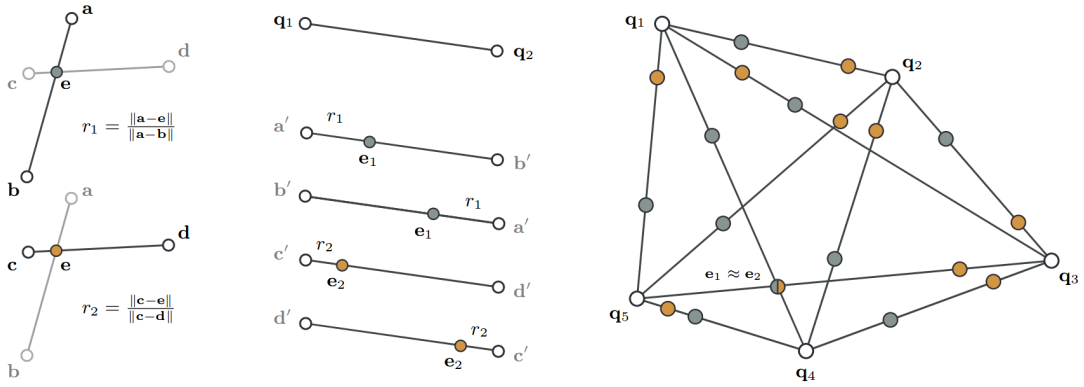


Figure II.6: *4PCS*: Extraction of congruent groups of 4 oriented 3D points. The wide baseline allows more accurate registration even in the presence of noise and outliers. Image taken from Aiger et al. [AMCO08].

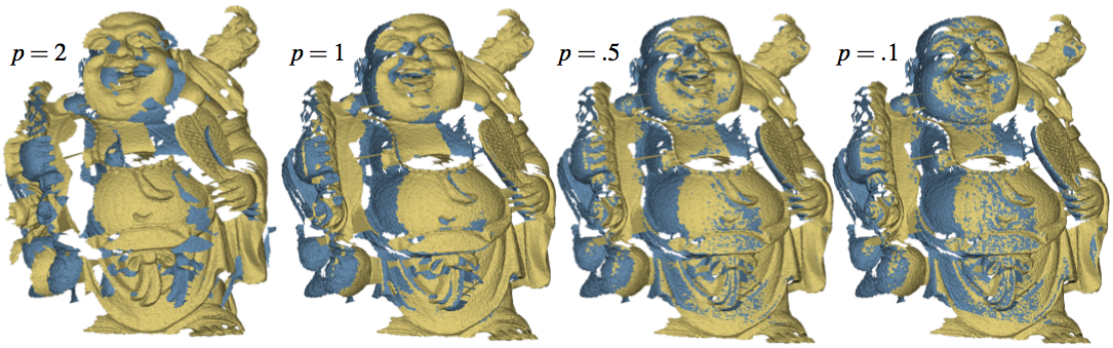


Figure II.7: Registration of partial point clouds using *SparseICP*. The p value is the norm used and $p = 2$ is the traditional least-squares *ICP*. The sparse optimization with $p \in [0, 1]$ allows detection and removal of outlier 3D points. Image taken from Bouaziz et al. [BTP13].

Iterative Closest Point

The *iterative closest point (ICP)* methodology was first described by Besl and McKay in 1992 [BM92] with the goal of registering general 2D and 3D geometric objects of multiple representations. *ICP* is widely used to register 3D point clouds and has seen numerous extensions. The first one was its combination with the point-to-plane distance defined by Chen and Medioni [CM91], by replacing the original point-to-point metric by the distance between a 3D point and the tangent plane at its corresponding oriented point.

Generalized ICP by Segal et al. [SHT09] as well as Viejo et al. [VHC08] define a *plane-to-plane ICP* by considering both source and target point clouds as oriented, thus having tangent planes at each location. Finally, *Sparse ICP* by Bouaziz et al. [BTP13] is an efficient variant of the original method, where outliers of the transformation are detected and removed during the process, as shown on Figure II.7.

For more details on *ICP* variants, a study was published by Rusinkiewicz and Levoy [RL01]. More recently, Bellekens et al. [BSBW14] also compared *ICP* variants to registration based on *principal component analysis* and *singular value decomposition*.

II.2.3 Offline Global Registration

While we have seen state-of-the-art methods to rigidly register pairs of overlapping RGB-D frames or point clouds, we now focus on the use of multiple images to register them together into a global model of the scene. Processing the full acquired collection of frames allows building a more complete model where missing data from some views is filled by other views. We consider offline global registration, in opposition to online reconstruction as presented in section II.3.

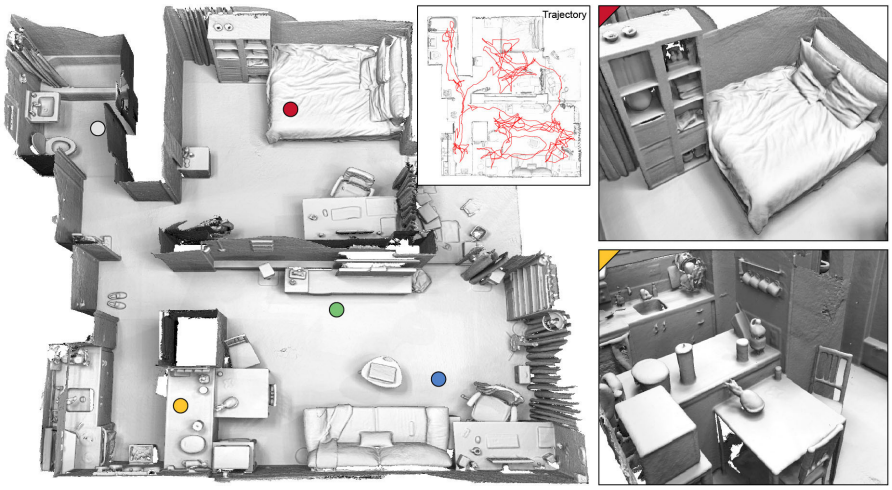


Figure II.8: Global offline reconstruction results. By aggregating many different overlapping views of the scene into a global optimization framework, the method is able to generate high quality geometry of the observed scene, with very few missing parts. Image taken from Choi et al. [CZK15].

Most methods first consider an initial pairwise transformation as previously described. Then, different strategies allow aggregating the information from all registered frame pairs into the global model. The *structure-from-motion* method of *SUN3D* [XOT13] first registers a list of RGB-D frames with each other using 2D keypoint descriptors associated with a RANSAC framework. Then, they define an energy based on geometry, appearance and semantics which is iteratively solved in order to get all frames into a globally consistent coordinate frame.

Methods by Choi et al. [CZK15] and Zhou et al. [ZPK16], whose implementations are both available in the *Open3D* library [ZPK18], use the *fast point feature histogram (FPFH)* descriptor to estimate initial pairwise alignments of partial point clouds. Then, a RANSAC framework allows discarding wrong initial alignments and a global bundle adjustment pass leads to high quality reconstruction results, as shown in Figure II.8.

II.2.4 Plane-based Registration

When looking to localize acquired parts of indoor scenes with each other, several methods make the observation that they are mostly composed of planar elements. Hence, introducing planar constraints into both pairwise and global registration improves robustness and reduces the drift due to data accumulation. In addition, using planar geometric features can improve localization in low textured areas of the scene.

In 2010, Pathak et al. [PBVP10] define a consistency-based framework to match extracted local plane features between multiple 3D views. By introducing the plane parameters into a *SLAM* update step, they obtain regular and accurate indoor scene reconstruction from a depth sensor mounted on a mobile robot.

In a similar fashion, Dou et al. [DGFF12a] detect large planes in indoor scenes and modify the RANSAC matching step to handle both points and planes. The generation of multiple hypothetic plane matches and their disambiguation using plane pairs and relative angles improves the robustness of the method. Again, the extension of *bundle adjustment* to planes greatly improves the accuracy of the reconstruction results.

Forstner and Khoshelham [FK17] assume knowledge of large matching planes in overlapping 3D views to define a plane-to-plane metric that is minimized to get the relative sensor positions and orientations. By explicitly modeling the uncertainty of detected planes, they leverage planar constraints and provide multiple formulations and solutions to the registration problem. Their method is fast and accurate to register pairs of frames, however gets confused and slower when multiple frames are simultaneously processed. The prior requirement of matched planes, which itself is a complex task, is a limit to the automation of this technique.

Halber and Funkhouser [HF17] define a *fine-to-coarse* registration framework that iteratively aggregates local to global planar features matched in subsets of the scene. Enforcing planar regularity while preserving local features fixes drifting issues appearing in regular optimization frameworks, as shown in Figure II.9. In addition, the local to global aggregation reduces the sensitivity to errors appearing in the frame-to-frame registration based on *SIFT* and *RANSAC*. However, as an offline method, it can not be applied to live registration because of the requirement for global structural information.

More recently, Shi et al. presented *PlaneMatch* [SXN+18], a learning approach to planar feature matching and registration in RGB-D frames. They predict local and global patch coplanarity in different RGB-D views of a scene and aggregate all coplanarity constraints as well as point correspondences into a robust optimization framework that achieves state-of-the-art results. The predictions are mostly accurate and of various nature, even with wide baselines, thanks to the large amount of training data. Although, as they rely on color, depth and normal information to match planar patches, the lack of discriminative features e.g. flat or low textured areas, can sometimes lead to false positive matches disturbing the global registration. In addition, the use of a *neural network* requires high performance hardware and leads to high computation times.

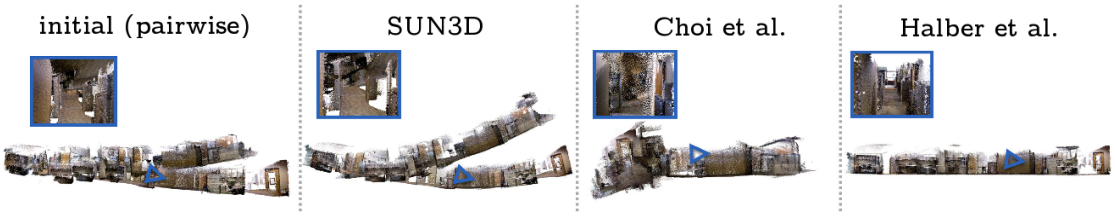


Figure II.9: Global reconstruction with planar constraints (right). Introducing planar regularization into the global reconstruction reduces registration drift and leads to more accurate models. Image taken from Halber and Funkhouser [HF17].

II.3 Indoor Scene Reconstruction

This section surveys the methods aiming at building a partial or full geometrical model of an indoor scene from a stream of depth or RGB-D frames. The particularity of these methods is their speed, as we focus on so-called *online* methods whose goal is to provide real-time interactive reconstruction of the scene while a sensor browses the surroundings. A recent survey by Zollhöfer et al. [ZSG+18] presents these methods in more details, with a focus on volumetric fusion and some surfel-based approaches. Here, we study three classes of methods:

- dense *simultaneous localization and mapping (SLAM)*;
- volumetric fusion of depth samples;
- offline surface regularization.

Several performance issues are raised when building such a map in real-time. First, accurate pairwise registration of nearby frames requires low baseline, which implies either slow motion of the sensor or a frame processing faster than the sensor provides new frames, to avoid delay, typically in less than 30 milliseconds. Second, the amount of memory needed to store and update the map and 3D model are usually significant, especially for volumetric methods, and require recent high end hardware.

II.3.1 Dense SLAM

Simultaneous localization and mapping (SLAM) jointly focuses on the localization of a moving sensor in a scene and the creation of a 3D map of the environment by aggregating multiple observed samples. While the information stored in the map grows with the motion of the sensor in the surroundings, more descriptive data of the scene can be tracked to accurately recover the sensor location and motion in time.

Figure II.10: 3D map built by *RGB-D SLAM*. Occupied voxels of the fixed 1cm grid are shown with averaged colors. Image taken from Endres et al. [EHS+14].

Pioneered by *PTAM* in 2007 [KM07] and first applied to *augmented reality*, these methods have evolved from sparse to dense with the apparition of commodity depth sensors in the early 2010s. Recent online *SLAM* methods aim at localizing multiple overlapping depth acquisitions of the scene with relation to a global model and aggregating their samples into this global representation.

One of the most common dense *SLAM* systems is *RGB-D SLAM*, presented by Endres et al. [EHS+14] in 2014. They describe an egomotion estimation method that uses point features detected in the color component of the RGB-D frames. After detecting and matching *SIFT*, *SURF* or *ORB* descriptors in subsequent color images, their 3D position in both frames is computed using the depth component. Using these matching 3D points, a robust RANSAC-based [FB81] estimation of the motion matrix allows discarding false positive matches. Sets of three matching points are randomly picked and the matrix transforming a set in the first frame into the second set is computed using a *least-squares* method [Ume91]. Inliers of the transformation are estimated using their 3D position and orientation and the one giving the most inliers is kept. After pairwise frame localization was successful, observations of keypoints are aggregated into a global camera pose model based on graph optimization of correspondence errors. Acquired scene samples are then accumulated into a voxel grid of fixed size 1cm to build the global map, as shown in Figure II.10. Other recent dense SLAM systems include *RTAB-Map* [LM14], *KDP SLAM* [HWZK17] or *ORB-SLAM2* [MAT17].

Figure II.11: Planar map creation with *dense planar SLAM*. The identification of large planar areas in the scene allows augmenting them with web content in real-time. Image taken from Salas-Moreno et al. [SMGKD14a].

Multiple methods have been developed to introduce higher level planar primitives in the SLAM system, either to smooth and improve the reconstruction [SMGKD14a, ERAB15] or improve the localization of the sensor [DGFF12b, Kae15, GZ15]. Results from *dense planar SLAM* [SMGKD14a] provide knowledge of large planar areas in the observed scene in real-time, which makes it suitable to add augmentations, such as web pages, over these areas, as shown in Figure II.11.

While these methods gain robustness by leveraging the descriptive character of color image features, the same features make them highly sensitive to illumination changes or low textured areas. In that sense, the introduction of planar features reduces this sensitivity while providing a lightweight regularization support for the generated map. This consistent support maintained throughout the processing can also be seen as the map itself, giving a solution to the issue of storage of a large set of 3D points, which are then transfered onto this planar structure through specific projection and update operators.

II.3.2 Volumetric Depth Fusion

Volumetric depth fusion is the construction of a volumetric representation by accumulating depth observations into a faithful model of the input scene. In 1996, pioneering research by Curless and Levoy [CL96] introduces depth fusion with the volumetric definition of a *signed distance function (SDF)* to the surface. A voxel grid is used as accumulator where 3D samples are merged to update values of signed distance to the nearest model surface. A mesh model can then be extracted as the isosurface of this implicit volumetric function.

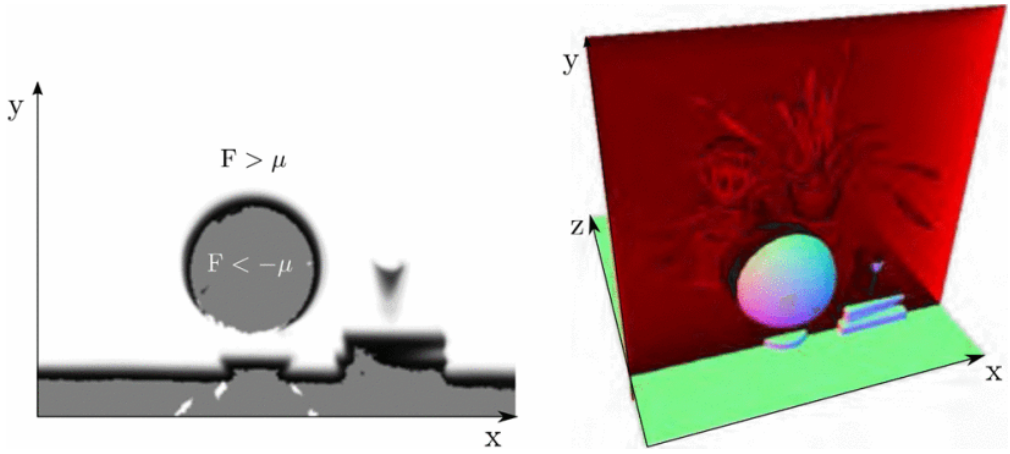
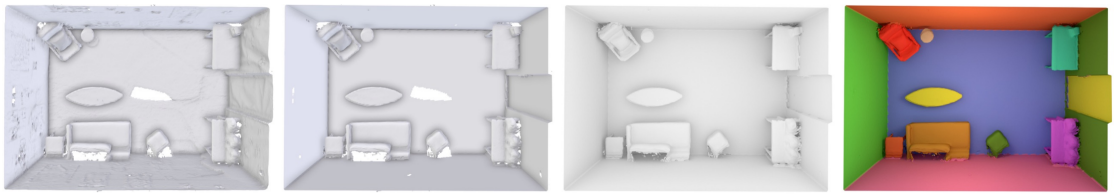


Figure II.12: *KinectFusion* integrates depth samples into a global *truncated signed distance function* grid. The *TSDF* slice on the left corresponds to the red plane in the 3D view. Image taken from Newcombe et al. [NIH+11].

In 2011, following the global availability of consumer grade depth sensor *Kinect*, *KinectFusion* [NIH+11] defines the new norm in volumetric 3D reconstruction with a truncated *SDF* and efficient implementation for live operation. First, dense frame-to-frame tracking is achieved by assuming high frame rate and applying *projective data association* [BL95] followed by point-to-plane metric optimization in an *ICP* scheme. No color information is required for the tracking, which enables depth only 3D reconstruction. Depth samples in a global coordinate frame can then be used to update *SDF* values in the voxel grid that we can see on Figure II.12. The progressive integration of depth samples into the volumetric representation creates a complete model that is ray casted to get live depth maps of the global model. In the following years, multiple performance improvements were published such as *VoxelHashing* [NZIS13], *DynamicFusion* [NFS15], *ElasticFusion* [WSMG+16] and more recently *BundleFusion* [DNZ+17b]. In 2015, Klingensmith et al. present *Chisel* [KDSX15], an embedded implementation of volumetric depth fusion for live large scale 3D reconstruction on a *Google Tango* tablet[6].

High level planar primitives can also be introduced in volumetric depth fusion in order to regularize and complete the data within the volume. Recent methods by Zhang et al. [ZXTZ15] or Dzitsiuk et al. [DSM+17] lead to smoother large scale reconstructions by maintaining a structural geometrical model of the surroundings to introduce spatial constraints in the voxel grid. In particular, the second method is based on the *Chisel* embedded implementation [KDSX15] and creates accurate, complete and semantically meaningful segmented 3D reconstructions in real-time, as shown on Figure II.13.



(a) Raw reconstruction (b) De-noised reconstruction (c) Hole filling (d) Object segmentation

Figure II.13: Planar regularization and segmentation of 3D reconstructions. From the raw scene reconstruction (left), plane regularization and hole filling lead to a more accurate model (middle) which can be easily segmented into planes and non planar connected components (right). Image taken from Dzitsiuk et al. [DSM+17].

While modern volumetric depth fusion enables live 3D reconstruction even on embedded devices, acceptable trade-offs between accuracy and performance often require high end hardware to get sufficient model resolution. Given the resolution of the voxel grid, the representation of a full set of rooms, however efficient, can need several *gigabytes* of memory. For instance, state-of-the-art live 3D reconstruction implementation *BundleFusion* [DNZ+17b] requires modern GPUs and more than 20 GB of CPU and GPU memory. In contrast, *point-based fusion* [KLL+13] is a variant of depth fusion that uses point accumulators to discard the full volumetric representation. Another aspect of performance requirements is the need for small baseline between subsequent frames to avoid losing track of the sensor, hence requiring processing images at least as fast as the sensor provides them.

II.3.3 Offline Surface Regularization

Online generation of full 3D models allows fast acquisition of rather accurate 3D images of scenes. However, the speed of live operation often leads to accumulation of noise and missing data in the representation. Using the exported 3D model as input, multiple methods apply structure analysis to the geometry to regularize and complete the scene representation. These processes are usually applied offline, after the acquisition is done, in order to take advantage of the complete knowledge of the scene. However, this complete scene data is heavy and processing it requires lots of computing memory and time, hence not adapted to embedded operation. As an example, the generation of a regularized model can take 30 minutes to 1 hour for 1000 images [ZCC16], to several hours for a full scene mesh [HDGN17].

[6]Google Tango project: http://en.wikipedia.org/wiki/Tango_(platform)

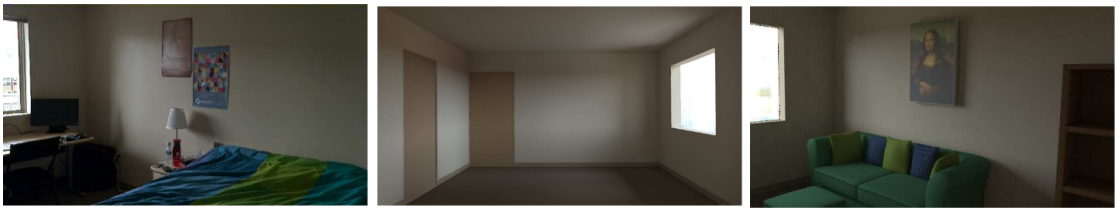


Figure II.14: Scene model and offline lighting and furniture editing. Left: Raw RGB image from an RGB-D frame. Middle: Empty room geometrical, lighting and appearance model. Right: Relighted and refurnished room. Image taken from Zhang et al. [ZCC16].

Zhang et al. [ZCC16] make use of planes to estimate the geometry of a room in order to remove furnitures and model the lighting of the environment. This allows relighting and refurnishing the room as desired, as shown on Figure II.14. In the same spirit, *3DLite* [HDGN17] turns the volumetric representation into a planar model of the observed scene and optimizes it to achieve a high quality texturing of the surfaces. Examples of such models can be seen in Figure V.6 (middle).

II.4 Shape Primitive Surface Recovery

This section aims at providing an overview, a classification and a comparison of the methods developed over the years to detect simple geometric primitives in 3D data, captured from different possible sources. As described by Woodford et al. [WPM+12], these methods are ”model fitting algorithms that, given a set of features (here, points), find the most likely model instance (here, primitives) that generated those features”. The concept of simple geometric primitives is further developed in section II.4.1. For a more complete study of primitive surface recovery, we refer the reader to our full paper [KYZB19].

Objectives and Motivations

The process of approximating and abstracting 3D shapes by a simple parameterization allows extreme simplification of the geometry while keeping an accurate representation of the input data. Therefore, explaining 3D data using simple geometric primitives is a way of representing it in a compact manner and makes easier any subsequent analysis that would be performed, with consequences on both performance and the ability to perform high level tasks. Figure II.15 shows the goals of the detection process with the different types of input, detailed in section II.4.1, and the expected output primitives with different layers of abstraction.

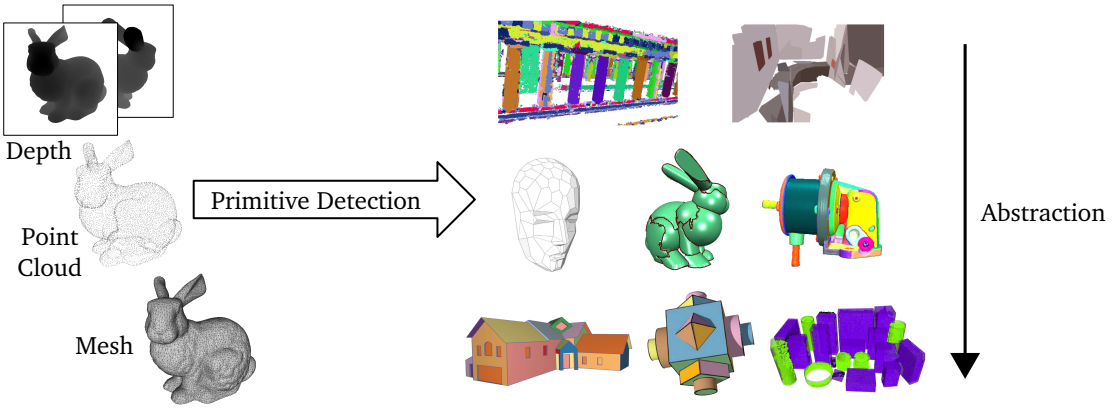


Figure II.15: Objectives of the detection procedure. Using one of the input data types discussed in section II.4.1 (left), the detection process outputs primitives with different levels of abstraction (right). Images taken from papers [LPRM02, WPM+12, TJRF13, CSAD04, WK05, SWK07, LWC+11, ASF+13, GMLB12].

In order to exploit raw captured 3D data in practical applications, one often has to reconstruct a high-level representation, of a single object or an entire scene, providing a visual summary which is similar to the understanding acquired by a human brain. This often amounts to the description of complex objects using only a couple of simple geometric primitives, such as spheres, cylinders, planes or boxes. Such visual abstractions not only simplify the geometry and topology of the input data, but also help clarify the spatial relationships between shape components, can act as economic substitutes for visibility queries, rendering effects and physics simulation, or be used as super-structures to quickly distribute filters, edits or enriched semantics over the data, on a per-component basis. Such compact primitive lists that summarize dense sampling sets are later used for processing, reasoning or interaction, with applications ranging from path finding in robotics to object placement in augmented reality, through domain meshing in CAD, control structures for free-form design and level-of-detail selection in game engines.

After giving fundamental information on 3D data and simple geometric shapes, this section presents the different theoretical concepts used as a basis for simple geometric primitive detection in existing methods. Note that a number of methods are based on more than a single theoretical paradigm. Table II.1 gives strengths and weaknesses for these different theoretical frameworks, which include:

- stochastic: RANSAC, local statistics;
- parameter spaces: Hough-like voting methods, parameter space clustering;
- other clustering techniques: primitive-driven region growing, Lloyd-like automatic clustering, primitive-oblivious segmentation followed by fitting.

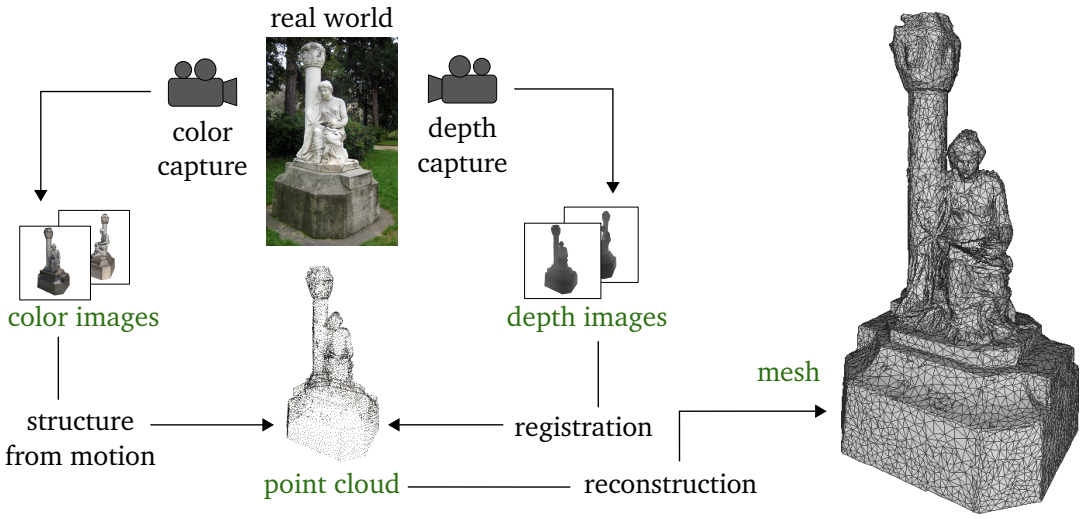


Figure II.16: 3D acquisition pipeline for the statue in the *Arenes de Lutece, Paris, France*. The green captions indicate potential bootstrap stages for primitive detection.

II.4.1 Fundamentals

Three Dimensional Data

Three dimensional data can be modeled with different representations, each of which favoring different families of processing methods. The specific forms of input data used for simple geometric primitive detection are further presented below, and the full acquisition pipeline is shown in Figure II.16. As presented in this figure, the data suitable as input for primitive detection can be formatted as color or depth images, point clouds, and polygonal meshes. Although most methods are developed for one particular type of data, some implementations handle any form of input by performing a conversion to their specific input format.

The raw format of acquired 3D data is the depth map directly extracted from depth sensors and represented by a single channel image. Depth sensors are often coupled with a regular RGB camera to provide additional color information. The natural 2D grid structure of these RGB-D images allows fast processing of 3D data. RGB-D data is presented in details in section II.1.

When scanning an object or a room, users can move a handheld sensor around them. The goal is to acquire as many views as possible and build the most accurate reconstruction. This generates a high number of 2D images that can be used as they are or post-processed. Typically, a sequence of RGB-D images gives a sequence of 2.5D colored point clouds sampling parts of the same items, which are then consolidated into a single 3D point cloud. On the other hand, a sequence of RGB images must first be processed via *stereo vision* algorithms to produce a 3D point cloud ready for primitive detection.

A point cloud is the most common representation of acquired 3D data, as it is simply a point sampling of the real world. It takes the form of a list of 3D positions, possibly with additional attributes, such as per-sample normals and color values. If computed from an image or an image sequence, the point cloud may be organized in multiple 2D grid structures, which allows fast browsing and point-wise neighborhood query using the sensor topology.

In *computer graphics*, numerous advanced processes exploit polygonal representations of surfaces, such as triangle or quad meshes. Their explicit topology makes easier operators such as filtering, resampling and rendering, either based on projection or intersection search. They are typically generated from acquired point clouds using surface reconstruction algorithms. These may use an inside/outside volumetric indicator functions, e.g., *moving least squares* [ABCO+03], *implicit multiple radial basis functions* [OBA+03] or *gradient-based Poisson* solutions [KBH06]. Other algorithms are based on a *Delaunay-based* triangulation, e.g., use tangent planes [HDD+92], *Voronoi filtering* [AB99] or *power crust* [ACK01]. See the work of Berger et al. [BTS+14] for a complete survey.

