Learning surface elements for shape representation

Théo Deprelle

École des Ponts ParisTech

THÈSE DE DOCTORAT

de l'École des Ponts ParisTech

Apprentissage d'éléments de surface pour la représentation de formes

École doctorale N° 405, Mathématiques & sciences et technologies de l'information et de la communication

Informatique

Thèse préparée au sein du Laboratoire d'Informatique Gaspard Monge (LIGM), UMR 8049, équipe A3SI (Imagine, Ecole des Ponts)

Thèse soutenue le 7 Octobre 2022, par

Théo DEPRELLE

Composition du jury:

Pierre, Alliez Directeur de recherche, INRIA Sophia-Antipolis	*Rapporteur*
Jean-Emmanuel, Deschaud Chargé de recherche, MINES ParisTech	*Rapporteur*
Maria, Vakalopoulo Professeure, Centrale Supélec	*Examinatrice*
Maks, Ovsjanikov Professeur, École Polytechnique	*Examinateur*
Mathieu, AUBRY Chercheur, École des Ponts ParisTech	*Directeur de thèse*
Pascal, MONASSE Chercheur, École des Ponts ParisTech	*Co-Directeur de thèse*

Learning surface elements for shape representation

Théo DEPRELLE

A dissertation submitted for the degree of

Doctor of Philosophy from École des Ponts ParisTech

École doctorale n°405 MSTIC

Mathématiques & sciences et technologies de l'information et de la communication

Presented on **October 7th, 2022** to a committee consisting of :

Mathieu AUBRY	Senior Researcher, École des Ponts ParisTech	Supervisor
Pascal MONASSE	Senior Researcher, École des Ponts ParisTech	Supervisor
Pierre Alliez	Head of Science, INRIA Sophia-Antipolis	Reviewer
Jean-Emmanuel Deschaud	Senior Researcher, MINES ParisTech	Reviewer
Maria Vakalopoulo	Professor at Centrale Supélec	Examiner
Maks Ovsjanikov	Professor, École Polytechnique	Examiner

École des Ponts ParisTech

LIGM-IMAGINE

6, Av Blaise Pascal - Cité Descartes

Champs-sur-Marne

77455 Marne-la-Vallée cedex 2

France

Université Paris-Est Marne-la-Vallée

École Doctorale Paris-Est MSTIC

Département Études Doctorales

6, Av Blaise Pascal - Cité Descartes

Champs-sur-Marne

77454 Marne-la-Vallée cedex 2

France

Abstract

Decades of public and industrial research were motivated by having a realistic virtual representation of our world and algorithms able to understand its semantics. From the first CAD software to modern deep learning techniques such as recent neural radiance fields, researchers and engineers have perfected 3D algorithms.

Nowadays many of the technologies we are using daily are 3D aware. They can be found in our smart-phones, in our cars, at the hospital, and in many other places. Recent advances use neural network techniques to perform tasks ranging from shape recognition, shape generation, to shape representation. In this thesis, we focus on the latest and we aim to represent the surfaces of 3D shapes using deformable elements. Many works from recent years have inspired this thesis. The most important is AtlasNet, a neural network architecture that learns how to deform a set of square patches to reconstruct the surface of any given shape in a "papier maché" like approach. We chose to start this thesis from AtlasNet, building upon its design to extend the philosophy of this approach to new tasks such as element discovery and joint shape collection parameterization. Our work was motivated by a range of industrial 3D-related challenges such as generating 3D representation, texturing, and many more.

This thesis presents two main technical contributions. The first one is to represent a collection of objects by a set of joint deformable elements that we call *elementary structures*. We designed these elementary structures to learn and represent consistent and meaningful parts of the objects in the collection. They are similar to primitives. We propose two models of elementary structures : (i) *patch deformation* and (ii) *point leaning*. The first learns parametric surfaces that enable

us to generate a continuous surface representation of the collection of shapes. The second learns a parametric set of points not constrained by topology, allowing us to generate more complex and interpretable elementary structures. An adjustment module learns to position and deform the elementary structures to reconstruct any shape of the collection. Using the elementary structures and the adjustment modules, we can generate an accurate reconstruction that preserves correspondences between shapes. The shape of those elements can evolve during training from a simple initialization to more complex shapes that describe the semantic part of the collection. This reconstruction strategy enables us to understand the underlying parts that best describe the collection's objects. We see it as an analysis-by-synthesis strategy.

The second technical contribution of this thesis builds upon elementary structures to learn a parameterization of a single shape or a collection's shapes. We combine the best features of the point learning and patch deformation modules in order to build a method that generates continuous surfaces not constrained by topology. This representation is conceptually similar to an atlas, i.e., a set of compatible homeomorphisms between a 3D shape and a 2D domain. We created a pipeline whose components mimic the defining elements of atlases. Namely, we learn a 2D domain homeomorphic to the surface, a parameterization which maps it to the surface and a chart which maps on it the surface. We show that we can merge the continuous surface aspect of the patch deformation modules and the flexibility of the point learning modules by representing the 2D domain as a probability density function. Our pipeline can represent several shapes using a single joint 2D domain while producing a joint atlas representation of a collection's shapes. Our method could allow using existing parameterization techniques such as texture mapping, displacement, roughness, ambient occlusion, height maps and many more on a collection of shapes.

Both methods presented in this thesis improve the reconstruction quality as measured on benchmarks with shapes of our collection and improved the coherence of the reconstruction.

Keywords : Deep learning, neural network, differentiable geometry, atlas, shape parameterization, primitive.

Résumé

Plusieurs décennies de recherches publiques et industrielles ont été motivées par le fait de disposer d'une représentation virtuelle réaliste de notre monde et d'avoir des algorithmes capables d'en comprendre sa sémantique. Des premiers logiciels de CAO aux techniques modernes d'apprentissage profond telles que les récents champs neuronaux de radiance, chercheurs et ingénieurs se sont efforcés de perfectionner les algorithmes 3D.

Aujourd'hui, de nombreuses technologies que nous utilisons au quotidien sont basées sur la 3D. On les trouve dans nos smartphones, dans nos voitures, à l'hôpital et dans de nombreux autres endroits. Les progrès récents en la matiere utilisent des techniques de réseaux de neuronnes profonds pour effectuer des tâches allant de la reconnaissance de formes et la génération de formes à la représentation de formes. Dans cette thèse, nous nous concentrons sur cette dernière et nous cherchons à représenter la surface de formes 3D en utilisant des éléments déformables. De nombreux travaux de ces dernières années ont inspiré cette thèse. Le plus important est AtlasNet, une architecture de réseau de neurones profond qui apprend à déformer un ensemble de patchs carrés pour reconstruire la surface d'une forme donnée ; une approche similaire au "papier maché". Nous avons choisi de construire cette thèse à partir de ce travail. Nous nous sommes appuyé sur cette approche et nous avons étendu sa philosophie à de nouvelles tâches telles que la découverte d'éléments et la paramétrisation de collections de formes conjointes. Ces travaux ont été motivés par une série de défis technologiques liés à la 3D, tels que la génération de représentations 3D, le texturage de modèle 3D et bien d'autres encore.

La première contribution technique de cette thèse est de représenter une collection d'objets par un ensemble conjoint d'éléments déformables que nous appelons

structures élémentaires. Ces structures élémentaires sont conçues pour apprendre et représenter une partie cohérente et significative des objets de la collection. Elles peuvent être considérées comme des primitives des représentations existantes. Au cours de l'apprentissage, la forme de ces structures élémentaires peut évoluer d'une simple initialisation à des formes plus complexes qui décrivent la partie sémantique de la collection. Nous proposons deux modèles de structures élémentaires : (i) *module de déformation de patch* et (ii) *module d'apprentissage de points*. Le premier apprend des surfaces paramétriques qui nous permettent de générer une représentation surfacique continue de la collection de formes. Le second apprend un ensemble paramétrique de points qui ne sont pas contraints par la topologie, ce qui nous permet de générer des structures élémentaires plus complexes et interprétables. Un module d'ajustement apprend à positionner et déformer les structures élémentaires afin de reconstruire chaque forme de la collection. En utilisant les structures élémentaires et les modules d'ajustement à l'unisson, nous sommes capables de générer une reconstruction précise qui préserve les correspondances entre les formes. Initialement, cette stratégie de reconstruction nous permet de comprendre les parties sous-jacentes qui décrivent le mieux les objets de la collection, et peut être comprise comme une stratégie d'analyse par synthèse.

La deuxième contribution technique de cette thèse s'appuie sur des structures élémentaires pour apprendre une représentation en deux dimensions d'une forme unique ou d'une collection de formes. Nous combinons les meilleures caractéristiques des modules d'apprentissage de points et de déformation de patchs, c'est-à-dire que nous voulons une méthode qui génère des surfaces continues non contraintes par la topologie. Cette représentation est conceptuellement similaire à un atlas, c'est-à-dire un ensemble d'homéomorphisme avec raccords entre une forme 3D et un domaine 2D. Nous avons créé un pipeline dont les composants imitent les éléments de définition des atlas, à savoir, pour une surface donnée, un domaine 2D homéomorphe

à la surface, une paramétrisation qui fait correspondre le domaine 2D à la surface et une carte qui fait correspondre la surface au domaine 2D. Nous montrons que nous pouvons fusionner l'aspect surfacique continu des modules de déformation de patchs et la flexibilité des modules d'apprentissage de points en représentant le domaine 2D comme une fonction de densité de probabilité. Notre pipeline peut représenter plusieurs formes à l'aide d'un seul domaine 2D conjoint, produisant ainsi une représentation atlasique conjointe d'une collection de formes. Puisque la méthode préserve les correspondances entre les formes de la collection, notre méthode pourrait permettre d'utiliser des techniques de paramétrisation existantes simultanément sur toute la collection. CCelles-ci étant, par exemple, l'application de texture, ou de détail, la rugosité, l'occlusion ambiante et bien d'autres.

Mot clés : Apprentissage profond, réseau de neurone, géométrie différenciable, atlas, paramétrisation de formes, primitives.

Contents

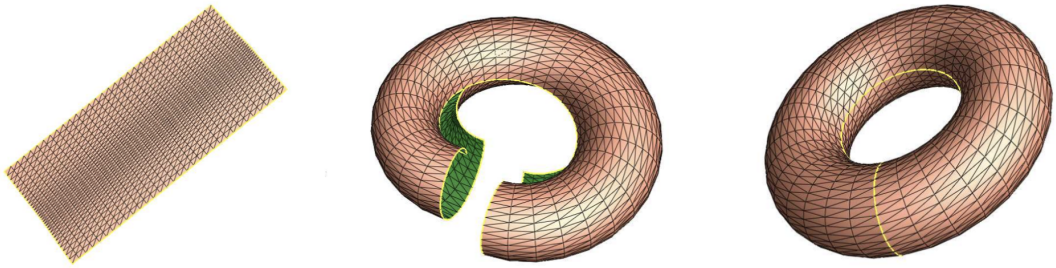
CHAPTER 1

Introduction

1.1 Goals

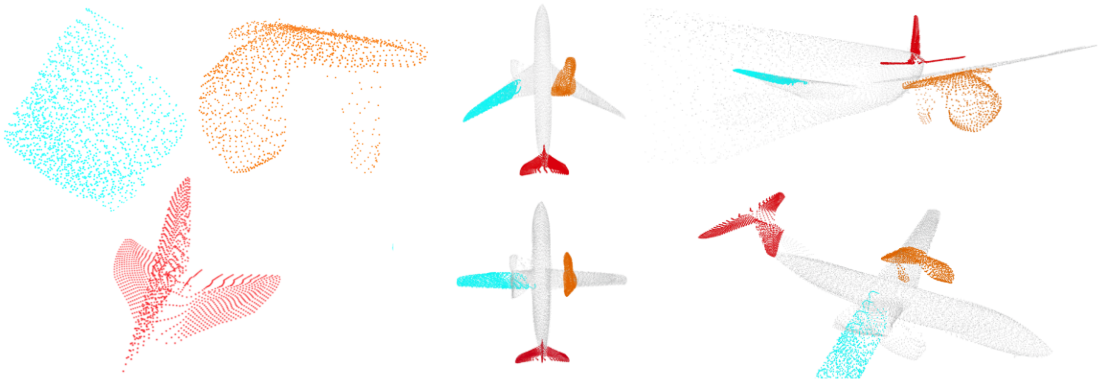
The goal of this thesis is **(i)** to improve the accuracy and the consistency of deep learning techniques for surface parameterization; **(ii)** to develop more interpretable representations of object surfaces.

Improving the accuracy and consistency of deep surface parameterization. For decades, the research community has proposed various methods of shape parameterization, i,e., mapping a 2D domain to the surface of a shape, see Figure 1.1a. With the recent democratization of neural networks, many works propose new approaches relying on deep learning techniques to produce surface parameterizations [Groueix et al., 2018b, Yang et al., 2018a, Bednarik et al., 2020]. Such deep learning methods have the advantage of parameterizing several shapes simultaneously. Consistency of parameterizations is one of the main aspects of these works. A consistent mapping is indeed able to generate the correspondences between shape surfaces.



2D domain *Intermediate mapped surface* *Final surface reconstruction*

(a) **Example of a parameterization of a torus.**. [Gu, 2003].



Interpretable elements *Reconstruction with the elements*

(b) **Interpretable surface elements.** See Chapter 3 for more details.

Figure 1.1: **Goals.** The goal of this thesis is to learn accurate surface parameterization using a set of interpretable learned deformable elements

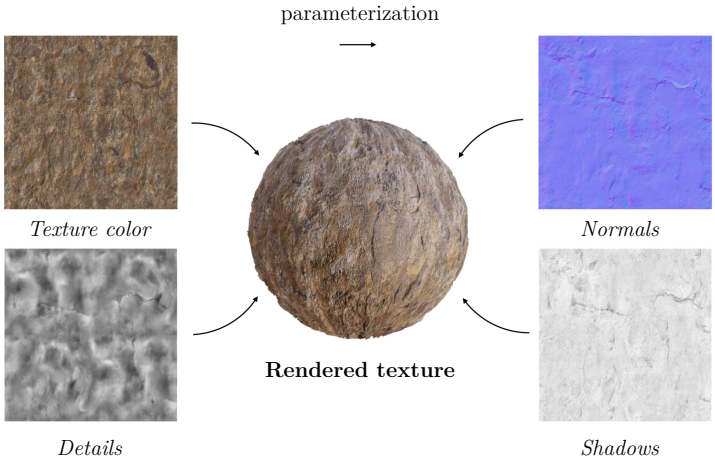
With correspondence between two parameterized shape surfaces, any point of the 2D domain maps to the corresponding points of the surfaces. In this thesis, we aim to build upon such methods. We focused our work in particular on AtlasNet [Groueix et al., 2018b] to generate more accurate and consistent shape parameterizations. AtlasNet is a neural network that produces a parameterization of any given shape surface by a set of square patches. We present a detailed description of AtlasNet in Section 2.4.3. It generates accurate results but has several downsides, mainly the consistency of the mappings and design restrictions. Leveraging the ability

of neural networks to learn the shape of our elementary surfaces, we propose to replace parts of the design of AtlasNet with new modules to improve accuracy and correspondences.