Simple Geometric Primitives

A simple 3D geometric primitive is defined as a 3D geometric shape with the following characteristics:

- fixed and limited number of global intrinsic parameters i.e., that only define the global size, orientation and position of the shape;
- convex (except for the torus);
- symmetric;
- basic shape which can be assembled with others to construct more complex shapes.

This definition matches the use of primitives in *constructive solid geometry* [Fol96], where complex shapes are built using Boolean compositions of these simple objects. We classify the geometric primitives used for detection in 3D data into four categories described in details below, from the most simple with few parameters to the most complicated ones. Figure Figure II.17 shows different simple primitives of varying complexity, sorted by category.

- planes;
- cuboids and boxes;
- spheres, cylinders and cones;
- other shapes: ellipsoids, tori, non-rectangular parallelepipeds.

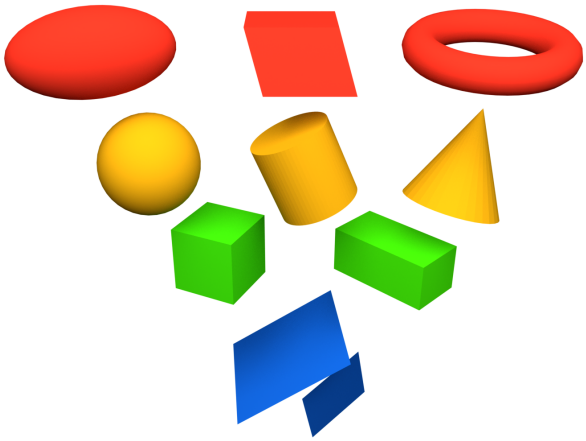


Figure II.17: Common geometric primitives.

Our interest lies in the fitting of primitive shapes to surfaces in order to approximate the boundary of objects and not their volume. We consider all simple primitive shapes as surface patches rather than as volumes of solid shapes. To be more specific, most of the methods that detect primitives in 3D data output trimmed shape surfaces, whose extent corresponds to that of the modeled object. Details on primitive trimming are given at the end of this section.

A *plane* is the most basic isotropic three-dimensional shape and the most commonly seen primitive in human-made environments. It can simply be defined by a normal vector $\vec{N}$ and its distance $d \in \mathbb{R}$ to the origin point of the reference frame. Methods using planar patches instead of infinite planes usually compute the convex hull or bounding rectangle of the set of plane inliers, or determine the limits of patches by clipping planes to each other.

Boxes and cuboids are sets of assembled orthogonal planes and are defined by their center C, orientation vector $\vec{N}$ and the three lengths of their sides. They can also be represented by their eight vertices or by the parameters of the planes forming them.

A *sphere* is an isotropic shape and can be defined by its center C and a scalar $r \in \mathbb{R}^+$ representing its radius. A *cylinder* can be parameterized with a point C belonging to its axis, a radius $r \in \mathbb{R}^+$ and an additional vector $\vec{A}$ representing its orientation. A *cone* is parameterized by its center C, called apex, an orientation vector $\vec{A}$ and an angle α between its axis and surface.

Ellipsoids, tori and non-rectangular parallelepipeds are geometric shapes of higher complexity. An ellipsoid is defined by its center C, axis $\vec{A}$ and three radii $a, b, c \in \mathbb{R}^+$, also called semi axes. A torus is defined by its center C, axis $\vec{A}$, minor and major radii m and M. A general parallelepiped is defined by its center C, orientation vector $\vec{N}$, the three lengths of its sides and three interaxial angles.

In the context of this study, simple primitive shapes are considered as surface patches whose goal is to approximate the bounded surface of objects. When parts of the data has been identified as belonging to a primitive shape, some methods stop the processing while the shape has an infinite (planes, cylinders, cones) or full (spheres, cuboids, boxes, parallelepipeds, ellipsoids, tori) extent. However, multiple algorithms go further in the analysis and estimate the extent of the fitted shape that models the actual object. In particular, Schnabel et al. [SWK07] define a regular grid on the surface of detected shapes in order to identify connected components. Such a grid can also be used to compute the *convex hull* of inliers in the space of the shape [And79]. Another solution is to fit smaller shape patches to the surface and merge them in the final model [BV11]. This extent appears naturally in region growing algorithms such as the method proposed by Feng et al. [FTK14].

II.4.2 Stochastic Methods

RANSAC-based

Random sample consensus (RANSAC) is a popular stochastic method used to estimate model parameters iteratively given a data set. It is known to be particularly robust to outliers. First introduced by Fischler and Bolles [FB81], RANSAC has been used in a variety of applications, especially in *computer vision* and *image processing*. The basic principle of the algorithm is to try many possible randomized models that could fit the data and evaluate how good this model is in order to find a consensus, i.e. an agreement of most of the data samples. Here, the term ”most” is to be defined depending on the application. Specifically, the algorithm is composed of two main steps. The first one is a randomized sampling over the data to find a minimum set of samples allowing to compute parameters for the model. The second step consists in counting how much of the dataset is represented by this model and keep the model that fits most of the data.

Variants of *RANSAC* usually keep the random sampling part but use a more elaborate method to compute the score and choose the best model, instead of the simple inlier count. The score to maximize or minimize can be the median of squared errors to the model [Rou84], a squared error function where outliers are given a fixed penalty in *MSAC* [TZ00] or the maximum likelihood in *MLESAC* [TZ00]. *Randomized RANSAC* [MC04] introduces a speed-up to the original *RANSAC* algorithm by running a pre-test on a few data points before score computation, called $T_{d,d}$ test, which allows detecting and discarding wrong models very early. *PROSAC* [CM05] makes a change to the random sampling strategy by selecting samples within a small subset of data, ordered by confidence values, and increasing its size until a suitable model has been found.

RANSAC for Shape Detection

The efficient iterative primitive detection method presented by Schnabel et al. [SWK07] makes use of the RANSAC paradigm to detect planes, spheres, cylinders, cones and tori given an unorganized oriented point cloud as input. Their implementation gives stochastic improvements to the critical steps of the algorithm in terms of complexity. For a regular RANSAC-based shape detector, minimal sets of three points would be randomly picked a fixed and large number of times. Then, the shape parameters are estimated from this minimal set and inliers of the estimated shape are computed. The shape with the highest score is kept, its inliers are removed from the point cloud and the algorithm is ran again on the remaining data. Schnabel et al. replace the fixed number of loops with a stochastic condition to stop looking for shapes in the dataset, based on the number of detected shapes and number of randomly picked minimal sets. Also, instead of searching the full point cloud for inliers of a given shape, they estimate this count in a random subset of the dataset and extrapolate it to the full point cloud. Other modifications allow improving the quality of detected shapes with a localized sampling and specific post-processing. This method and its implementation have since been used in many follow ups, as it is designed to be efficient by giving a stochastic answer to the problem of RANSAC iteration count.

Its direct application [SWWK08] uses the primitives to build a topology graph and match shapes in 3D point clouds. Another application by Li et al. [LWC+11] adds a regularization step for mechanical objects. Assuming regular relations of coplanarity, coaxiality and orthogonality between object parts, the high-level modeling made of geometric primitives is optimized to form a cleaner and more regular model. Following the same paradigm, the *multiBaySAC* algorithm [KL15] uses Bayes rule to randomly generate multiple primitive hypothesis that fill up a parameter space in which the best candidates are identified.

Local Statistics

The definition of occupancy probabilities at space locations allows inferring local primitive parameters from these distributions. Mostly, such methods aim at bounding detected objects with boxes or cylinders. The probabilities can be defined at all or some locations in space using e.g., a simple Gaussian, Gaussian mixture models or Bayesian inference [BFF15]. The analysis of this probabilistic field enables the detection of object positions.

For example, Carr et al. [CSM12] create occupancy maps from registered RGB views, where each ground location is associated with an occupancy probability for vehicles or pedestrians. By deconvolving these maps with primitive specific kernels corresponding to the projection of boxes and cylinders, the algorithm is able to highlight object locations. A mean shift procedure then allows recovering the position and orientation of cuboids and cylinders that bound vehicles and pedestrians in the scene.

Bagautdinov et al. [BFF15] identify objects, especially persons, standing on the floor of indoor rooms acquired through depth sensors. The algorithm builds a Bayesian generative model of probabilistic occupancies at each location. Its optimization using current depth observations makes it converge towards a discrete number of positions on the ground. Boxes are then fitted to the detected objects to form a clear boundary in image space.

II.4.3 Parameter Spaces

Hough Transform

The *Hough* transform defines an accumulation space built upon a parameter space in which similar geometric elements coincide. After quantizing this space into regular bins, all samples of the data cast votes for all geometric elements of which they are inliers. The most voted parameter set identifies the object that best explains the input data. Named after Paul Hough's 1962 patent [Hou62], and originally used to detect lines in images [DH72], it has since been generalized [Bal81] to detect 2D and 3D common geometric objects, such as circles. In particular, it can be used to detect 3D shapes [WPM+14]. In spite of its generic nature, one of the most important drawbacks of this method is the lack of boundary of the parameter space. Given its potential dimension, it can become an issue in terms of memory consumption and processing time.

Research has been conducted in order to improve the performance and usability of the Hough transform. The *probabilistic Hough tansform* [KEB91] gives a stochastic answer to the complexity of the Hough transform but keeps its one-to-many mapping from image to parameter space, thus only solves the speed issue. The authors show that only a random subset of points in the image is enough to correctly recover the models, although the size of the subset is not automatically defined. In contrast to the usual algorithmical definition of the Hough transform, Stephens [Ste91] derives an analytic formulation based on the maximum likelihood method which leads to continuous values in Hough space and allows the use of known mathematical tools for locating maxima. Introduced in 2008, the *kernel-based Hough transform* [FO08] forms approximately collinear image edge pixel clusters in which the best fitting line and its error kernel are computed. All kernels are then merged into the Hough space and a peak detection allows identifying the correct candidates within a simplified parameter space.

Planes can be detected using their spherical parameterization. Each 3D point and its associated normal extracted from depth maps contributes to only one voxel in the discrete Hough space. By smoothing this space, local maxima and candidate planes are identified. Through time, correct candidates form peaks in a time-global Hough space and get activated [HSSM14]. The detected planes can then be refined using *singular value decomposition* over the inlier sets [WO02]. Figure Figure II.18 shows a three-dimensional Hough space where each 3D Euclidean location builds a surface. The intersection of these surfaces yields the plane passing through all three points.

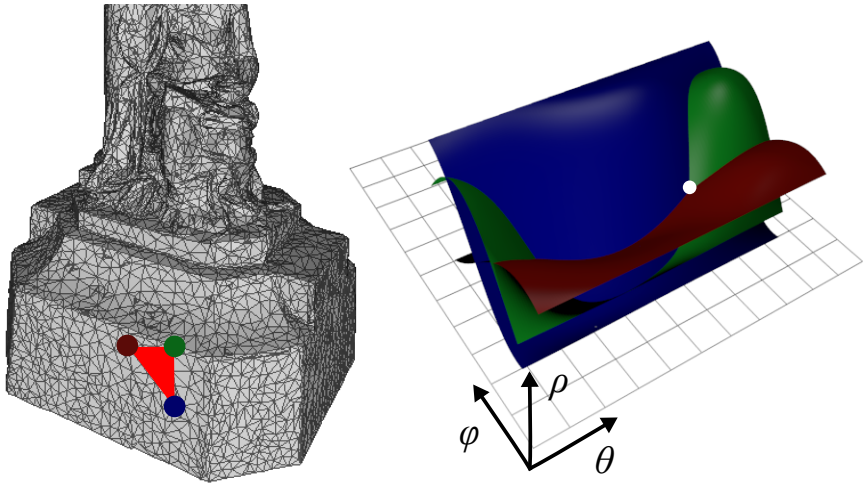


Figure II.18: 3D Hough space for plane detection. Each 3D Euclidean location (left) builds a surface defined by spherical coordinates (right). The 3D Hough points at the intersections of these surfaces give the planes passing through many points (white dot).

In order to simplify a given mesh model, *billboard clouds* [DDSD03] can also be created using a voxelized 3D Hough space. The local maxima of this space give the planes that approximate the shape, called *billboards* (textured planar polygons) that can be used to render the shape using very few primitives. The resolution of the voxel grid gives control over the approximation of the mesh. A coarse voxel grid tends to merge many triangles into a single billboard, and a finer grid will produce more billboards for a better approximation. Rabbani et al. [RVDH05] use a Hough-based method to detect cylinders in point clouds. First, the orientation of the cylinder is detected using a 3D Hough parameter space lying upon the Gauss sphere. From the orientation of the cylinders, the corresponding inliers are projected on the orthogonal plane and the Hough transform for circle fitting in 2D allows recovering the radius and position of the cylinders.

Clustering Parameter Space

Some methods directly exploit and analyze parameter spaces to detect simple geometric primitives. These methods mainly detect plane clusters in point clouds, as the parameter space for planes can be divided into two simple disjoint spaces. These are the parameter space for normal vectors i.e., Gauss map modeling the plane orientation as a point on the unit sphere – and the Euclidean distance to the origin.

By clustering the space of normal orientations over a hemisphere using voxels [HHRB11] or a mean shift procedure [CC08], dominant orientations can be found. The individual plane clusters can be extracted using a threshold on the distance in Euclidean space [HHRB11] or point density peaks along the detected directions [FCSS09]. Parameters can then be estimated using *principal component analysis* within the clusters [CC08].

II.4.4 Clustering

Primitive-driven Region Growing

Several methods use known segmentation and clustering techniques to discover primitives in 3D data. As described by Lukacs et al. [LMM98], "segmentation is most commonly treated as a local-to-global aggregation problem with similarity constraints employed to control the process". In the context of 3D geometric primitive detection, these similarity constraints drive the detection towards different paradigms.

The *region growing* algorithm is used to extract connected components in a depth map or a point cloud with neighborhood information (e.g., k-nearest neighbors). A label is assigned to a seed sample and its neighbors are iteratively analyzed and assigned the seed's label if their characteristics are similar enough to the seed's. These characteristics, such as color, depth or normal orientation are considered similar given a threshold which is usually application-dependent. We call *primitive growing*, the region growing procedure in the context of primitive detection.

The processing is started by assigning seeds to random [OLA16, LMM98] or regular [ZYH+15] positions in the data. The growing of points into regions can be performed through a neighbor search using efficient data structures such as a neighbor graph [FTK14, AEH15]. A shape-based flood filling can also be used [LLL+12, ZXTZ15]. Some methods over-segment the data into patches, *super-points* or *super-regions* [LGZ+13] that can be joined together to form the final segmentation. Algorithms used to merge these patches include rectangle fitting [MPM+14, OLA16], candidate generation and selection [MMBM15], linear interpolation [TGB13] or automatic merge of neighboring shapes [LMM98, AFS06, ZYH+15]. Making use of the efficient structure of images, a few methods offer a speed-up over point cloud based methods [TGRC13, KHB+15, AEH15]. A refinement step is usually applied to estimate plane parameters. It can be based on *least squares fitting* [TGRC13, BSG+11], *principal component analysis* [SMGKD14b], *RANSAC* [LLL+12, SXZ+12, AEH15] or shape-driven pixel-wise region growing [FTK14].

Points are aggregated into regions using similarities with their neighbors in terms of different heuristics related to the primitive shapes to detect. For plane detection, planar heuristics include Euclidean distance [SMGKD14b, SXZ+12, LGZ+13], normal orientation [MMBM15, OLA16, TGRC13] and surface curvature [ZXTZ15, MPM+14, BSG+11]. More general methods often use the primitive fitting error, computed as the mean square error resulting of a potential aggregation [AP10, XZZ+11, LMM98]. Other metrics include the *spherical quadric error* [TGB13], *cylindricity measure* [ZYH+15] and tensors capturing the local dimensionality of the data [Sch04]. More details on fitting error are given in section II.4.5. Gelfand and Guibas [GG04] compare point clusters in terms of *slippable motions*, defined as rigid transformations that, applied to a simple geometric shape, will not form any gaps between the original and transformed shapes.

A special kind of primitive growing algorithms uses a tree representation of the data [AP10, TGB13]. Points are initialized as separate clusters and ordered according to the cost of aggregating them with nearby clusters into a single primitive, based on previously discussed metrics. Iterative aggregations are then performed, starting from the least costly, in order to build a hierarchical partition of the data. The hierarchical model allows navigating through different levels of approximation, that can be used as an intermediate control structure to deform the input model.

Automatic Clustering

Automatic clustering methods in machine learning are often based on Lloyd's clustering algorithm [Llo82], developed in 1957 to create a partition of point sets in Euclidean spaces. This algorithm proceeds iteratively in order to cluster all points into regularly distributed regions. The two main Lloyd-based algorithms are the *k-means* and *mean shift* clustering methods, described below.

The *k-means* procedure, first defined by MacQueen [Mac67], takes as input a spatial data set and a fixed number k of clusters to detect within this set. After selecting k random data points called *means*, the remaining points are assigned to the closest mean according to a chosen distance metric. Using the so-initialized clusters, new means are created at their respective centroids. The cluster assignment and means update steps are repeated until convergence. These two steps can also be seen as iterations of *expectation and maximization* steps, making the k-means algorithm a variant of the *EM* method.

In contrast to the k-means algorithm, the *mean shift* algorithm does not require a prior number of clusters to be found. Originally presented by Fukunaga and Hostetler [FH75], and first applied to *computer vision* by Comaniciu and Meer [CM02], it creates partitions of a feature space. The data is considered as samples of a probability distribution made of kernel instances and points are assigned to the corresponding kernels. In other words, this algorithm is seeking the modes of a distribution modeled by kernels, among which the two commonly used ones are the flat and Gaussian kernels. These kernels are recovered by iteratively shifting each point towards a kernel center using weighted contributions of the original points in the dataset. The kernels then appear naturally to form the data clusters.

Several methods were inspired by these automatic clustering algorithms to fit geometric primitives to 3D data. They were pioneered by *variational shape approximation (VSA)* [CSAD04], which aims to find a fixed number of planar proxies to simplify a meshed object with the best possible approximation. Randomly picked data samples are used as initial shapes to bootstrap the process. Then, the optimization starts with an iteration of geometry partitioning and shape fitting that is repeated until convergence of the partitions. The partitioning is done using region growing started from the centers of the current shapes and using their parameters. Regions are grown based on distortion error between adjacent samples and the surface of the shapes. The fitting step computes the shapes that minimize the error to their associated samples and the partitioning starts again with the newly computed shapes. Yan et al. [YWLY12] developed a similar method to estimate quadric surfaces from a given mesh model. In the specific case of the method proposed by Woodford et al. [WPM+12], points are assigned by expanding or contracting neighboring primitives, leading to an unspecified number of primitive shapes.

Primitive-oblivious Segmentation

Several methods apply a segmentation step to the data prior to fitting, in order to reduce the number of outliers, thus getting a more accurate model. The segmentation of three-dimensional data, in the form of a point cloud or depth map, is carried out through algorithms such as region growing, watershed by flooding, classification or other clustering methods presented below. In this context, the segmentation methods do not take into account primitive search: they are completely *oblivious* to the primitive detection. Instead, they use heuristics such as location, color, distance or other application-specific features.

Region growing and the search for connected components is the most commonly used technique to segment images and 3D data. This method, presented earlier with the use of primitive-specific neighbor comparison, can also be applied to the raw data with a comparison based on heuristics such as depth discontinuities or color. For example, Martinovic et al. [MKRVG15] first label a point cloud for different architectural elements, independently of primitive shape considerations. Connected components are then extracted for each element of the facade using architecturally-based features.

Supervised *classification* methods use previously labeled data to characterize points into a number of classes. Depending on the application, these semantic classes are defined using features that discriminate them well from one another. In scene analysis, they are typically used to segment buildings from vegetation [LM12], walls from desks, or object parts [KLM+13]. For example, Lalonde et al. [LVHH06] use local geometry features in natural outdoor scenes to label flat areas as trails, linear areas as trees and scattered areas as leaves. Other geometric attributes include elevation or horizontality [VLA15].

The *watershed* algorithm is a segmentation method that uses the gradient magnitude image computed from an input image and considers it as a height map inside which water would be dropped. Starting from a number of markers in the image, the watershed by flooding technique [BL79] makes the water level rise up from these locations to find local maxima of the height map. These local maxima represent the watershed of the image gradient and thus are the limits of the segments in the image. Such flood filling algorithms are most suitable for depth maps as input images, in the context of 3D data segmentation [WGC99].

Using 3D data as input, one can apply geometry rules to detect different parts in the data. These rules can be based on the height or size of clusters in the data. For example, different tables of an indoor scene can be clustered using horizontality and distance. Then, objects are projected on the table planes and clustered in 2D to form object segments [RBMB09, GMLB12].

In some of the primitive detection methods, *interactive* techniques are used to drive the segmentation prior to fitting. The user can be asked to assign labels to regions [OVWK14, WGC99] or drive the segmentation using strokes [CZS+13, SXZ+12, SAG+13].

Finally, many segmentation methods have been developed when using meshes [Sha08], usually by adapting known image processing algorithms to 3D data. In particular, data-driven methods [XKH+16] that perform segmentation based on databases of labeled meshes are more and more used thanks to the availability of this segmented data. Using powerful descriptors, shapes can be segmented very efficiently and robustly using the diversity of the many segmented mesh examples.

Fitting Primitives to Segments

In order to compute the parameters of geometric primitives, a fitting method is finally applied to the individual detected clusters. To do so, one common method is to use *principal component analysis* to extract these parameters [WGC99, KLM+13]. By computing the covariance matrix of all points in the segment, an *eigen* analysis allows recovering the primitive information. For instance, the normal of an optimal plane fitting the data is the eigenvector of the covariance matrix with the lowest corresponding eigenvalue. The covariance matrix for a set of N 3D points $X_i = (x_i, y_i, z_i)_{i=1...N}$ with their centroid $\bar{X} = \frac{1}{N}\sum_{i=1...N} X_i$ is given by

$$\frac{1}{N}\sum_{i=1...N}(X_i - \bar{X})(X_i - \bar{X})^T. \tag{II.1}$$

One can also apply the RANSAC algorithm to the segments, which allows getting a faster and more accurate model as the segmented objects or parts often correspond to one primitive shape [RBMB09, GMLB12].

Finally, the primitive fitting problem can often be modeled as an unconstrained optimization problem. Thus, it can be solved using *least-squares* fitting by minimizing an energy representing the distance from the model to the data. Different energies can be used and are detailed in section II.4.5. For instance, Lukacs et al. [LMM98] provide per-primitive energies in the context of *least-squares* fitting, based on the efficient parameterization of primitive shapes brought by Marshall et al. [MLM01]. Depending on the complexity of the energy, the minimizing parameters can be estimated using standard linear system solvers such as *Cholesky* and *QR* matrix factorization or *singular value decomposition* [GVL96]. Complex energies are minimized with iterative optimization methods like Newton's method or a *gradient descent* [Avr76].

II.4.5 Metrics and Evaluation

This section aims at giving insights into ways to evaluate simple geometric detection methods, as well as metrics used to model the error in these methods.

Evaluation Methodology

In order to evaluate the quality of a modeling instance made of simple geometric primitives, different metrics can be used depending on the application and the performance objective, including:

- fitting error (detailed below);
- processing time measured in milliseconds or number of processed frames per second (fps);
- simplicity of the model and over-detection: number of primitives.

Some metrics, such as segmentation correctness, need ground truth information to be computed. This usually requires prior manual and user-assisted work on the data, which includes:

- segmentation and spatial consistency: objects are correctly separated by primitives and modeled by one instance each;
- camera poses (in the context of scene analysis).

Framework	Strengths	Weaknesses	Reference algorithm
RANSAC	simple, general, accurate, robust to outliers	many parameters to tune, dependent on a minimum set, no spatial consistency	Fast RANSAC [SWK07]
Local statistics	application-specific	model-dependent	Monocular Occupancy Maps [CSM12]
Hough transform	handles missing data, supports many model instances, relatively robust to noise	unbounded space size, dependent on parameter space quantization	Primitive-based registration [RDvdHV07]
Clustering parameter space	robust to outliers	restricted to low dimensions	Cluster Normal Space [HHRB11]
Primitive growing	meaningful segmentation, spatial consistency	slow, local, sensitive to initial conditions (seeds), noise and outliers	Hierarchical Modeling [AP10]
Automatic clustering	no prior on location, few parameters	dependent on seeds, sensitive to outliers, can require numerous clusters (k-means)	Quadric Surface Fitting [YWLY12]
Segmentation then fitting	vast literature for segmentation	can merge different primitives, sensitive to noise, sensitive to outliers	Hybrid City Representation [LM12]

Table II.1: Strengths and weaknesses of the main theoretical frameworks. The reference algorithm is the most representative of each framework and usually has the most features and the best quality output.

Evaluation Metrics

In order to evaluate the quality of an output model made of simple geometric primitives compared to the input data, different metrics have been used to estimate the error made with the detected set of primitives by measuring the distance to the model, or fitting error:

- the simplest metric is the *sum of squared distances* from points to the their corresponding primitives. For primitives $S_i, i \in [0, N]$ gathering inliers $P_j^i, j \in [0, M]$, the fitting error is

$$\epsilon = \sum_{i=0}^{N} \sum_{j=0}^{M} \left\| P_j^i - proj(P_j^i, S_i) \right\|^2 \tag{II.2}$$

  with $proj(P_j^i, S_i)$ modeling the projection, i.e. the closest point, of point P_j^i on its corresponding primitive shape S_i;

- the *Hausdorff* distance [CRS98] defined for two sets of points $a \in A$ and $b \in B$ is the highest distance among all points a to the corresponding closest point of B, considering $d()$ as a given real distance function:

$$H_{AB} = max_{a \in A}\{min_{b \in B} \ d(a, b)\}. \tag{II.3}$$

Processing Metrics

The following metrics are used to drive algorithms, although their values do not make much sense to evaluate of the quality of the output.

- the *quadric error metric*, introduced by Garland and Heckbert [GH97] in order to simplify meshes. By summing squared point-to-plane distances, a quadratic form appears and allows efficient evaluation of the error at any point in space. For a vertex v and N planes $P_i, i \in [1, N]$ with normals $p_i, i \in [1, N]$:

$$\epsilon = \sum_{i=1}^{N} dist(v, p_i)^2 = \sum_{i=1}^{N} (p_i^T v)^2 = \sum_{i=1}^{N} v^T p_i p_i^T v = v^T (\sum_{i=1}^{N} p_i p_i^T) v = v^T Q v \ . \tag{II.4}$$

  In the scope of the original work [GH97], summing the Q matrices associated with the two vertices of an edge allows evaluating the error produced by the collapse of this edge. This provides a global ordering of edges to collapse for progressive mesh simplification. In the field of geometric primitive detection, Yan et al. [YWLY12] perform Lloyd-like iterations based on the quadric fitting error on an input triangle mesh;

- an extension of the quadric error metric, called the *spherical quadric error metric (SQEM)* [TGB13] allows to iteratively collapse edges to spheres, with potential null radii, progressively moving from a surface to a volume representation as the model is simplified. The *SQEM* represents the distance from a sphere to an oriented plane and is minimized to identify the best sphere approximation for a set of triangles, with the resulting mesh of spheres connected by edges and triangles being called a *sphere mesh*. This is instrumental for extreme approximation, shape editing and, through its later extensions, animated mesh analysis [TGBE16] and hand recognition [TPT16].

II.4.6 Discussion

Common geometric shapes, such as planes, cuboids, spheres, cylinders, cones, tori, ellipsoids and parallelepipeds are the building blocks of most of the objects present in man-made environments and of some natural elements as well. Their simplicity makes them a perfect tool for the analysis of heavy and complex 3D data acquired from noisy 3D scanners, as they allow both reduction of the size of the data and complexity of the model for a computer.

For scene modeling in the context of robotics, a simple geometric primitive-based representation allows faster and more accurate processing for real-time applications and autonomous navigation. For the automatic reconstruction of objects or buildings, geometric primitives can help recover the regularity of the scanned items. In the area of *computer graphics*, shape processing can also make use of geometric primitives to simplify objects and apply simple algorithms for deformation or animation.

In this section, we have described the principles of the most recent methods aiming at detecting such simple geometric primitives in captured 3D data. We categorized the detection of simple geometric primitives in 3D data such as depth images, point clouds or polygonal meshes using several well established theoretical foundations that make use of stochastic paradigms, parameter spaces or clustering and segmentation techniques.

Towards Spatial Reasoning

Spatial reasoning enriches data in terms of space and organization through structural information. In the case of 3D space modeling, spatial reasoning is possible by acquiring qualitative and quantitative knowledge of spatial locations in the observed scene. In the particular context of this section, these are represented by the positions and orientations of detected objects or parts, modeled by simple geometric primitives, with relations to each other.

Several recent methods have looked into quantitatively describing these relations, in order to infer information about the scene structure. They can be represented by a graph of objects where the edges model rigid transformation matrices. Li et al. [LWC+11] define a graph of geometric relations between parts of objects modeled as simple primitives, which is simplified to get a regular model based on relations of coplanarity, coaxiality and orthogonality between object parts. To model relations between objects in a closed room, Rusu et al. [RBM+07] define relations between detected objects with 3D rigid transformation matrices, while Silberman et al. [SHKF12] hierarchically segment the scene into objects and infer adjacency relations between them. In order to model full building interiors, Ochmann et al. [OVWK14] detect doors and windows to create a graph of connectivity between rooms. Monszpart et al. [MMBM15] assume a regular structure between walls and floors in the building and find regular arrangements of planes to model that structure.

Research Challenges

The problem of simple geometric primitive detection in captured 3D data raises numerous challenges in the context of modern applications. Although consumer depth cameras represent a great opportunity for many applications, they still raise many issues, as shown in section II.1.2. Indoor scene modeling methods based on streams of depth data seem to perform generally well, as the repetition of observations of a closed environment allows building a noise-free and consistent model through time.

Most methods aim at modeling man-made environments using planes or even model curved objects with this primitive shape, because of its simplicity and the fact that it can be easily identified with known geometric heuristics such as normal orientations. Fewer methods detect more complex primitives in order to build a more reliable model of the data which allows even lighter representations for a similar quality.

Therefore, future research challenges lean towards the improvement of results in terms of completeness and consistency of the model. In particular, completeness can take the form of more complex primitives, although they need to stay generic and not data-specific. Meanwhile, overall performance and compatibility with real time constraints remain a key enabler for future applications.

II.5 Research Challenges

In this chapter, we presented existing methods to perform some essential 3D scene analysis tasks and solve arising issues. Recent work on RGB-D data processing, registration and reconstruction exhibits robust results in the presence of noise, outliers and missing data, often at the expense of requirements in computing resources. Abstracting such acquired 3D data with simple geometric shapes leads a lightweight and intuitive representation of the space, and can be used to assist or drive registration and reconstruction methods.

However, recent methods have limitations and cannot cope with the constraints to reach our goals, expressed in section I.2. In particular, dynamic scenes, variability and lifelong operation are mostly ignored when considering feature matching for registration or reconstruction. In addition, our hardware and performance constraints cannot be solved with point or volumetric approaches, and surface based methods do not accurately represent the topology in shape space. We detail below the research axes that we followed to satisfy our requirements.

II.5.1 Frame-wise View Registration

As explained in introduction (section I.1), the 3D interaction framework requires spatial knowledge of the local zones with relation to the camera, even when the sensor is moved. However, the embedded device has limited computing power that prevents us from tracking the camera frame after frame similarly to e.g., *simultaneous localization and mapping*. Thus, we need to perform offline registration of RGB-D frame pairs, after the device is done moving (see section A.1.3 for details). Our other constraint of lifelong operation means that compared frames can have modifications of object placement, visibility or illumination.

We saw in section II.2 that there exists a number of ways to register pairs of overlapping frames. 2D descriptors (section II.2.1) detected in the color component are too sensitive to illumination changes, missing data or moving objects. Geometric methods based on 3D descriptors and point comparison e.g., *iterative closest point* (section II.2.2), need to iterate over the whole point set and cannot run fast enough on embedded devices.

In that sense, comparing and matching planes instead of pixels or points between views allows fixing issues of performance and sensitivity to changes. Although, existing plane-based methods showed in section II.2.4 mostly integrate matching plane parameters into global formulations to register a full set of frames, and cannot be applied to our frame-wise scenario.

Hence, we propose in chapter III a novel approach that leverages planes detected in two overlapping 3D views to register pairs of RGB-D frames in a robust and lightweight way.

II.5.2 Lightweight Global Reconstruction

Our embedded context for 3D interaction first required the visualization of the sensor view in an intuitive way for non experienced users. The remote display on mobile phone also added a constraint on the model weight that needs to stay low for real-time feedback. Again, lifelong operation of the device implies static and dynamic motion of objects as well as illumination and texture changes, which have to be considered in the model. To that end, we aim to build an understandable reconstruction of the observed scene that takes into account these constraints, which would lead to a more complete and intuitive visualization for the end user.

State-of-the-art reconstruction methods such as *dense SLAM* (section II.3.1) and *volumetric fusion* (section II.3.2) have high requirements of computing resources that cannot be met on our embedded platform. They also generate heavy detailed models that are not suitable for live wireless transmission. Offline surface regularization as detailed in section II.3.3, while offering lighter models, is currently too slow for our application, as it can require several hours to compute on a desktop computer. These offline methods are usually based on the output of an online point or volume reconstruction, which adds the computing resource constraint as one of their limitations.

We make the observation that there exists no *surface* accumulation structure to aggregate acquired point sets into a global model on the fly. In consequence, we propose in chapter IV a new geometric superstructure, based on simple geometric shapes as explained in section II.4. While simple shapes have a few 3D parameters, the representation of their extent at the surface can be problematic, as it needs to be both accurate and lightweight. In order to take into account both the limits of the extent, holes and salient areas existing at the surface as well as illumination and object arrangement changes, our structure defines local regularization on a simple grid that aggregates incoming depth and color samples into compressed statistics.

We show in chapter V that our simple shape-based reconstruction is lightweight and fast to compute with a moderate memory footprint, while staying accurate enough to be understood and operated by non-expert users.

III

PLANE-BASED REGISTRATION

This chapter focuses on an essential task in the process of recording the real world in 3D, namely the registration of multiple 3D views. We aim to introduce geometric constraints under the form of planes into the registration problem, in order to reduce complexity and take into account the structure of the scene. We present methods to match planes between overlapping 3D views in section III.2 and use these matches to estimate the motion matrix between the views in section III.3. Experiments of 3D view registration based on detected planes are shown in section III.4.

The specificity of our approach is the structure analysis we implicitly perform when categorizing planes given their orientation in the scene and comparing them by pair instead of one by one. In addition, our plane-based motion estimation leverages this classification to separately estimate the different degrees of freedom of the sensor based on plane orientation matching and *quadric error* minimization. Existing methods that ignore the scene structure are more prey to confusion between geometrical elements and require a certain amount and configuration of constraints in order to estimate the full camera motion.

III.1 Context and Motivation

The low amount of structural and high level information contained in a single depth view can be increased by aggregating several views. However, this requires knowledge of the relative position and orientation of the sensor while capturing these views. The problem of image registration, while studied for several decades, gets more complex when considering 3D data, as shown in section II.2. In this study, we showed that estimating the motion between different 3D views is mostly carried out using local 3D descriptors such as *fast point feature histograms (FPFH)* [RBB09], or 2D descriptors using the 3D information of the depth component, when color image is available. Instead, our approach makes use of detected planes in overlapping 3D views to infer relative position and orientation of sensors.

In order to stay as general as possible and as we cannot assume having control over the acquisition of the views, we consider registration of 3D views by single pairs that overlap between roughly 20% and 80% in *intersection over union* of the 3D geometry e.g., image surface or 3D point locations. Empirically, these values give sufficient overlapping features while keeping challenging geometrical differences. We do not aim to use any kind of global optimization as we only consider two views at a time and not the full acquisition. Our algorithm, being based on detected planes, is particularly suited for indoor environments composed of multiple horizontal and vertical planar structures, such as floors, ceilings, walls, doors or tables. More specifically, scenes that follow the *Manhattan world* orientation assumption [CY99] allow better performance.

The inference of sensor motion from the scene structure gives robustness to variability in indoor scenes where small objects might often be moved. Hence, such a registration of scenes acquired at different times is robust to movements of objects, which could confuse local descriptors.

Overview

Our goal is to leverage the scene structure to estimate the motion T of a sensor capturing an indoor scene. We decompose the scene into different plane arrangements, e.g. horizontal and vertical, using our probabilistic framework based on geometric characteristics of planes, such as relative angles α and distances d. By defining a local coordinate frame, we are able to reduce the dimensionality of the computation and separately estimate the degrees of freedom of the sensor. Figure III.1 is a representation of the elements involved in our plane-based registration procedure.

Contributions

We present different strategies to match planes between 3D views and estimate the relative transformation based on the following contributions:

- the definition of a probabilistic framework to analyze the scene structure and separate different arrangements of planes, e.g. horizontal and vertical;
- the comparison of planes by pairs instead of single matches to reduce complexity of the search;
- the definition of multiple heuristics based on geometry and appearance to prevent wrong matches;
- a quadric minimizer between multiple vertical planes to estimate the horizontal translation of the sensor;
- the estimation of a prior motion matrix to gracefully degrade to the state-of-the-art in cases where not enough plane information is available.

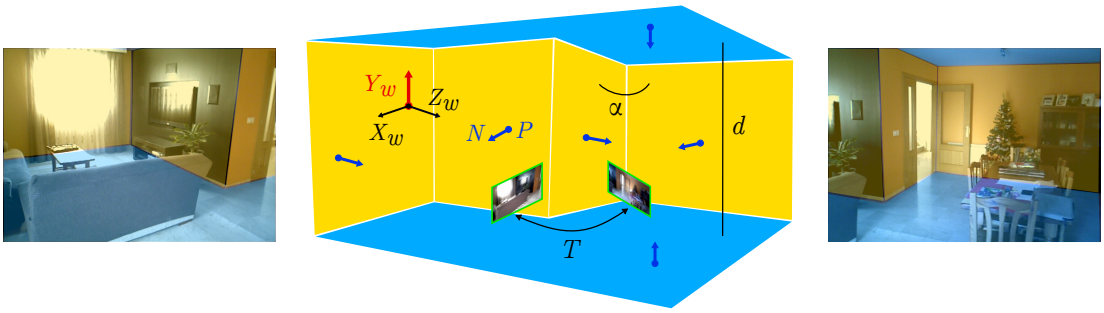


Figure III.1: Overview of the plane-based registration of two RGB-D views. Two RGB-D views are represented by their color component with planar areas overlaid (left and right). In this example, the observed indoor scene (middle) is composed of vertical planes, the walls (orange) and horizontal planes, the floor and ceiling (light blue). A plane is represented by its normal vector $\vec{N}$ and a 3D point P at its surface (dark blue arrows and circles). The scene was captured by the views at the locations shown in green frames when the sensor moved of a motion $T \in \mathbb{R}^{4\times4}$. Using our analysis of the scene, we define the world local frame $\vec{X}_w, \vec{Y}_w, \vec{Z}_w$ where Y_w is the *Up* vector aligned with the *gravity*. Our method is able to recover the sensor motion using plane arrangements constraints such as relative angles α and distances d.

III.2 Plane Matching

In this section, we present strategies to match planes between two views of a scene, modeled as a depth map and more generally as a point cloud. The first step is to estimate the parameters of the planar elements of the scene in both views. This can be done through different methods, detailed in section II.4. In our experiments, we use the *RANSAC-based* plane detection of Schnabel et al. [SWK07]. At the end of this step, we have a list of detected planes, their associated parameters and inlier point positions. In case we have a prior coarse estimation of the motion between the views, we can track the planes instead of detecting and matching them, as shown in section III.2.1.

In the general case where we have no prior knowledge of the motion between the views, we first classify planes following absolute and relative geometry rules, as explained in section III.2.2. In particular, we separate horizontal and vertical planes relatively to the gravity orientation and classify them as parallel or non-parallel. Grouping planes following these arrangements allows reducing the complexity of the matching problem. Then, the matching is done by considering planes by pairs and comparing their relative orientation or distance, as explained in section III.2.3.

III.2.1 Plane Tracking

A straightforward way to match planes between views is to use a prior transformation matrix between the views, that could be estimated through any registration method presented in section II.2. Then, we can run any plane detection method on the first view, apply the prior motion matrix to their parameters, and check if samples of the planes are present in the second view. If the number of inliers that are close enough to the planes in the new view is high enough, then the plane is considered as seen in the new frame and its parameters can be refined using the new inlier set.

III.2.2 Plane Arrangements

In the following, we will consider planes and pairs of planes as belonging to geometric categories with relation to the orientation of the scene, such as:

- horizontal plane, i.e. orthogonal to a reference direction;
- vertical plane, i.e. parallel to a reference direction;
- pair of parallel planes;
- pair of non-parallel planes.

Reference Direction

For a given scene sampled as an RGB-D frame or point cloud, we consider the reference direction to be the gravity vector in the sense of *Newton's law* of universal gravitation in physics. We motivate this choice after observing that most of the components of indoor scenes are either orthogonal or parallel to the gravity vector. In particular, this direction has the advantage of being consistent in the scene and among the objects composing it, from any point of view and representation.

For the rest of this chapter, we consider that the direction of the gravity is known and can be acquired by either one of these means:

- identification of a near horizontal plane among the detected planes;
- inertial device such as an accelerometer, whose orientation in the coordinate frame of the data is known;
- pre-computed and available in a file for loading.

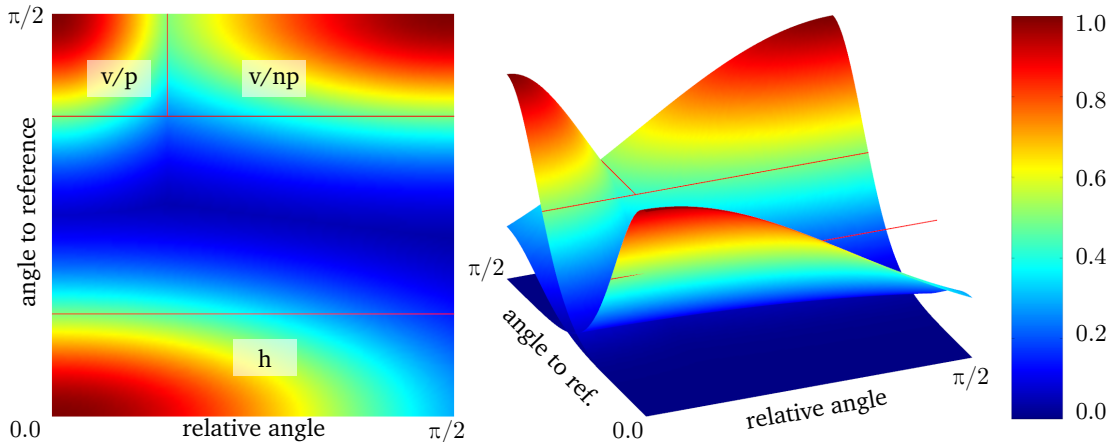


Figure III.2: Detection of plane arrangements using Gaussian distributions. The category of the plane or pair of planes is given by its highest probability. Categories are horizontal (h), vertical parallel (v/p) and vertical non-parallel (v/np). In this illustration, the y axis is the angle of a plane with relation to the reference Up vector. The x axis is the relative angle between two planes. Control over the angle thresholds is given by the standard deviation values of the Gaussian functions. The red lines show the actual angle thresholds in radians at a probability of 0.5.

Plane Classification

In order to classify planes as horizontal or vertical and parallel or non-parallel, we define the following probabilistic framework. Figure III.2 shows visual representation of the plane classification probabilities. For a plane of angle deviation $\alpha_{up} = \arccos(|\vec{N}.\vec{Y}_w|)$ with the reference direction $\vec{Y}_w$ and a pair of planes (i, j) of relative angle deviation $\alpha_{rel} = \arccos(|\vec{N}_i.\vec{N}_j|)$, the class of the plane is defined by Equation III.1.

$$k(\alpha_{up}, \alpha_{rel}) = \arg\max_k \; g(\alpha_{up}, \mu^k_{up}, \sigma^k_{up})\; g(\alpha_{rel}, \mu^k_{rel}, \sigma^k_{rel}) \tag{III.1}$$

Here, $g(\alpha, \mu, \sigma)$ is the value at α of a Gaussian function centered on μ with standard deviation σ, as shown in Equation III.2.

$$g(\alpha, \mu, \sigma) = e^{\frac{-(\alpha-\mu)^2}{2\sigma^2}} \tag{III.2}$$

The probability distributions $g(\alpha)$ for the three categories are centered at the following values:

$$\begin{array}{lll} \text{horizontal} & \mu_{up} = 0 & \mu_{rel} = 0 \\ \text{vertical parallel} & \mu_{up} = \pi/2 & \mu_{rel} = 0 \\ \text{vertical non-parallel} & \mu_{up} = \pi/2 & \mu_{rel} = \pi/2 \end{array} \tag{III.3}$$

The value of σ for each category is set to control the values of the angle thresholds by arbitrarily fixing the probability at 0.5. In practice, we evaluate the standard deviation value σ for each Gaussian function using an angle threshold value α_{thresh} and the fixed probability threshold 0.5, as shown in Equation III.4.

$$\sigma(\alpha_{thresh}) = \sqrt{\frac{-(\alpha_{thresh} - \mu)^2}{2\ \log(0.5)}} \tag{III.4}$$

In that formulation, the σ values represent uncertainties associated with the plane classification, as they model the extent of the higher probability values over the angle domain. Tuning these values gives control over the angle thresholds as well as the uncertainty of the classification.

III.2.3 Plane Association

We now consider that we have a list of planes categorized as horizontal or vertical and parallel or not. In order to match planes that correspond to the same part of the actual scene, we will use this classification to reduce the amount of possible matches. Hence, we first group horizontal and vertical planes together. While we could simply try to match planes with each other in both views, we choose to consider planes as pairs and not as single planes, in order to further reduce potential wrong matches.

Plane Pairs Generation

We generate all possible pairs of vertical and horizontal planes, regardless of the order. Then, we compare all pairs in views a and b and estimate whether or not the two pairs (i^a, j^a) and (i^b, j^b) are composed of two plane matches (i^a, i^b) and (j^a, j^b). A first test is performed using a simple penalty e in plane pair space, that is agnostic to the view:

- for pairs of non-parallel planes, i.e vertical non-parallel, the plane pair penalty is the difference of relative angles α between the normals of the planes:

$$e_{(i^a,i^b),(j^a,j^b)} = |\alpha_{i^a j^a} - \alpha_{i^b j^b}| = |\arccos(|\vec{N}_{i^a}.\vec{N}_{j^a}|) - \arccos(|\vec{N}_{i^b}.\vec{N}_{j^b}|)| \ ; \tag{III.5}$$

- for pairs of parallel planes, i.e vertical parallel or horizontal, the plane pair penalty is the difference of relative distance d in the normal directions of the planes:

$$e_{(i^a,i^b),(j^a,j^b)} = |d_{i^a j^a} - d_{i^b j^b}| = |\vec{N}_{i^a}.(P_{i^a} - P_{j^a}) - \vec{N}_{i^b}.(P_{i^b} - P_{j^b})| \ . \tag{III.6}$$

For each pair of planes in the two views, if this penalty e is below a given threshold, we keep it and further check if the pair is an actual match, based on several heuristics, as shown below. In practice, we keep a pair match if the difference is below 10 degrees in angle or 10 cm in distance. These loose thresholds allow accounting for the noise disturbing the plane parameters between acquisitions.

Plane Pairs Validation

First, we compute the motion matrix transforming the planes from one view to another, using the method from section III.3. For pairs of vertical planes, we compute the rotation (section III.3.1) and horizontal translation (section III.3.2). For pairs of horizontal planes, we compute the vertical translation (section III.3.3). If the magnitude of the computed transformation is too high, we consider the match as wrong and discard it. If it is low enough, we apply it to the second pair of planes and validate the match using plane-wise comparisons of:

- the similarity of normal orientations and distances to origin;
- the overlap of convex hulls;
- the average Euclidean distance of 100 inliers of each plane to the other plane;
- the similarity of color histograms as described by Dou et al. [DGFF12a].

After comparing all generated plane pairs, in case plane pair matches create more than one match for a single plane, we keep the match with the closest plane distance. If there are not enough planes to generate pairs, or if no plane pairs have been matched, we consider single planes and compare them one by one using the previously described heuristics. Finally, we recompute the motion between the two views using all plane matches derived from the pair matches, using the method described in section III.3.

III.3 Transformation Computation

In this section, we describe a novel method to estimate the global transformation between two 3D views composed of matching planes, based on the plane structural classification defined in section III.2.2. We consider planes as horizontal and vertical and split the degrees of freedom of the transformation the same way. In particular, while the rotation is computed in camera coordinate frame, the translation is first computed in world coordinate frame, considering the known Up vector as Y axis, and then converted to camera coordinate frame. Hence, the computation of the transformation is divided into three steps:

- estimation of the rotation in camera space X, Y, Z;
- estimation of the horizontal translation with vertical planes along X_w, Z_w;
- estimation of the vertical translation with horizontal planes along Y_w.

As an optional pre-processing step, we refine the Up vector in each view by taking the median value of deviations of horizontal plane normals to the current Up vector.

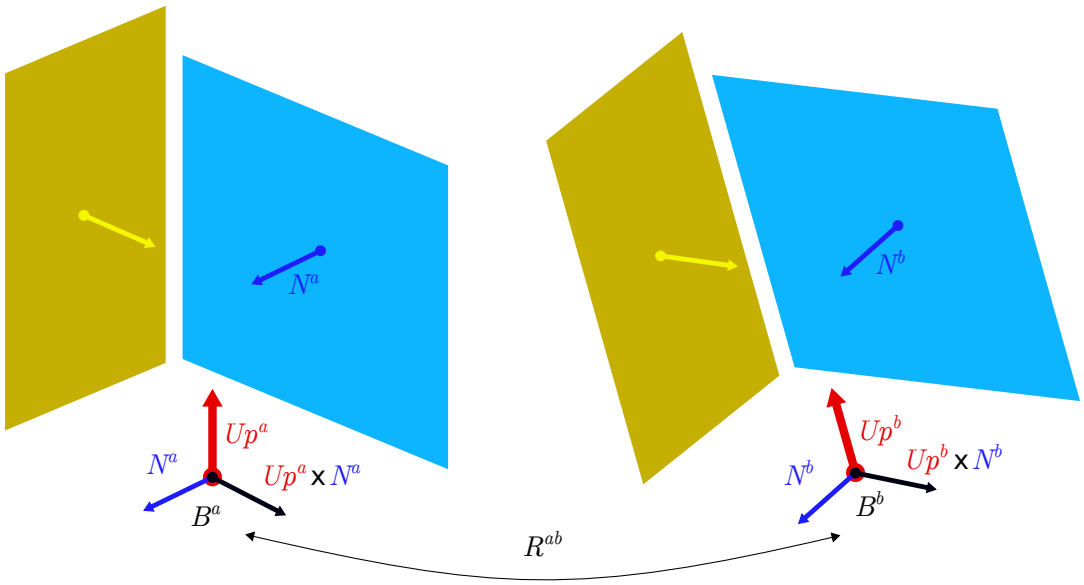


Figure III.3: Rotation computation using a matching vertical plane. As the Up vector is known in both views, knowledge of a matching vertical plane of normal N is enough to build a local coordinate frame $\{Up,\ N,\ Up \times N\}$. Associating the local coordinate frames in both views a and b leads the complete rotation matrix between the views, as shown in Equation III.7. The same computation can be done for the second plane and rotations R^{ab} can then be averaged.

III.3.1 Rotation Computation

For views a and b of reference Up vectors Up^a and Up^b, the local coordinate frames for a plane of normal N are given by B^a and B^b. The rotation matrix in world space can be inferred as R^{ab}, as shown in Equation III.7 and illustrated in Figure III.3.

$$\begin{array}{rcl} B^a & = & (N^a\ Up^a\ Up^a \times N^a) \in \mathbb{R}^{3\times3} \\ B^b & = & (N^b\ Up^b\ Up^b \times N^b) \in \mathbb{R}^{3\times3} \\ R^{ab} & = & {B^a}^T B^b \in \mathbb{R}^{3\times3} \end{array} \tag{III.7}$$

III.3.2 Horizontal Translation

We use the vertical planes matching between two views to compute the translation in the horizontal plane X_w, Z_w of the world space. First, we apply the rotation computed in the previous step and we project the plane normals and positions in this 2D space. At this point, matching planes are aligned in orientation and are only transformed by a translation.

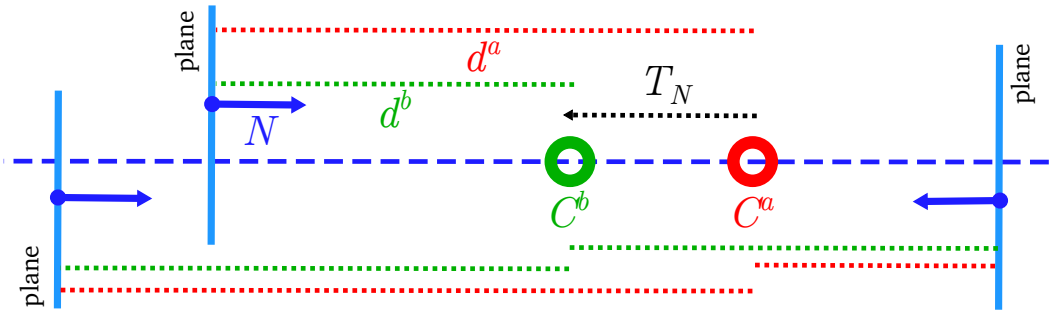


Figure III.4: Computation of the translation in horizontal 2D X_w, Z_w world space with planes (light blue) aligned along a single direction, here represented by the blue dotted line. C^a and C^b are the center of cameras in the two views, projected on this line. The known distances to origin d^a and d^b (dotted red and green lines) of planes in the views allow recovering the value of the translation T_N in the direction N of the planes.

We have knowledge of the planes 2D normals and distances to origin in the two views, and the goal is to estimate the relative position of the camera origins in 2D horizontal space. Given the relative orientations of the vertical planes, we have two choices:

- if all planes are parallel and have the same direction, we can only compute the 2D translation in this direction;
- if at least one pair of planes is non-parallel, we use a *quadric error minimizer* to compute the full translation in X_w, Z_w world space.

Parallel Planes

In the case where all vertical planes are parallel as defined in section III.2.2, we can only compute the translation of the camera along their normal direction in horizontal X_w, Z_w world space, as illustrated in Figure III.4. As we know the distances to origin of the plane in both views d^a and d^b, we can compute their difference and apply it to the common direction to get a translation vector $T_N = (d^b - d^a)\ N$. This vector can then be converted into world space as $T_{X_w,Z_w} = [T_N.X_w\ T_N.Z_w]$.

Non-Parallel Planes

In the case where there is at least one pair of non-parallel vertical planes, we can compute the full translation of the camera in horizontal X_w, Z_w world space. Figure III.5 illustrates the setup in horizontal X_w, Z_w world space where multiple planes have non-parallel directions.

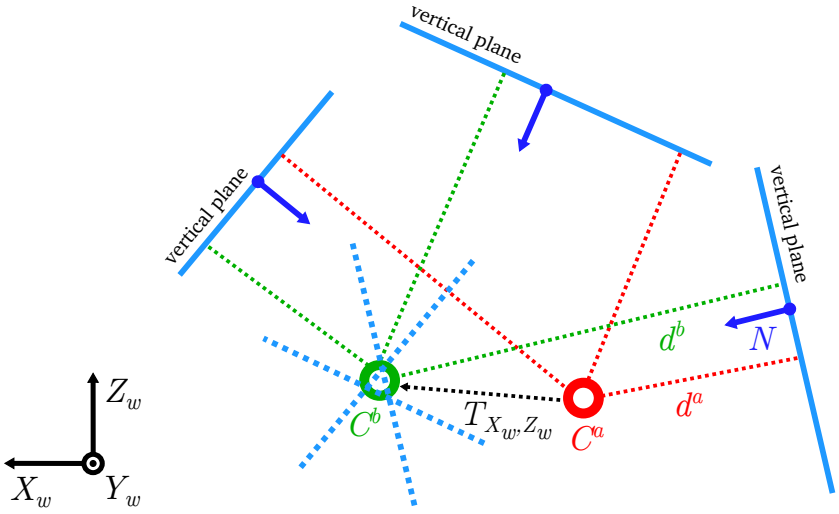


Figure III.5: Computation of the translation in horizontal X_w, Z_w world space with non-parallel plane directions. C^a and C^b are the center of cameras in the two views. Blue lines with arrows represent tracked vertical planes projected in 2D X_w, Z_w world space. Dotted blue lines are the planes displaced of the distance to origin. Dotted red and green lines represent the distances d^a and d^b of vertical planes to the center of the camera in the first and second frames, respectively. The quadric minimizer for those displaced planes leads the translation of the camera in horizontal world space T_{X_w,Z_w}.

In order to minimize the error, we make use of the quadric error to the planes and their minimizer point as described by Garland and Heckbert [GH97]. For each vertical plane of normal N, we compute the fundamental error quadric in 2D space X_w, Z_w as

$$K = pp^T \text{ with } p = [a\ b\ -d]^T = [N.X_w\ N.Z_w\ -(d^a - d^b)]^T\ . \tag{III.8}$$

By defining the distance d as the difference between distances to the origin of the planes in the first and second views, we simulate the displacement of the planes to the origin of the second view, while taking the origin of the first as reference. After summing the K matrices for all planes, we solve for the minimizer as described by Garland and Heckbert, which describes the translation along the X_w and Z_w axes in world space T_{X_w,Z_w}.

III.3.3 Vertical Translation

For the vertical translation, we consider the parallel case of the horizontal translation as illustrated in Figure III.4, with the Y world Up vector as common direction. Using all horizontal planes, we compute the displacement along the Y axis as in section III.3.2 with $N = Y_w$.

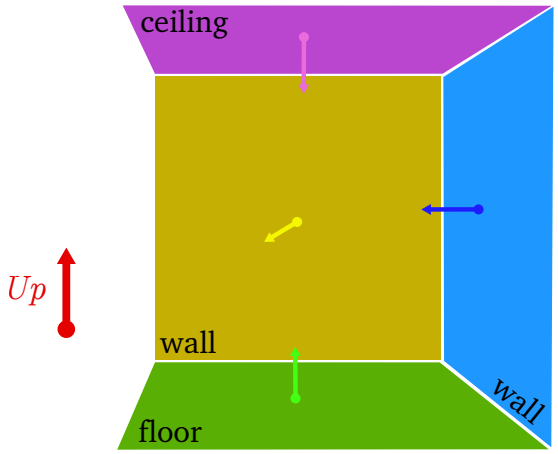


Figure III.6: Toy example used to validate the plane-based transformation computation. The red arrow is the Up vector. The blue and yellow planes are orthogonal vertical ones. The green and purple are horizontal planes, modeling the floor and ceiling respectively.

III.4 Experiments

In section III.4.1, we validate our motion computation method using a toy example. Then, we evaluate both our plane matching and plane motion estimation strategies on recent public datasets presented in section III.4.2. We compare our methods to state-of-the-art 3D view registration methods.

III.4.1 Toy Example

In order to validate the computation of the transformation as detailed in section III.3, we designed a toy example, illustrated in Figure III.6. The example models views of an indoor scene composed two orthogonal vertical planes, that can be seen as a left and far walls, and two horizontal planes to represent the floor and ceiling.

With fixed initial plane parameters in one view, we randomly compute values of rotation of the camera with relation to the Up vector, as well as a transformation matrix that we apply to the parameters, in order to run the registration algorithms on two views.

We show experiments on this toy example while adding uniform noise in Figure III.7. To simulate sensor noise, we sample a set of 3D points from the generated plane parameters and add uniform noise to their positions to re-estimate the parameters from this noisy inlier set. We can see that with no noise added to the plane inliers, the transformation matrix is always recovered with no error, which validates our motion estimation error. Although, we notice that the accuracy of the transformation quickly drops after 30% of added synthetic noise, which highlights the sensitivity of our algorithm to noisy planes.