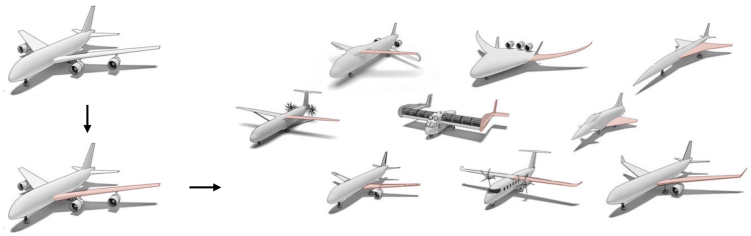
Interpretable surface elements. We also aim at generating interpretable surfaces elements for representing shape collections. Figure 1.1b illustrates this idea. Many works nowadays use neural networks and complex architectures to solve problems and focus mainly on improving performances on different metrics and benchmarks. Often, increasing performances come at the cost of over-complicated architecture or design choice lacking interpretability. Having an interpretable representation is a general and important challenge for deep learning approaches beyond 3D data. While having an accurate representation is one of our goal, we believe that for specific tasks, like the ones tackled in this thesis, having an interpretable design is desirable. To that end, one objectif of this thesis is in to developed accurate and especially interpretable approaches.

1.2 Motivations

Since the beginning of the digitization of our world, professionals and day-to-day users have been thriving on 3D data. Many artists have shifted their previous analog tasks to digital 3D ones. The first animated movie, *Fantasmagorie* by Emile Cohl (1908), was drawn by hand one frame at a time. Today CGI (Computer Graphics Imagery) artists generate entire films or video games using 3D graphics software. Other professionals such as engineers or architects went from handcrafting wooden or clay prototypes to 3D modeling software. The academic and industrial research on 3D geometry created the bedrock of many industrial processes. In recent years, the democratization of 3D software like Unity/Unreal or Blender enables anyone to work on complex 3D digital data. In this context, the thesis is motivated by several industrial applications in the creative industries and other aspects of our digital world.



(a) **Rendering**



(b) **Transferring details**

(c) **Discovering elements**

Figure 1.2: **Motivations**, The motivations of this thesis range from parameterization techniques for rendering, transferring details from a shape to a collection, and discovering recurrent elements in an unlabeled 3D scene. 1.2a textured from [Kat, 2022], 1.2b airplanes from Shapenet [Chang et al., 2015a] to 1.2c scene from historical data from [Ico, 2022].

CGI artist tools for creative industry: A 3D artist aims to reconstruct aspects of our world within computer software. From natural images to sketches, they use inspirations combined with their knowledge of the tools to generate animation, photo-realistic reconstruction, video game characters, and many more. One of the main processes is the 3D model generation depends on the style, historical context, or realism. To that end, the artist uses shape parameterization in many steps like the geometry, the details refinement, or the texturing process before rendering (cf Figure 1.2a).

Editing multiple 3D objects at the same time Some CGI applications in film or video games industries handle hundreds of similar 3D models during specific tasks such as background generation or crowded rendering. The most common practice is to reuse and copy-past a few dozens of distinct models since it would take simply too much time to process every single 3D model individually. In this thesis, we aim at developing an approach that would allow a collection of 3D models to be represented by a single joint parameterization. This could enable a scenario where artists use techniques like the ones seen in Figure 1.2a (texture mapping, normal mapping, shadows) to edit entire collection of 3D models instantly and without having to edit every model independently (see Figure 1.2b).

Keeping a virtual trace of our word: With the democratization of drones and remote sensors over the last years, scanning historical monuments at any given time have become possible. Because of the uncertainty of the coming decades (global warming, wars, disasters), we need more digital representations of our world. During this thesis, the architects in charge of rebuilding Notre Dame de Paris after its destruction in 2019 used such scans. Another French company, ICONEM, was mandated by UNESCO to scan historical and architectural monuments for preservation purposes. They produced, for instance, several detailed reconstructions of monuments in Alep, Syria, that were destroyed by ISIS soon after. Today these

scans are the only traces we have left and will be used to preserve historical heritage for the next generations. Such scans contain billions of unlabeled data points. Many historians spend a significant amount of time understanding the scans. We could imagine using the methods developed in this thesis to discover recurrent shape elements across the scans, similar to a clustering technique (cf Figure 1.2c) that could help historians or other professionals studying these scenes.

1.3 Context

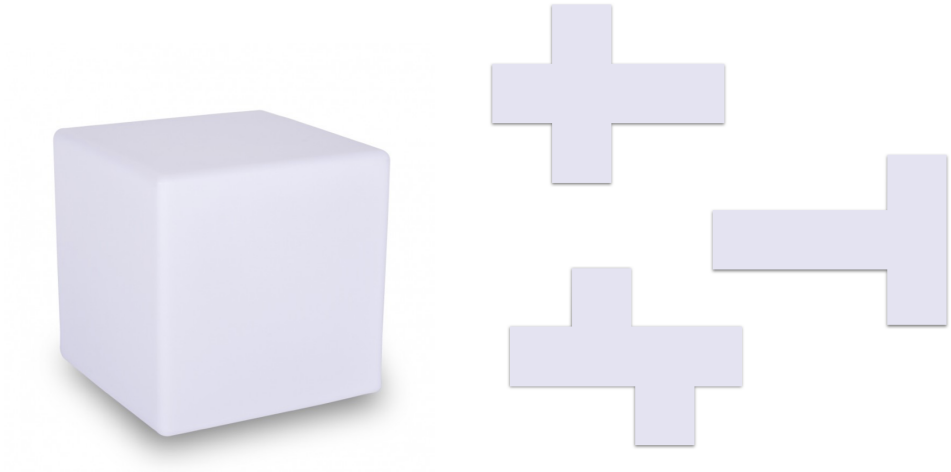
The development of parameterization techniques over the years solved many issues in the creative industry. For decades the CGI artist has used such techniques for several tasks ranging from generating 3D models, editing details, texturing, and rendering while creating films or video games. With the rise of 3D data over the last decade, more and more professionals have faced issues initially related to the creative industry. Thus, parameterization techniques have become ever more popular. Today, with the democratization of deep learning approaches and 3D computer vision algorithms, professionals have new powerful tools that enable them to solve new problems. Recently, many approaches have leveraged deep learning to solve parameterization problems. With the new methods, handling a collection of shapes have become easier. We argue that the most important of these methods is AtlasNet by Groueix et al. [Groueix et al., 2018b]. This method uses a set of parametric functions represented by neural networks aiming to deform a set of 2D dimensional surface elements to reconstruct the surface of a given shape. Designed for single-view reconstruction, meaning generating a 3D representation from an input image, AtlasNet introduced deep learning for the task of parameterization of shape surfaces to a pure 3D pipeline like the ones developed in this thesis.

1.4 Challenges

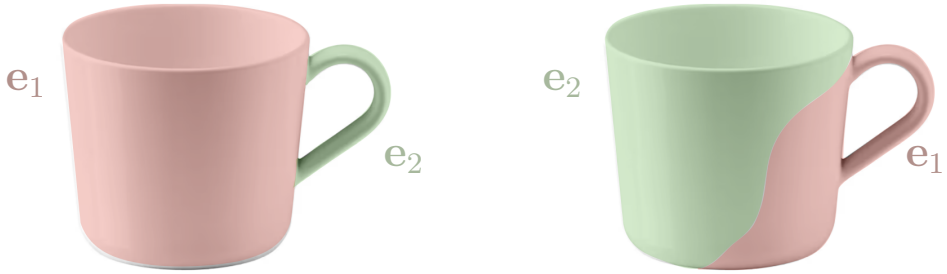
In this section we discuss the main challenges we faced during this thesis, namely the lack of annotated data and difficulty of making neural network approaches interpretable.

Lack of annotated training dataset The simplest approach to a lot of Deep Learning problems is supervised learning. In this case, we have a vast collection of data and all the ground truth annotations to train a network. Given an input, we know without ambiguity what the results should be. In this thesis, it was not the case. Thus, the concept of ground truth itself is not well defined in our case since there is no clear solution for most of the problems we want to solve. For instance in chapter 4, we want to learn a parameterization of a surface's shape. The example in Figure 1.3a illustrates the ambiguity of the solutions to this problem and the fact that there are multiple "*correct*" parameterizations of a surface shape. In Chapter 3 we aim to reconstruct an object using deformable surface elements. In this case there are again several possible solutions. The issue is illustrated in Figure 1.3b. Hence for the parts of our design where we cannot define a ground truth, we cannot have supervision. The methods developed in this thesis rely on point coordinates samples on the shape surface. But to train our method, we needed specific data annotations such as normal coordinate or correspondence ground truth points to improve the consistency and reconstruction accuracy. Thus, we needed benchmarks of the 3D model with annotations which are challenging to get since such annotations are hard to generate automatically and often require human supervision (cf Figure 1.3c).

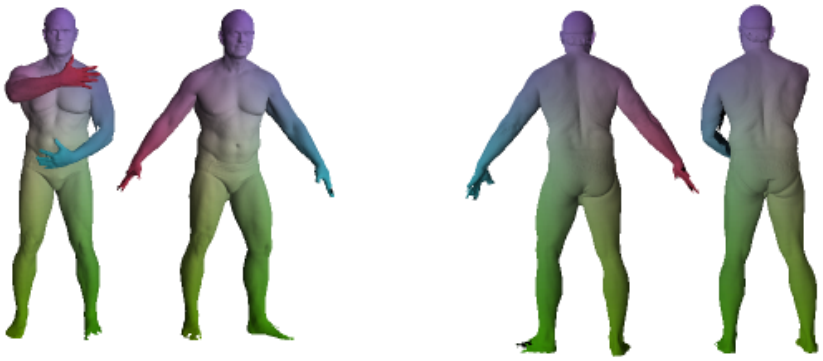
Making deep learning interpretable. If one has the choice between an interpretable model and another better in terms of performance while being much more complex and cryptic to understand, the interpretable model is often the better solution. In particular, we cannot simply trust a complex system that we do not



(a) Three different 2 domains that can be used to parameterize a cube



(b) Two decompositions of a mug using two elements e_1 and e_2



(c) Correspondence annotation [Bogo et al., 2014]

Figure 1.3: **Lack of annotated training data**. In this thesis we face lack of annotation, either because there are multiple correct solutions and annotations or because the annotation required manual annotation.

truly understand to design drugs or control the autopilot of our cars and airplanes. There is a reason why we are using a proof management system for a critical aspect of industrial software instead of using a deep learning approach. Knowing this, we believe further research towards interpretability will improve the level of trust regarding deep learning and the process that leads to a result. To train a neural network, one needs to define a set of design constraints such as the energy function, the data types fed to the pipeline, and several hyperparameters like the number of layers. Those choices are usually hand-made carefully by experienced researchers and engineers. They define how the network performs. For every task, there is a set of optimal variables. Often the more complex the design and variables are, the better the model performs. Having an interpretable deep learning model allows us to understand better the processes and details of the neural network architecture. For some applications, it is critical to understand how a model produces a given result. The main issue regarding interpretability is that it is not natural for a neural network. They often represent the data in a parameter space too complex for any of us to understand. When a feature space of more than a thousand feature elements describes an input, it is hard to understand which feature corresponds to what and how one feature element impacts the results. There have been many works trying to understand feature representation or make them more interpretable. Many generative neural networks approach like [Karras et al., 2017] aimed to control the generation process of random faces by a set of features, but it is not clear that any of them influences a specific characteristic of generated faces such as hair color or face geometry and gender (cf Figure 1.4).

1.5 Contributions

In this section we list the main contributions of this thesis. First we proposed novel approaches to learn a shape representation using a set of deformable elements.

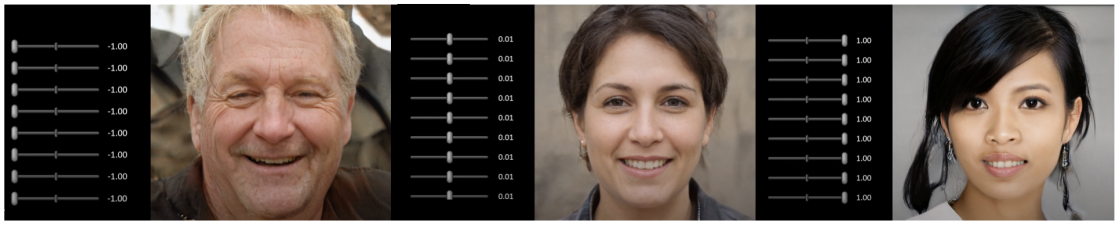


Figure 1.4: **Interpretability of the features**, each slider controls the generative process of faces but they are not interpretable [Karras et al., 2017]

Second, we propose various methods to control the deformations in order to improve the intepretability of the representations.

Shape representation using deformable surfacic elements. In this thesis we propose three representations of deformable elements. Table 1.1 summarize the following paragraph and Figure 1.5 illustrates the three different approaches.

(i) Our patch deformation approach learns parametric surfaces that enable us to generate continuous surface shapes. It maps a fixed 2D domain, a two dimensional unit square, to a more complex surface. This mapping is modeled as a neural network which takes as input the coordinates of the point samples on the 2D domain and generates a set coordinates. This method is restricted by the topology of the initial 2D domain meaning that we can only learn surface shape elements homeomorphic to a plane.

(ii) Our point learning approach learns a set of points with no notion of topology or surface hence allowing us to generate more complex elementary structures. The points are initialized randomly on a two dimensional unit square and we treat their spatial coordinate as network parameters which are optimised during the training. They overall yield more accurate results than the patch deformation modules but the surfacic aspect of the elementary structures is lost with this approach.