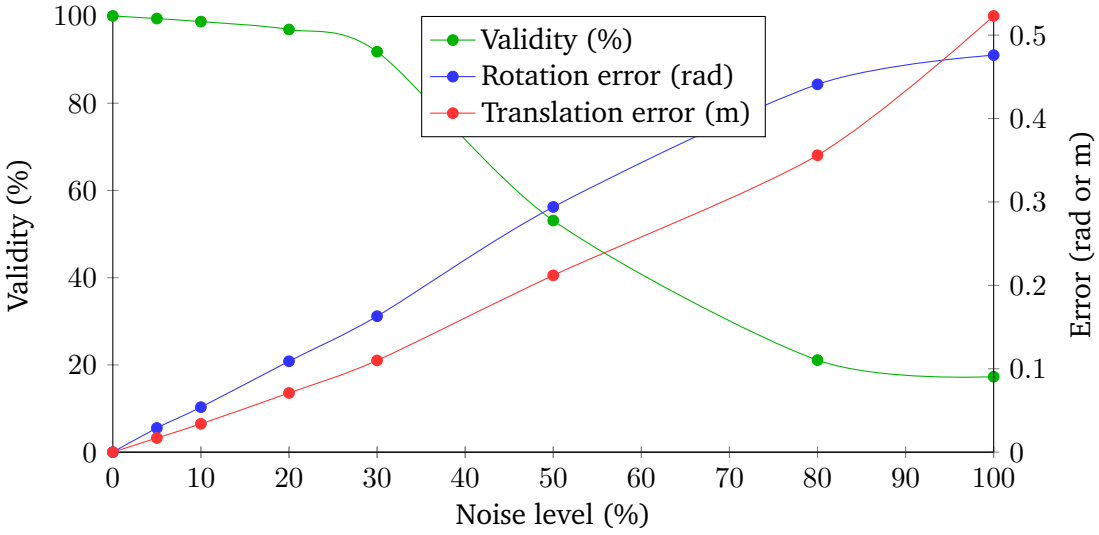


Figure III.7: Adding noise on the toy example for registration. After applying the random transformation to the planes in both views, we sample the plane surface with 400 random 3D point locations to which we apply uniform noise displacement as a percentage of a maximum value of 2m, which ensures covering most noise and outlier values observed in practice. Although this noise model is not the one observed on samples from depth sensors, it gives a first insight on the influence of noise on a controllable example, while staying simple and general. We then re-estimate the normals and centers of the planes using *singular value decomposition* of the noisy point set and use them in the plane-based registration algorithm. The rotation and translation errors are the norm along the three axes. A registration is considered valid if its rotation angle error is below 20 degrees and its translation error is below 20cm. Each noise level experiment was ran 10000 times and averaged. This experiment first validates our motion estimation in the perfect case where no noise is present. Then, it shows a good robustness to noisy point sets up to about 30% of noise, which is already a high amount. The use of planes even in noisy depth maps allows smoothing the error and recovering correct information in many cases.

In this experiment, we generate 400 noisy points to re-estimate plane parameters, but in practice with real captured data, the aggregation of thousands of inlier point positions to compute the plane parameters allows the noise in the observed data to stay low.

III.4.2 Dataset

In order to evaluate the accuracy of our plane-based registration, we make use of the publicly available RGB-D dataset *SUN3D* [XOT13]. In particular, we use the benchmark provided by the authors of *fine-to-coarse* [HF17] and *3DMatch* [ZSN+17]. The former provides ground truth correspondences in overlapping RGB-D frames, which we use to compute error values and evaluate the accuracy of our registration. The latter, similar to the synthetic benchmark of Choi et al. [CZK15], provides scene fragments generated from 50 fused RGB-D frames at 6mm resolution, associated with ground truth transformations and correspondences.

As specified by Choi et al., we consider a registered pair as valid if the average error of all correspondences after applying the transformation, computed as the averaged Euclidean distance between transformed corresponding 3D point locations, falls below the threshold of *20cm*. We then attempt to register pairs of frames and fragments with available ground truth motion and correspondences.

We quantitatively evaluate the registration with the following metrics:

- *success*: the amount of pairs that the algorithm is able to register;
- *recall*: the amount of correct pairs that the algorithm successfully registers and are valid;
- *precision*: the amount of registered pairs that are valid, i.e. with a correspondence error below the threshold;
- *root mean squared error (RMSE)*: the average error of the ground truth correspondences for all registered pairs;
- *median absolute error (MAE)*: the median error of the ground truth correspondences for all registered pairs;
- *time*: the average total number of milliseconds required to register a pair.

We compare our plane-based registration to state-of-the-art *pairwise* registration methods using *ORB* keypoint descriptors in RGB image [RRKB11] and *fast point feature histograms (FPFH)* [RBB09]. In the following tables, we present the evaluation metrics of our methods with relative difference to the accuracy of *ORB* and *FPFH*.

Method	Success (%)	Rec (%)	Prec (%)	RMSE (m)	MAE (m)	Time (ms)
ORB	78.1	42.7	53.2	0.771	0.133	8
FPFH	73.5	41.8	55.6	0.649	0.141	2470
Plane pairs	38.7	11.4	27.1	1.01	0.483	396
vs ORB	-50%	-73%	-49%	+31%	+263%	+5K%
vs FPFH	-47%	-73%	-51%	+56%	+243%	-84%

Table III.1: Registration of single RGB-D frames using our plane matching strategy.

Method	Success (%)	Rec (%)	Prec (%)	RMSE (m)	MAE (m)	Time (ms)
FPFH	90.2	54.1	68.2	1.13	0.100	3004
Plane pairs	73.7	14.3	21.0	1.254	0.865	987
vs FPFH	-18%	-74%	-69%	+11%	+765%	-67%

Table III.2: Registration of scan fragments using our plane matching strategy.

III.4.3 Plane Matching

To evaluate our plane matching strategy presented in section III.2, we run plane detection in both frames separately and feed the planes to the algorithm to match them and estimate the motion between the views. The method will be successful as long as the frames have a common vertical plane that is correctly identified. However, the validity of the estimated motion matrix – or precision – is evaluated using the ground truth motion and correspondences provided by the datasets.

Table III.1 shows quantitative evaluation of single RGB-D frames registration using our plane matching method, while Table III.2 shows evaluation of scan fragments registration. In both cases, while the plane-based method shows significant speed-up over *FPFH*, we can see a noticeable degradation of the accuracy. We put that degradation down to the lack of distinguishable features between geometrically similar planes, even when taking appearance histograms into account. In consequence, we consider a prior registration estimate, however coarse, as needed to match the planes before computing the transformation.

III.4.4 Motion Computation

As we notice that geometric planes are not characteristic enough by themselves to be matched even when classified and grouped by pairs, we first compute a coarse transformation matrix to track the planes as presented in section III.2.1, and then refine this estimate using our method presented in section III.3. The method will be successful as long as the prior registration is successful and the frames in the pair have a common horizontal or vertical plane. In the case where some planes are missing to compute the full 6 degrees of freedom of the transformation, we use the prior estimate to fill in for the components of the rotation and translation that we can not compute.

Method	Success (%)	Rec (%)	Prec (%)	RMSE (m)	MAE (m)	Time (ms)
ORB	78.1	42.7	53.2	0.771	0.133	8
FPFH	73.5	41.8	55.6	0.649	0.141	2470
ORB + pl.	52.4	34.4	63.3	0.732	0.122	228
vs ORB	-33%	-19%	+19%	-5%	-8%	+3K%
vs FPFH	-29%	-18%	+14%	+13%	-13%	-91%
FPFH + pl.	63.7	33.6	52.0	0.837	0.169	2668
vs ORB	-18%	-21%	-2%	+9%	+27%	+33K%
vs FPFH	-13%	-20%	-6%	+29%	+20%	+8%

Table III.3: Registration of single RGB-D frames using planes tracked with a prior estimate. Using *ORB* as prior allows improving accuracy, while showing significant speed-up over *FPFH*.

Method	Success (%)	Rec (%)	Prec (%)	RMSE (m)	MAE (m)	Time (ms)
FPFH	90.2	54.1	68.2	1.13	0.100	3004
FPFH + pl.	90.1	28.3	35.8	1.26	0.365	3855
vs FPFH	-0%	-48%	-48%	+12%	+265%	+28%

Table III.4: Registration of scan fragments using planes tracked with a prior estimate. The higher amount of information present in fragments allows *FPFH* to perform better than the plane-based method.

Again, we compute evaluation metrics for the registration of single RGB-D frames in Table III.3 and scan fragments in Table III.4. For RGB-D frames, computing a prior transformation with *ORB* keypoints leads to a 10 to 20% increase in precision, while reducing the median error of about 10%. The significant speed-up of 90% over *FPFH* shows a real interest of the plane-based method when coupled with *ORB*. However, using *FPFH* as a prior registration method, while allowing more frame pairs to succeed, reduces the accuracy of the the transformation. For scan fragments, the plane-based method lacks accuracy improvement, as the prior *FPFH* motion estimate is likely accurate enough. While scan fragments aggregate the information from 50 single frames, more distinctive geometric features should overlap between the views, allowing the *FPFH* descriptor to perform well.

Figure III.8 shows corresponding planes in two views used to compute the transformation. Figure III.9 shows the two registered views in a common space. In this example, we can see that even low overlap is sufficient to register the view if enough common planes are present. Here in both frames, the floor plane allows recovering the gravity vector and its association with the wall planes leads the full three angles rotation. Translation is then recovered using the floor (vertical translation) and wall (horizontal translation).

Figure III.8: Corresponding planes in two RGB-D views used to compute their relative transformation. In this example, we use frames 301 and 401 of scene *home_at/home_at_scan1_2013_jan_1* from the *SUN3D* database. Top: plane inliers overlaid on RGB images. Bottom: side and top views showing detected planes in 3D with their normal vectors in thin green and the world frame in red, green and blue lines. Only the blue horizontal and purple vertical planes are common in the two views, but they are sufficient to compute the sensor motion.

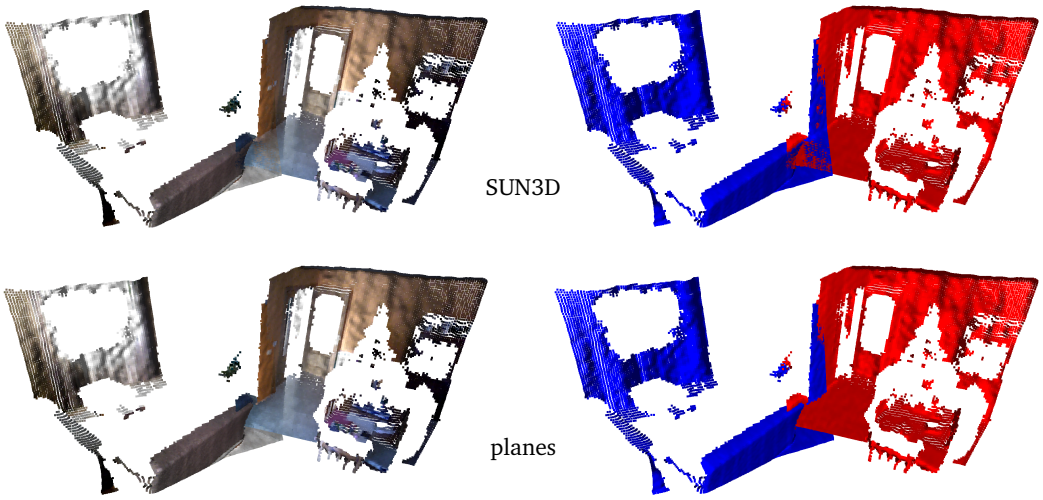


Figure III.9: Plane-registered RGB-D views in a common space using *SUN3D SfM* [XOT13], giving matrices as ground truth in the dataset (top), and our plane-based method (bottom). In these two frames, the overlap is rather low and the overlapping region does not contain many distinctive features, which are essential to the performance of regular 3D descriptors. Here, recovering two overlapping planes allows us to compute the full transformation between sensor positions. We can see that the difference with the ground truth is subtle, showing the high quality of the plane-based method.

III.5 Discussion and Perspectives

In this chapter, we introduced planar geometric and structural constraints into the problem of 3D view registration. We have presented multiple ways to make use of planar structures seen in overlapping 3D views in order to estimate the motion between them. We exploit the *Manhattan world* structure of indoor scenes to recover the motion of the sensor with relation to this structure. This allows the methods to be lightweight and makes them agnostic to subtle changes in the positions of objects. While results could be improved, we like to see these contributions as incremental and designed to be paired with other existing methods in order to reach state-of-the-art accuracy and performance. We discuss them below and give hints towards improvement of robustness and accuracy.

In the following chapters, we enforce temporal and spatial consistency by tracking planes as defined in section III.2.1. This tracking will enable consistent knowledge of geometric elements of the scene in time and allow building a complete superstructure representing the surroundings, as shown in chapter IV.

III.5.1 Discussion and Limitations

The analysis of the scene structure, agnostic to subtle changes in scene geometry of e.g., small objects, is particularly well suited to lifelong 3D information aggregation. In that sense, our use of large planar structures to register views with each other is an advantage over state-of-the-art local descriptors. In particular, the analysis of relations between these structures and the definition of plane pair distances that are agnostic to the point of view adds robustness to the registration process.

Computing the sensor motion from separated horizontal and vertical planes to split computation of rotations and translations works well in practice when used as a refinement step after a prior coarse estimation method. We show a significant improvement in accuracy over the prior method, as well as fast processing compared to 3D descriptors. However, the requirement for overlapping planar structures in the two views leads to a lower success rate.

For view matching, detected planes do not seem to be strong distinguishable features in an indoor scene, where more complex 3D descriptors give better results. The grouping of planes by pairs, while reducing complexity of the problem, does not prevent wrong plane matches as much as expected. On the other hand, the low success rate can be explained by the requirement for overlapping *Manhattan world* structures in the views, which are not always present.

III.5.2 Towards Robust Plane-based Registration

While geometric planes offer a more stable support than 3D points to estimate the motion of a sensor in a scene, they also embed less information that regular 3D descriptors and can lead to some confusion. We give here multiple research paths and challenges for the improvement of robustness in plane distinction and estimation of the transformation matrix, in order to improve current results.

Plane Matching

In order to ease the distinction of different geometrically similar planes and prevent confusion, we could use stronger heuristics. When color information is available, the comparison of color keypoints detected in the 2D space of the plane could help discarding wrong matches.

While we presented associations of two planes to form pairs and their relative distances, we could imagine grouping planes by three or four. We would need to design specific descriptors based on geometric information in order to discriminate the correct groups from each other.

Finally, we could add a quality check to discard wrong plane pair matches. By applying the estimated transformation for a plane pair match to other already detected plane pair matches, we could identify whether the motion matrix correctly transforms planes from a view to another, and discard the new plane pair if not.

Transformation Computation

In order to reduce error on the translation, when possible, we could use an horizontal plane associated with two non-parallel planes. After applying the rotation, their unique intersection point will allow computing the translation of the sensor.

We could imagine lowering the quality of the prior 3D descriptor method to speed it up, while keeping enough accuracy for plane tracking. This would allow faster processing with a plane-based refinement step to reach state-of-the-art accuracy.

Evaluation

In order to further understand the behavior of both algorithms in failure cases, we could investigate specific cases by implementing visual quality check. In addition, the model of the noise we add to the toy example is rather far from the noise observed in captured data. We could imagine using a more advanced depth sensor simulator on our synthetic toy dataset in order to get a more meaningful evaluation.

IV GEOMETRIC SUPERSTRUCTURE

We saw in chapter III that camera and plane tracking lead to a temporally and spatially consistent set of geometric shapes representing an observed scene. In this chapter, we focus on turning these consistent geometric objects into a multi-shape geometric superstructure that models structural elements of the scene. In particular, we define a spatially consistent grid to model the extent of the shapes and gather local statistics.

IV.1 Context and Motivation

As shown in section II.1, modern commodity depth sensors provide real time RGB-D stream which has end applications in multiple domains, from human computer interaction and augmented reality to industrial design. Although, recent advances in sensor technology are not sufficient and their impact is still limited by the low quality and unreliability of their data stream. In particular, limitations include low resolution of the frames and inherent noise as well as incompleteness and temporal inconsistency due to single view capture.

In this chapter, we present a new multi-shape geometric superstructure to improve real time RGB-D streams by analyzing them. A sparse set of detected 3D shapes are parameterized to record statistics extracted from the stream and form a structure that we call *proxy*. This superstructure substitutes the RGB-D data and approximates the geometry of the scene. In particular, our contributions are:

- a stable and lightweight multi-shape geometric superstructure for RGB-D data (section IV.3);
- construction and updating methods which are spatially and temporally consistent (section IV.2);
- a collection of RGB-D enhancement methods based on our structure which run on the fly (section V.2).

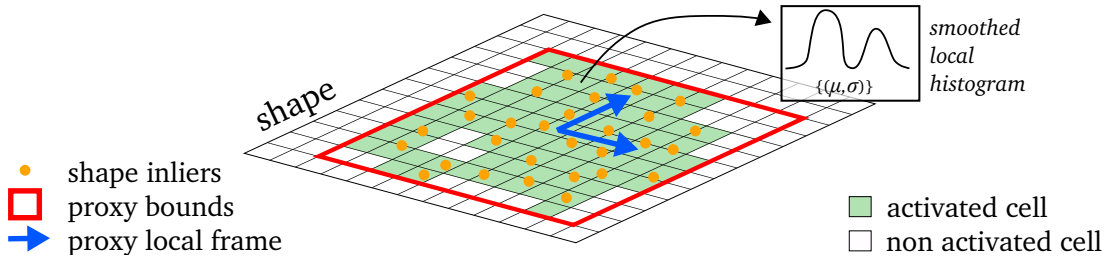


Figure IV.1: Geometric superstructure model. Built upon a shape in 3D space (e.g., a plane), our proxy model is made of a local frame, bounds and a grid of cells which contain statistics. These statistics are a collection of mean μ and variance σ values representing a *smoothed local histogram*, as well as quantized color information, all gathered from the RGB-D data. They are detailed in section IV.3.3. Activated cells are the ones containing inliers from many frames.

IV.2 Multi-shape Modeling

Basically, our model represents RGB-D data which is often seen and consistent through frames and space, hence revealing the dominant structural elements in the scene. To do so, they take the form of a multi-shape geometric superstructure, where each proxy is equipped with a local frame, bounds and, within the bounds, a regular 2D grid of rich statistics, mapped on the shape and gathered from the RGB-D data. Figure IV.1 gives visual insight of a geometric proxy. A proxy can have the shape of a plane, a cylinder or a sphere. Our implementation is based on the *efficient RANSAC* method by Schnabel et al. [SWK07], which is detailed in section II.4.2 (paragraph *RANSAC-based*).

IV.2.1 Shape Detection and Tracking

We build geometric proxies on the fly and update them through time using solely incoming raw RGB-D frames from the live stream. More precisely, for each new RGB-D image $X_t = \{I_t, D_t\}$ (color and depth), we run the procedure described in Algorithm 1 and Figure IV.2. Figure IV.3 shows the different steps of the construction of the proxies on one specific example.

The initial depth filtering (step 1) is based on a bilateral convolution [TM98] of the depth map using a Gaussian kernel associated with a range check. This allows discarding points further than a depth threshold from the current point, which could create artificial depth values if taken into account. In our experiments, we choose to set this threshold to *20cm*, which allows filtering together parts of the same object, while ignoring the influence of unrelated objects. Due to the embedded processing constraint, we estimate the normal field (step 2) through the simple computation of the depth gradient at each pixel, using the sensor topology as domain.

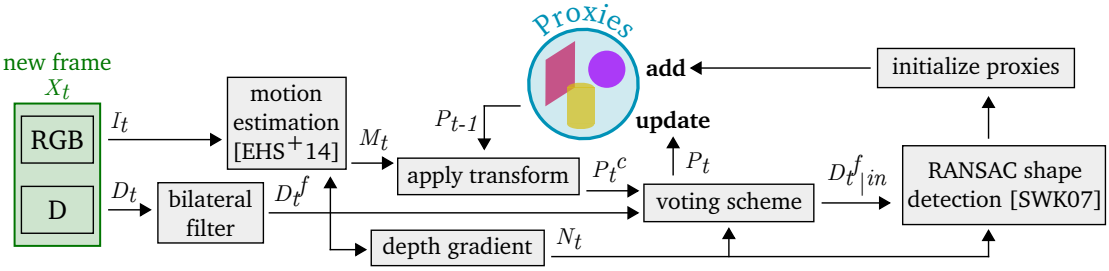


Figure IV.2: *Overview* of proxy structure construction. This procedure is ran for each new RGB-D image X_t at frame t, made of a color image I_t and a depth map D_t. The proxy update allows *accumulating* samples from D_t in the *superstructure* to compute statistics, as detailed in section IV.3. D_t^f: low-pass filtered version of D_t; M_t: camera motion matrix; N_t: normal vectors associated with D_t; P_{t-1}: proxies detected at frame $t-1$; P_t^c: candidate proxies; P_t: proxies at frame t; $D_t^f{}_{|in}$: low-pass filtered depth points without inliers from P_t.

The estimation of the camera motion from the previous frame (step 3) is inspired from *RGB-D SLAM* introduced by Endres et al. [EHS+14], using point features from I_t. However, any egomotion estimation algorithm can be used at this step, as all we need is the values of the six degrees of freedom localizing an RGB-D camera in its environment. Examples of such algorithms are given in section II.2.

In order to keep or discard previously detected proxies (step 4.2), we define a voting scheme where samples of X_t which are inliers of a given previous proxy cast their vote to this proxy and are marked. Then, the per-proxy vote count in the new frame indicates whether the proxy is preserved or discarded (step 4.3). Preserved proxies are updated with X_t, hence see their parameters refined and occupancy statistics updated with new inliers. Discarded proxies are placed in *probation* state for near-future recheck with new incoming frames, and purged if discarded for too long. However, in order to avoid losing information on non-observed but important parts of the scene, we do not purge proxies that have been seen a large number of times, which stay in probation instead.

When new proxies have been detected (step 5.1), similar ones are merged together in order to avoid modeling different parts of geometric surfaces with multiple proxy instances (step 5.2). The proxy is then initialized (step 5.3) with a bounding rectangle and a local frame computed to be aligned with the scene orientation (more details in section IV.4.1). Using the global scene axes to compute the local frame leads to a fixed resolution and spatial consistency for the grid of all proxies and allows efficient recovery and fusion (step 5.2).

Algorithm 1 Proxy structure construction

Notations:
X_t: current RGB-D frame;
I_t: current RGB frame;
D_t: current depth frame;
P_t: current proxies;

$P_t \leftarrow \varnothing$
function BUILDPROXIES($X_t = \{I_t, D_t\}$)
1. filter D_t with a bilateral depth convolution;
2. estimate the normal field N_t from D_t;
3. estimate the camera motion M_t from X_{t-1} [EHS+14];
4. search for previous proxies in X_t:
   4.1 register previous frame proxies to X_t using M_t;
   4.2 cast votes from samples of X_t to previous proxies;
   4.3 given the vote count for each previous proxy:
      keep it, update it with X_t and add it to P_t;
      discard it and place it in probation state;
      purge it if it has been discarded for too long;
5. detect new proxy shapes in $X_t \setminus inliers(P_t)$:
   5.1 RANSAC-based shape detection [SWK07];
   5.2 post-detection shape fusion;
   5.3 compute the local frame;
   5.4 initialize the new proxy with X_t;
   5.5 register new proxy to global space using M_t.

return P_t
end function

Finally, initial occupancy statistics are computed using X_t (step 5.4), by recovering the coordinates of the cell in the proxy grid corresponding to each inlier. In order to take into account the point of view when modeling the scene, grid cell coordinates for an inlier depth point are computed by projecting it upon the detected shape following the direction between the camera and the point. This projection is illustrated in Figure V.3. As a last step, proxies are transformed from local depth frame into global 3D space in order to be tracked in the next frames (step 5.5).

IV.2.2 Spatial Extent Modeling

In order to model the observed elements as faithfully as possible, we need to take into account the extent of these elements at the surface of the shape. In addition, we wish to store local information on parts of the shape to avoid smoothing small details, which requires quantization of the shape surface.

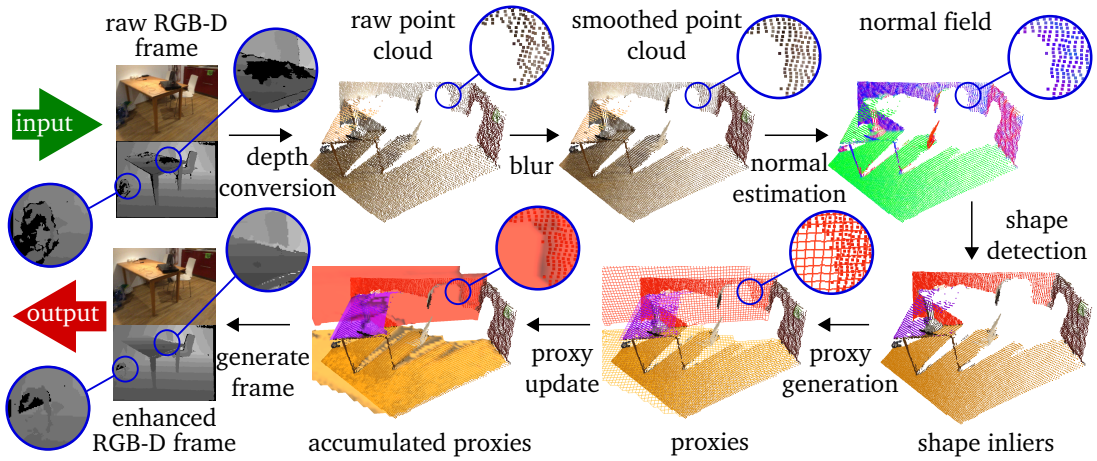


Figure IV.3: Example of proxy structure construction. The input RGB-D frame is converted into a raw point cloud to which a low-pass filter is applied, followed by normal estimation (top). RANSAC-based shape detection [SWK07] is applied and used as a basis for the construction or the update of our proxies, following the structure shown in Figure IV.1 (bottom right). Accumulated proxy cells (bottom) are visualized with colors weighted by their occupancy probabilities: darker cells have a low probability and represent low confidence areas, whereas brighter cells represent areas with high confidence. Proxies are then used to generate enhanced RGB-D frames, as detailed in section V.2. The whole process runs on the fly.

When proxy shapes are detected, the shape is parameterized based on its local frame in order to define a fixed grid with relation to the actual object. This parameterization from 3D space at the surface of the shape to 2D space is described below for planes and revolution surfaces such as cylinders and spheres.

Planes

We define local axes in the space of a plane based on the reference orientation of the world, as detailed in section IV.4.1. This leads to a consistent orientation of the grid of all proxies, where local axes $\vec{X}$ and $\vec{Y}$ both belong to the plane. This local frame allows us to compute a consistent parameterization of the shape at any given frame. In particular, a 3D point P belonging to a plane of origin point C will have coordinates in the local frame of the proxy defined as

$$\begin{cases} u = (P - C).\vec{X} \\ v = (P - C).\vec{Y} \end{cases} \tag{IV.1}$$

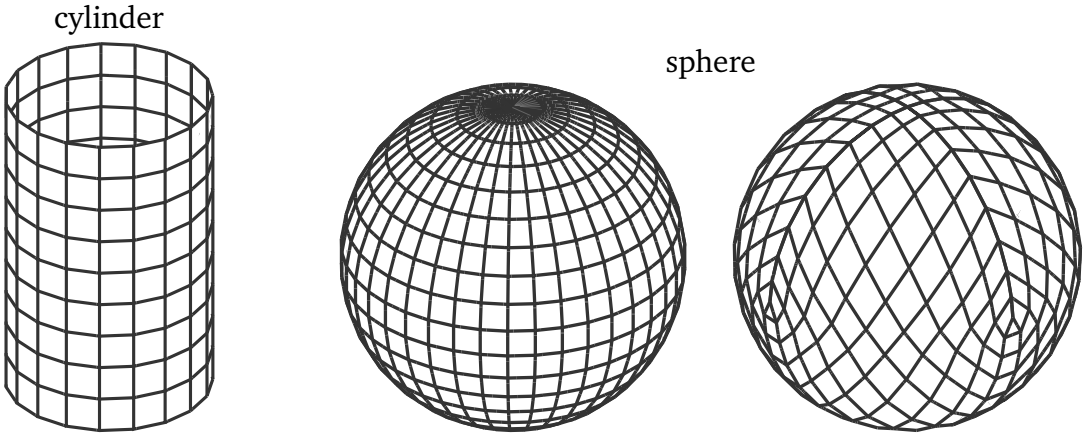


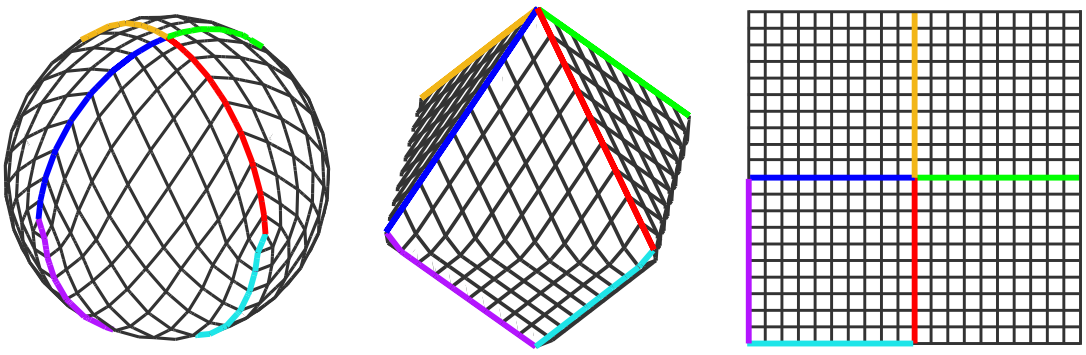
Figure IV.4: Proxy grid parameterization for cylinders and spheres. Cylinders are parameterized with the cylindrical coordinates (left). As parameterizing a sphere with spherical coordinates creates heavy distortion at the poles (middle), we choose to use the octahedral parameterization (right), which reduces stretch between grid cells.

Revolution Shapes

For the the shapes of revolution such as cylinders and spheres, we want to maintain a unified representation allowing the use of the same 2D operators as for the plane. In that regard, we define a parameterization for both of these shapes, also based on their local axes. Figure IV.4 shows the parameterization of the revolution shapes.

The cylinder has its $\vec{X}$ and $\vec{Y}$ local axes orthogonal to its direction axis $\vec{A}$, allowing the use of *cylindrical coordinates*. Hence, the local axes define the *angular position* around the cylinder, while the axis $\vec{A}$ defines the *height* of a point along the cylinder. On a cylinder of origin point C and radius r, a 3D point P would have its local shape coordinates defined as

$$\begin{cases} u = r(\pi + \text{arctan2}(\vec{pc}.\vec{Y}, \vec{pc}.\vec{X})) \in [0, 2\pi r] \\ v = \vec{pc}.\vec{A} \end{cases} \quad \text{(IV.2)}$$
$$\text{with } \vec{pc} = P - C \ .$$



octahedral sphere octahedron flattened octahedron

Figure IV.5: Sphere octahedral parameterization. This parameterization defined by Praun and Hoppe [PH03] allows reducing stretch at the surface of the sphere. The 2D grid on the right fully contains the extent of the shape.

The straightforward solution to parameterize a sphere would be to use spherical coordinates. Local axes $\vec{X}$ and $\vec{Y}$ of the sphere are defined in that sense, where they allow computing the *azimuthal* angle. In that parameterization, the *polar* angle is computed from the *zenith* direction defined as $\vec{X} \times \vec{Y}$. However, as we can see on the middle of Figure IV.4, the use of spherical coordinates implies strong distortion at the poles.

As we want to discretize the shape extent to locally represent the surface with statistics, it would be more meaningful to have a grid with similar cell size. In consequence, we choose to parameterize a sphere with a *octahedron*, as defined by Praun and Hoppe [PH03], which strongly reduces stretch. Figure IV.5 shows the octahedron-based spherical parameterization that we use.

On a sphere of center point C and radius r parameterized as a octahedron, a 3D point P would have its local shape coordinates defined as

$$
\begin{cases} x = \vec{pc}.\vec{X} \\ y = \vec{pc}.\vec{Y} \\ z = \vec{pc}.(\vec{X} \times \vec{Y}) \end{cases}
\text{with } \vec{pc} = \frac{P-C}{\|P-C\|}
$$

$$
\begin{cases}
u = \frac{\pi r}{2} \begin{cases} \frac{x}{n} & \text{if } z \geq 0 \\ 1 - \frac{|y|}{n} & \text{if } z < 0 \wedge x \geq 0 \\ \frac{|y|}{n} - 1 & \text{if } z < 0 \wedge x < 0 \end{cases} \\
v = \frac{\pi r}{2} \begin{cases} \frac{y}{n} & \text{if } z \geq 0 \\ 1 - \frac{|x|}{n} & \text{if } z < 0 \wedge y \geq 0 \\ \frac{|x|}{n} - 1 & \text{if } z < 0 \wedge y < 0 \end{cases}
\end{cases}
\text{with } n = |x| + |y| + |z| \ . \tag{IV.3}
$$

Shape Extent Discretization

From the 2D local coordinates of each shape as described above, we define a fixed-size grid on top of the shape surface. We simply discretize the coordinate values of a given point at the surface of shape using a fixed cell size. We set the size to 5cm x 5cm, which corresponds to about four times the area of a depth pixel at a typical distance of 8 meters from the sensor. The area of a pixel at given depth z is given by $a(z) = \tan(\frac{fov_H}{res_H})\tan(\frac{fov_V}{res_V})z^2$. With $fov = (60°,45°)$ and $res = (320, 240)$, we have $a(8m) \approx 0.00068539m^2 \approx (2.6cm)^2$. Hence, this size ensures a minimum sampling of proxy cells by depth points even at far capture distances. In practice, the real case capture distance will not go beyond 5 to 6 meters due to the size of indoor rooms and limitations of the capture device. In consequence, the potential amount of visiting depth point per frame for a cell is guaranteed to be more than four.

Cells are activated when their visitation percentage over the recent frames (the last 100 frames in our experiments) is greater than a threshold (25% in practice). Once activated, a cell stays so until the end of the processing. We consider a cell as visited as soon as it admits at least one inlier data point i.e., a data point located within a threshold distance to the cell under a projection in the direction from the sensor origin to the point. This activation threshold allows modeling the actual geometry of the observed scene, while discarding outliers observations due to the low quality of the sensor.

IV.3 Accumulation Structure

IV.3.1 Accumulation of Depth Samples

As shown in section IV.2, we track 3D shapes in time and space to maintain a consistent geometric representation of the surroundings. This temporal and spatial consistency gives information on local geometry at its fixed location in the real world. Hence, we leverage the 3D sampling of this geometry as provided by the depth sensors, to refine our model with time. At each new frame, shape tracking provides a list of 3D sample points belonging to a shape. These 3D points bear geometrical and appearance information such as distance to the shape, orientation deviation, curvature and color. The definition of our local extent model at the shape surface, as explained in section IV.2.2, allows accumulating this information into a unified geometric representation.

IV.3.2 Shape-wise Statistics

The first use of these accumulated depth samples at the shape surface is to compute global shape statistics. At each frame, the new set of inliers is used to compute shape parameters which are averaged with the previous parameters to get more consistent shapes. In addition, the standard deviation of shape parameters gives insight on their distribution and can be seen as a confidence value on the shape.

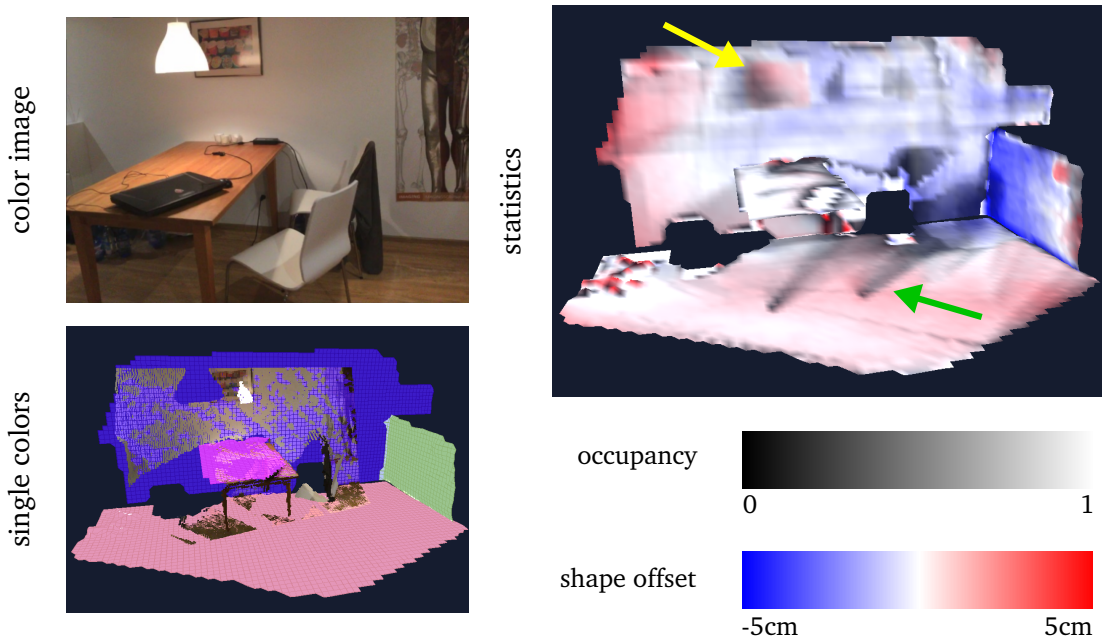


Figure IV.6: Real data example of proxy statistics. Top left: Color image provided for information in order to understand the structure of the scene. Bottom left: Four planar proxies are represented in 3D with a single color per shape, on top of the current 3D point cloud. Right: Proxy cells are represented with a color code showing the current values of their statistics. Intensity represents occupancy probability, and hue – from blue (-5cm) to red (5cm) through white (0cm) – represents the distance from the shape. We can notice that cells behind table legs have a lower occupancy with darker shade, as they were not observed very often (green arrow). The geometric details due to the frame on the wall above the table are correctly recovered with a positive shape offset (yellow arrow).

IV.3.3 Local Statistics

The consistent grid defined at the surface of a shape based on a shape-specific parameterization was designed to gather statistics on local parts of the shape, in order to model small details. In each cell of the grid, a probability of *occupancy* is computed to get knowledge of the visitation rate of this cell. This allows understanding whether a cell models actual geometry in the scene, or if it just corresponds to noise or flickering parts and should be ignored. Figure IV.6 shows an example of statistics of occupancy and distance stored at the surface of shapes.

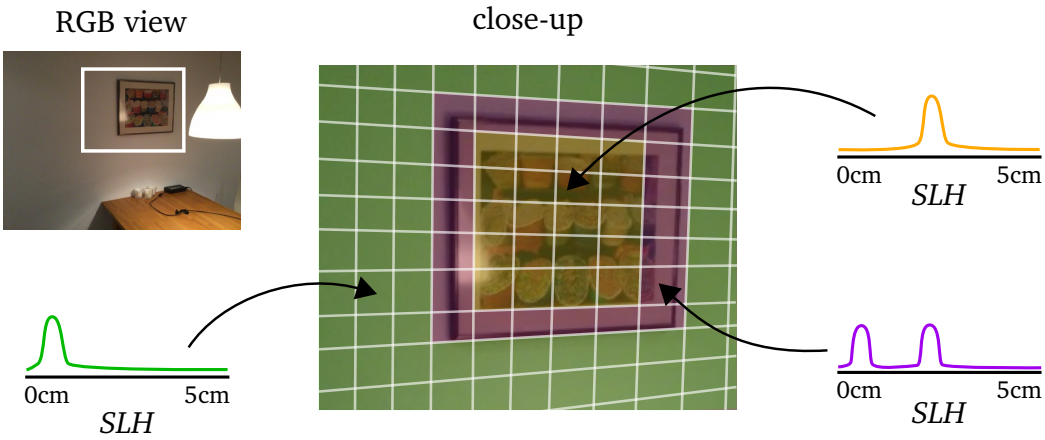


Figure IV.7: Close-up example of local geometry stored in the *smoothed local histogram (SLH)* of each proxy cell. Accumulating depth samples into the histogram naturally reveals distribution modes indicating the local nature of the geometry. Here, cells over the wall and frame have a single mode, as all samples have a similar distance to the plane (light green and orange). Cells at the limit between wall and frame accumulate samples that can have the plane distance of either the wall or frame, hence containing two histogram modes (light purple).

Shape Distance

Each cell of the grid includes a statistical model of the depth values gathered from the RGB-D data. These statistics are composed of a collection of mean μ and variance σ values of the distance to the shape. This local distribution represents a *smoothed local histogram* [KS10] made of Gaussian kernels. Using such a compressed histogram representation allows recording samples into a compact but faithful model made of a short list of normal distribution parameters, as well as smoothing out outlier depth values. The contribution of an inlier p of distance $d(p)$ to the proxy is given as

$$h_p(s) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(s-d(p))^2}{2\sigma^2}} \qquad h'_p(s) = -\frac{s-d(p)}{\sigma^2} h_p(s)\ . \tag{IV.4}$$

This compressed model stores the repartition of shape inliers distances to the proxy and makes possible estimating the *diversity* of the values within each cell by counting the *number of modes* in the distribution, appearing naturally when building the histogram. If it has a single mode, then all values are similar and the surface of the proxy within the cell is most likely flat. If the distribution has two or more modes, then the values belong to different groups and the cell likely overlaps a salient area of the surface. Figure IV.7 shows a close-up example of flat and salient proxy areas with representation of the associated histogram and distribution modes.

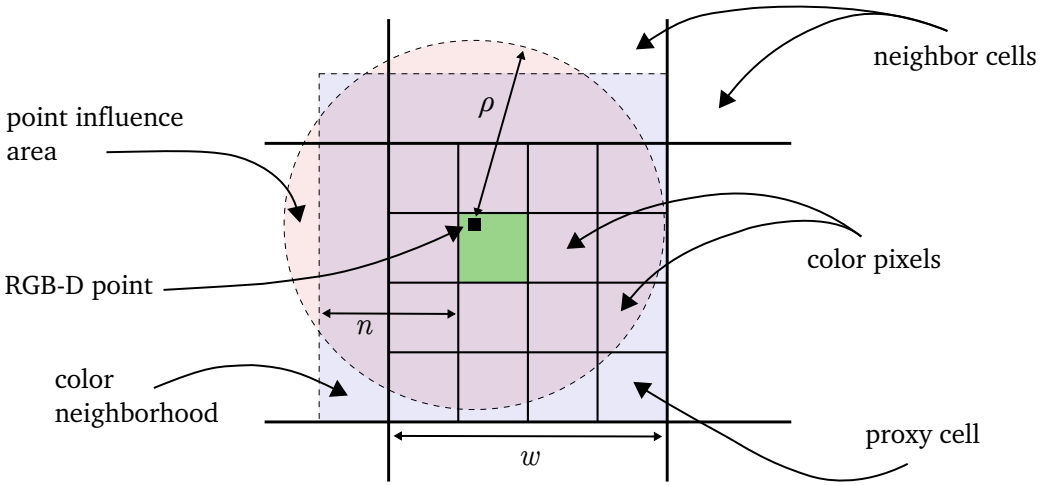


Figure IV.8: Updating color points of a proxy cell from the color component of an RGB-D point sample. An RGB-D point sample projected on the 2D proxy grid falls within a discretized color point (light green). w is the cell width in meters. Here, 2^r color points are defined with $r = 2$. Radii of influence area ρ (light red) and color point neighborhood n (light blue) are computed from the point depth (see Equation IV.5). All color points within the computed neighborhood n are updated with a weight depending on the distance to the point and the size of the influence area (see Equation IV.6).

Color

In order to accumulate faithful color information within our coarse grid, we define color samples at higher resolution than the cells. To do so, we take inspiration from the *mesh colors* methodology [YKH10] which defines discrete color positions on faces of triangle meshes. In our framework, each cell of grid contains a fixed color sub-resolution grid which is updated by each RGB-D sample point falling into the cell. Figure IV.8 details the update of color in the grid from an RGB-D sample.

For each RGB-D sample falling within the bounds of a given proxy cell, we compute a neighborhood n of color positions which will be updated with this sample's color value. In order to update as much color positions as covered by a sample point, we compute an *influence radius* of the point based on its *distance to the sensor*. The topology of depth sensors implies that the further the depth point, the larger the size of the image pixel when unprojected to 3D. Hence, the influence radius is computed as the size of this depth pixel in 3D and discretized to get the size of the color point neighborhood at the surface of the shape.

For a 3D point sample of depth $z \in \mathbb{R}^+$, with a width of grid cells defined as $w \in \mathbb{R}^+$ and a color point resolution defined by its power of two $r \in \mathbb{N}$, the point influence radius ρ and discrete color neighborhood radius n are given by

$$\begin{aligned} \rho &= \tfrac{z}{2} \tan(\tfrac{fov_H}{res_H}) \in \mathbb{R}^+ \\ n &= \left\lfloor \rho \tfrac{2^r}{w} \right\rfloor \in \mathbb{N} \,. \end{aligned} \tag{IV.5}$$

Then, for each color point within the computed neighborhood n, including points belonging to neighboring cells, the average color is updated with the point's color. In order to take into account the uncertainty of information brought by depth samples far from the camera, we *weight* the color contribution with both the *distance* from the color point to the projected depth point and the color *neighborhood size*, which itself is computed from the point depth, as shown in Equation IV.5. Weighting the color *inversely* to the point depth allows further points to give a less important contribution to the color model. For color point neighbor $(u, v) \in [\![-n, n]\!] \times [\![-n, n]\!]$, the color contribution weight of a given depth point is defined in Equation IV.6. In practice, we empirically define the standard deviation factor $\alpha = 3$, which gives a good balance between smoothness and sharpness of the textures.

$$\frac{1}{1 + 2n^2} e^{-\frac{\|(u,v)\|^2}{2\sigma^2}} \text{ with } \sigma = \alpha 2^r, \alpha \in \mathbb{R}^+ \tag{IV.6}$$

IV.4 Experiments

IV.4.1 Implementation

Our proxies are implemented through hardware and software components. The hardware setup is made of a computer with Intel Core i7 at 3.5GHz and 10GB memory. No GPU is used. The software setup has a client-server architecture, where the server runs within an embedded environment with low computational power and limited memory to trigger the sensor and process the data. The client's *graphical user interface* allows controlling the processing parameters and getting a real time feedback of the stream. A limited range of intuitive parameters allows the user to control the trade-off between quality of the output and performance of the processing. More details on the graphical interface are given in section A.1.2. In order to achieve a better quality and efficiency when building proxies, a few minor optimizations have been implemented, as shown below.

Parallelization

Parallelization through *OpenMP* is used to process shapes faster. When tracking shapes, as more and more previous shapes have to be checked in new frames, their score in the new frame is computed in parallel. The update of proxies with the new shape observation is also conducted in parallel for all proxies. However, computations such as proxy clipping or cell update within a proxy can not be parallelized because of concurrent access and inter-dependencies. Optimization along these axes would require heavy refactoring of the current implementation.

Sensor-dependent Tuning

The axial and lateral noise introduced by the consumer depth cameras lead to artificial and erroneous data samples in the input. Based on the noise model from Nguyen et al. [NIL12], we estimate a noise threshold at a given distance to the camera, under which differences in geometry will not be considered as actual differences in the observed surfaces. Hence, in order to prevent wrong geometry from being modeled within the proxies, we modulate the distance thresholds throughout the processing by the noise estimated at the corresponding distance to the camera origin.

Scene Orientation

We aim at building a model which is consistent and intuitive with relation to the structure of the observed scene. We therefore orient the local frames of the proxies along the orientation of the scene, assuming that its structure follows the *Manhattan world* assumption [CY99]. This allows the cells on the walls to be oriented along the directions of the floor and orthogonal walls, and the cells on the floor and ceiling to be oriented along the directions of both walls. In order to detect the structural orientation of walls and floor in the scene, we look for nearly horizontal and vertical planes at the beginning of the processing, and use their orientations as a prior.

IV.4.2 Dataset

We run all of our experiments on the *3DLite* [HDGN17] dataset[1], containing 10 scenes acquired with a *Structure* sensor[2] under the form of RGB-D image sequences. This choice was motivated by the availability of ground truth poses along with the visual data, as well as result meshes and performance metrics provided from processing with both *BundleFusion* [DNZ+17a] and *3DLite* [HDGN17], with which we compare our method in section V.3.1.

[1] 3DLite dataset: http://graphics.stanford.edu/projects/3dlite/#data
[2] Structure sensor: http://structure.io

Scene	# proxies	# cells	Avg # proxies / fr.	Avg # cells / fr.
apt	51	62K	3.8	3139
offices	86	123K	3.7	3958
office0	50	44K	4.0	2491
office1	56	53K	4.5	3108
office3	56	50K	3.9	2332
scene0220_02	41	42K	4.7	3369
scene0271_01	32	33K	4.4	3340
scene0294_02	31	39K	4.5	4441
scene0451_05	48	42K	4.7	3984
scene0567_01	42	48K	4.4	3354

Table IV.1: Statistics on the geometric proxies. Total number of proxies and cells for each scene of the *3DLite* [HDGN17] dataset. The average number of observed proxies and cells per frame (/ fr.) is also given.

IV.4.3 Building Proxies

Geometric statistics on the generated proxies are available in Table IV.1 for all processed scenes. We also provide in Figure IV.9, a plot of the increment of the average depth over time, to show the fast convergence of the proxy statistics after about 30 accumulated samples. The accuracy of the proxy representation can be quantitatively assessed through the PSNR values in Table V.1.

The time required to build and update geometric proxies using our current implementation is around 150 ms for an input depth image of 320x240 pixels. A detailed graph presenting the repartition of the processing time for all steps is available in Figure IV.10.

IV.4.4 Validation on Synthetic Data

We define a synthetic data processing framework, in order to validate RGB-D data modeling with our geometric proxies while testing all kinds of shapes, even when not commonly seen in human made environments. We use the *Blender* tool[3] to generate a synthetic RGB-D image set from known 3D objects[4] of planar, cylindrical and spherical shapes, along with ground truth camera poses. Figure IV.11 shows the proxies generated from this artificial dataset, along with ground truth renderings at the same positions.

[3]Blender is an open source 3D modeler and renderer: https://www.blender.org/
[4]3D models taken from https://sketchfab.com/adrksr/collections/simple-shapes

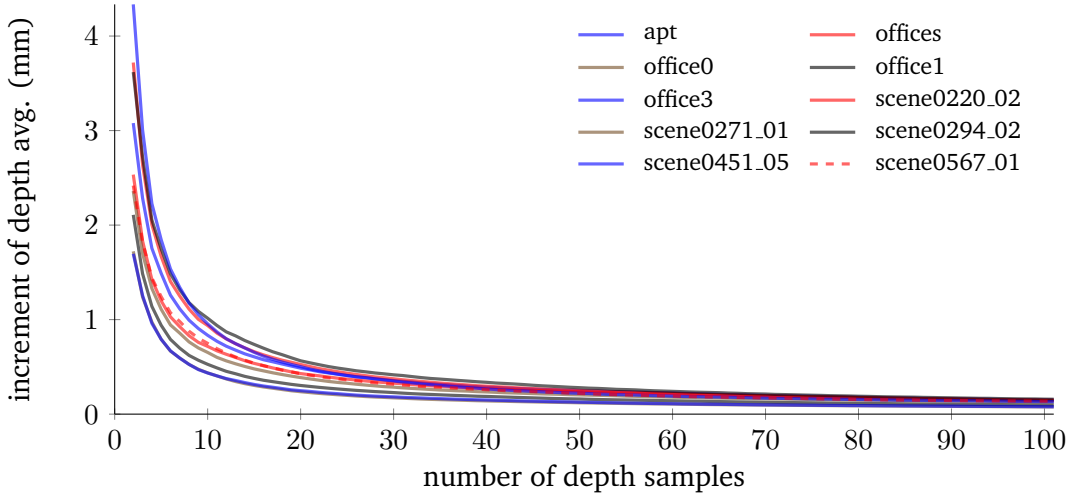


Figure IV.9: Increment of the depth average in proxy cells over time. Absolute value of the increment of the depth average with relation to the number of depth samples accumulated within the proxy cell. At each frame, we compute the absolute difference between the previous and current average values of distance to shape and average it over all cells in the scene. We observe the fast convergence of the depth average after about 30 accumulated samples, where the change in distance to shape value falls below 0.5mm. This shows that only a few seconds are needed for the statistics in the proxy to become stable.

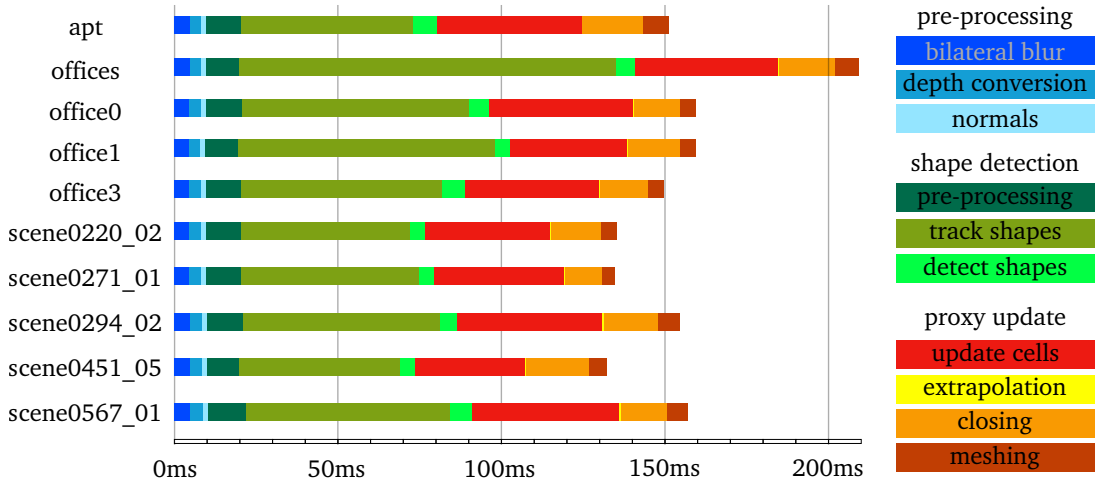


Figure IV.10: Timing repartition for geometric proxy generation, averaged over all frames of each scene. Proxy update operations *extrapolation* and *closing* are described in section V.2.2 and *meshing* in section V.3.1. Shape tracking is the longest processing with about 40% of the total time, as it iterates over all previous proxies and depth image samples. Proxy cell update is the second longest with about 30% due to the iterations over cells and color points for each cell. We can see that the tracking takes more time for the largest scene *offices* (8518 frames), as its size implies more previous shapes to iterate over.

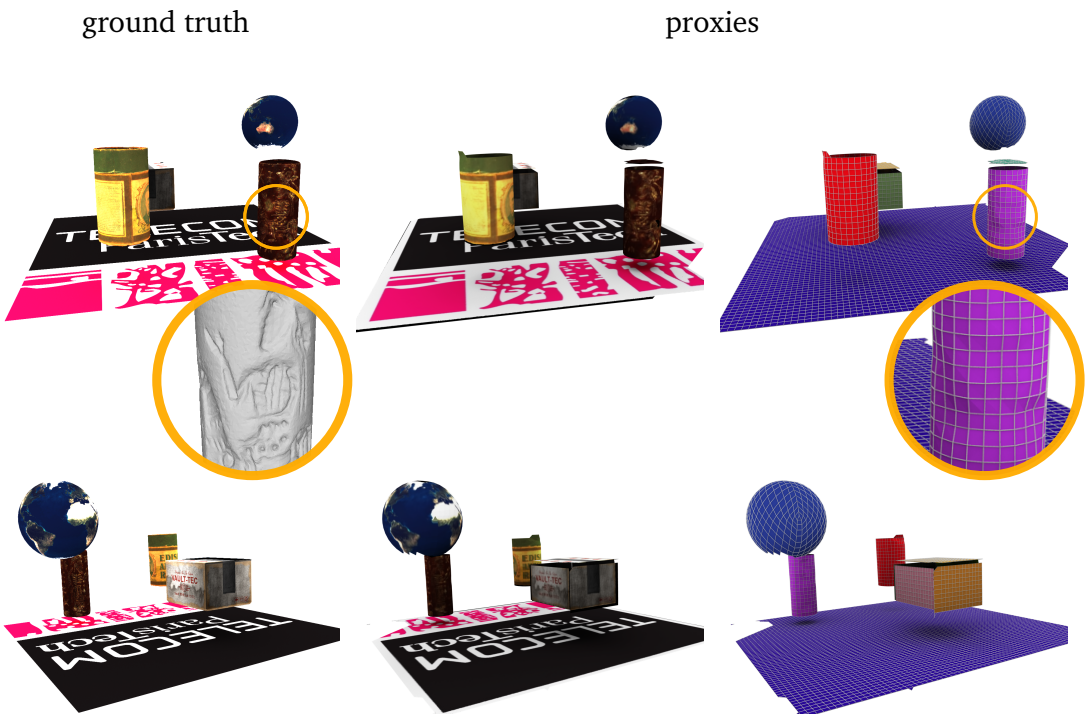


Figure IV.11: Simple shapes modeling from synthetic data, shown at two camera locations. From a set of rendered RGB-D images (left), proxies are built for planar, cylindrical and spherical synthetic elements. Both geometry and texture are recovered at lower resolution than the original while keeping a good visual quality (middle). For these simple shapes, a coarse proxy grid resolution allows recovering most of the geometry (right). The yellow circle close-ups show how proxies record local geometric information. Although at lower resolution, we can notice slight irregularities on some proxy cells at the surface of the purple cylinder, corresponding to fine geometric details present in the original 3D model.

In particular, processing data generated from synthetic models allows comparing texture information between proxy-generated and ground truth images from 3D models. Figure IV.12 shows qualitative comparison of texture images used to generate the synthetic data and computed with the color point model of our proxies. As we can see for the revolution surfaces, our proxies allow re-parameterization of the texture information from *bin packing* for the cylinder or *spherical* for the sphere, into more meaningful and less distorted *cylindrical* or *octahedral* solutions.

Figure IV.12: Simple shape proxies modeling textures. Left: Texture images from 3D models used to generate the RGB-D sequence in *Blender*. Right: Our local color point model embedded within the 2D grid of the proxies allows recovering textures of 3D objects. Black areas on the proxies textures were not observed by the camera e.g., the poles of the sphere at the center and corners of the octahedron square (bottom right). As expected, the parameterization of our proxies leads to more consistent and less distorted textures as we can see between e.g., the octahedral and spherical sphere parameterizations (bottom). In addition, our proxies converted the texture of the cylinder from an automatic bin packing to a more meaningful cylindrical parameterized representation (middle).

IV.5 Discussion and Perspectives

This chapter introduced a new geometric superstructure for RGB-D streams, based on spatially and temporally consistent shape instances. By tracking geometric shapes over time and space, we define a shape-wise accumulation structure to record information brought by RGB-D samples. The use of surface shapes instead of a volumetric structure lightens the modeling while keeping a faithful representation of most elements seen in human-made environments. The statistics stored in the proxies maintain knowledge of the local geometry and appearance of the observed scene to keep as much detail as possible with the lowest possible hardware cost. In the following, we list paths to improvement towards better stability and performance.

The frame-wise on the fly construction and update of the superstructure enables continuous understanding of the surroundings. In addition, the statistics of multiple nature stored within the proxy grid can be thought of as a consistent support to apply operations to the incoming data. In that sense, chapter V presents both local and global tools for improvement and consolidation of the data on the fly, based on the described geometric superstructure.