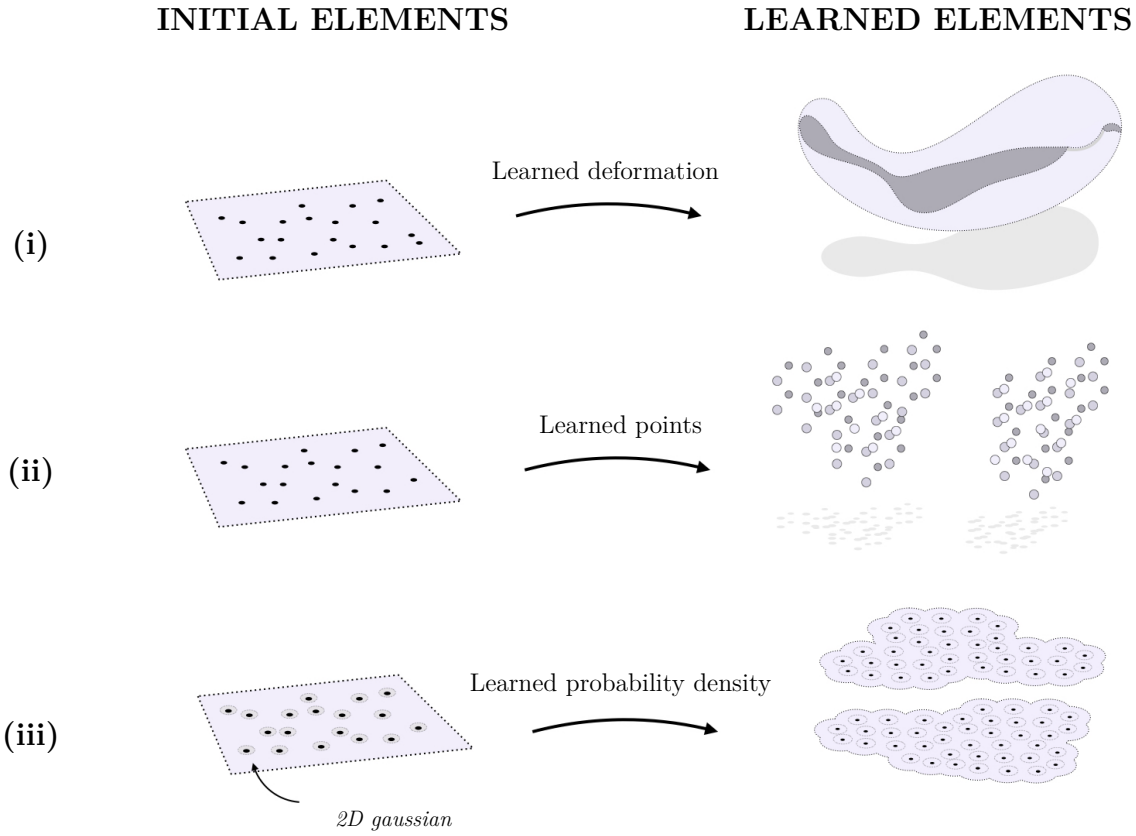


Figure 1.5: **Deformable surfacic elements.** **(i)** patch deformation - **(ii)** point learning - **(iii)** probability distribution function

	Flexible topology	Surfacic representation
Patch deformation	no	yes
Point learning	yes	no
Probability density	yes	yes

Table 1.1: **Deformable elements representation recapitulation.**

(iii) Our probability distribution function (PDF) learns a set of surface element in dimension two by composing a set of Gaussians with a learnable two dimensional mean and a fixed standard deviation. Like in the point learning approach, the means of the Gaussians are treated as parameters of the network and are optimized. By thresholding the PDF we can generate a set of dimension two surface elements with learned topology. This combines the best of the patch deformation and point learning approaches since this approach enables

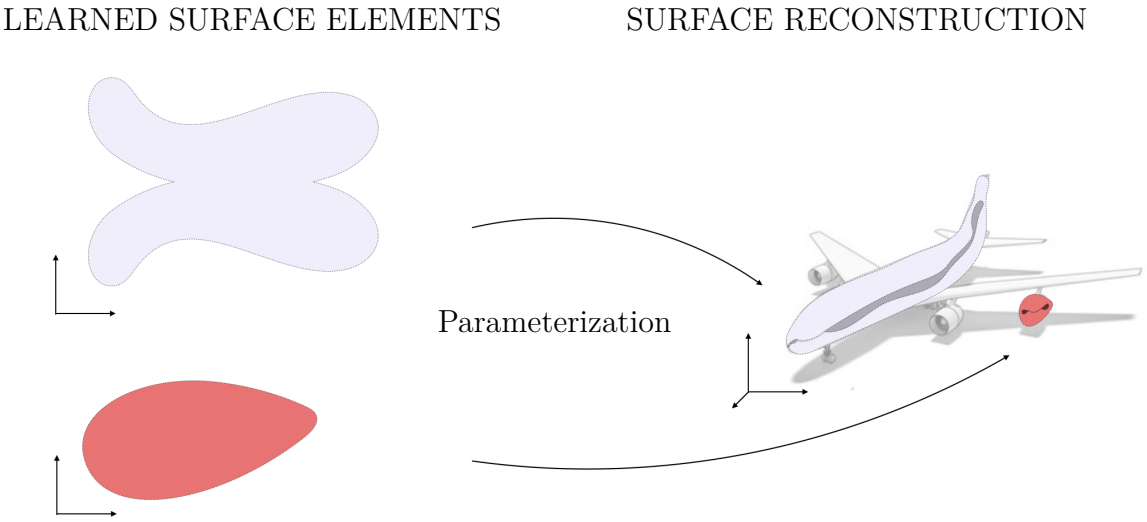


Figure 1.6: **The adjustment modules position the elementary shape surfaces.**

us to have a surfacic representation with no topological constraints while being limited in the number of Gaussians.

Regularisation of the deformable elements shapes In this thesis, we train our neural network architectures to learn surface elements while generating a representation of shape surfaces. In order to learn meaningful and interpretable elements, we added additional constraints to their shapes. These constraints are applied mainly to the parameterization modules that map the learned surface elements onto the target shape surface.

(i) We use a set of linear functions to parameterize our learned surface elements. Their weights are predicted using an MLP which only takes as input a feature representation of the target shape surfaces generally extracted using a PointNet module [Qi et al., 2017a, Qi et al., 2017b] and generates a 3x3 coefficient matrix and 1x3 intercept. We then use the set of the learned linear functions to map the surface elements onto the target surface. They enable us to learn more interpretable and meaningful surface elements while having the lowest reconstruction accuracy of the methods we propose.

(ii) We also use a set of non-linear parametric functions represented using an MLP which takes as input a set of points sampled on our elementary surfaces and feature representation similar to the linear approaches to predict a set of point coordinates on the target surface. Overall, the MLP yields a better reconstruction metric-wise but learned surfaces element lack interpretability. To solve this issue, we can add additional constraints, such as normal coordinates and additional regularization of the weight of the MLP making the parametric functions as much isometric as possible. This allows us to control the shape of the 2D domain by forcing two subsets of the target surface with opposite normals and similar spatial coordinates to be parameterized by two distinct regions of the 2D domain.

1.6 Thesis outline

This thesis is organised as follows:

Chapter 2 In Chapter 2 we cover the background of this thesis by providing an overview of deep learning, then of the different shape representations, and finally we focus on prior methods performing shape reconstruction and shape paramerization using atlases and primitive.

Chapter 3 This chapter introduces our new shape representation based on deformable primitives that we called *elementary structures*. We first explain how our network learns the shape of those deformable elements. We then explain how it adjusts them with respect to each other in order to reconstruct shapes of the collection on which it it is trained. We empirically compare our representation with other 3D representations on the tasks of shape reconstruction on the ShapeNet [Chang et al., 2015a] and SURREAL [Varol et al., 2017] benchmark and on the task of predicting shape correspondences on the FAUST benchmark [Bogo et al., 2014].

In particular, we show that elementary structures yield better reconstruction and correspondences while learning interpretable elements describing categories they has been trained on.

Chapter 4 In this chapter we develop the idea of generating joint surface atlases of a collection of shapes. We first explain how we model the three constituent elements of the atlas: a chart-mapping, a parameterization and a two dimensional domain. We then explain how these elements can be learned jointly for a collection of shapes and how such modelization is able to answer the limitations of the *elementary structures*. We perform a qualitative and quantitative ablation study of the different training parameters and features and empirically compare our representation with prior work on the task of shape reconstruction on the SHREC [Giorgi et al., 2007] benchmarks and the shapes from [Williams et al., 2019]. Especially, we show that our joint atlas representation yeilds better correspondance results while being much closer than the previous work to be a proper atlas, i.e., one that defines an homeomorphism between 3D shape and a 2D domain.

Chapter 5 This chapter summarizes the contributions of the thesis and suggests some directions of future work.

1.7 Publications list

The work presented in this thesis have been described in two papers:

- Learning elementary structures for 3d shape generation and matching - **Theo Deprelle**, Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. *Advances in Neural Information Processing Systems 32* (NeurIPS 2019)

- Learning Joint Surface Atlases - **Théo Deprelle**, Thibault Groueix, Noam Aigerman, Vladimir G. Kim, Mathieu Aubry. *arXiv preprint arXiv:2206.06273* (2022)

The source code are available:

- http://imagine.enpc.fr/deprellt/atlasnet2

- https://imagine.enpc.fr/deprellt/joint-surface/

CHAPTER 2

Related work

2.1 Deep Learning

A detailed presentation of deep learning is out of the scope of this thesis. Instead, in this section, we give a short historical perspective of the technique and then list the key elements used across this thesis.

2.1.1 Historical perspective

Neural network theory emerged in the 50s with the first models such as the Perceptron from Rosenblatt [Rosenblatt, 1958]. From the 50s onwards, the scientific community set up the pillars on which the entire deep learning community relies on today; from the first version of continuous backpropagation introduced by Henry J. Kelley [Kelley, 1960] to the original deep convolutional neural network (CNN) architecture brought in by Fukushima [Fukushima, 1988] to finally the first practical demonstration of backpropagation on an MLP developed by Lecun [LeCun et al., 1998]. Before the 2010s, the community faced technical limitations and lacked computing power.

Neural networks are complex systems relying on reasonably simple and parallelizable operations. With the rise of the Graphics Processing Unit (GPU), chips designed to handle lots of parallelizable numerical operations, such technical limitations have dramaticly reduced. In 2012, Krizhevsky et al. [Krizhevsky et al., 2012] were the first to leverage the parallelizable computing power of the GPUs to train a neural network. Then open source libraries such as Torch, Caffe, Keras, Lua, Pytorch [Paszke et al., 2019], or TensorFlow [Abadi et al., 2015] were developed and simplified the development of the process.

2.1.2 Data collection

In deep learning, the most popular paradigm is to train the model on a large amount of data, such as the classification dataset ImageNet [Deng et al., 2009], to generalize on an unseen test dataset. Some of the approaches developed in this thesis follow this paradigm. For instance, we trained our models on a large scale 3D data such as ShapeNet [Chang et al., 2015a] and SURREAL [Varol et al., 2017] and then evaluate on unseen objects. In most of this thesis, we chose a different paradigm. We use neural networks in a pure optimization setup where we are not interested in the generalization of the model anymore but rather in the performances of the model on a given smaller dataset.

2.1.3 Loss function

Given a dataset, we define a loss function as a metric measuring how well the model performs. With shape generation, for example, the loss function will measure how far the generated shape is from the actual target. Most loss functions used in our works measure the distance between two sets of coordinates. If we know the ground truth in advance, we used supervised loss functions like the L2 norm on the

coordinates. On the contrary, where the ground truth is not well defined, we work with unsupervised loss functions such as the Chamfer distance or the Earth Mover Distance on the coordinates.

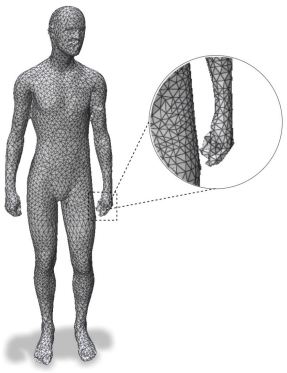
2.1.4 Neural networks

The neural networks in this thesis are Multi layer Perceptrons (MLP), modeled after the first preceptron from Rosenblatt [Rosenblatt, 1958]. An MLP alternates linear and non-parameterized non-linear operations. We optimize the parameters of the linear layer to minimize the loss, and we mainly use the rectified linear activation unit (ReLU) for the non-parameterized non-linear ones. We also include several other modules in the network parameters list. The most important are the parameters we introduced to define our learnable surface elements. We also add the shape codes of the auto-decoder modules [Kingma and Welling, 2013] of the architecture in Chapter 4.

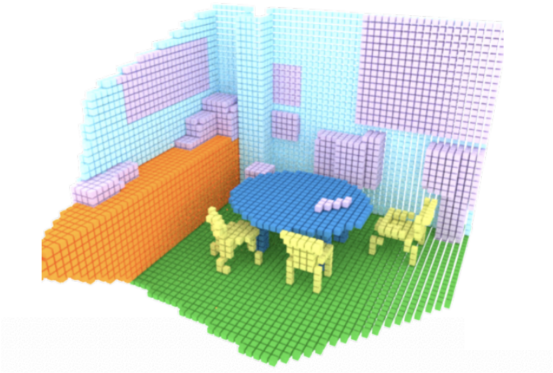
2.1.5 Optimization

A neural network can include millions of parameters. They are typically randomly initialized. The losses are non-convex functions of the parameters. To minimize the loss, they are thus optimized with variant stochastic gradient descent approaches [Robbins and Monro, 1951] such as RSMprop [Tieleman et al., 2012], and Adam [Kingma and Ba, 2014]. In this thesis, we use the implementation of the Adam optimizer from the Pytorch library [Paszke et al., 2019]. This optimization is often referred to as the training of the network or the learning of its parameters.

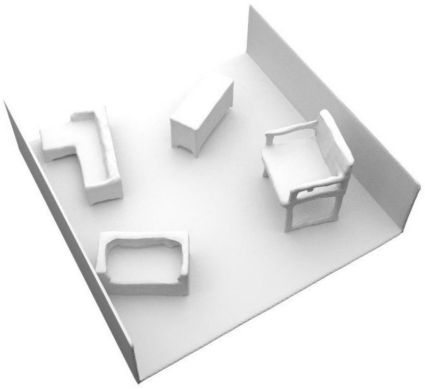
(a) **Point cloud.** [Liu et al., 2020]



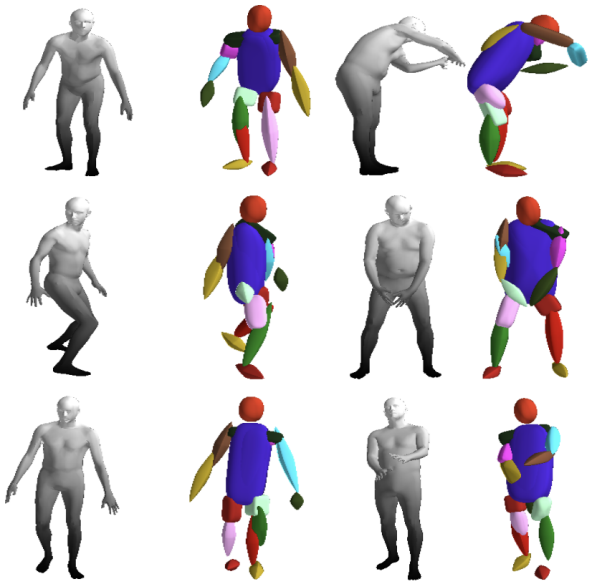
(b) **Mesh.** [Potamias et al., 2022]



(c) **Voxels.** [Song et al., 2017]



(d) **Implicit functions.** [Peng et al., 2020]



(e) **Primitives.** [Paschalidou et al., 2019a]

Figure 2.1: **3D shape representations**. The five shape representation the most related to this thesis are the surface representation (point cloud and mesh), the volumetric representations (voxels and implicit functions) and the primitive representations.

2.2 3-dimensional shape representations

In this section, we discuss one of the main questions of this thesis: *How should we represent 3D data?* We will give a brief overview of volumetric, surface, and primitive-based representations. These representations are the most related to our work. Examples of different representations can be found in Figure 2.1

2.2.1 Volumetric representation

Voxels. The equivalent of the image pixel in 3D is called voxel (cf Figure 2.1c). While the pixels in two-dimension correspond to a 2D grid, in 3D, we can use a grid where each cell, called a voxel, contains volumetric information about the shape. Voxels can represent shapes, signed distance function, occupancy, material, and many more. Storing the values of the voxels requires a lot of memory. The octree representation, introduced in 1982 by Meagher [Meagher, 1982] tries to solve this issue by merging voxels into larger ones. The voxel representation has been used in medical application [Ashburner and Friston, 2000, Norman et al., 2006, Ashburner and Friston, 2005], object detection [Maturana and Scherer, 2015], and shape reconstruction [Wang et al., 2017, Riegler et al., 2017]. Recent methods use neural network with voxels e.g. [Maturana and Scherer, 2015, Wang et al., 2017] and octree e.g. [Riegler et al., 2017, Tatarchenko et al., 2017, Häne et al., 2017] representation.