IV.5.1 Towards More Stable Proxies

While our proxies are continuously made more accurate by averaging observations in RGB-D frames, they are highly sensitive to the quality of the spatial consistency. We enforce spatial and temporal consistency based on the computation of the camera motion at each frame, which itself is a hard problem. As the proxies are agnostic to the camera motion computation method, tracking is sometimes lost and new proxies are generated for the same physical element. Introducing the shape proxies into the camera motion estimation using e.g., their parameters or geometric statistics, could allow more robustness and stability of the shape tracking.

IV.5.2 Performance Improvements

While we currently use *OpenMP* to improve performance of shape processing, we could develop a finer parallel implementation and leverage mobile GPUs, e.g. *ARM Mali*[5], to achieve a higher processing rate on embedded platforms. In addition, our current proxy model stores statistics on a uniform (yet sparse) grid, which could be improved using a sparse adaptive structure such as *random access trees* [LH07].

[5] https://en.wikipedia.org/wiki/Mali_(GPU)

V

INDOOR SCENE ANALYSIS

As shown in chapter IV, we maintain a multi-shape geometric superstructure modeling structural elements of an observed indoor scene and storing rich statistics to accurately represent it. As such, it can be seen as a consistent model of the surroundings and a support to apply operations and processing components of 3D scene analysis. In particular, section V.2 explores applications of the superstructure to solve issues of noise and instability in RGB-D data, missing information or heavy weight in the context of transmission and visualization. In section V.3, we show how our stable superstructure and its statistics can be used as an accumulation structure to consolidate information while navigating an indoor scene.

V.1 Context

V.1.1 Motivation

We define a geometric framework using the time-evolving statistics stored in our proxies and based on their consistent spatial support that can be then seen as a geometric ”scaffold”. Its purpose is to improve the RGB-D stream on the fly by reinforcing features, removing noise and outliers or filling missing parts, under the memory-limited and real time embedded constraints of mobile capture in indoor environments (Figure V.1).

As shown in the previous chapter, we designed such a lightweight geometric superstructure to be stable through time and space, which gives priors to apply several signal-inspired processing primitives to the RGB-D frames. They include filtering to remove noise and temporal flickering, hole filling or resampling (section V.2).

This allows structuring the data and can simplify or lighten subsequent operations, e.g. tracking and mapping, measurements, data transmission, rendering or physical simulations. While our primary goal is the enhancement of the RGB-D data stream, our framework can additionally be applied to compression and scene reconstruction (section V.3), as the generated structure is a representation of the observed scene.

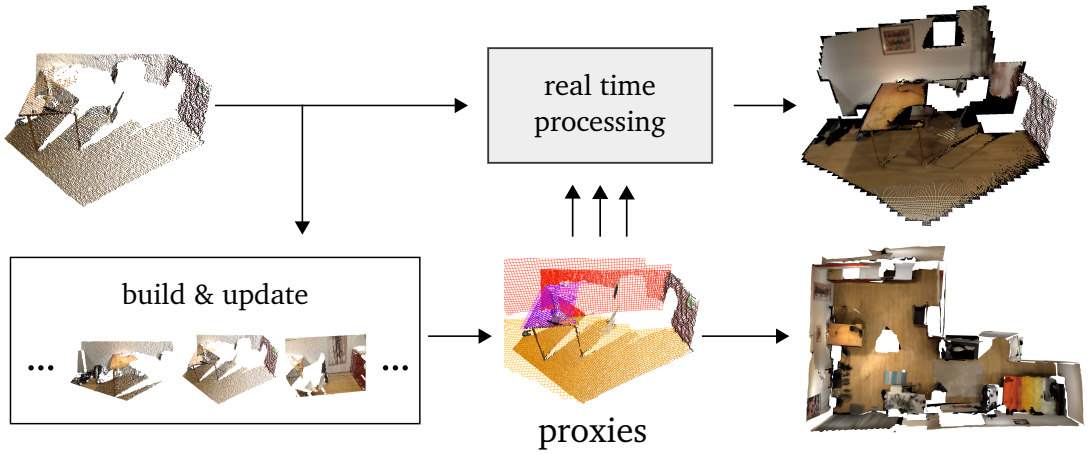


Figure V.1: *Overview* of our geometric scene analysis framework. From a stream of RGB-D frames (left), proxies are built on the fly and updated over time (bottom) and used to apply different real-time processing primitives to the incoming RGB-D frames (top). The system outputs an enhanced data stream and a geometric model of the observed scene (right). The ”build & update” procedure is detailed in Figure IV.2.

V.1.2 Overview

We leverage the stable and consistent nature of our geometric proxies in two different applications to process incoming RGB-D frames into a more regular, complete and stable data set, namely:

- *local* processing for frame-wise improvement and compression (section V.2);
- *global* consolidation and reconstruction into a complete 3D representation (section V.3).

In practice, our system takes a raw RGB-D stream as input to build and update a set of geometric proxies on the fly, as shown in chapter IV. It outputs an enhanced stream together with a model of regularly-shaped (e.g., planar) areas in the observed scene (Figure V.2).

On the contrary to previous approaches, which mostly rely on a full volumetric reconstruction to consolidate data as shown in section II.3.2, our approach is lightweight, with a moderate memory footprint and a transparent interfacing to any higher level RGB-D pipeline.

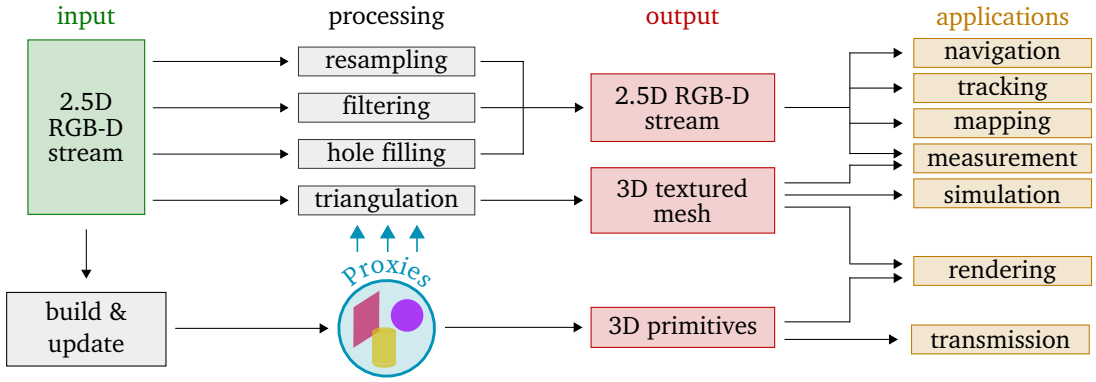


Figure V.2: Geometric scene analysis *workflow*. From a stream of 2.5D RGB-D frames, proxies are built on the fly and updated through time (section IV.2). They are used as priors to process incoming frames through filtering, resampling or hole filling, which can lead to better tracking, mapping, automated navigation or measurement. A selection of proxies based on the current RGB-D frame can be used for lightened transmission of the data. Proxies can be used as priors for triangulation and fast depth data meshing, with applications to rendering or simulation. Eventually, the consolidation of the depth stream within the proxies leads to a reconstruction of regularly-shaped areas in the observed scene. Processing modules are described in section V.2 and the consolidation of data is presented in section V.3.

V.2 Live Frame Processing

Our first use of the consistent geometric proxy structure is as a support for filtering operators with the goal of:

- *removing noise* to smooth data while keeping small details and local saliency;
- *filling holes* due to specular elements or unseen parts;
- *resampling* the incoming point cloud at any desired resolution at the surface of shape proxies;
- using the proxies as a *compressed version* of the input for transmission or visualization.

V.2.1 Filtering

While projecting the sensor's data points onto their associated proxy would allow removing the acquisition noise and quantization errors due to the sensor, this would lead to the *flattening* of all shape inliers. In order to minimize the loss of details on the shape surfaces while keeping a lightweight data structure, we instead use the geometric proxies as a simple *collaborative filter* model.

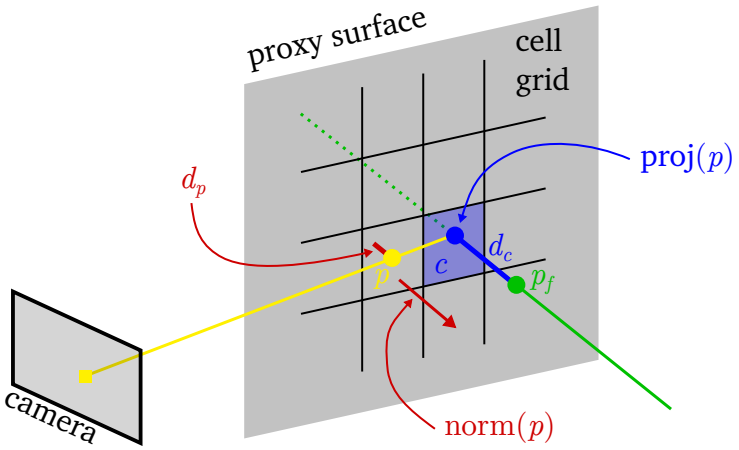


Figure V.3: Filtering 3D points using statistics from the proxy. Example of our custom filtering on a planar proxy inlier point p (yellow circle). Here, 3D point p has normal $\text{norm}(p)$ at the proxy surface (red arrow) with orthogonal distance d_p (red line), used to update the average d_c. Point p is projected along the camera ray (yellow line) into $\text{proj}(p)$ (blue circle), which allows retrieving its cell c (light blue). In this specific example, we assume that c is a cell whose *smoothed local histogram* has one mode $m_c = 1$, hence we apply the shape offset d_c (blue line) along the surface normal direction (green line) to get the filtered 3D point p_f (green circle).

To that end, we designed a custom filter to leverage the *smoothed local histograms* stored in each cell of the proxies. As explained in section IV.3.3, the number of detected modes allows *distinguishing* flat areas of the proxy surface from salient ones. For flat cells whose distribution has a single mode, we project the depth points on the shape along the direction between the camera and the point. We offset the points of the average distance to the shape only if it is above the noise threshold at the corresponding distance to the camera (see details on the noise threshold in section IV.4.1). This allows *smoothing* surface areas that are exactly on the shape while *keeping* flat areas offset from the shape as they are in the scene. For cells whose distribution has two or more modes, we do not perform any projection in order to keep the saliency and details of the surface.

Equation V.1 details the *smoothed local histograms*-based filtering of inlier p to p_f, belonging to cell c with m_c modes and an average distance to the proxy of d_c, and a noise threshold of α.

$$p_f = \begin{cases} \text{proj}(p) & \text{if } m_c = 1 \wedge d_c \leq \alpha \\ \text{proj}(p) + d_c \, \text{norm}(p) & \text{if } m_c = 1 \wedge d_c > \alpha \\ p & \text{if } m_c \neq 1 \end{cases} \tag{V.1}$$

The function proj(p) represents the projection of p on the proxy along the camera direction, which can also be seen as the intersection between the proxy shape and the camera ray passing through the pixel that generated point p. The function norm(p) represents the normal vector on the shape surface at the projected point location. Figure V.3 illustrates the filtering process on a planar example where, for a plane of normal $\vec{N}$ and distance to origin l, proj$(p) = \frac{l}{p.\vec{N}}p$ and norm$(p) = \vec{N}$.

In addition, the proxy can also be used as a high level range space for *cross bilateral filtering* [KCLU07], where inliers of different proxies will not be processed together.

Temporal Flickering Removal

Based on time-evolving data points, the proxies consolidate the stable geometry of the scene by accumulating observations in multiple frames. Averaging those observations over time removes temporal flickering, after a few frames only. In particular, Figure IV.9 shows that the changes in statistics of distance to the shape fall below 0.5mm after about 30 accumulated samples.

V.2.2 Hole Filling

Missing data in depth is often due to specular and transparent surfaces such as glass or screens. With our framework, observed data is *reinforced* over multiple frames from the support of stable proxies, *augmenting* the current frame with samples from previous ones. In practice, the depth data that is often seen in incoming frames creates activated cells with sufficient occupancy probability to survive within the model even when samples for these cells are missing.

This hole filling, stemming naturally from the proxy structure, is completed by two additional steps. First, the extent of the proxies is *extrapolated* to the intersection of adjacent proxies – this is particularly useful to complete unseen areas under furniture for example. Second, we perform a *morphological closing* [Ser83] on the grid of cells, with a square structural element having a fixed side of seven cells. This corresponds to closing holes of maximum 35cm by 35cm, which allows filling missing data due to small specular surfaces, e.g. computer screens or glass-door cabinets, while keeping larger openings such as windows or doors.

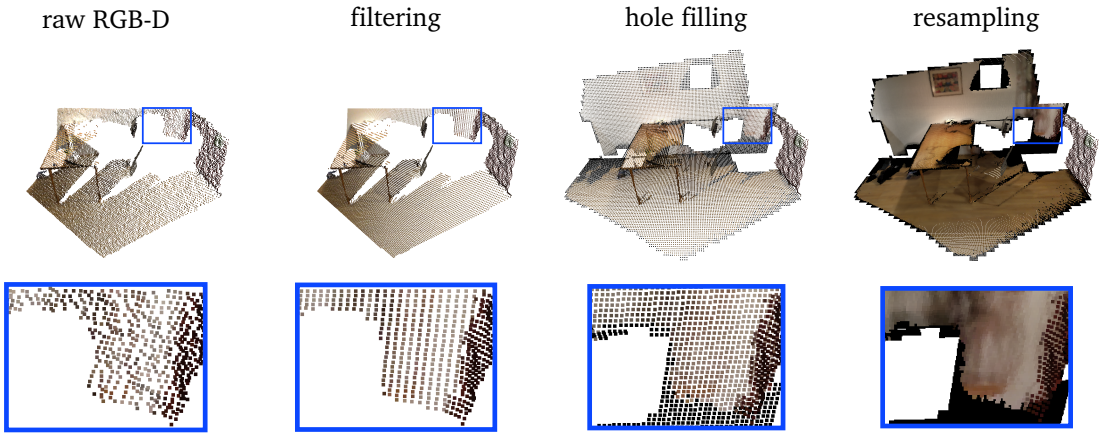


Figure V.4: Data improvement using geometric proxies. Raw RGB-D and proxy-improved data showing results of real time filtering, hole filling and resampling. Hole filled and resampled point clouds are generated by sampling the proxy surface with respectively 2x2 and 4x4 points per cell. Blue surrounded areas highlight a region where improvement using the proxies is significant compared to the low quality input RGB-D frames.

V.2.3 Resampling

Our proxies can be used to *super-sample* RGB-D streams on the fly. The low definition geometric component of raw frames can be enriched by the higher resolution information structured in the proxies. Our 3D structure can *guide the process* using both its color and geometric components, whose smoothness and stability *appear naturally* with the accumulation of samples. In addition, the local nature of statistics stored in the proxy cells allows generating high resolution data while keeping geometrical details and salient areas and *avoid over-smoothing*. The sub-resolution color component of the proxies can be used to assist the sampling process to recover even more detail. This results in high definition RGB-D data with *controllable* point density on the surface of the shapes.

V.2.4 Experiments

Figure V.4 shows examples of data improvement using the processing modules of our framework. Experiments show that the proxies are particularly efficient to remove noise over walls and floors while keeping salient parts, and helps reducing holes due to unseen areas, specular areas such as lights or glass, and low confidence areas such as distant points. Resampling the point cloud allows recovering structure if the sensor did not give enough data samples, e.g. on lateral surfaces. Timings for hole filling operations are given in Figure IV.10.

Scene	Proxies			JPEG export		H.264 [NSS14]	
	frame r.	scene r.	PSNR	ratio	PSNR	scene r.	PSNR
apt	5.40	**790.5**	**38.0dB**	8.09	33.8dB	11.1	25.0dB
offices	4.29	**1173.7**	**38.0dB**	9.22	33.8dB	10.5	22.1dB
office0	6.78	**2376.6**	21.4dB	6.60	20.6dB	14.4	**30.3dB**
office1	5.44	**1828.0**	**41.2dB**	6.70	34.7dB	10.5	26.9dB
office3	7.24	**1286.2**	**33.8dB**	8.58	31.0dB	12.3	29.0dB
scene0220_02	5.02	**814.7**	**36.7dB**	6.29	31.9dB	11.1	18.3dB
scene0271_01	5.07	**977.8**	**43.8dB**	5.82	36.7dB	10.5	18.8dB
scene0294_02	3.82	**1024.3**	**43.4dB**	7.27	36.8dB	9.4	17.2dB
scene0451_05	4.25	**694.9**	**42.1dB**	5.44	34.7dB	7.9	15.5dB
scene0567_01	5.05	**729.8**	**40.2dB**	9.00	35.3dB	15.6	20.0dB

Table V.1: Proxy compression metrics for all processed scenes. Compression ratios (frame-wise and scene global) are based on the raw size of a 320x240 depth map and the size of the proxies without the outliers. The JPEG export ratio is between the sizes of the raw and exported proxies. The *peak signal-to-noise ratio (PSNR)* is computed using the average *root mean square error (RMSE)* between raw depth points and *proxy*-filtered ones. We compare our compression performance to a state-of-the-art method based on H.264 [NSS14] with a quality profile of 50.

V.2.5 Compression

Compression of the input data is achieved by using directly the proxies as a compressed, lightweight geometric substitute to the huge amount of depth data carried in the stream, avoiding storing uncertain and highly noisy depth regions, while still being able to upsample back to high resolution depth using a bilateral upsampling. In particular, this is convenient to broadcast captures of indoor scenes where planar regions are frequent.

Substituting the geometric proxies to the RGB-D stream provides a simple yet effective lossy compression scheme for transmission, with the practical side effect of removing many outliers. Our efficient data structure leads to good compression ratios while keeping high *peak signal-to-noise ratio (PSNR)* and being fast for compression and decompression. Table V.1 and Table V.2 show evaluation metrics of the compression using proxies.

The proxies are stored as simple grids of statistics with a local frame and bounding rectangle. As such, the compressed structure itself, i.e. the proxies, can benefit from image-based compression schemes such as *JPEG* [Wal92] for offline export and storage, for which we report compression ratios and PSNR values in Table V.1. The *JPEG* export and load procedure for all proxies of a scene takes an average of about 40 ms.

Scene	Proxies (ms)		H.264 [NSS14] (sec)	
	comp.	decomp.	comp.	decomp.
apt	150	32	267	232
offices	213	30	828	926
office0	161	19	692	902
office1	160	27	693	692
office3	150	32	424	444
scene0220_02	134	41	165	175
scene0271_01	133	39	165	162
scene0294_02	152	39	238	200
scene0451_05	133	41	190	138
scene0567_01	154	28	170	151

Table V.2: Proxy compression timings for all processed scenes. The compression time is the building of proxies averaged over all frames, while the decompression time is the generation of a depth map by applying visibility tests to the proxies. The compression and decompression times for the method based on H.264 [NSS14] are given in seconds for the whole frame set.

In addition to the bandwidth saving, the compressed proxy representation enables smooth *super-sampling* of the geometric data, where the output point cloud density over proxy surfaces *can be increased as desired*. The geometric surface parameterization of each proxy offers a suitable domain for point upsampling operators, while a similar approach performed directly on the RGB-D stream is blind to the scene structure.

V.3 RGB-D Stream Consolidation

V.3.1 Regular Shape Reconstruction

While being lightweight and fast to compute, the proxies represent a superstructure modeling the dominant regular geometric elements of an indoor scene. In addition to being used to filter the input point cloud and generate an enhanced RGB-D stream as output, proxies themselves are a way to *consolidate* the input RGB-D frames. Hence, meshing the proxy cells leads to a lightened organized structure and aggregating all proxies in the global space allows reconstructing a higher quality surface model of the observed scene, generated on the fly. The time required to mesh the full proxy set is below 10ms, as shown in Figure IV.10.

In this section, we compare the performance and quality of scene reconstruction using our proxies to state-of-the-art methods *BundleFusion* [DNZ+17a] and *3DLite* [HDGN17]. As in the previous chapter, we run all of our experiments on the *3DLite* dataset composed of 10 RGB-D scenes captured with a *Structure* sensor (section IV.4.2).

Mesh Closing

In practice, our meshing process iterates over all active cells of a given proxy to connect adjacent cells. However, revolution surfaces such as cylinders and sphere have a periodic nature and cells at opposite limits of the parameterized domain are actually adjacent in the Euclidean domain. Hence, to avoid discontinuity of the proxy surface mesh, we designed a closing methodology through a range check where cells at extremities of the 2D domain are connected to cells at the other extremities. In particular, we can see these connections in the lower part of the octahedral sphere in Figure IV.4.

V.3.2 Experiments

Qualitative Results

Figure V.5 presents the reconstructed geometric models based on the corresponding proxies. As we can see, most large planar surfaces such as walls and floors are modeled with a *single* proxy instance. At scene scale, the relatively low color resolution of our proxy is sufficient to identify most elements of the scene. Figure V.6 and Figure V.7 compare reconstructions with Proxies, *BundleFusion* and *3DLite* from global and local point of views. Qualitatively, proxy meshes are similar to *BundleFusion* while offering the structural regularity of *3DLite* models.

Quantitative Results

Table V.3 and Table V.4 present performance and quality metrics for the 10 scenes of the dataset. The accuracy of the reconstruction using proxies can be quantitatively assessed and compared to *3DLite* through the values of *RMSE* with *BundleFusion* as a reference. These metrics show that the lightweight and simple structure of the proxies leads to better performance both in timing and memory consumption, while keeping a quality comparable to that of state-of-the art methods.

With their low runtime and memory needs, our proxies offer a lighter alternative to most recent reconstruction methods characterized by *volumetric* or *deep learning* approaches, which have high requirements in computation costs and memory consumption. The generic format and implementation of the proxies avoid the need for tedious platform-specific tuning and make them well suited for embedded operation and *modern mobile applications*. In addition to the fact that our proxies are built and updated on the fly, the processing does *not* run on a GPU and requires far less memory than modern embedded devices offer.

Figure V.5: Examples of reconstructed scenes of the *3DLite* [HDGN17] dataset using our geometric proxies. On the left, each proxy is shown with a different random color. On the right, textures have been generated from the color points stored in the proxies (see section IV.3.3). The meshes are made of quads when four activated cells are adjacent, and triangles otherwise.

Figure V.6: Qualitative comparison of reconstruction using our Proxies, *BundleFusion* [DNZ+17a] and *3DLite* [HDGN17]. *BundleFusion* meshes (top), while being the most accurate and containing lots of details, are a heavy model and keep some noise and outliers from the incoming data. In addition, their lack of knowledge of the scene structure and global geometry prevents filling missing data in unobserved areas. *3DLite* meshes (middle) have geometric details smoothed out because of the strong planar regularization. However, the textures are sharper and appearance details are of better quality. With a *good balance* between the two, proxy meshes (bottom) are aware of the scene structure and composed of meaningful geometric elements. The *local* nature of the grid and its *accumulated* statistics allow keeping saliency details at the surface of the shapes.

Figure V.7: Close-up views of reconstructions using our Proxies, *BundleFusion* [DNZ+17a] and *3DLite* [HDGN17]. Left: As in the global view, we can see the high detail but also high irregularity of the *BundleFusion* mesh (top). The *3DLite* model (middle) has sharp texture for high appearance quality, although no geometric details. The proxy mesh (bottom) has a lower color resolution but sufficient to identify elements in the scene. Its local information allows keeping geometric details smoothed out by *3DLite*, such as the laptop on the table. Right: The *BundleFusion* mesh (top) is shown with only 30% of the original geometry for better visualization, but we can see that the high details imply a large amount of polygons. *3DLite* is aware of the geometry and models large planar areas with large triangles, refining it at the limits of elements. The regular grid of the proxies is lightweight while storing accurate geometry at all locations of the shapes e.g., for the frame on the wall above the table.

Scene	#Frames	Processing Time			Memory Consumption	
		BF (ms)	3DL (h)	Prox (ms)	BF (GB)	Prox (MB)
apt	2865	–	5.5	150	–	226
offices	8518	–	10.8	213	–	432
office0	6159	**26.4**	3.6	161	21.4	**239**
office1	5730	**27.7**	3.1	160	21.1	**241**
office3	3820	**27.7**	4.5	150	16.9	**232**
scene0220_02	2026	–	2.8	134	–	183
scene0271_01	1904	–	3.0	133	–	178
scene0294_02	2369	–	5.0	152	–	177
scene0451_05	1719	–	5.1	133	–	189
scene0567_01	2066	–	3.1	154	–	207

Table V.3: Quantitative comparison of proxy scene reconstruction processing. The metrics are compared between *BundleFusion* (BF), *3DLite* (3DL) and our proxies (Prox). The processing time to process one frame is averaged over all frames in the scene. The memory consumption is the maximum used memory during processing.

Scene	Geometry Size (vertices/faces)			Model Size (MB)			RMSE (m)	
	BF (M)	3DL (K)	Prox (K)	BF	3DL	Prox	3DL	Prox
apt	1.7/3.3	62/93	62/58	70	**9.8**	10.7	**0.14**	0.15
offices	1.4/2.8	116/174	123/115	58	**19**	21.8	**0.14**	0.18
office0	5.7/11.3	42/63	44/40	238	**6.3**	8.2	0.14	**0.11**
office1	6.0/11.8	46/69	53/48	251	**7.2**	10.3	**0.09**	0.15
office3	6.4/12.6	42/64	50/46	266	**6.2**	9	**0.15**	0.19
scene0220_02	0.3/0.6	55/83	42/39	12	9.1	**7.6**	0.33	**0.22**
scene0271_01	0.2/0.4	39/59	33/33	9	5.8	**5.6**	0.11	**0.10**
scene0294_02	0.3/0.5	39/59	39/40	10	**6.1**	6.6	0.19	**0.14**
scene0451_05	0.3/0.6	60/90	42/38	12	9.2	**8.3**	**0.10**	0.13
scene0567_01	0.3/0.4	29/43	48/45	9	**4.5**	8.3	**0.10**	0.12

Table V.4: Quantitative comparison of proxy scene reconstruction data. We compare our proxies (Prox) to *BundleFusion* (BF) and *3DLite* (3DL). The *root mean square error (RMSE)* is computed using the *Metro* tool [CRS98] between the *BundleFusion* mesh, taken as reference, and the *3DLite* and proxy meshes. We can see that the model generated by the proxies is similar in size and quality to the one generated by *3DLite*, i.e. orders of magnitude lighter than the *BundleFusion* mesh, while requiring much less computing resources.

V.4 Discussion and Perspectives

We introduced a new unified geometric framework for real-time processing of RGB-D streams. It is based on stable proxies modeling the dominant geometric scene structure, introduced in chapter IV. Our method leverages the compact, lightweight and consistent spatio-temporal support of geometric proxies within the processing primitives designed to enhance data or lighten subsequent operations. It runs at interactive rates on mobile platforms and allows fast enhancement and transmission of the captured data. Our structure can be meshed and used as a model of the observed scene, generated *on the fly*. Its implementation makes possible real-time feedback and its control relies on a limited range of parameters. Compared to *BundleFusion* and *3DLite*, reconstruction using our proxies provides a *good balance* between processing time, memory consumption and approximation quality.

V.4.1 Towards a More Faithful Proxy Model

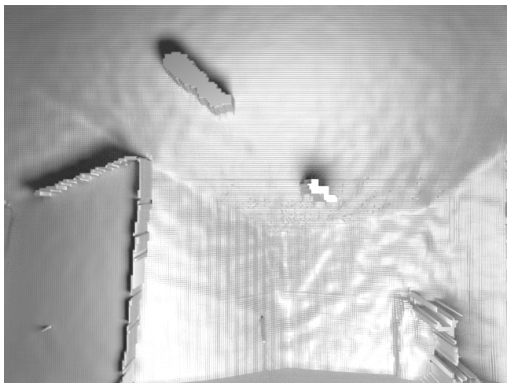
More Geometry

While our proxy grid allows recovering unseen geometry at the surface of the shapes, the heuristic of hole size is not strong enough to make the decision of filling it or not. Missing data at the location of a door or window should not be filled, and the fact that this empty space is observed by the sensor could allow modeling it as ”non-fillable” space. In that sense, we could update non observed cells when crossed by a ray going from the sensor to another proxy's cell to get a negative occupancy probability indicating the absence of physical elements. In addition, we could imagine strengthen this heuristic using the color component to derive semantic information on elements at the locations of holes.

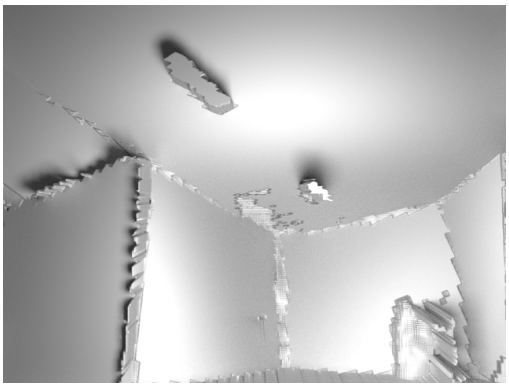
While geometric proxies model most components of indoor scenes, e.g. planar, our current implementation simply ignores non regular shaped elements, such as most small objects. In order to complete this missing information, we could imagine defining a *sparse* voxel grid of the same size as proxy cells. While most sensor data would be modeled as shape proxies, at least in regular indoor scenes, the few remaining depth samples could be used to *locally* update voxels at observed locations. These could store the same geometric and appearance statistics as proxy cells, and their low amount and coarse resolution would only add low overhead. To export them, we could imagine running reconstruction algorithms such as *marching cubes* [LC87] or *dual contouring* [JLSW02] to generate a polygonal mesh.

More Colors

While filling holes in the proxy grid allows recovering unseen geometry, we do not currently recover unseen appearance. This could be done using *inpainting* techniques such as *ImageMelding* [DSB+12], to compute pixel values at locations where the sensor did not provide color information. As we store proxy appearance as simple texture images, it would be straightforward to add a color filling step before exporting the mesh.



raw depth map



after proxy filtering

Figure V.8: Surface reconstruction on raw depth and proxy-filtered data. This preliminary result shows how geometric proxies can be used as a smoothing operator in the context of mesh reconstruction. Here, we apply *Poisson surface reconstruction* [KBH06] to raw depth point cloud (left) and point cloud generated by sampling our proxies (right). We can see that the planar elements (door on the left, ceiling at the top, and walls) are smoothly reconstructed and do not exhibit the artifacts seen on the raw depth due to sensor limitations.

V.4.2 Proxies to Regularize Reconstruction

Our proxies are a geometric superstructure that stores accumulated data from the RGB-D stream to model regular elements of indoor scenes. As we showed in section V.3, it can be used directly as a regularized reconstruction of the scene by meshing the proxy cells. However, we could imagine using its shapes as *regularization priors* to be integrated into existing reconstruction frameworks.

For example, we could obtain a smoother model by applying *Poisson surface reconstruction* [KBH06] on a regularized point cloud. Figure V.8 shows preliminary results of surface reconstruction applied before and after proxy filtering. One could also use directly the geometric proxy information to regularize the mesh within the reconstruction process. Accumulating data points can be achieved using e.g., an *Octomap* [HWB+13] where additional voxel or vertex attributes could link to proxy shapes bearing additional geometric information used for regularization. As such, the proxy superstructure becomes a smoothing operator used before or during reconstruction.

VI

CONCLUSION

VI.1 Summary

VI.1.1 Concluding Remarks

In this thesis, we defined a methodology which introduces geometry processing and analysis tools into the problem of RGB-D based indoor scene analysis. As planned in section I.2, we focused on the construction of a scene superstructure based on simple geometric shapes detected in the raw 3D data, which can be used as *geometric scaffold* for *live processing* of RGB-D frame sets as well as substitute for the creation of efficient global *3D representations* of the surroundings. In particular, our approach shows robustness in the constrained context where positions of sensors and objects as well as illumination and scene setup can change in a time frame that we do not control.

We started this thesis with the observation that geometric abstraction of 3D data would make sense to process captured scenes as a simple, lightweight and meaningful representation. As we show in section II.4, abstracting real 3D elements with simple parameterized shapes allows extreme simplification of the geometry while keeping an accurate representation of the input data. Hence, our lightweight geometric proxies model the dominant structural elements in the scene while adapting well to changes. They provide a consistent spatio-temporal support for the typical processing primitive designed to *localize, enhance or consolidate* data and lighten subsequent operations. As shown in section V.3, geometric proxies as 3D reconstruction are well balanced between performance and accuracy, with the goal of being efficient and sufficiently usable for modern 3D applications running on mobile devices.

Our contributions allow bringing Ayotle's efficient process of local 3D interaction to a consumer mobile device by tackling the issues of device motion, dynamic scenes and lifelong operation. The continuous integration of scene analysis software components into the company's code base (section A.1.1), with the guidance of software engineers, was key to discovering new issues due to *real life* conditions. In addition, a large amount of engineering effort was used to build a demonstrator interface allowing real-time remote feedback and control over the processing on the embedded platform (section A.1.2). Eventually, iterations across all components of product development were necessary to tailor the output of our scene analysis framework towards better performance and usability. Specific details on the integration of our results into the company's product are given in section A.1.3.

VI.1.2 Contributions

As we showed in introduction (section I.1), real-time analysis of indoor scenes in an embedded computing environment raises several challenges that have to be addressed in order to run modern interactive 3D applications. The amount of incoming data as well as its low quality and resolution require specific algorithms to process RGB-D streams in real-time and infer structural information on the observed scene. The use of *geometry processing* on this 2.5D data allows lightening and improving subsequent operations of the *computer vision* and *computer graphics* frameworks such as localization, reconstruction and visualization.

In particular, we made several contributions listed below:

- an *overview and classification* of recent methods to generate simple geometric shape abstraction from captured 3D data (section II.4);
- a fast and lightweight methodology to *register* overlapping 3D images of indoor scenes based on planar representation (chapter III);
- a *consistent model* based on simple geometric shapes to aggregate 3D samples of captured scenes in a global superstructure (chapter IV);
- its use in an efficient live *processing and consolidation* framework for RGB-D streams (chapter V).

We communicated on these contributions during this PhD under different forms, summarized in a complete list of scientific productions available in section A.2.

VI.2 Open Problems

The scene abstraction as defined here is efficient and allows tackling some challenges raised by our context and needs. However, abstracting the scene with simple geometric shapes has some limitations, both in terms of accuracy and performance. Hence, the current state of our scene analysis framework could benefit from some modifications and evolutions for which we give hints in the next sections. In addition, a number of applicative opportunities can be exploited on top of it, in particular regarding the integration into an interaction tool.

VI.2.1 Accuracy

While representing real indoor scene elements with simple shapes leads to a good trade-off between efficiency and accuracy, the latter may be improved to get results closer to state-of-the-art. The current shape detection sometimes fails to distinguish nearby objects and often generates a single shape instance for parts of different objects. More advanced considerations on the extent of these shapes could allow splitting more accurate connected components, for which the current method has quite a coarse resolution. In addition, leveraging recent performance of semantic identification of objects using *neural networks* [RDGF16, SIVA17, HSS18] could allow separating parts of different objects.

When considering the variability of inhabited indoor environments, our current geometric proxy modeling lacks stability and robustness. Objects moving on a short term e.g. people or pets, or long term e.g. small furniture or bags, might be modeled as structural elements with simple shapes, even with the safeguards and requirements of long term presence added to our implementation. Again, more advanced reasoning on the nature of elements and their motion might help discarding this type of non-structural parts from the model.

In section V.4.1, we give hints towards the construction of more faithful proxy models. In particular, our shape-based modeling of indoor scenes lacks knowledge of non-regular elements for which depth samples could be accumulated into an additional sparse local structure such as a voxel grid or octree. Hole filling using the proxies as a prior, could be more efficient by identifying empty space at door and window locations, in order to make stronger decisions when recovering missing data. Eventually, the proxies could be used as a smoothing operator to be integrated into existing surface reconstruction methods. Introducing such a regularization prior into e.g., *Poisson surface reconstruction* [KBH06] could lead to more regular reconstruction, leveraging the structure awareness brought by geometric information.

The *multi-scale* nature of indoor scenes is still to be considered within our geometric proxies. We could generate knowledge of global and local scene geometry at multiple levels, whose analysis could reveal scene information of e.g., similarities or clusters. First, at the level of proxies, analyzing the local geometry within each cell could reveal regular structures whose modeling by a compact representation would allow generating more faithful scene model with finer details. This idea is close to the *volume-surface trees* [BHGS06], where volume nodes of a 3D hierarchical representation switch to surface nodes when their local geometry is close to a regular shape, e.g. planar. Second, at scene level, the currently unique set of proxies representing the scene at a single scale could be replaced by multiple sets of proxies each modeling specifically and faithfully different scales of the scene. Decomposing a proxy shape into sub-shapes or aggregating nearby similar proxies would lead to multiple levels of detail representing the surroundings, providing more accurate geometrical insight into the scene structure at different scales. In such a hierarchical representation, local proxy statistics could be shared between levels, similarly to *covariance trees* [GAB14], where multivariate Gaussian kernels model spatial data locally and hierarchically and share distribution information between tree levels.

When using planes to localize 3D acquisitions from each other as in chapter III, we saw that two main issues prevented the methods from giving superior results. First, planes, even associated with shape space extent and color histogram, do not contain enough discriminative features and information to be unequivocally matched between views. As we explain in section III.5.2, improvement of plane matching could be allowed through the use of stronger heuristics, a grouping of the planes by three or four, or a step of quality check. Second, a limitation of our approach is the need for overlapping planar structures in two views, which is not always the case, in particular when comparing single frames from sensors with no control over their fields of view. In order to increase the success rate of our method, we could imagine using not only planes, but also objects present in the scene, that would be modeled by simple shapes as well. Shapes such as cylinders, spheres, boxes or cones, are simple and lightweight but embed more information than planes. By exploiting the parameters of these simple shapes matched in overlapping views, in the same spirit as with planes, we could infer sensor motion more accurately.

VI.2.2 Performance

While we designed our framework to be fast and lightweight, the different operations required to process the data and maintain our superstructure still prevent full real-time processing. The current implementation within the interaction system leverages multi-threading to process the data at a lower frame rate for scene analysis, while maintaining real-time interaction. This setup is viable, as a model of the scene structure does not need to be updated 30 times per second, however a better performance would mean taking better advantage of available resources to run more functionalities on the device. In that sense, model update could be run every second or so and be fast enough to leave time for other processing to run.

While representing indoor scenes with simple lightweight geometric shapes, our current model can itself benefit from improvement on its structure. The objective of such an optimization of our data structure is twofold. First, a more efficient model means a lower memory footprint on the device, which again leaves more resources to other processes. Second, and more importantly, a lighter model is lighter to transmit through wireless network for e.g. visualization or further processing. The intuitive improvement to implement would be a smarter organization of local proxy grid cells, whose goal is to accurately represent the spatial limits of the object in shape space as well as the saliency at the surface of the shape. When looking at the proxy scene models in Figure V.7 (bottom right), we can see large planar areas which contain many similar quads. We could imagine building a sparse adaptive structure where similar basic grid cells are grouped together into a higher level cell that gathers information from lower level cells. This multi-scale grouping would generate different levels of detail that are key to enable faster transmission and visualization on low end networks and devices. Such an advanced cell quantization mechanism, associated with a smart re-organization of the *smoothed local histograms*, can be seen as a form of local learning of the nature and distribution of the geometry, which could then be used to modulate the aggregation of samples into the structure.

VI.2.3 Interaction

Integrating our scene analysis framework into the existing embedded interaction system, as described in section A.1.3, enables efficient localization of the sensor as well as visualization of the data. However, these improvements do not allow the interaction to leverage the knowledge of the surroundings acquired by our framework. We can imagine seamlessly using the simplified model as input of the interaction system, provided that processing is fast enough, to increase the performance and usability of interactive tools. In the following, we present two possible uses of the our lightweight and simplified scene representation.

First, our scene model naturally defines a distinction between permanent structural elements of the scene such as walls or furniture and temporary data clusters such as people interacting with the environment. We could use the structural information stored in our proxies to remove RGB-D data points belonging to the scene structure, hence building a *change detection* mechanism. Removing this structural data from the input of the interaction engine greatly reduces the amount of data to process. Such a use of geometric proxies for change detection simplifies the operations required to analyze the interactions, as we focus on the motion of people with relation to fixed local interaction zones. Again, this requires real-time identification of the structure data points in order to maintain real-time interaction responsiveness.

Second, this distinction between structural and moving elements gives a natural prior as to where and how to define the interaction zones. Using the temporally consistent parameterization of the proxies as a stable interaction space would allow creating smart control interfaces. For example, we could define interaction templates that would be automatically setup when a specific organization of the space or elements are detected. In addition, the proxy support could allow displaying 3D data onto the surfaces of the observed scene and open new opportunities for *augmented reality* applications in the context of 3D interaction.

APPENDIX

A.1 Implementation

A.1.1 Software Components

The components of the proxy-based scene analysis pipeline have been implemented as modules within a prototype application running on an embedded device. Existing methods used in the processing, such as scene tracking [EHS+14] and geometric primitive detection [SWK07], have been reimplemented following the corresponding articles. Basic functionality such as system tools or 3D vectors and matrices and the corresponding operations were used directly from *Ayotle*'s code base. Time profiling and memory footprint monitoring modules have been included in order to control the usability of the tool. In total, about 55K lines of C++ code were written to implement all components and experiments of this thesis.

A.1.2 Demonstrator Interface

A *graphical user interface* has been designed to control most parameters of the processing and get a real-time feedback of the data (Figure A.1). This demonstrator has a client-server architecture where the server is a module of the prototype application and updates its data at each new frame. The client runs within the graphical interface to update the displayed data. The initial goal of this interface was to display the raw data acquired by the sensor as well as the enhancement and consolidation obtained through the processing. The specificity of the project being the need for embedded operation prevents direct access to the computer's graphical interface and led to the development of such a client-server architecture, where only wireless connection is needed to get remote feedback. Later, the communication between the demonstrator and the processing device was modified for the device to receive information from the interface. This allowed modifying parameters within the interface and getting feedback of their impact on the data in real time.

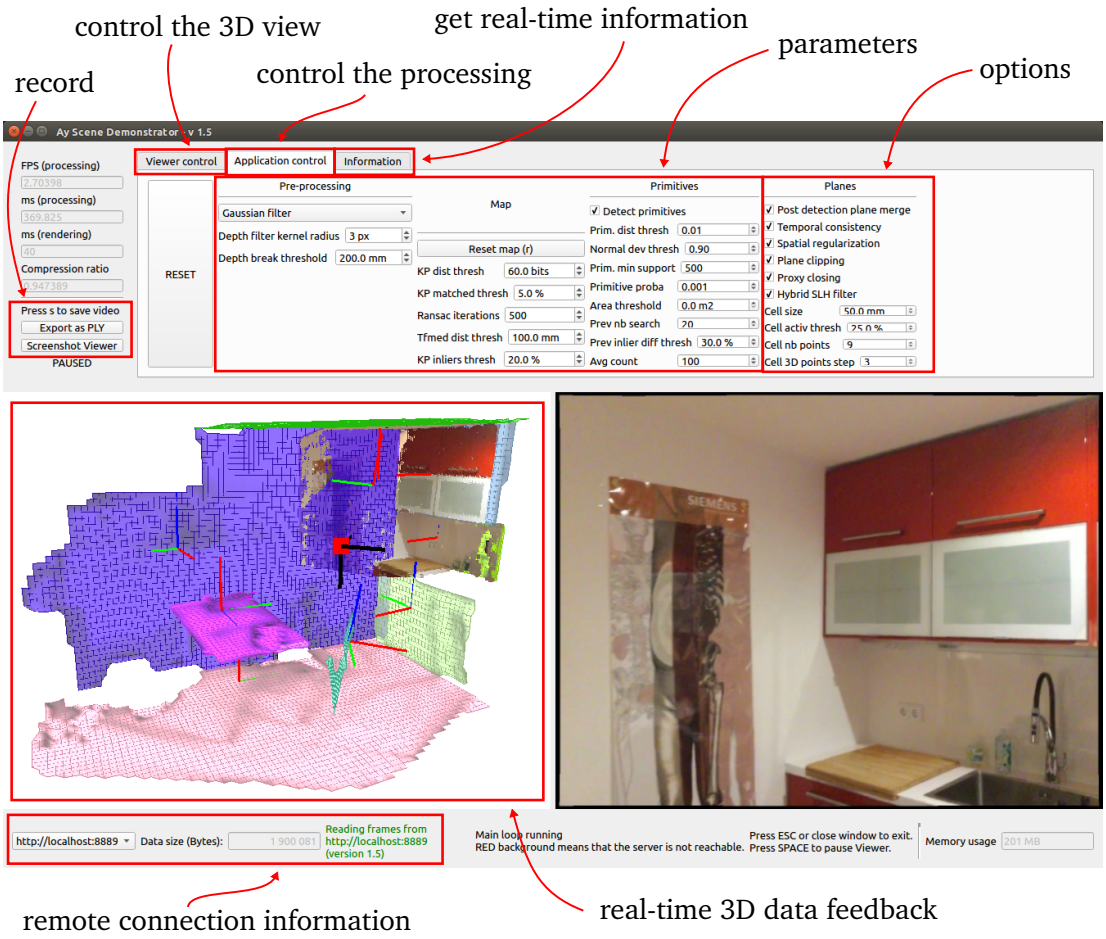


Figure A.1: Demonstrator Interface. The graphical interface was developed to display the current data seen and processed by the sensor (middle), as well as controlling a number of parameters to analyze their impact on the results (top). The client-server architecture allows remote analysis of the results when working with embedded devices (see bottom banner).

A.1.3 Technology Transfer

In the industrial context of this thesis, research presented in chapter III, chapter IV and chapter V has been integrated into the company's product *Hayo*, presented in section I.1.1. First, the plane-based 3D view registration method showed in chapter III was used to solve the need for spatial consistency of the local interaction zones when the device is moved. Second, the scene reconstruction offered by the geometric proxy representation (section V.3) was integrated into both the device and the mobile app to allow more complete and simplified visualization of the sensor view for more intuitive positioning of the interaction zones.

Efficient RGB-D Sensor Relocalization

In the use case of 3D interaction, when the mobile device is moved in the scene, the sensor needs to be localized with relation to a reference, here being its last position. As the performance of recent frame-to-frame tracking methods (section II.3) is not high enough to be accurate on embedded devices such as a Raspberry Pi board, we need to use so-called *relocalization* methods, as if the sensor was lost. In practice, we detect motion of the sensor and wait until motion has finished to trigger the computation. We first tried to use the visual sensor to detect the motion of the device, but its high sensitivity to objects moving in the field of view led to the use of an accelerometer. When the device stops moving, we run the plane-based registration algorithm from chapter III (and described in our patent [KYZBC19]) several times and keep the motion matrix with the highest confidence.

Live Simplified Visualization of RGB-D Stream

The *Hayo* mobile app was designed to allow non-expert users to position and configure local interaction zones activated with the *Hayo* sensor. The view of the sensor would be shown as a 3D point cloud computed from its depth map. However, the sparse and incomplete depth map provided by commodity depth sensors, as explained in section II.1.2, leads to point structures that are non intuitive to understand for inexperienced users. In consequence, we decided to use our geometric representation of the scene, described in chapter V, to improve the user's perception of the 3D sensor view. The light weight and simplicity of our structure is well suited to live streaming to a mobile device through wireless network. As shown in Figure A.2, depth points belonging to the structure are identified and removed from the data to stream. They are replaced by textured and meshed proxy grid cells which offer lighter and more complete display, leading to a hybrid polygon and point visualization.

Figure A.2: Geometric proxies in *Hayo* mobile app. In the 3D viewer of the companion app for the *Hayo* camera, proxy inliers are replaced by the textured proxy mesh to offer a more complete and intuitive visualization during the process of placing interaction zones (light green sphere). In addition, streaming the compressed proxy version of the data increases the refreshing rate of the 3D information.

A.2 Scientific Productions

This section presents the different productions made during this PhD, under the form of journal articles, conference proceedings, talks, patents and industrial conferences. Full references for the journal and conference communications are listed on the ORCID identifier 0000-0002-5998-3932.

A.2.1 Peer-Reviewed Journal Articles

- Adrien Kaiser, Jose Alonso Ybanez Zepeda and Tamy Boubekeur:
  A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data [KYZB19]
  Computer Graphics Forum, Volume 38 (2019), number 1 (February) pp. 167 – 196
  Presented at *Eurographics* 2019, May 6 – 10, Genova, Italy (1h30 talk)
  DOI: 10.1111/cgf.13451

A.2.2 Peer-Reviewed Conference Proceedings

- Adrien Kaiser, Jose Alonso Ybanez Zepeda and Tamy Boubekeur:
  Proxy Clouds for RGB-D Stream Processing: A Preview [KYZB17a]
  Eurographics Posters Program
  Presented at *Eurographics* 2017, April 24 – 28, Lyon, France (1h30 poster session)
  DOI: 10.2312/egp.20171039

- Adrien Kaiser, Jose Alonso Ybanez Zepeda and Tamy Boubekeur:
  Proxy Clouds for RGB-D Stream Processing: An Insight [KYZB17b]
  SIGGRAPH Technical Talks Program
  Presented at *SIGGRAPH* 2017, July 30 – August 3, Los Angeles, CA, USA (20 min talk)
  DOI: 10.1145/3084363.3085031

- Adrien Kaiser, Jose Alonso Ybanez Zepeda and Tamy Boubekeur:
  Proxy Clouds for Live RGB-D Stream Processing and Consolidation [KYZB18]
  European Conference on Computer Vision (ECCV)
  Presented at *ECCV* 2018, September 8 – 14, Munich, Germany (2h poster session)
  DOI: 10.1007/978-3-030-01231-1_16

A.2.3 Invited Talks

- *Proxy Clouds for RGB-D Stream Processing: A Preview* (*Eurographics* 2017 poster)
  Research Day of the *Image, Signal, Data* department of *Telecom Paris*
  July 6, 2017, Paris, France (1h15 poster session)

- *Geometric Proxies for Live RGB-D Stream Processing and Consolidation*
  Research Day of the *Image, Signal, Data* department of *Telecom Paris*
  July 5, 2018, Paris, France (20 min talk)

- *Simple Geometric Primitives Detection for Captured 3D Data*
  IMAGINE research group of ENPC
  October 22, 2018, ENPC, Champs-sur-Marne, France (1h talk)

- *Proxy Clouds for Live RGB-D Stream Processing and Consolidation*
  Research seminar of *NPM3D* class in *MVA master* of *ENS Paris-Saclay*
  February 22, 2019, Mines ParisTech, Paris, France (20 min talk)

A.2.4 Patents

- Adrien Kaiser, Jose Alonso Ybanez Zepeda, Tamy Boubekeur and Alain Courteville:
  Procédé de Recalage d'Images de Profondeur [KYZBC19]
  French patent request number FR1901827, submitted on February 22, 2019.

A.2.5 Industrial Conferences

- Presentation of Ayotle's *Hayo* camera [1]
  Laval Virtual 2017, March 22 – 26, Laval, France
- Presentation of Ayotle's *Hayo* camera
  Futur en Seine 2017, June 8 – 10, Paris, France

[1] hayo.io

BIBLIOGRAPHY

[AB99] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, 1999. II.4.1

[ABCO+03] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. Computing and rendering point set surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 9(1):3–15, March 2003. II.4.1

[ACK01] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266, June 2001. II.4.1

[AEH15] Mahdi Alehdaghi, Mahdi Abolfazli Esfahani, and Ahad Harati. Parallel ransac: Speeding up plane extraction in rgbd image sequences using gpu. *Computer and Knowledge Engineering (ICCKE)*, pages 295–300, October 2015. II.4.4

[AFS06] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193, March 2006. II.4.4

[AFT14] Guillem Alenyà, Sergi Foix, and Carme Torras. Using ToF and RGBD Cameras for 3D Robot Perception and Manipulation in Human Environments. *Intelligent Service Robotics*, 7(4):211–220, October 2014. II.1.1

[AII+18] Mykhaylo Andriluka, Umar Iqbal, Eldar Insafutdinov, Leonid Pishchulin, Anton Milan, Juergen Gall, and Bernt Schiele. Posetrack: A benchmark for human pose estimation and tracking. In *Proceedings of*

the IEEE Conference on Computer Vision and Pattern Recognition, pages 5167–5176, 2018. I.1.3

[AMCO08] Dror Aiger, Niloy J Mitra, and Daniel Cohen-Or. 4-points congruent sets for robust pairwise surface registration. *ACM Transactions on Graphics (TOG)*, 27(3):85, 2008. II.2.2, II.6

[And79] Alex M Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979. II.4.1

[AP10] Marco Attene and Giuseppe Patanè. Hierarchical structure recovery of point-sampled surfaces. *Computer Graphics Forum*, 29(6):1905–1920, September 2010. II.4.4, II.4.4

[ASF+13] Murat Arikan, Michael Schwärzler, Simon Flöry, Michael Wimmer, and Stefan Maierhofer. O-snap: Optimization-based snapping for modeling architecture. *ACM SIGGRAPH*, 32(6):6:1–6:15, November 2013. II.15

[Avr76] M. Avriel. Nonlinear programming: analysis and methods. *Prentice-Hall series in automatic computation*, 1976. II.4.4

[Bal81] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981. II.4.3

[BDTL15] Pierre Buyssens, Maxime Daisy, David Tschumperlé, and Olivier Lézoray. Superpixel-based depth map inpainting for rgb-d view synthesis. In *IEEE International Conference on Image Processing (ICIP)*, pages 4332–4336. IEEE, 2015. II.1.3

[BFF15] Timur Bagautdinov, Francois Fleuret, and Pascal Fua. Probability occupancy maps for occluded depth images. *Computer Vision and Pattern Recognition*, June 2015. II.4.2

[BHGS06] Tamy Boubekeur, Wolfgang Heidrich, Xavier Granier, and Christophe Schlick. Volume-surface trees. *Computer Graphics Forum*, 25(3):399–406, 2006. VI.2.1

[BL79] S. Beucher and C. Lantuejoul. Use of watersheds in contour detection. *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, Rennes, France.*, September 1979. II.4.4

[BL95] Gérard Blais and Martin D. Levine. Registering Multiview Range Data to Create 3D Computer Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):820–824, 1995. II.3.2

[BM92] Paul J Besl and Neil D McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,

14(2):239–256, 1992. II.2.1, II.2.2

[BRR15] Akash Bapat, Adit Ravi, and Shanmuganathan Raman. An iterative, non-local approach for restoring depth maps in rgb-d images. In *Twenty First National Conference on Communications (NCC)*, pages 1–6. IEEE, 2015. II.1.3, II.3

[BSBW14] Ben Bellekens, Vincent Spruyt, Rafael Berkvens, and Maarten Weyn. A Survey of Rigid 3D Pointcloud Registration Algorithms. In *AMBIENT 2014: the Fourth International Conference on Ambient Computing, Applications, Services and Technologies, August 24-28, 2014, Rome, Italy*, pages 8–13, August 2014. II.2.2

[BSG+11] Roseline Bénière, Gérard Subsol, Gilles Gesquière, François Le Breton, and William Puech. Recovering primitives in 3D cad meshes. *IS&T/SPIE Electronic Imaging*, pages 78640R–78640R, 2011. II.4.4

[BTP13] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Sparse Iterative Closest Point. *Computer Graphics Forum (Symposium on Geometry Processing)*, 32(5):1–11, 2013. II.7, II.2.2

[BTS+14] Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Joshua Levine, Andrei Sharf, and Claudio Silva. State of the art in surface reconstruction from point clouds. *EUROGRAPHICS star reports*, pages 161–185, April 2014. II.4.1

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417. Springer, 2006. II.2.1

[BV11] Joydeep Biswas and Manuela Veloso. Fast sampling plane filtering, polygon construction and merging from depth images. *Robotics: Science and Systems Conference (RSS)*, June 2011. II.4.1

[BV12] Joydeep Biswas and Manuela Veloso. Planar polygon extraction and merging from depth images. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3859–3864. IEEE, 2012. II.1.3

[CC08] Jie Chen and Baoquan Chen. Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision*, 78(2-3):223–236, July 2008. II.4.3

[CHL15] Miguel Heredia Conde, Klaus Hartmann, and Otmar Loffeld. Adaptive high dynamic range for time-of-flight cameras. *IEEE Transactions on Instrumentation and Measurement*, 64(7):1885–1906, 2015. II.1.3

[CL96] Brian Curless and Marc Levoy. A Volumetric Method for Building

Complex Models from Range Images. *ACM SIGGRAPH*, pages 303–312, August 1996. II.3.2

[CLL12] Li Chen, Hui Lin, and Shutao Li. Depth image enhancement for kinect using region growing and bilateral filter. In *21st International Conference on Pattern Recognition (ICPR)*, pages 3070–3073. IEEE, 2012. II.1.3

[CM91] Yang Chen and Gérard Medioni. Object Modelling by Registration of Multiple Range Images. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 2724–2729. Elsevier, 1991. II.2.2

[CM02] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, May 2002. II.4.4

[CM05] Ondrej Chum and Jiri Matas. Matching with prosac-progressive sample consensus. *Computer Vision and Pattern Recognition*, pages 220–226, June 2005. II.4.2

[CRS98] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: Measuring Error on Simplified Surfaces. *Computer Graphics Forum*, 17(2):167–174, August 1998. II.4.5, V.4

[CS12] Massimo Camplani and Luis Salgado. Adaptive spatio-temporal filter for low-cost camera depth maps. In *IEEE International Conference on Emerging Signal Processing Applications (ESPA)*, pages 33–36. IEEE, 2012. II.1.3

[CSAD04] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Transactions on Graphics (TOG)*, 23(3):905–914, August 2004. II.15, II.4.4

[CSM12] Peter Carr, Yaser Sheikh, and Iain Matthews. Monocular object detection using 3d geometric primitives. *ECCV*, pages 864–878, October 2012. II.4.2, II.4.4

[CY99] James M Coughlan and Alan L Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *Proceedings of*

the Seventh IEEE International Conference on Computer Vision, volume 2, pages 941–947. IEEE, 1999. III.1, IV.4.1

[CZK15] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust Reconstruction of Indoor Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. II.8, II.2.3, III.4.2

[CZS+13] Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 3-sweep: Extracting editable objects from a single photo. *ACM Transactions on Graphics (TOG)*, 32(6):195, November 2013. II.4.4

[DBI18] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPFNet: Global Context Aware Local Features for Robust 3D Point Matching. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*, 2018. II.2.2

[DDSD03] Xavier Décoret, Frédo Durand, François X Sillion, and Julie Dorsey. Billboard clouds for extreme model simplification. *ACM Transactions on Graphics (TOG)*, 22(3):689–696, August 2003. II.4.3

[DGFF12a] Mingsong Dou, Li Guan, Jan-Michael Frahm, and Henry Fuchs. Exploring High-Level Plane Primitives for Indoor 3D Reconstruction with a Hand-held RGB-D Camera. In *Asian Conference on Computer Vision*, pages 94–108. Springer, 2012. II.2.4, III.2.3

[DGFF12b] Mingsong Dou, Li Guan, Jan-Michael Frahm, and Henry Fuchs. Exploring high-level plane primitives for indoor 3d reconstruction with a hand-held rgb-d camera. In Kyoung Mu Lee, Yasuyuki Matsushita, James M. Rehg, and Zhanyi Hu, editors, *Asian Conference on Computer Vision*, volume 7724 of *Lecture Notes in Computer Science*, pages 94–108. Springer, Berlin, Heidelberg, November 2012. II.3.1

[DH72] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, January 1972. II.4.3

[DNZ+17a] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 36(3):24, 2017. IV.4.2, V.3.1, V.6, V.7

[DNZ+17b] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. BundleFusion: Real-Time Globally Consistent 3D Reconstruction using Online Surface Reintegration. *ACM Transactions on Graphics (TOG)*, 36(4):76a, 2017. II.3.2, II.3.2

[DSB+12] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. Image Melding: Combining Inconsistent Images

using Patch-based Synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):82–1, 2012. V.4.1

[DSM+17] Maksym Dzitsiuk, Jürgen Sturm, Robert Maier, Lingni Ma, and Daniel Cremers. De-noising, Stabilizing and Completing 3D Reconstructions On-the-go using Plane Priors. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3976–3983. IEEE, May 2017. II.3.2, II.13

[DUNI10] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model Globally, Match Locally: Efficient and Robust 3D Object Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 998–1005. IEEE, June 2010. II.2.2

[EAG17] Sergio Escalera, Vassilis Athitsos, and Isabelle Guyon. Challenges in multi-modal gesture recognition. In *Gesture Recognition*, pages 1–60. Springer, 2017. I.1.3

[Ebr15] Mostafa Ebrahim. 3d laser scanners' techniques overview. *International Journal of Science and Research (IJSR)*, 4:5–611, 10 2015. II.1.1

[EGD+12] Kyis Essmaeel, Luigi Gallo, Ernesto Damiani, Giuseppe De Pietro, and Albert Dipandà. Temporal denoising of kinect depth data. In *Eighth International Conference on Signal Image Technology and Internet Based Systems (SITIS)*, pages 47–52. IEEE, 2012. II.1.3

[EHS+14] Felix Endres, Jürgen Hess, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. 3D Mapping with an RGB-D Camera. *IEEE Transactions on Robotics*, 30(1):177–187, 2014. II.10, II.3.1, IV.2.1, 11, A.1.1

[ERAB15] Hakim Elchaoui Elghor, David Roussel, Fakhreddine Ababsa, and El Houssine Bouyakhf. Planes Detection for Robust Localization and Mapping in RGB-D SLAM systems. In *International Conference on 3D Vision (3DV)*, pages 452–459. IEEE, October 2015. II.3.1

[FB81] Martin A Fischler and Robert 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, June 1981. II.2.1, II.3.1, II.4.2

[FCSS09] Yudai Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Manhattan-world stereo. *CVPR*, pages 1422–1429, June 2009. II.4.3

[FH75] Keinosuke Fukunaga and Larry D Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition.