Implicit function Implicit functions are another volumetric description of 3D shapes (cf Figure 2.1d). Where voxels are a discrete volumic representation, implicit functions are continuous. They use a function f that can be defined for every point $x \in \mathbb{R}^3$. In most of the cases it determines occupancy, that is whether x is inside or outside of a shape. It describes the concept of occupancy. Given Ω a subset of $\mathbb{R}^3$ and a distance metric d, the signed distance function can be defined by :

$$f(x) = \begin{cases} d(x, \partial\Omega) & \text{if} \quad x \in \Omega \\ -d(x, \partial\Omega) & \text{if} \quad x \in \Omega^c \end{cases}, \tag{2.1}$$

where $\partial\Omega$ denotes the boundary of Ω and $d(x, \partial\Omega) = \inf_{y \in \partial\Omega} d(x, y)$ for any $x \in \mathbb{R}^3$ with inf denoting the infimum. Recent work uses signed distance function for shape reconstruction [Park et al., 2019, Mescheder et al., 2019] or 3D rendering [Mildenhall et al., 2020].

2.2.2 Surfacic representation

Point clouds Many acquisition sensors produce a set of points to represent a surface. For example, LiDAR uses lasers which emit a pulse of light that will bounce off surrounding object surfaces and return to the sensor. In this case, a discrete set of 3D coordinates represents a scanned surface (cf Figure 2.1a). Large real-world datasets thus often use this representation. For example, in ScanNet [Dai et al., 2017] Dai et al. used infrared scanners to generate a large dataset of point clouds of indoor scenes. Additional information such as color [Luo et al., 2015], normals [Serafin and Grisetti, 2015] and semantic [Landrieu and Simonovsky, 2018] can also be associated with the data points. PointNet [Qi et al., 2017a, Qi et al., 2017b] introduced a deep learning-based method to extract and condense information from a point cloud into a feature vector that can be used as the output of many other neural networks. Recent works use point clouds for task ranging from scene reconstruction [Fan et al., 2017, Wang et al., 2019], shape correspondences [Groueix et al., 2018c], semantic segmentation [Hackel et al., 2017], etc.

Meshes The continuous aspect of a surface is lost with a descrete point cloud. To better represent a surface, one can use a mesh (cf Figure 2.1b). Meshes are constituted of a point cloud (*vertices*), a set of connections (*edges*) between the

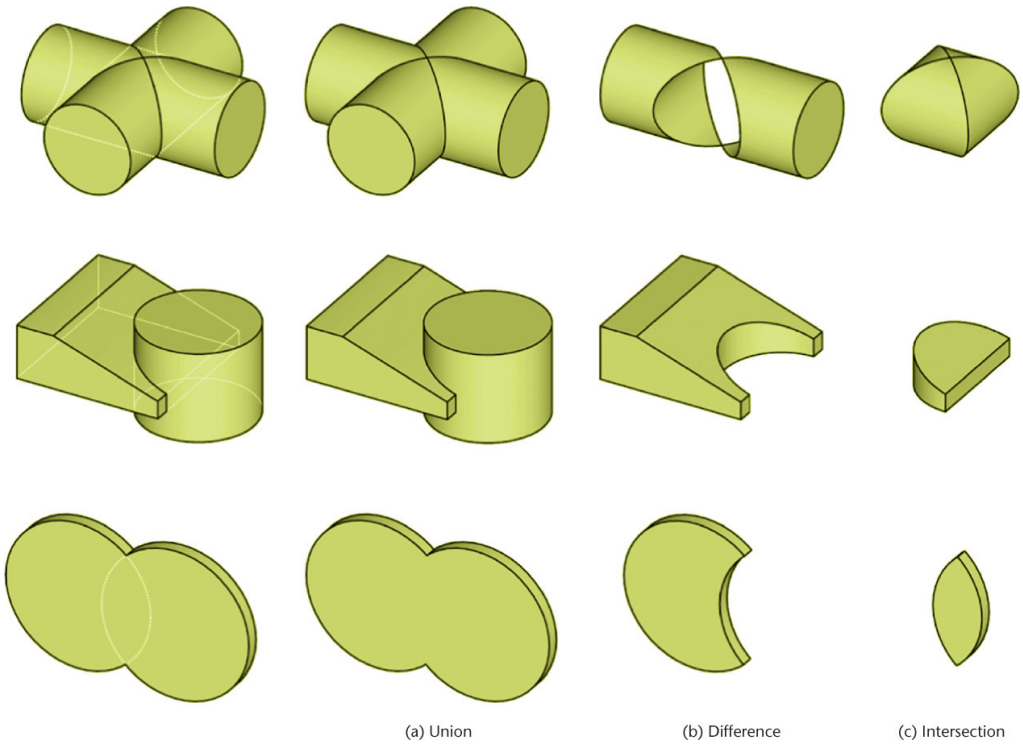


Figure 2.2: **CAD model generation**. Many CAD model are generated using a combination of simple shapes often referred to as primitives. [Giesecke et al., 2016]

points, and a set of polygons (the *faces*), often triangles, formed by three vertices. They are vastly used in the creative industry by the CGI artist, and many research have been done on this subject. Recent work like ShapeNet [Chang et al., 2015a] or ABC [Koch et al., 2019] built large-scale mesh datasets that are used daily by the community. Most recent point-cloud-based methods used the vertices of a mesh and are able to generate meshes at inference [Groueix et al., 2018b, Park et al., 2019]. Other applications use parametric mesh surface to represent an object. This, as well as several applications such as parameterization and UV-mapping, will be discussed in details in details in Section 2.4.

2.3 Representing shapes with primitives

With the democratization of CAD models, many professionals started to generate 3D shapes by combining simple primitive shapes. Figure 2.2 illustrates this process.

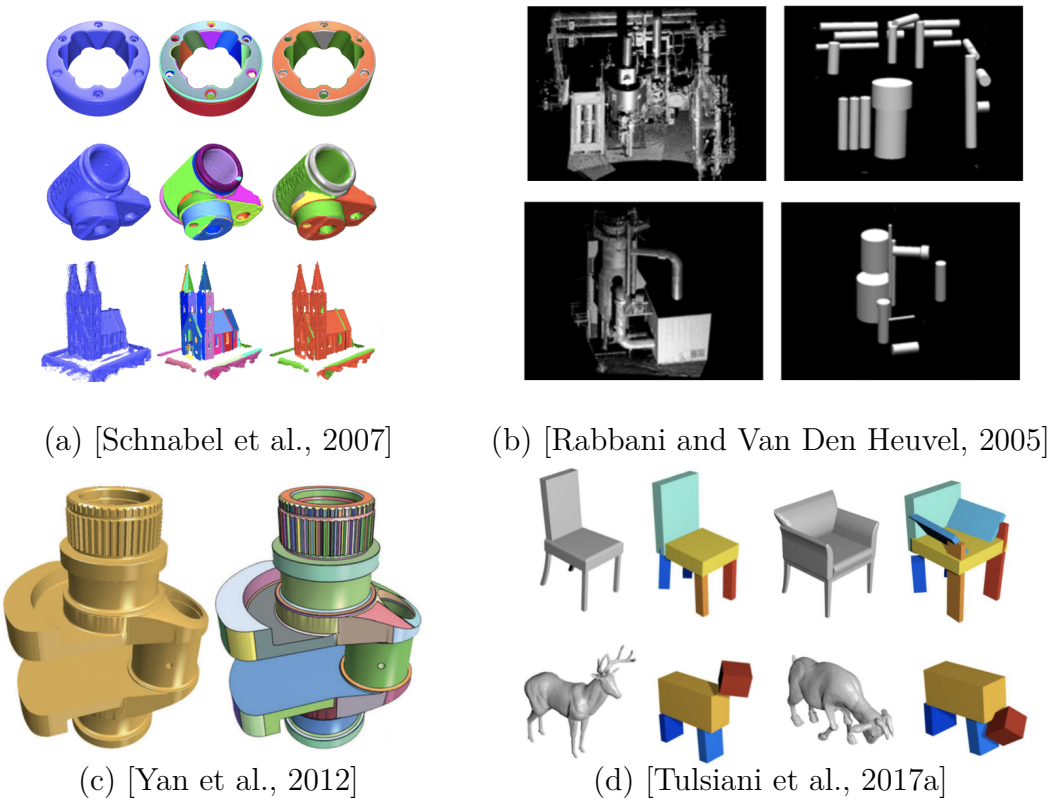
Yet, this representation is not new and was one of the oldest computer vision tasks. The primitive representation has several advantages. The most important one is interpretability. They were introduced by the "block world" hypothesis of [Roberts, 1963]. Primitives are a simplified, non-realistic, and interpretable representation of the world. Recently, many works have tried to fit primitives into other representations. In this section, we will list four different approaches for primitives fitting. We will discuss stochastic approaches, Hough-like voting methods, clustering methods, and recent neural network approaches. Examples of those methods can be found in Figure 2.3.

2.3.1 Stochastic approaches.

Several approaches use stochastic methods to estimate the position and parameters of the primitives model. Most of them are consensus-based and rely heavily on the RANSAC algorithm introduced by Fishler et al. [Fischler and Bolles, 1981]. Other variants such as [Matas and Chum, 2004, Rousseeuw, 1984, Torr and Zisserman, 2000, Chum and Matas, 2005] were developed over the years. For example Schnabel et al. [Schnabel et al., 2007] (see Figure 2.3a) detects planes, spheres, cylinders, cones and tori given a point cloud with normal annotations.

2.3.2 Hough-like voting approaches.

Other methods represent the primitives in a parameter spaces and use Hough-like voting methods to detect them. Initially designed for line detection in 1962 by Hough [Hough, 1962], the method was extended to more complex primitives in 2D and 3D by Barallard in 1981 [Ballard, 1981] or cylinder by Rabbani et al. [Rabbani and Van Den Heuvel, 2005] (see Figure 2.3b). Other methods improve the efficiency [Xu et al., 1990], the memory footprint [Kiryati et al., 1991] and voting procedure [Fernandes and Oliveira, 2008] of the Hough-like approaches.



(a) [Schnabel et al., 2007] (b) [Rabbani and Van Den Heuvel, 2005]

(c) [Yan et al., 2012] (d) [Tulsiani et al., 2017a]

Figure 2.3: **Primitive fitting.** Many works of the last decades aim to fit primitive to various kind of data.

2.3.3 Clustering-based approaches.

Another way to tackle the primitive fitting task is to cluster the point into groups that will be each represented by a primitive afterward. The most popular clustering methods are *K-Means* [Lloyd, 1982] and *Mean sift* [Fukunaga and Hostetler, 1975]. Once detected, each cluster of points can be represented by a primitive. Yan et al. estimated quadratic surface [Yan et al., 2012] from mesh representation and Woodford et al [Woodford et al., 2012] used such clustering algorithms to fit geometric primitive to large real-world point clouds (see Figure 2.3c).

2.3.4 Neural network-based approaches.

Neural networks can be trained to predict accurately the position and dimension parameters of a set of given primitives to reconstruct a target shape. Several methods use cuboids with reinforcement learning [Tulsiani et al., 2017a] (see Figure 2.3d) or recurrent neural network [Zhang et al., 2020]. More complex superquadrics shapes can be used as well [Paschalidou et al., 2019a]. [Huang et al., 2021] manipulates primitives with an adversarial loss. HPNet [Yan et al., 2021] segment points into patch primitives. [Li et al., 2019] supervised the fitting of a varying number of geometric primitives to 3D point clouds using neural networks. In [Mo et al., 2019], the authors perform a hierarchical semantic segmentation using learned primitives.

2.4 Surface parameterization

Parameterizing a surface, i.e. a mapping of a surface on the 2D Euclidien plane to a 3D surface, as many applications like applying a 2D texture image on a 3D shape or finding correspondences between shapes. Those are common problems of many professionals in the creative industry and many other fields. In this section we will first explain relevant differentiable geometry terminilogy, then how the classical approaches use shape parameterization, followed by how recent methods used neural networks, and we will finish by listing a few example of their applications.

2.4.1 Differentiable geometry terminology

In this section we explain the concepts of atlas, chart, parameterization, first fundamental form and finally discuss different type of regularisation. Atlases and shape parameterization are vast research subjects and we encourage the readers to dive into the two following surveys [Floater and Hormann, 2005, Sheffer et al., 2007b] in order to find a broader explanation of the concepts.

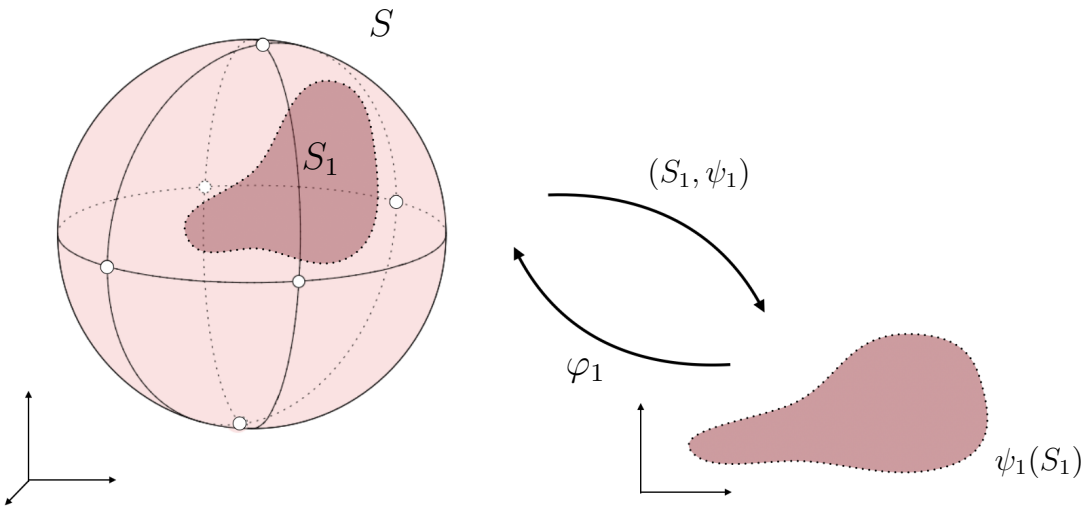


Figure 2.4: **Charts & parameterization of a manifold** A subset S_1 of 2D-manifold S is mapped in 2D using a chart (S_1, ψ_1).

2D-Manifold The term surface is technically less specific but is often used synonymously for a 2D-manifold. A 2D-manifold is a topological space with the same local topological properties as the Euclidean geometry plane.