Information Theory, IEEE Transactions on, 21(1):32–40, January 1975. II.4.4

[FK17] Wolfgang Förstner and Kourosh Khoshelham. Efficient and Accurate Registration of Point Clouds with Plane to Plane Correspondences. In *3rd International Workshop on Recovering 6D Object Pose*, pages 2165–2173. IEEE, 2017. II.2.4

[FO08] Leandro AF Fernandes and Manuel M Oliveira. Real-time line detection through an improved hough transform voting scheme. *Pattern recognition*, 41(1):299–314, 2008. II.4.3

[Fol96] J.D. Foley. 12.7 Constructive Solid Geometry. *Computer Graphics: Principles and Practice, Addison-Wesley systems programming series*, pages 557–558, 1996. II.4.1

[FTK14] Chen Feng, Yasuhiro Taguchi, and Vineet R Kamat. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. *ICRA*, pages 6218–6225, June 2014. II.4.1, II.4.4

[GAB14] Thierry Guillemot, Andrés Almansa, and Tamy Boubekeur. Covariance trees for 2d and 3d processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 556–563, 2014. VI.2.1

[GBS+16] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A Comprehensive Performance Evaluation of 3D Local Feature Descriptors. *International Journal of Computer Vision*, 116(1):66–89, January 2016. II.2.2

[GG04] Natasha Gelfand and Leonidas J Guibas. Shape segmentation using local slippage analysis. *Eurographics, symposium on Geometry processing*, pages 214–223, July 2004. II.4.4

[GH97] Michael Garland and Paul S Heckbert. Surface Simplification using Quadric Error Metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., August 1997. II.4.5, II.4.5, III.3.2

[GMLB12] Lucian Cosmin Goron, Zoltan-Csaba Marton, Gheorghe Lazea, and Michael Beetz. Robustly segmenting cylindrical and box-like objects

in cluttered scenes using depth cameras. *Proceedings of ROBOTIK 2012*, pages 1–6, May 2012. II.15, II.4.4, II.4.4

[GVL96] G.H. Golub and C.F. Van Loan. Matrix computations. *Johns Hopkins Studies in the Mathematical Sciences*, 1996. II.4.4

[GZ15] Xiang Gao and Tao Zhang. Robust rgb-d simultaneous localization and mapping using planar point features. *Robotics and Autonomous Systems*, 72:1–14, 2015. II.3.1

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics and Applications*, 26(2), March 1992. II.4.1

[HDGN17] Jingwei Huang, Angela Dai, Leonidas Guibas, and Matthias Niessner. 3dlite: Towards commodity 3d scanning for content creation. *ACM Transactions on Graphics (TOG)*, 36(6):203, 2017. II.3.3, IV.4.2, IV.1, V.3.1, V.5, V.6, V.7

[HF17] Maciej Halber and Thomas Funkhouser. Fine-To-Coarse Global Registration of RGB-D Scans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. II.2.4, II.9, III.4.2

[HHRB11] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-time plane segmentation using rgb-d cameras. *RoboCup 2011*, pages 306–317, July 2011. II.4.3, II.4.4

[Hou62] Paul V. C. Hough. Method and means for recognizing complex patterns. *US Patent 3,069,654*, December 1962. II.4.3

[HSS18] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. VI.2.1

[HSSM14] Rostislav Hulik, Michal Spanel, Pavel Smrz, and Zdenek Materna. Continuous Plane Detection in Point-cloud Data based on 3D Hough Transform. *Journal of visual communication and image representation*, 25(1):86–97, January 2014. II.4.3

[HWB+13] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 2013. Software available at `http://octomap.github.com`. V.4.2

[HWZK17] Ming Hsiao, Eric Westman, Guofeng Zhang, and Michael Kaess. Keyframe-based dense planar slam. In *IEEE International Conference on*

Robotics and Automation (ICRA), pages 5110–5117. IEEE, May 2017. II.3.1

[JH99] Andrew E. Johnson and Martial Hebert. Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, May 1999. II.2.2, II.5

[JLSW02] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual Contouring of Hermite Data. In *ACM transactions on graphics (TOG)*, volume 21, pages 339–346. ACM, 2002. V.4.1

[Kae15] M. Kaess. Simultaneous Localization and Mapping with Infinite Planes. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4605–4611, May 2015. II.3.1

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. *Proceedings of the fourth Eurographics symposium on Geometry processing*, 7, June 2006. II.4.1, V.8, V.4.2, VI.2.1

[KCLU07] Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics (ToG)*, 26(3):96, 2007. II.1.3, V.2.1

[KDSX15] Matthew Klingensmith, Ivan Dryanovski, Siddhartha Srinivasa, and Jizhong Xiao. Chisel: Real Time Large Scale 3D Reconstruction Onboard a Mobile Device using Spatially Hashed Signed Distance Fields. *Robotics Science and Systems*, July 2015. II.3.2, II.3.2

[KEB91] Nahum Kiryati, Yuval Eldar, and Alfred M Bruckstein. A probabilistic hough transform. *Pattern recognition*, 24(4):303–316, 1991. II.4.3

[KHB+15] Salman H. Khan, Xuming He, Mohammed Bennamoun, Ferdous Sohel, and Roberto Togneri. Separating objects and clutter in indoor scenes. *Computer Vision and Pattern Recognition*, June 2015. II.4.4

[KL15] Zhizhong Kang and Zhen Li. Primitive fitting based on the efficient multibaysac algorithm. *PloS one*, 10(3):e0117341, 2015. II.4.2

[KLL+13] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *International Conference on 3D Vision (3DV)*, pages 1–8. IEEE, June 2013. II.3.2

[KLM+13] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning part-based

templates from large collections of 3d shapes. *Transactions on Graphics (Proc. of SIGGRAPH)*, 32, November 2013. II.4.4

[KM07] Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE Computer Society, 2007. I.1.2, II.3.1

[KS10] Michael Kass and Justin Solomon. Smoothed local histogram filters. *ACM Transactions on Graphics (TOG)*, 29(4):100, 2010. IV.3.3

[KYZB17a] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. Proxy Clouds for RGB-D Stream Processing: A Preview. In *Proceedings of Eurographics 2017 - Posters*, April 2017. A.2.2

[KYZB17b] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. Proxy Clouds for RGB-D Stream Processing: An Insight. In *Proceedings of SIGGRAPH 2017 - Talks*, July 2017. A.2.2

[KYZB18] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. Proxy Clouds for Live RGB-D Stream Processing and Consolidation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. I.3.1, A.2.2

[KYZB19] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data. *Computer Graphics Forum*, 38(1):167–196, February 2019. I.3.1, I.3.2, II.4, A.2.1

[KYZBC19] Adrien Kaiser, Jose Alonso Ybanez Zepeda, Tamy Boubekeur, and Alain Courteville. Procédé de Recalage d'Images de Profondeur, February 2019. FR Patent pending. I.3.1, A.1.3, A.2.4

[LC87] William E Lorensen and Harvey E Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *ACM SIGGRAPH*, volume 21, pages 163–169. ACM, 1987. V.4.1

[LCK16] Suolan Liu, Chen Chen, and Nasser Kehtarnava. A computationally efficient denoising and hole-filling method for depth image enhancement. In Nasser Kehtarnavaz and Matthias F Carlsohn, editors, *SPIE Conference on Real-Time Image and Video Processing*. SPIE, April 2016. II.1.3

[LDY+16] Ran Liu, Zekun Deng, Lin Yi, Zhenwei Huang, Donghua Cao, Miao Xu, and Ruishuang Jia. Hole-filling based on disparity map and

inpainting for depth-image-based rendering. *International Journal of Hybrid Information Technology*, 9(5):145–164, 2016. II.1.3

[LGZ+13] Hui Lin, Jizhou Gao, Yu Zhou, Guiliang Lu, Mao Ye, Chenxi Zhang, Ligang Liu, and Ruigang Yang. Semantic decomposition and reconstruction of residential scenes from lidar data. *ACM Transactions on Graphics, (Proc. of SIGGRAPH)*, 32(4), November 2013. II.4.4

[LH07] Sylvain Lefebvre and Hugues Hoppe. Compressed random-access trees for spatially coherent data. In Jan Kautz and Sumanta Pattanaik, editors, *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 339–349. Eurographics Association, 2007. IV.5.2

[Li14] Larry Li. Time-of-Flight Camera - An Introduction. In *Technical White Paper SLOA190B*, May 2014. II.1.1

[Li16] Larry Li. Filtering for 3d time-of-flight sensors. Technical Report SLOA230, Texas Instruments, January 2016. II.1.3

[LJW14] Anh Vu Le, Seung-Won Jung, and Chee Sun Won. Directional joint bilateral filter for depth images. *Sensors*, 14(7):11362–11378, 2014. II.1.3, II.3

[LLL+12] Tae-kyeong Lee, Seungwook Lim, Seongsoo Lee, Shounan An, and Se-young Oh. Indoor mapping using planes extracted from noisy rgb-d sensors. *Intelligent Robots and Systems (IROS)*, pages 1727–1733, October 2012. II.4.4

[Llo82] Stuart P Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, March 1982. II.4.4

[LM12] Florent Lafarge and Clément Mallet. Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation. *International journal of computer vision*, 99(1):69–85, August 2012. II.4.4, II.4.4

[LM14] Mathieu Labbé and François Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 2661–2666. IEEE, September 2014. II.3.1

[LMM98] Gabor Lukács, Ralph Martin, and Dave Marshall. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. *ECCV*, pages 671–686, June 1998. II.4.4, II.4.4

[Low99] David G Lowe. Object Recognition from Local Scale-invariant Features. *International Conference on Computer Vision*, 99(2):1150–1157, 1999. II.2.1

[LPRM02] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG)*, 21(3):362–371, July 2002. II.15

[LVHH06] Jean-Francois Lalonde, Nicolas Vandapel, Daniel Huber, and Martial Hebert . Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics*, 23(10):839 – 861, November 2006. II.4.4

[LWC+11] Yangyan Li, Xiaokun Wu, Yiorgos Chrysanthou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics*, 30(4):52:1–52:12, July 2011. II.15, II.4.2, II.4.6

[Mac67] James MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1(14):281–297, 1967. II.4.4

[MAM14] Nicolas Mellado, Dror Aiger, and Niloy J. Mitra. Super4PCS: Fast Global Pointcloud Registration via Smart Indexing. *Computer Graphics Forum*, 33(5):205–215, 2014. II.2.2

[MAT17] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. II.3.1

[MC04] Jiri Matas and Ondrej Chum. Randomized ransac with t d, d test. *Image and vision computing*, 22(10):837–842, September 2004. II.4.2

[MGSCVM+18] Vicente Morell-Gimenez, Marcelo Saval-Calvo, Victor Villena-Martinez, Jorge Azorin-Lopez, Jose Garcia-Rodriguez, Miguel Cazorla, Sergio Orts-Escolano, and Andres Fuster-Guillo. A Survey of 3D Rigid Registration Methods for RGB-D Cameras. *Advancements in Computer Vision and Image Processing*, pages 74–98, 2018. II.2.1

[MKRVG15] Andelo Martinovic, Jan Knopp, Hayko Riemenschneider, and Luc Van Gool. 3d all the way: Semantic segmentation of urban scenes from

start to end in 3d. *Computer Vision and Pattern Recognition*, June 2015. II.4.4

[MLD12] Dongbo Min, Jiangbo Lu, and Minh N Do. Depth video enhancement based on weighted mode filtering. *IEEE Transactions on Image Processing*, 21(3):1176–1190, 2012. II.1.3

[MLM01] David Marshall, Gabor Lukacs, and Ralph Martin. Robust segmentation of primitives from range data in the presence of geometric degeneracy. *PAMI*, 23(3):304–314, March 2001. II.4.4

[MMBM15] Aron Monszpart, Nicolas Mellado, Gabriel Brostow, and Niloy Mitra. RAPter: Rebuilding man-made scenes with regular arrangements of planes. *ACM SIGGRAPH*, August 2015. II.4.4, II.4.6

[MPM+14] Oliver Mattausch, Daniele Panozzo, Claudio Mura, Olga Sorkine-Hornung, and Renato Pajarola. Object detection and classification from large-scale cluttered indoor scans. *Computer Graphics Forum*, 33(2):11–21, July 2014. II.4.4

[NFS15] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. *Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, June 2015. II.3.2

[NIH+11] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time Dense Surface Mapping and Tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136. IEEE, October 2011. II.2.1, II.12, II.3.2

[NIL12] Chuong V Nguyen, Shahram Izadi, and David Lovell. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In *Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 524–530. IEEE, October 2012. II.1.2, IV.4.1

[NSS14] Fabrizio Nenci, Luciano Spinello, and Cyrill Stachniss. Effective compression of range data streams for remote robot operations using h. 264. In *IEEE/RSJ International Conference on Intelligent Robots and*

Systems (IROS), pages 3794–3799. IEEE, September 2014. V.2.5, V.1, V.2.5, V.2

[NZIS13] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):169, 2013. II.3.2

[OBA+03] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM SIGGRAPH 2003*, page 173, July 2003. II.4.1

[OLA16] Sven Oesau, Florent Lafarge, and Pierre Alliez. Planar shape detection and regularization in tandem. *Computer Graphics Forum*, 35(1):203–215, 2016. II.4.4

[OVWK14] Sebastian Ochmann, Richard Vock, Raoul Wessel, and Reinhard Klein. Towards the extraction of hierarchical building descriptions from 3d indoor scans. *EUROGRAPHICS Workshop on 3D Object Retrieval*, April 2014. II.4.4, II.4.6

[PBVP10] Kaustubh Pathak, Andreas Birk, Narunas Vaskevicius, and Jann Poppinga. Fast Registration Based on Noisy Planes with Unknown Correspondences for 3D Mapping. *IEEE Transactions on Robotics*, 26(3):424–441, 2010. II.2.4

[PH03] Emil Praun and Hugues Hoppe. Spherical Parametrization and Remeshing. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 340–349. ACM, 2003. IV.5, IV.2.2

[RBB09] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast Point Feature Histograms (FPFH) for 3D Registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009. III.1, III.4.2

[RBM+07] Radu Bogdan Rusu, Nico Blodow, Zoltan Marton, Alina Soos, and Michael Beetz. Towards 3d object maps for autonomous household robots. *Intelligent Robots and Systems (IROS)*, pages 3191–3198, October 2007. II.4.6

[RBMB08] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning Point Cloud Views using Persistent Feature Histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391. IEEE, 2008. II.2.2

[RBMB09] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments.

Intelligent Robots and Systems (IROS), pages 1–6, October 2009. II.4.4, II.4.4

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-time Object Detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. VI.2.1

[RDvdHV07] Tahir Rabbani, Sander Dijkman, Frank van den Heuvel, and George Vosselman. An integrated approach for modelling and global registration of point clouds. *ISPRS journal of Photogrammetry and Remote Sensing*, 61(6):355–370, February 2007. II.4.4

[RL01] Szymon Rusinkiewicz and Marc Levoy. Efficient Variants of the ICP Algorithm. *3DIM*, pages 145–152, May 2001. II.2.2

[Rou84] P.J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871–880, January 1984. II.4.2

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *International Conference on Computer Vision*, pages 2564–2571. IEEE, 2011. II.2.1, III.4.2

[RVDH05] Tahir Rabbani and Frank Van Den Heuvel. Efficient hough transform for automatic detection of cylinders in point clouds. *ISPRS journal of Photogrammetry and Remote Sensing*, 3:60–65, September 2005. II.4.3

[SA12] Mashhour Solh and Ghassan AlRegib. Hierarchical hole-filling for depth-based view synthesis in ftv and 3d video. *IEEE Journal of Selected Topics in Signal Processing*, 6(5):495–504, 2012. II.1.3

[SAG+13] Alex Shtof, Alexander Agathos, Yotam Gingold, Ariel Shamir, and Daniel Cohen-Or. Geosemantic snapping for sketch-based modeling. *Computer Graphics Forum (Proc. EUROGRAPHICS)*, 32(2pt2):245–253, May 2013. II.4.4

[Sch04] Hanns-F Schuster. Segmentation of lidar data using the tensor voting framework. *ISPRS*, 35(B3):1073–1078, July 2004. II.4.4

[SDK09] Ruwen Schnabel, Patrick Degener, and Reinhard Klein. Completion and reconstruction with primitive shapes. *Computer Graphics Forum*, 28(2):503–512, March 2009. II.1.3

[Ser83] Jean Serra. *Image analysis and mathematical morphology*. Academic Press, Inc., 1983. V.2.2

[SFC+11] J Shotton, A Fitzgibbon, M Cook, T Sharp, M Finocchio, R Moore, A Kipman, and A Blake. Real-time Human Pose Recognition in Parts

from Single Depth Images. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1297–1304. IEEE Computer Society, 2011. I.1.3

[Sha08] Ariel Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008. II.4.4

[SHKF12] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. *ECCV*, October 2012. II.4.6

[SHKZ14] Ling Shao, Jungong Han, Pushmeet Kohli, and Zhengyou Zhang, editors. *Computer vision and machine learning with RGB-D sensors*. Advances in Computer Vision and Pattern Recognition. Springer International Publishing, 2014. II.1.3

[SHT09] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. *Robotics Science and Systems*, 5, June 2009. II.2.2

[SIVA17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections On Learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. VI.2.1

[SJ13] Michael Schmeing and Xiaoyi Jiang. Color segmentation based depth image filtering. In X. Jiang, O. R. P. Bellon, D. Goldgof, and T. Oishi, editors, *Advances in Depth Image Analysis and Applications (WDIA 2012)*, volume 7854 of *Lecture Notes in Computer Science*, pages 68–77. Springer, Berlin, Heidelberg, 2013. II.1.3, II.3

[SMGKD14a] Renato F Salas-Moreno, Ben Glocken, Paul HJ Kelly, and Andrew J Davison. Dense planar slam. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 157–164. IEEE, September 2014. II.11, II.3.1

[SMGKD14b] Renato F Salas-Moreno, Ben Glocken, Paul HJ Kelly, and Andrew J Davison. Dense planar slam. *ISMAR*, September 2014. II.4.4

[Ste91] Richard S Stephens. Probabilistic approach to the hough transform. *Image and vision computing*, 9(1):66–71, 1991. II.4.3

[SWK07] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2):214–226, June 2007. II.15, II.4.1, II.4.2, II.4.4, III.2, IV.2, 20, IV.3, A.1.1

[SWWK08] Ruwen Schnabel, Raoul Wessel, Roland Wahl, and Reinhard Klein. Shape recognition in 3d point-clouds. *The 16-th International*

Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, 8, 2008. II.4.2

[SXN+18] Yifei Shi, Kai Xu, Matthias Nießner, Szymon Rusinkiewicz, and Thomas Funkhouser. PlaneMatch: Patch Coplanarity Prediction for Robust RGB-D Reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. II.2.4

[SXZ+12] Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics (TOG)*, 31(6):136, November 2012. II.4.4

[SZ07] Alexander Shpunt and Zeev Zalevsky. Depth-varying Light Fields for Three Dimensional Sensing, 2007. Granted patent US20080106746A1. I.1.3, II.1.1

[TGB13] Jean-Marc Thiery, Emilie Guy, and Tamy Boubekeur. Sphere-meshes: Shape approximation using spherical quadric error metrics. *ACM Transactions on Graphics, (Proc. of SIGGRAPH Asia)*, 32(6), November 2013. II.4.4, II.4.5

[TGBE16] Jean-Marc Thiery, Émilie Guy, Tamy Boubekeur, and Elmar Eisemann. Animated mesh approximation with sphere-meshes. *ACM Trans. Graph.*, 35(3):30:1–30:13, 2016. II.4.5

[TGRC13] Alexander JB Trevor, Suat Gedikli, Radu B Rusu, and Henrik I Christensen. Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration (SPME)*, May 2013. II.4.4

[TJRF13] Yasuhiro Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. Point-plane slam for hand-held 3d sensors. *Robotics and Automation (ICRA)*, pages 5182–5189, May 2013. II.15

[TM98] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision*, pages 839–846. IEEE, 1998. II.1.3, IV.2.1

[TPT16] Anastasia Tkach, Mark Pauly, and Andrea Tagliasacchi. Sphere-meshes for real-time hand modeling and tracking. *ACM Trans. Graph.*, 35(6):222:1–222:11, 2016. II.4.5

[TZ00] Philip HS Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, April 2000. II.4.2

[Ume91] Shinji Umeyama. Least-squares estimation of transformation

parameters between two point patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 13(4):376–380, 1991. II.2.1, II.3.1

[VHC08] Diego Viejo Hernando and Miguel Cazorla. 3D Model Based Map Building. In *IX Workshop de Agentes Fisicos (WAF'2008), Vigo, Spain*, September 2008. II.2.2

[VLA15] Yannick Verdie, Florent Lafarge, and Pierre Alliez. Lod generation for urban scenes. *ACM Transactions On Graphics (TOG)*, 2015. II.4.4

[Wal92] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992. V.2.5

[WGC99] Ross T Whitaker, Jens Gregor, and PF Chen. Indoor scene reconstruction from sets of noisy range image. *3-D Digital Imaging and Modeling (3DIM)*, pages 348–357, October 1999. II.4.4

[WK05] Jianhua Wu and Leif Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24(3):277–284, September 2005. II.15

[WO02] Jianning Wang and Manuel M Oliveira. Improved scene reconstruction from range images. *Computer Graphics Forum*, 21(3):521–530, September 2002. II.4.3

[WPM+12] Oliver J Woodford, Minh-Tri Pham, Atsuto Maki, Riccardo Gherardi, Frank Perbet, and Björn Stenger. Contraction moves for geometric model fitting. *ECCV*, pages 181–194, October 2012. II.4, II.15, II.4.4

[WPM+14] Oliver J Woodford, Minh-Tri Pham, Atsuto Maki, Frank Perbet, and Björn Stenger. Demisting the hough transform for 3d shape recognition and registration. *International Journal of Computer Vision*, 106(3):332–341, February 2014. II.4.3

[WSMG+16] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. ElasticFusion: Real-Time Dense SLAM and Light Source Estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016. II.3.2

[WZN+14] Chenglei Wu, Michael Zollhöfer, Matthias Nießner, Marc Stamminger, Shahram Izadi, and Christian Theobalt. Real-time shading-based

refinement for consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 33(6):200, 2014. II.1.3

[XKH+16] Kai Xu, Vladimir G Kim, Qixing Huang, Niloy Mitra, and Evangelos Kalogerakis. Data-driven shape analysis and processing. *SIGGRAPH ASIA 2016 Courses*, page 4, 2016. II.4.4

[XOT13] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. SUN3D: A Database of Big Spaces Reconstructed using SfM and Object Labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1625–1632, December 2013. II.2.3, III.4.2, III.9

[XWW18] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 466–481, 2018. I.1.3

[XZZ+11] Junhao Xiao, Jianhua Zhang, Jianwei Zhang, Houxiang Zhang, and Hans Petter Hildre. Fast plane detection for slam from noisy range images in both structured and unstructured environments. *Mechatronics and Automation (ICMA)*, pages 1768–1773, August 2011. II.4.4

[YKH10] Cem Yuksel, John Keyser, and Donald H. House. Mesh Colors. *ACM Transactions on Graphics*, 29(2):15:1–15:11, 2010. IV.3.3

[YWLY12] Dong-Ming Yan, Wenping Wang, Yang Liu, and Zhouwang Yang. Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design*, 44(11):1072–1082, November 2012. II.4.4, II.4.4, II.4.5

[YZ13a] José Alonso Ybanez Zepeda. Methods and Systems for Controlling a Virtual Interactive Surface and Interactive Display Systems, January 2013. Granted patent US9880728B2. I.1.3

[YZ13b] José Alonso Ybanez Zepeda. Virtual Sensor Systems and Methods, January 2013. Granted patent US9182812B2. I.1.3

[ZCC16] Edward Zhang, Michael F Cohen, and Brian Curless. Emptying, Refurnishing, and Relighting Indoor Spaces. *ACM Transactions on Graphics (TOG)*, 35(6):174, 2016. II.3.3, II.14

[ZPK16] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast Global Registration. In *European Conference on Computer Vision*, pages 766–782. Springer, 2016. II.2.3

[ZPK18] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847*, January 2018. II.2.3

[ZSG+18] Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt,

Matthias Nießner, Reinhard Klein, and Andreas Kolb. State of the Art on 3D Reconstruction with RGB-D Cameras. *Computer Graphics Forum (Proceedings of EUROGRAPHICS)*, 37(2):625–652, May 2018. II.3

[ZSN+17] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. In *CVPR*, 2017. II.4, II.2.1, III.4.2

[ZXTZ15] Yizhong Zhang, Weiwei Xu, Yiying Tong, and Kun Zhou. Online Structure Analysis for Real-time Indoor Scene Reconstruction. *ACM Transactions on Graphics (TOG)*, 34(5):159, November 2015. II.3.2, II.4.4

[ZYH+15] Yang Zhou, Kangxue Yin, Hui Huang, Hao Zhang, Minglun Gong, and Daniel Cohen-Or. Generalized cylinder decomposition. *ACM Transactions on Graphics (TOG)*, 34(6):171, November 2015. II.4.4

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Analyse de Scène Temps Réel pour l'Interaction 3D

Mots clés: analyse visuelle de scène, capteurs de profondeur, analyse géométrique, abstraction haut niveau

Résumé: Cette thèse porte sur l'analyse visuelle de scènes intérieures capturées par des caméras de profondeur dans le but de convertir leurs données en information de haut niveau sur la scène. Elle explore l'application d'outils d'analyse géométrique 3D à des données visuelles de profondeur en termes d'amélioration de qualité, de recalage et de consolidation. En particulier, elle vise à montrer comment l'abstraction de formes permet de générer des représentations légères pour une analyse rapide avec des besoins matériels faibles. Cette propriété est liée à notre objectif de concevoir des algorithmes adaptés à un fonctionnement embarqué en temps réel dans le cadre d'appareils portables, téléphones ou robots mobiles. Le contexte de cette thèse est l'exécution d'un procédé d'interaction 3D temps réel sur un appareil mobile. Cette exécution soulève plusieurs problématiques, dont le placement de zones d'interaction 3D par rapport à des objets environnants réels, le suivi de ces zones dans l'espace lorsque le capteur est déplacé ainsi qu'une utilisation claire et compréhensible du système par des utilisateurs non experts. Nous apportons des contributions vers la résolution de ces problèmes pour montrer comment l'abstraction géométrique de la scène permet une localisation rapide et robuste du capteur et une représentation efficace des données fournies ainsi que l'amélioration de leur qualité et leur consolidation. Bien que les formes géométriques simples ne contiennent pas autant d'information que les nuages de points denses ou les ensembles volumiques pour représenter les scènes observées, nous montrons qu'elles constituent une approximation acceptable et que leur légèreté leur donne un bon équilibre entre précision et performance.

Title: Real-Time Scene Analysis for 3D Interaction

Keywords: visual scene analysis, commodity depth sensors, geometric analysis, high level abstraction

Abstract: This PhD thesis focuses on the problem of visual scene analysis captured by commodity depth sensors to convert their data into high level understanding of the scene. It explores the use of 3D geometry analysis tools on visual depth data in terms of enhancement, registration and consolidation. In particular, we aim to show how shape abstraction can generate lightweight representations of the data for fast analysis with low hardware requirements. This last property is important as one of our goals is to design algorithms suitable for live embedded operation in e.g., wearable devices, smartphones or mobile robots. The context of this thesis is the live operation of 3D interaction on a mobile device, which raises numerous issues including placing 3D interaction zones with relation to real surrounding objects, tracking the interaction zones in space when the sensor moves and providing a meaningful and understandable experience to non-expert users. Towards solving these problems, we make contributions where scene abstraction leads to fast and robust sensor localization as well as efficient frame data representation, enhancement and consolidation. While simple geometric surface shapes are not as faithful as heavy point sets or volumes to represent observed scenes, we show that they are an acceptable approximation and their light weight makes them well balanced between accuracy and performance.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France