Charts & parameterization of a manifold. A *chart* is a homeomorphic function ψ_i that map an open subset S_i of a 2D-manifold S onto a Euclidian space. That is to say that a chart is *continuous*, *bijective* and that it has an inverse continuous function φ_i so that $\psi_i \circ \varphi_i$ is the identity function. Charts are usually define by a set pair (S_i, ψ_i). In this thesis they are generally defined by functions from $\mathbb{R}^3$ to $\mathbb{R}^2$. The inverse mapping φ_i of a chart is called a *parameterization*. Figure 2.4 illustrates these concepts.

Atlases In this thesis we use *atlases* to map a 2D-manifold S to a 2D domain. An atlas $\mathcal{A}$ is defined by an indexed family of charts such as

$$\mathcal{A} = \{(S_i, \psi_i) : i \in I\} \quad , \tag{2.2}$$

with

$$S = \bigcup_{i \in I} S_i \quad . \tag{2.3}$$

Note that the subsets S_i can overlap and are associated to open subsets in the mathematical definitions. In this thesis, we use atlases with a slightly different meaning, considering closed and non-overlapping sets S_i.

Tangent plane. Let us consider the parametric subset S_i defined by the parameterization $\phi_i(u, v)$ of the (u, v) coordinates subset of $\mathbb{R}^2$ referred as a 2D domain. For every points of S_i,

$$\mathbf{x}_u = \frac{\partial \varphi_i}{\partial u} \qquad \text{and} \qquad \mathbf{x}_v = \frac{\partial \varphi_i}{\partial v} \quad , \tag{2.4}$$

are the tangent vectors; the tangent plane is defined by the local basis $(\mathbf{x}_u, \mathbf{x}_v)$ and every tangent vector can be expressed as a linear combination of both vectors.

First fundamental form. Let four real coefficients a, b, c, d in $\mathbb{R}$, then the inner product of two tangent vectors

$$\mathbf{v}_1 = a\mathbf{x}_u + b\mathbf{x}_v \quad \mathbf{v}_2 = c\mathbf{x}_u + d\mathbf{x}_v \tag{2.5}$$

can be written as :

$$\begin{aligned} \mathbf{I}(\mathbf{v}_1, \mathbf{v}_2) &= \mathbf{I}(a\mathbf{x}_u + b\mathbf{x}_v, c\mathbf{x}_u + d\mathbf{x}_v) && (2.6) \\ &= ac\langle \mathbf{x}_u, \mathbf{x}_u \rangle + (ad + bc)\langle \mathbf{x}_u, \mathbf{x}_v \rangle + bd\langle \mathbf{x}_v, \mathbf{x}_v \rangle \quad . && (2.7) \end{aligned}$$

The values

$$E = \langle \mathbf{x}_u, \mathbf{x}_u \rangle, \quad F = \langle \mathbf{x}_u, \mathbf{x}_v \rangle \quad G = \langle \mathbf{x}_v, \mathbf{x}_v \rangle \tag{2.8}$$

are called the coefficients of the *first fundamental form*. The matrix

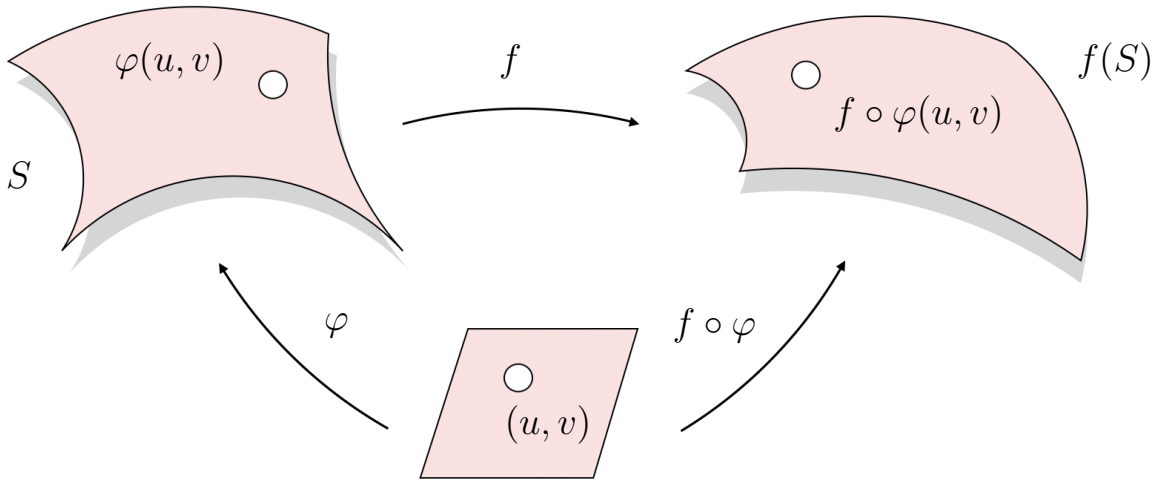


Figure 2.5: **Deformation and parameterization of a 2D-manifold.** We define the notation corresponding to the deformation of a 2D-manifold S by a deformation function f and how a joint parameterization between S and $f(S)$ can be achieved.

$$\mathbf{I} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} \tag{2.9}$$

is often referred to as the first fundamental form itself. For two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ on the tangent plane :

$$\mathbf{I}(\mathbf{v}_1, \mathbf{v}_2) = \mathbf{v}_1^T \begin{pmatrix} E & F \\ F & G \end{pmatrix} \mathbf{v}_2 \quad . \tag{2.10}$$

Regularisation We can use the first fundamental form to express many properties of a 2D-manifold such as curvatures, area, and distortion. Let us assume that we deform a manifold S to another one $f(S)$, the deformation being defined by a homeomorphism f and that we have a parameterization φ mapping a 2D domain to the first manifold S. We can define the parameterization $f \circ \varphi$ of $f(S)$ so that every origin points on S and image points on $f(S)$ have the same coordinates (u, v) on the 2D domain see Figure 2.5. We can use the first fundamental form $\mathbf{I}(\varphi)$ and $\mathbf{I}(f \circ \varphi)$ to control the distortion of the deformation induced by f :

- **Isometric deformation**. The deformation f is a *length preserving* defor-

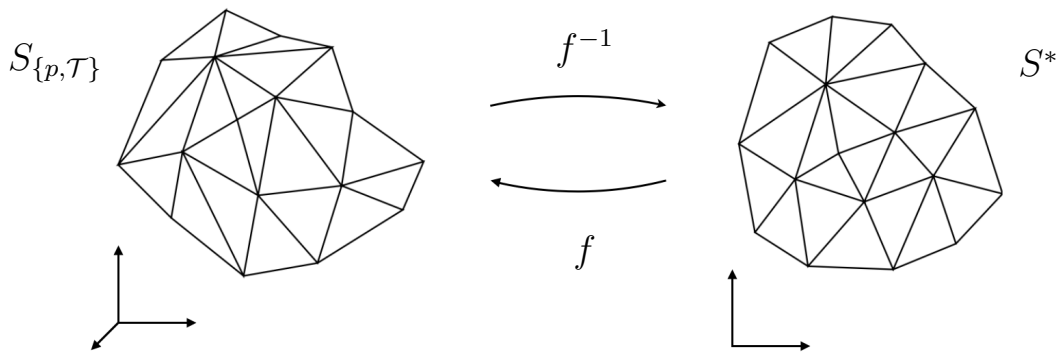


Figure 2.6: **Piecewise linear mapping.** [Floater and Hormann, 2005]

mation, or *isometric* deformation if the length of any arc of S is equal to its image of $f(S)$. In this case the two first fundamental forms are equal

$$\mathbf{I}(\varphi) = \mathbf{I}(f \circ \varphi) \quad . \tag{2.11}$$

- **Conformal mapping**. The mapping is said to be *conformal* if any given angle between pair of arc on S is equal to its image on $f(S)$. In this case the coefficient of the first fundamental form are proportional

$$\mathbf{I}(\varphi) = \alpha(u,v)\ \mathbf{I}(f \circ \varphi) \quad \text{with} \quad \alpha(u,v) \in \mathbb{R}, \quad \alpha(u,v) \neq 0 \quad . \tag{2.12}$$

Discrete paramaterization The most common way to approximate in a discrete way a surface S is to use a triangular mesh $S_{\{p,\mathcal{T}\}}$. We can characterize it by a set of 3D vertices p and a set of index triplets $\mathcal{T}$ with

$$p = \{p_i \in \mathbb{R}^3 : 1 \leq i \leq N\} \quad , \tag{2.13}$$

and

$$\mathcal{T} = \{\mathcal{T}_j = (v_j^1, v_j^2, v_j^3) : 1 \leq j \leq M, \quad \forall i \in \{1,2,3\} \quad 1 \leq v_j^i \leq N\} \quad , \qquad (2.14)$$

where each $\mathcal{T}_j$ defines a triangle composed of three vertices, and N and M are the number of vertices and triplets of indexes of the mesh. Given a mesh the task of parameterization corresponds to finding a domain $S^* \subset \mathbb{R}^2$ and a piecewise linear mapping $f : S^* \mapsto S_{\{p,\mathcal{T}\}}$ of every triangle of the mesh. This mapping can be entirely defined by the mapping $f^{-1}(p) \in S^*$ of the vertices p of the mesh. Figure 2.6 illustrate this process. S^* is often called a UV-map.

2.4.2 Classical approaches for surface parameterization

In this subsection, we will discuss mesh parameterization. The task is to map the faces of a mesh to a 2D domain. The parameterization is a piece-wise linear function from 2D to 3D. It maps every 2D face to its corresponding face on the mesh. The parameterization aims to produce a bijection (invertible mapping) such that each point of the planar domain corresponds to a single point on the mesh. The sum of the angles around a mesh vertex is always equal to 2π in 2D and can vary in 3D. Thus, the parameterization introduced some distortion. Minimizing distortion or controlling it is one of the main problems the research community has spent time trying to solve, and it is still an open problem. We will discuss several parameterization techniques. We will start with the early parameterization that ignores distortion, then the ones minimizing the angular distortion, followed by the ones focusing on stretch distortion, and we will finish with the methods introducing cuts to reduce distortion. Initially thought for texture mapping, mesh parameterization application became broader over the last decade. We will discuss several applications in subsection 2.4.4.

Parameterization without distortion regularisation Tutte [Tutte, 1963] introduced a method to generate generic graph embedding that can be extended to meshes. The first step of this method maps the boundary vertices of a surface to the planar plane, and the position of the rest of the vertices are obtained by solving a linear system that do not generate bijectivity. The circle packing theorem [Rodin and Sullivan, 1987, Collins and Stephenson, 2003] gives another way to produce planar embedding constituted of a collection of circles whose centers correspond to the vertices embedded in two dimensions. Two circles are tangent iff an edge exists between their corresponding vertices on the mesh. The two dimensional embedding mesh is obtain by connecting the center of the circles. This method has intersesting properties such as bijectivity and uniqueness of the solution under some constraints.

Conformal regularisation Riemann's theorem [Do Carmo, 2016] says that for any given differentiable surface, a mapping with zero angular distortion exists. Since meshes are surface approximations it is likely to produce planar parameterization with minimal angular distortion. Building upon Tutte's approach [Tutte, 1963], several works [Eck et al., 1995, Floater, 1997, Floater, 2003] change the weight in the linear system in order to generate conformal embedding. The most popular weights are *harmonic* or *cotangent* [Eck et al., 1995, Pinkall and Polthier, 1993a]. Initially non-bijective, such embedding can be under specific contitions such as the Delaunay criterion proven to be bijective [Kharevych et al., 2006]. *Shape-preserving parameterization* [Floater, 1997, Guskov, 2004] are also by nature bijective and are another way to tackle conformal parameterization. The *mean-values parameterization* introduced by Floater [Floater, 1997] is another minimal angular distortion technique that guaranties bijectivity. All previous methods relying on Tutte's approach [Tutte, 1963] perform poorly when the mesh has non-convex boundaries. Several methods propose to add part of the boundaries to the linear system. For example [Lee et al., 2002, Kós and Várady, 2003, Zhang et al., 2005] use Floater [Floater, 1997]

approaches with free-boundary specifications. Others such as LSC [Lévy et al., 2002] or DCP [Desbrun et al., 2002] opted for free-boundaries harmonic parameterizations by only fixing two vertices to avoid degenerate solutions. Other methods such as MIPS [Hormann and Greiner, 2000, Hormann et al., 1999] start with a fixed-boundary harmonic solution and optimize the position of the vertices one by one in a manner that ensures bijectivity. Instead of defining the embedding by the vertices coordinates like all the previous methods, ABF approaches [Sheffer and De Sturler, 2000, Sheffer and de Sturler, 2001, Sheffer et al., 2005] parameterization rely on the angle of the faces on the plane. They constrain the parameterization to guarantee no flipped faces, but those methods are weak to overlaps. With additional constraints [Zayer et al., 2005] ABF can be proven bijective.

Isometric regularisation Unlike the conformal mappings, parameterization with zero *stretch* only exists for developable surfaces. For the rest, the isometric regularization can only reduce the isometric distortion. The earlier approaches complexity make the optimisation difficult [Bennis et al., 1991, Lévy and Mallet, 1998, Maillot et al., 1993]. Following methods like [Sander et al., 2001] introduced *stretch* metrics and use similar approaches to MIPS [Hormann and Greiner, 2000, Hormann et al., 1999] where one starts with a shape-preserving parameterization [Floater, 1997] and moves vertices around to reduce the stretch of the 2D triangulation faces while enforcing bijectivity. Others [Zigelman et al., 2002, Zhou et al., 2004] used methods based on geodesic distance (length of the minimal path between two points that follow the object surface) but tend to produce non-bijective parameterization for complex surfaces.

Mesh segmentation and seams The close surfaces or the ones with a genus greater than zero need to be unfolded first and then parameterized. For this, one needs to introduce cuts that are also referred to as seams. Using seams allows reducing the overall distortion of the parameterization for all kinds of metrics used

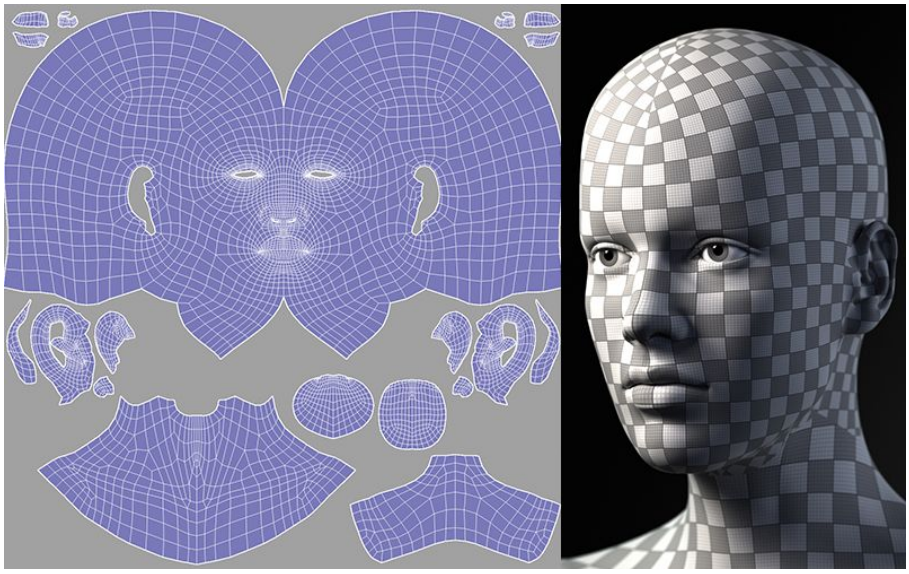


Figure 2.7: **Unfolding and cuts of a 3D mesh**

to measure the distortion. After "*cutting*" the shape, each patch can be mapped in 2-dimension using a family of chart functions (one per patch). Figure 2.7 illustrates this. In this process, finding the least number of cuts and the shortest ones while keeping to minimize the distortion is essential. Several methods approach the problem using mesh segmentation. For instance [Cohen-Steiner et al., 2004, Garland et al., 2001, Maillot et al., 1993, Sander et al., 2001, Sander et al., 2003], try to decompose the mesh into near planar patches easy to parameterize with minimal distortion. Those methods tend to produce a large number of charts. Other methods tackle the mesh segmentation with more complex surface decomposition by using mean curvature [Lévy et al., 2002], region growing base methods [Julius et al., 2005], spectral analysis [Zhou et al., 2004] or graph analysis [Zhang et al., 2005]. It is also possible to cut the mesh without segmenting it into parts. It generally yields a shorter seams length. The cuts can be generated by hand [Piponi and Borshukov, 2000], detected during the parameterization optimization using a distortion threshold [McCartney et al., 1999, Sorkine et al., 2002], computed from an existing shape-preserving parameterization [Gu et al., 2002], or selected on region of interest found using differentiable geometry techniques such as Gaussian curvature [Sheffer,

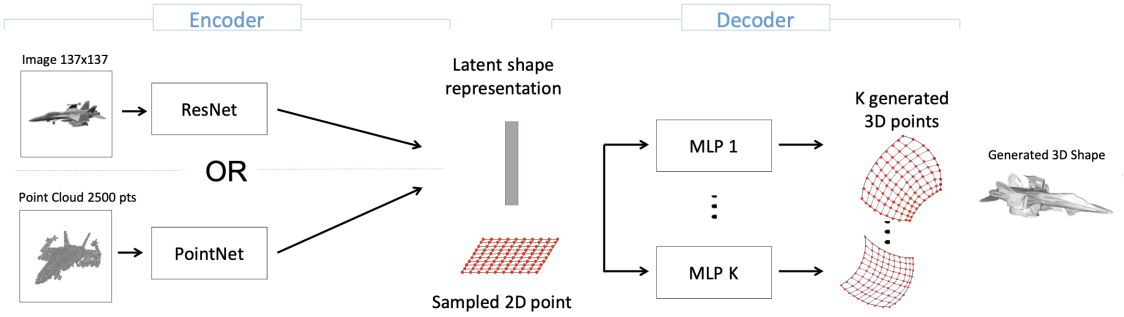


Figure 2.8: **Parameterization-mapping.** AtlasNet parameterization architecture mapping a set of 2D patch in 3D. Image courtesy of [Groueix et al., 2018b].

2002, Sheffer and Hart, 2002].

2.4.3 AtlasNet and Deep surface parameterization

In this subsection, we discuss AtlasNet and the other atlas-like approaches for deep shape parameterization. These works used a neural network to learn a family of parameterization mapping for a set of 2D domains to generate an object surface. AtlasNet [Groueix et al., 2018b] and FoldingNet [Yang et al., 2018a] were the first methods to propose deep surface parameterization. Both methods use a set of unit square patches, and they define a collection of continuous parametric functions that deform and position those patches in 3D in a "papier maché"-like approach illustrated in Figure 2.8. These mapping functions are MLPs that take as input the coordinates of 2D points sampled on the patches and a shape code extracted by a PointNet module [Qi et al., 2017a], and they output 3D point spatial coordinates. After being trained on a large collection of 3D shapes such as [Chang et al., 2015a], the two methods can generate a parameterization to reconstruct any given shape. They can also produce meshes by mapping the vertices of dense triangularization of the square patches in 3D while keeping the two-dimensional edges information. While the shape reconstruction is accurate, such methods often produce various overlaps between the images of the patches in 3D, which mean they do not produce

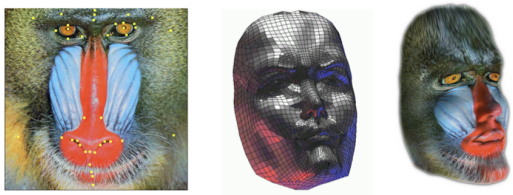
bijective mappings. Other parameterization approaches build upon AtlasNet and FoldingNet. In Deep Geometric Prior [Williams et al., 2019] the authors showed that bijective parameterization could also be optimized on individual shapes. Deng et al. [Deng et al., 2020] improve the global arrangement of the different parts of the parameterization via a normal-aware reconstruction loss and a stitching loss to reduce the overall overlap. Finally, DSR [Bednarik et al., 2020] regularizes the smoothness of the reconstructed surface by optimizing conformal energy based on the Jacobians of the mappings and introduces an area-preserving loss so that the patch reconstructs the shape with minimal overlap.

2.4.4 Applications

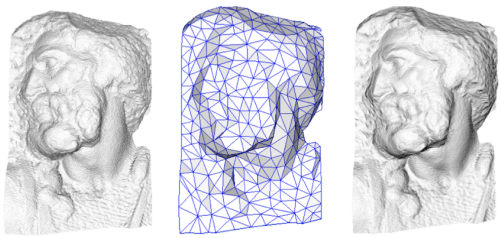
Initially designed for texture mapping and uv-mapping [Bennis et al., 1991, Maillot et al., 1993, Lévy, 2001] (see Figure 2.9), the parameterization of shape have become in the last decades a more generic tool with several other applications such as details generation, morphing of shapes, mesh completion and mesh editing.

Generating rendering details

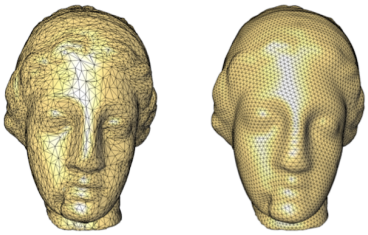
While meshes have shown benefits for approximating a surface, they come with smoothness and details issues when rendered. Typically, the normal orientations are one of the most important aspect for rendering shadows, and without any additional work the faces of the mesh will be seen *flat* when rendered since normals are constant across each one of them. Several works like [Blinn, 1978, Sheffer et al., 2005] propose to use texture mapping methods to add a perturbation of the normal orientations during the rendering. It yields a more realistic rendering while preserving the geometry of the mesh.



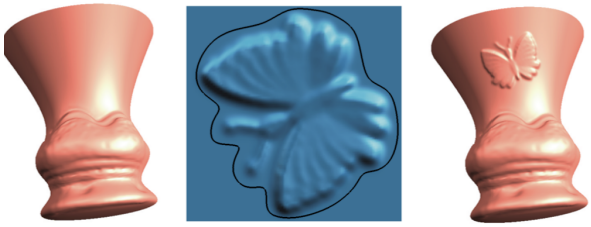
(a) **Texture mapping.** [Lévy, 2001]



(b) **Normal maps.** [Commons, 2005]



(c) **Remeshing.** [Alliez et al., 2008]



(d) **Mesh editing.** [Biermann et al., 2002]

Figure 2.9: **Applications of parameterization** : **(a)** a mesh is textured using an Mandrill image texture, **(b)** a target mesh on the left is reconstructed on the right using a normal map and a low resolution mesh in the middle, **(c)** the left mesh is re-meshed using a uniform triangularisation, **(d)** a detail in the middle is inserted to the left mesh to produce the one on the right.

Mesh completion

Some mesh scans can have missing data or holes from the acquisition process. Many works have used parameterization techniques to fill the voids by using a planar patch mapped onto the mesh [Lévy, 2003]. Other have to use the parameterization of specific human templates as a piece of prior knowledge to fill the gaps of human mesh [Allen et al., 2003a, Anguelov et al., 2005], and other works such as [Kraevoy and Sheffer, 2005] developed approaches that use more generic template shapes.

Mesh editing

Many works have developed methods leveraging parameterization to modify the geometry of a mesh locally. Local adjustments are usually stored like a texture and mapped onto the mesh using parameterization [Pedersen, 1995, Praun et al., 2000, Soler et al., 2002, Turk, 2001, Wei and Levoy, 2001, Ying et al., 2001]. [Biermann et al., 2002] have used a shared map between two surfaces to copy-past local detail from one shape to another. Other works [Lévy et al., 2002, Sorkine and Alexa, 2007] use local parameterization of the region of interest on two meshes and overlap the parameterizations.

Remeshing

For a given continuous surface, there is not a single mesh representation. Depending on the task, one would like a denser, sparser, or more uniform triangulation. In some works, a parameterization of the given mesh is generated, and a different triangulation with the desired characteristics is mapped back onto the surface using the generated parameterization. The most used triangulations are planar and regular grids [Gu et al., 2002], regular subdivision of the faces [Guskov et al., 2000, Khodakovsky et al., 2003, Lee et al., 2000] and Delauney triangulation [Desbrun et al., 2002]. More details of the techniques can be found in this survey [Alliez et al., 2008].

CHAPTER 3

Learning elementary structures for 3d shape generation and matching

Abstract

We propose to represent shapes as the deformation and combination of learnable elementary 3D structures, which are primitives resulting from training over a collection of shapes. We demonstrate that the learned elementary 3D structures lead to clear improvements in 3D shape generation and matching. More precisely, we present two complementary approaches for learning elementary structures: (i) patch deformation learning and (ii) point translation learning. Both approaches can be extended to abstract structures of higher dimensions for improved results. We evaluate our method on two tasks: reconstructing ShapeNet objects and estimating dense correspondences between human scans (FAUST inter challenge). We show 16% improvement over surface deformation approaches for shape reconstruction and outperform FAUST inter and intra challenge state of the art by 2% and 7%, respectively.

3.1 Introduction

Current surface-parametric approaches for generating a surface or aligning two surfaces, such as AtlasNet [Groueix et al., 2018b] and 3D-CODED [Groueix et al., 2018a], rely on alignment of one or more shape primitives to a target shape. The shape primitives can be a set of patches or a sphere, as in AtlasNet, or a human template shape, as in 3D-CODED. These approaches could easily be extended to other parametric shapes, such as blocks [Roberts, 1963], generalized cylinders [Binford, 1971], or modern shape abstractions [Li et al., 2017, Sharma et al., 2018, Tulsiani et al., 2017b]. While surface-parametric approaches have achieved state-of-the-art results for (single-view) shape reconstruction [Groueix et al., 2018b] and 3D shape correspondences [Groueix et al., 2018a], they rely on hand-chosen parametric shape primitives tuned for the target shape collection and task. In this chapter, we ask – what is the right set of primitives to represent a collection of diverse shapes?

To address this question, we seek to go beyond manually choosing shape primitives and automatically learn what we call "learnable elementary structures" from a shape collection, which can be used for shape reconstruction and matching. The ability to automatically learn elementary structures allows the shape generator to find a better set of primitives for a shape collection and target task. We find that learned elementary structures correspond to recurrent parts among 3D objects. For example, in Figure 4.1, we show automatically learned elementary structures roughly corresponding to the tail, wing, and reactor of an airplane. Moreover, we find that learning the elementary structures leads to an improvement in shape reconstruction and correspondence accuracy.

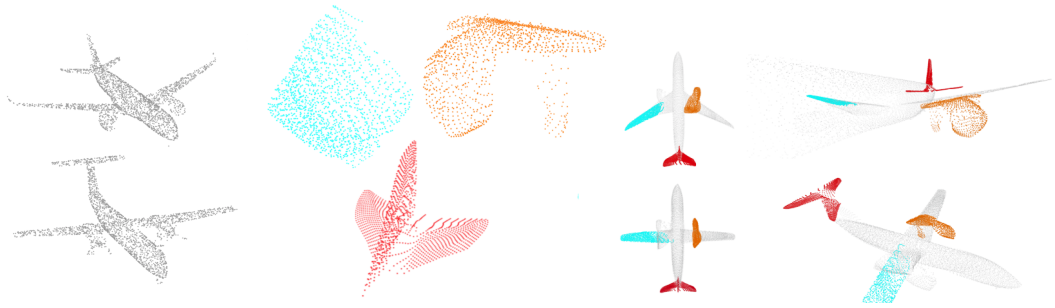
We explore two approaches for learning elementary structures – *patch deformation learning* and *point translation learning*. For patch deformation learning, similar to AtlasNet [Groueix et al., 2018b], we start from a surface element, such as a 2D square, and deform it into the learned structure using a multi-layer perceptron

[Rosenblatt, 1958]. This approach has the advantage that the learned elementary structures are continuous surfaces. Its key difference with respect to AtlasNet is that the deformations, and thus the elementary structures, are common to all shapes. For point translation learning, starting from a fixed set of points, we optimize their position to reconstruct the target objects. The drawback of this approach is that it does not produce a continuous surface – only a finite set of points. However, this approach is more flexible since it can, for example, change the topology of the structure.

We show how to deform and combine our learnable elementary structures to explain a given 3D shape. At inference, given the learned elementary structures, we learn to position the structures by *adjustment* – a linear (projective) transformation will lead to maximum interpretability, while a complex transformation parameterized by a multi-layer perceptron will make our approaches generalizations of prior shape reconstruction methods [Groueix et al., 2018b, Groueix et al., 2018a] using optimized instead of manually defined templates. Moreover, such representation allows for disentanglement of the structure's shape and pose. We include structure learning in a deep architecture that unifies shape abstraction and deep surface deformation approaches.

We demonstrate that our architecture leads to improvements for 3D shape generation and matching – 16% relative improvement over AtlasNet for generic object shape reconstruction and 7% and 2% over 3D-CODED for human shape matching on Faust [Bogo et al., 2014] Intra and Inter challenges, respectively, achieving state of the art for the latter task. Our code is available on our project webpage[1]

[1]http://imagine.enpc.fr/ deprellt/atlasnet2



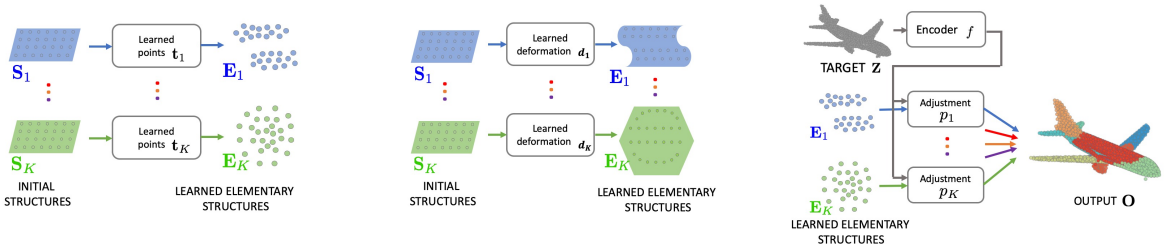
(a) Input target shapes (b) Learned elementary structures (c) Our reconstructions

Figure 3.1: **Problem statement.** We seek to automatically learn a set of primitives (called "learned elementary structures") for shape reconstruction and matching. (a) Input target shapes to reconstruct. (b) Learned elementary structures roughly corresponding to the tail, wing, and reactor of airplanes. (c) Our output reconstructions with learned elementary structures highlighted.

3.2 Related work

Primitive fitting is a classic topic in computer vision [Roberts, 1963], with a large number of methods targeting parsimonious shape approximations, such as generalized cylinders[Binford, 1971] and geons [Biederman, 1987]. Efficient fitting of these primitives attracted a lot of research efforts [Kaiser et al., 2018, Li et al., 2011, Schnabel et al., 2009, Schnabel et al., 2007]. Since these methods analyze shapes independently, they are not expected to use the primitives consistently across different objects, which makes the result unsuitable for discovering a common structure in a collection of shapes, performing consistent segmentation, or correspondence estimation. To address these limitations some methods optimize for consistent primitive fitting over the entire shape collection [Kim et al., 2013], or aim to discover a consistent set of parts [Golovinskiy and Funkhouser, 2009, Huang et al., 2011, Sidi et al., 2011]. The resulting optimization problems are usually non-convex, and thus existing solutions tend to be slow, require heuristics, and are prone to being stuck in

local optima. Learning-based techniques offer a promising alternative to hand-crafted heuristics. Zhu *et al.* [Zou et al., 2017] use a Recurent Neural Network supervised by a traditional heuristic-based algorithm for cuboid fitting. Tulsiani *et al.* [Tulsiani et al., 2017b] use reconstruction loss to predict parameters of the cuboids that approximate an input shape, and thus do not require any direct supervision. Several recent techniques, concurrent to our work, extend this approach by using more complex primitives that can better approximate the surface, such as anisotropic 3D Gaussians [Genova et al., 2019], categorie specifique morphable model [Kanazawa et al., 2018] or superquadrics [Paschalidou et al., 2019b]. All of these techniques use a collection of simple hand-picked parametric primitives. In contrast, we propose to learn a set of deformable primitives that best approximate a collection of shapes. One can further improve reconstruction by fitting a diverse set of primitives [Li et al., 2018] or constructive solid geometry graphs [Sharma et al., 2018]. These methods, however, usually do not produce consistent fitting across different shapes, and thus cannot be used to discover common shape structures or inter-shape relationships. On the other side of the spectrum, instead of simple primitives, some techniques fit deformable mesh models [Allen et al., 2002, Allen et al., 2003b, Loper et al., 2015, Zuffi and Black., 2015]. While they can capture complex structures, these techniques are also prone to being stuck in local optima, due to large number of degrees of freedom (e.g., mesh vertex coordinates). Neural network architectures have been used to facilitate the mesh fitting [Groueix et al., 2018a], learning to predict the deformation of a template to reconstruct unstructured input point cloud. This approach is sensitive to the choice of the template. We demonstrate that our method improves the quality of the fitting by learning the structure of the reference shape. Neural mesh fitting has been also employed for geometrically and topologically diverse datasets that do not have a natural template. In these cases, meshed planes or spheres can be deformed into complex 3D structures [Groueix et al., 2018b, Yang et al., 2018b]. We extend this line of work by proposing a technique



(a) Points translation learning module

(b) Patch deformation learning module

(c) Elementary structures combination model

Figure 3.2: **Approach overview.** At training time, we learn (a) translations t_i or (b) deformations d_i that transform points from the unit square S_i into shared learned elementary structures (c). At evaluation time, we transform each elementary structure E_i to target shape Z using learned shape-dependent adjustment networks p_i that produce points on the surface of the output shape O.

for learning the base shapes that are further used to approximate the shapes in the collection. Learning these elementary structures enables us to more accurately and consistently reconstruct the shapes in the collection.

3.3 Approach

We aim to learn shared elementary structures to reconstruct a set of 3D shapes. We visualize an overview of our approach in Figure 3.2. We formulate two ways to learn elementary structures – via *patch deformation learning* and *point translation learning* modules. The elementary structures are learned over the entire training set and do not depend on the input during testing. At test time, the elementary structures are deformed by *adjustment modules* to create the output 3D shape. These modules take as inputs features computed from the input via an encoder network and the coordinates of the elementary structure points and output the 3D coordinates of the deformed primitives.

For the task of 3D shape reconstruction, we assume that we are given a training set $\mathcal{Z}$

of target shapes $\mathbf{Z} \in \mathcal{Z}$. Our goal is to reconstruct the target shapes using a set of K learned elementary structures $\mathbf{E}_1, \ldots, \mathbf{E}_K$, which are deformed via shape-dependent adjustment modules $p_1, \ldots, p_K$. We represent each shape by a feature vector $f(\mathbf{Z})$ computed by a point set encoder f (defined later in this section). Each adjustment module p_k takes as inputs the coordinates of a point in the associated elementary structure $\mathbf{e} \in \mathbf{E}_k$ and the feature vector of the target shape $f(\mathbf{Z})$ and outputs 3D coordinates of the corresponding point. The output shape $\mathbf{O} = p(\mathbf{Z})$ can thus be written as the union over learned and adjusted elementary structures,

$$\mathbf{O} = p(\mathbf{Z}) = \bigcup_{k=1}^{K} \bigcup_{\mathbf{e} \in \mathbf{E}_k} p_k(\mathbf{e}, f(\mathbf{Z})). \tag{3.1}$$

If the elementary structures were unit squares or a unit sphere, then this equation would describe exactly the AtlasNet [Groueix et al., 2018b] model. On the other hand, the 3D-CODED model [Groueix et al., 2018a] uses an instance of $\mathcal{Z}$ as a single elementary structure. Generalizing these approaches, our goal is to automatically learn the elementary structures $\mathbf{E}_k$ over a shape collection. The intuition behind our approach is that if the elementary structures $\mathbf{E}_k$ have useful shapes to reconstruct the target, the adjustment p_k should be easier to learn and more interpretable.

3.3.1 Learnable elementary structures

For each $k \in \{1, \ldots, K\}$, we start from an initial surface $\mathcal{S}_k$ on which we sample N points to obtain an initial point cloud $\mathbf{S_k}$. We then pass each sampled point $\mathbf{s}_{k,i} \in \mathbf{S_k}$ for $i \in \{1, \ldots, N\}$ through elementary structure learning modules ψ_k. We consider two types of elementary structure learning module ψ_k.

The first type, patch deformation learning module, learns a continuous mapping d_k to obtain deformed points $\mathbf{e}_{k,i} = d_k(\mathbf{s}_{k,i})$ starting from sampled point $\mathbf{s}_{k,i}$. The intuition behind the deformation module is that elementary structures E_k should be surface elements, and can thus be deduced from the transformation of the original surfaces $\mathbf{S}_k$. Alternatively, we consider a point translation learning module which translates

independently each of the points $\mathbf{s}_{k,i}$ by a learned vector $\mathbf{t}_{k,i}$, $\mathbf{e}_{k,i} = \mathbf{t}_{k,i} + \mathbf{s}_{k,i}$. This module thus allows the network to update independently the position of each point on the surface. The result of either module results in a set of elementary structure points $\mathbf{e}_{k,i} = \psi_k(\mathbf{s}_{k,i})$, and we write the elementary structure $\mathbf{E}_k$ as the union of the independently deformed or translated points $\mathbf{s}_{k,i} \in \mathbf{S_k}$.
In Section 3.3.3 we will show that different choices here can be desirable depending on the application domain.

Dimensionality of the elementary structures. While it is natural to consider elementary structures as sets of 3D points, we can extend the idea to other dimensions. We experimented with 2D, 3D, and 10D elementary structures and show that while they are less interpretable, higher-dimensional structures lead to better shape reconstruction results.

3.3.2 Architecture details

The following describes more details of our final network.

Shape encoder. We represent the input shape as a point cloud, and we use as shape encoder a simplified version of the PointNet network [Qi et al., 2017a] used in [Groueix et al., 2018a, Groueix et al., 2018b]. We represent each 3D point of the input shape as a 1024 dimensional vector using a multi-layer perceptron with 3 hidden layers of 64, 128 and 1024 neurons and ReLU activations. We then apply max-pooling over all point features followed by a linear layer, producing a global shape feature used as input to the adjustment modules.

Patch deformation learning module. The patch deformation learning modules are continuous-space deformations that we learn as multi-layer perceptrons with 3 hidden layers of 128, 128 and 3 neurons and ReLU activations. This module takes as input coordinates of points in the initial structures and can compute not only a

set of points [Groueix et al., 2018b] but the full image of a surface. If this module is used, we can densely sample points on the generated surface.

Point translation learning module. The point translation learning modules learn a translation for each of the N points of the associated initial structure. While this step gives more flexibility than generating points through the patch deformation learning module, it can only be applied for a fixed number of points, similar to point-based shape generation [Fan et al., 2017].

Adjustment module. The goal of the adjustment modules p_k is to reconstruct the input shape by positioning each elementary structure. The intuition is that this adjustment should be relatively simple. However, we can expect the quality of the reconstruction to increase using more complex adjustment modules. In this chapter, we consider two cases:

- *Linear adjustment*: each adjustment module applies an affine transformation to the corresponding elementary structure. The parameters of this transformation are predicted by a multi-layer perceptron that takes as input the point cloud feature vector generated by the encoder. We use three hidden MLP layers (512, 512, 12), ReLU activation, BatchNorm layers and a hyperbolic tangent at the last layer for this module.

- *MLP adjustment*: each adjustment module uses a multi-layer perceptron (MLP) that takes as inputs the concatenation of the coordinates of a point from the associated elementary structure and the shape feature predicted by the shape encoder and outputs 3D coordinates. We use the same architecture as [Groueix et al., 2018b] for this network to obtain comparable results.

3.3.3 Losses and training

We now discuss two scenarios in which we tested our approach.

Training with correspondences. In this scenario, we assume point correspondences across all training examples and a common template that we can use as an initial structure for all shapes. More precisely, we assume that each training shape $\mathbf{Z}$ is represented as an ordered set of N 3D points $\mathbf{z}_1, \ldots, \mathbf{z}_N$ in consistent locations on all shapes. Since all shapes are in correspondence, we consider a single elementary structure S_1 ($K = 1$) and N sampled points on the shape $\mathbf{s}_{1,1}, \ldots, \mathbf{s}_{1,N}$. We then train our network to minimize the following squared loss between sampled points $\mathbf{z}_i$ on each training shape to reconstructed points starting from sampled template points $\mathbf{s}_{1,i}$:

$$\mathcal{L}_{\mathbf{sup}}(\theta) = \sum_{\mathbf{Z} \in \mathcal{Z}} \sum_{i=1}^{N} \left\| \mathbf{z}_i - p_1(\psi_1(\mathbf{s}_{1,i}), f(\mathbf{Z})) \right\|^2 \tag{3.2}$$

where θ are the parameters of the networks. Note that at inference, we do not need to know the correspondences of the points in the test shape, since they are processed by the point set encoder which is invariant to the order of the points. Instead, the points in the reconstruction shapes will be in correspondence with the elementary structure and by extension with each other. We use this property to predict correspondences between test shapes, following the pipeline of [Groueix et al., 2018a]. Learning the elementary structures is the difference between our approach and 3D-CODED [Groueix et al., 2018a] in this scenario, which leads to improved reconstruction and correspondence accuracy.

Training without correspondences. We are also able to train our system when no correspondence supervision is available during training. In this case, there are many options for our choice of elementary structures. To be comparable with AtlasNet [Groueix et al., 2018b], we will assume we have K elementary structures and that each initial structure $\mathcal{S}_k$ is a unit 2D square patch. For a given training shape $\mathbf{Z}$, we compute the output shape $\mathbf{O} = p(\mathbf{Z})$ according to Equation 3.1, and train our network's parameters to minimize the symmetric Chamfer distance [Fan

	Single-category training		Multi-category training		
	Airplanes	Chairs	Airplanes	Chairs	All
Linear adjustment					
AtlasNet [Groueix et al., 2018b]	1.57	4.14	2.22	3.72	3.07
Deformation	1.16	2.76	1.49	2.52	2.26
Points	1.04	2.00	1.35	2.47	2.11
MLP adjustment					
AtlasNet [Groueix et al., 2018b]	0.91	1.64	0.81	1.50	1.45
Deformation	0.87	1.56	0.81	**1.25**	1.43
Points	**0.79**	**1.43**	**0.71**	**1.25**	**1.22**

	Multi-category training	
	Points	Def.
MLP adjustment		
2D	1.28	1.42
3D	1.22	1.43
10D	**1.21**	**1.39**
Linear adjustment		
2D	2.45	2.75
3D	2.11	2.26
10D	**1.66**	**1.90**

Table 3.1: **ShapeNet reconstruction.** We evaluate variants of our method for single- and multi-category reconstruction tasks. *Left*: Linear vs MLP adjustment, Patch Deformation vs Points Translation with 3D elementary structures. *Right*: different template dimensionality and deformation vs points learning modules in the multi-category setup with MLP-adjustement. We report Chamfer distance (multiplied by 10^{-3}). AtlasNet uses 10 patch primitives, which is the same as our approach, without the learned elementary structures.

et al., 2017] between the point clouds $p(\mathbf{Z})$ and $\mathbf{Z}$.

$$\mathcal{L}_{\mathbf{unsup}}(\theta) = \sum_{\mathbf{Z}\in\mathcal{Z}} \sum_{\mathbf{z}\in\mathbf{Z}} \min_{k\in\{1,\dots,K\},\ i\in\{1,\dots,N\}} \|\mathbf{z} - p_k(\psi_k(\mathbf{s}_{k,i}), f(\mathbf{Z}))\|^2 + \sum_{\mathbf{Z}\in\mathcal{Z}} \sum_{k=1}^{K} \sum_{i=1}^{N} \min_{\mathbf{z}\in\mathbf{Z}} \|\mathbf{z} - p_k(\psi_k(\mathbf{s}_{k,i}), f(\mathbf{Z}))\|^2 \quad (3.3)$$

where θ are the parameters of the networks. In all of our experiments, we used $K = 10$.

Training details. We use the Adam optimizer with a learning rate of 0.001, a batch size of 16, and batch normalization layers. We train our method using input point clouds of 2500 points when correspondences are not available and 6800 points when correspondences are available. When training using only the deformation modules d_k, we resample the initial surfaces S_k at each training step to minimize overfitting. At inference time, we sample a regular grid to allow easy mesh generation. We train our model on an NVIDIA 1080Ti GPU, with a 16 core Intel I7-7820X CPU (3.6GHz), 126GB RAM and SSD storage. Training takes about 48h for most experiments. Using the trained models from the official implementation on all

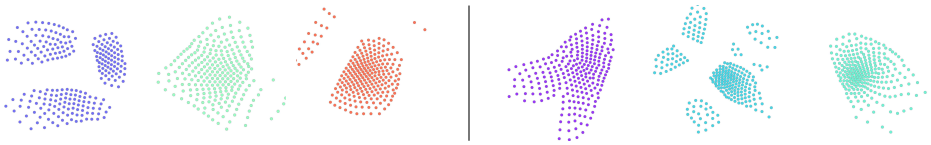
categories, AtlasNet-25 performance is 1.56 (see also Table 1 in the Atlasnet paper). Using the released code to train AtlasNet-10 yields an error of 1.55. By adding a learning rate schedule to the original implementation we decreased this error to 1.45 and report this improved baseline (see Table 1).
In this section, we show qualitative and quantitative results of our approach on the tasks of shape reconstruction and shape matching.

3.3.4 Generic object shape reconstruction

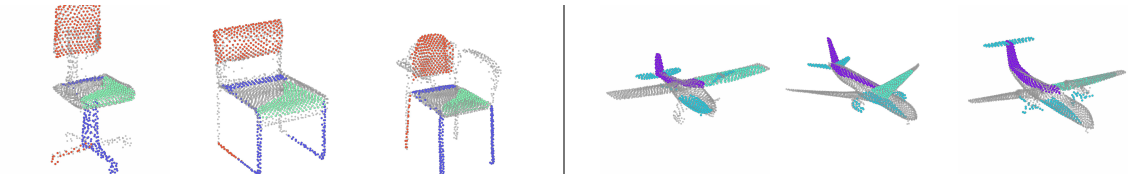
We evaluate our approach on non-articulated generic 3D object shapes for the task of shape reconstruction. We use the training setting without correspondences described in Section 3.3.3.

Dataset, evaluation criteria, baseline. We evaluate on the ShapeNet Core dataset [Chang et al., 2015b]. For single-category reconstruction, we evaluated over airplane (5424/1360 train/test shapes) and chair (3248/816) categories. For multi-category reconstruction, we used 13 categories – airplane, bench, cabinet, car, chair, monitor, lamp, speaker, firearm, couch, table, cellphone, watercraft (31760/7952). We report the symmetric Chamfer distance between the reconstructed and target shapes. All reported Chamfer results are multiplied by 10^{-3}. As a baseline, we compare against AtlasNet [Groueix et al., 2018b] with ten unit-square primitives.

Single-category shape reconstruction. For our first experiment, we trained separate networks for the different ShapeNet Core categories. Figure 3.3a demonstrates learned 2D elementary structures using ten 2D unit squares as initial structures $\mathbf{S_k}$. In Figure 3.3b, we show shape reconstructions using our points translation learning module with MLP adjustments. Note the emergence of symmetric and topologically complex elementary structures.

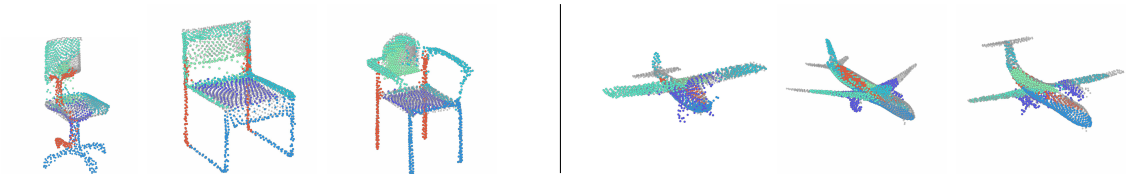


(a) Category-specific 2D elementary structures (3 out of 10 structures) learned for chairs (left) and plane (righ).

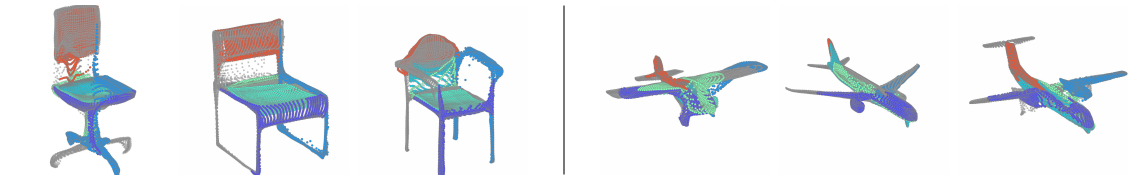


(b) Reconstructions using elementary structures with category-specific training.

(c) 2D elementary structure learned from all categories (7 out of 10 structures are shown).

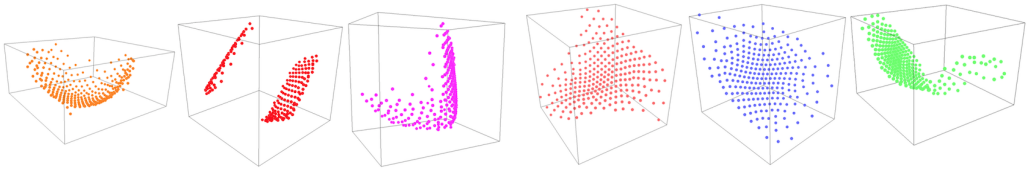


(d) Our reconstructions using 2D elementary structures trained on all categories.



(e) AtlasNet reconstruction using square patch primitives trained on all categories

Figure 3.3: We visualize elementary structures using point learning and MLP adjustment modules. For all reconstruction results, we show in color the points corresponding to the visualized 2D primitives. For AtlasNet, the primitives are unit squares (so we do not show the elementary structures), and we visualize seven of them for the reconstruction (similarly to our method). Contrary to AtlasNet, our learned elementary structures have limited overlap in the reconstructions and better reconstructs the shapes.



(a) MLP adjustment (b) Linear adjustment

Figure 3.4: Three (out of ten) learned 3D elementary structures learned by the point translation learning approach when training on all ShapeNet categories.

3.4 Experiments

Multi-class shape reconstruction. We now evaluate how well our method generalizes when trained on multiple categories, again using 2D elementary structures with point translation learning module and MLP-adjustements. As in single-category case, we observe discovery of non-trivial 2D elementary structures (Figure 3.3c) that are used to accurately reconstruct the shapes (Figure 3.3d), with higher fidelity than the baseline performance of AtlasNet with ten 2D square patches (Figure 3.3e). Note how AtlasNet is less faithful to the topology of reconstructed shapes, incorrectly synthesizing geometry in hollow areas between the back and the seat. Our quantitative evaluation in Table 3.1 confirms that AtlasNet provides less accurate reconstructions than our method.

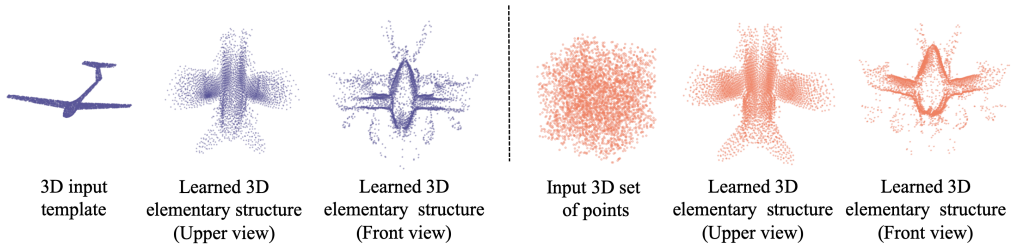


Figure 3.5: 3D elementary structure obtained with point learning when initializing the training from a template shape (left) or a random set of points (right). See text for details.

Linear vs MLP adjustment. We evaluated networks trained in both the single- and multi-category settings with linear and MLP adjustment modules using 3D learned elementary structures (Table 3.1 left, Figure 3.4). In all experimental setups, we observe that the MLP adjustment offers significant quantitative improvements over restricting the network to use linear transformations of the elementary structures. This result is expected as linear adjustment allows only limited adaptation of the elementary structures for each shape. Similar to shape abstraction methods [Tulsiani et al., 2017b], linear adjustment allows a better intuition of the shape generation process but limits the reconstruction accuracy. Using MLP adjustments, however, offers the network more flexibility to faithfully reconstruct the shapes.

Patch deformation vs points translation modules. We compare using patch deformation vs points translation modules in Table 3.1. The patch deformation learning module does not allow topological changes and discontinuities in mapping, and produces inferior results in comparison to points translation learning. On the other hand, learning patch deformations enables the estimation of the entire deformation field. Thus one can warp an arbitrary number of points or even tessellate the domain and warp the entire mesh to generate the polygonal surface, which is more amenable to tasks such as rendering.

Higher-dimensional structures. We experimented with the dimensionality of the learned elementary structures. Elementary/Figures 3.3a and 3.3c suggest that learned 2D elementary structures can capture interesting topological and symmetric aspects of the data – splitting, for instance, the patch into two identical parts for the legs of the chairs. note also the variable point density. Similarly, learned 3D elementary structures with linear adjustment and patch deformation learning modules are shown in Figure 4.1 for the airplane category. Note that they roughly correspond to meaningful parts, such as wings, tail and reactor. Figure 3.4 shows 3D elementary structures inferred from all ShapeNet categories, where the learned

	Chairs	Table
AtlasNet	1.64	4.70
Patch.	1.56	4.82
Point.	**1.34**	**4.45**

Table 3.2: **Category generalization.** Chamfer distance for networks trained on chairs and tested on either the chairs or tables test sets.

	Param.	Chamfer
AtlasNet	1.8×10^8	1.45
6-layer AN	3.9×10^8	1.35
Patch.	1.8×10^8	1.43
Point.	1.8×10^8	**1.22**

Table 3.3: **Number of parameters.** Impact of number of parameters on reconstruction chamfer error.

structures include non-trivial elements such as symmetric planes, sharp angles, and smooth parabolic surfaces. The learned structures are often correspond to consistent parts in the reconstructions. In our quantitative evaluations (Table 3.1, right) we found that the results improve with the dimensionality. The improvement diminishes for higher-dimensional spaces and are more difficult to visualize and interpret.

Consistency in template elementary structures. We experimented with several initializations of our elementary structures on the ShapeNet plane category. We used the point translation learning method and a single 3D elementary structure. In Figure 3.5, we show our results when initializing the elementary structure with either a plane 3D model (left) or a set of random 3D points sampled uniformly (right). Notice that the learned 3D elementary structure is similar regardless of the initial template shape.

Generalization to new categories. To test the generality of our approach, we trained on the chair category using ten 2D elementary structures and tested on the table category. As shown in Figure 3.2, point translation learning outperforms both patch deformation learning and AtlasNet. Figure 3.6 shows qualitatively how the elementary structures are positioned on chairs and tables. Notice how the chair and

table legs are reconstructed by the same elementary structures.

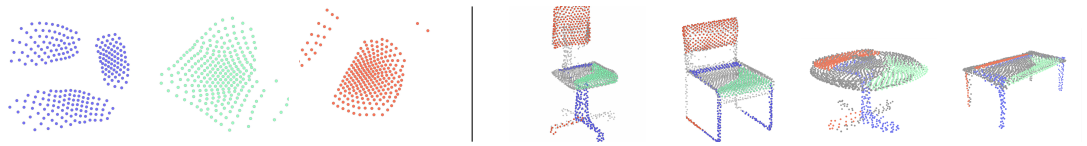


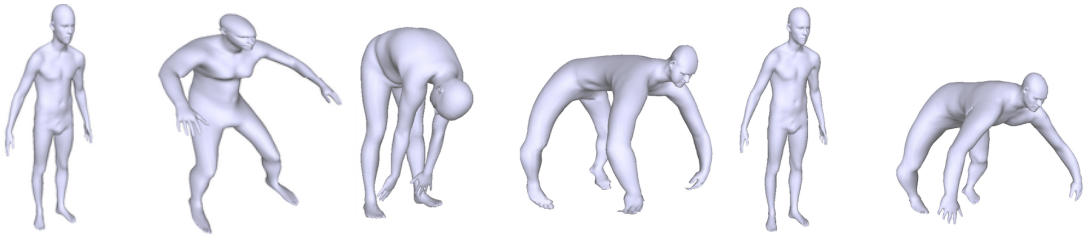
Figure 3.6: Elementary structures learned on chairs (**left**) used to reconstruct chairs and tables (**right**).

Number of parameters. In Figure 3.3, we show the number of parameters for AtlasNet and our method. Our method has less than 1% additional parameters to learn the elementary structures – 2.0×10^6 and 2.5×10^3 for patch deformation and point translation, respectively (orders of magnitude smaller than 1.8×10^8 for the full network). During inference, our approach has the same complexity as AtlasNet as the elementary structures are precomputed and remain fixed for all shapes. We also tried training AtlasNet with six layers (6-layer AN), which significantly increases the number of parameters. Our approach with points translation learning outperforms all methods.

3.4.1 Human shape reconstruction and matching

We now evaluate our approach on 3D human shapes for the tasks of shape reconstruction and matching using the training setup with correspondences described in Section 3.3.3. For this task, we use a single elementary structure for the human body using one of the meshes as the initial structure $\mathbf{S}_1$. Since we use a single elementary structure and the shapes are deformable, we only report results using the MLP-adjustment.

Datasets, evaluation criteria, baselines. We train our method using the SURREAL dataset [Varol et al., 2017], extended to include some additional bend-over poses as in 3D-CODED [Groueix et al., 2018a]. We use 229,984 SURREAL



(a) Learned points (b) Learned deformation (c) Learned deformation

Figure 3.7: Initial shape (left) and learned elementary structure (right) using the deformation or points learning modules. Notice the similarity between the elementary structure learned with the different approaches.

meshes of humans in various poses for training and 224 SURREAL meshes to test reconstruction quality. To evaluate correspondences on real data, we use the FAUST benchmark [Bogo et al., 2014] consisting of 200 testing scans with $\sim 170k$ vertices from the "inter" challenge, including noise and holes which are not present in our training data. As a baseline, we compared against 3D-CODED [Groueix et al., 2018a].

Results. Figure 3.7 shows learned elementary structures using deformation or points translation learning and different initial surfaces. We observe that the learned templates are inflated, bent, and with their arm and legs in a similar pose, suggesting a reasonable amount of consistency in the properties of a desirable primitive shape for this task.

As before, we found that points translation learning provides the best reconstruction (see SURREAL column in Table 3.4). Both of our approaches also provide lower reconstruction loss than 3D-CODED.

We used reconstruction to estimate correspondences by finding closest points on the deformed elementary structure as in 3D-CODED [Groueix et al., 2018a]. We report correspondence error in the "FAUST" column in Table 3.4. We observe that deformation learning provides better correspondences than points learning, also yielding state-of-the-art results and clear improvement over 3D-CODED. This result

	SURREAL [Varol et al., 2017]	FAUST [Bogo et al., 2014] Inter | Intra
3D-CODED	1.32	2.64 | 1.747
Deformation	1.44	2.58 | 1.742
Points	1.00	2.71 | 1.626

	SURREAL [Varol et al., 2017]	
	Points	Deform.
2D	1.54	6.76
3D	1.00	1.44
10D	1.06	1.18

Table 3.4: **Human correspondences and reconstruction.** We evaluate different variants of our method (with deformation vs points translation learning and different template dimensionality) for surface reconstruction (SURREAL column) and matching (FAUST column). We report Chamfer loss for the former and correspondence error for the latter (measured by the distance between corresponding points). Results in the left table are with 3D elementary structures, and the only difference with the 3D-CODED baseline is thus the template/elementary structure learning. The table on the right shows results with elementary structures of different dimensions.

is not surprising because understanding the deformation field for the entire surface is more relevant for matching and correspondence problems.

Elementary structure dimension. Similar to generic object reconstruction, we evaluate with 2D, 3D and 10D elementary structures (Table 3.4, right). Note that when using the patch deformation learning module we control the output size and therefore it is easy to map the input 3D template to higher- or lower-dimensional elementary structure. On the other hand the points translation learning module does not allow to change dimensionality of the input template. Hence, for 2D elementary structures we project the 3D template (front-facing human in a T-pose) to a front plane, and for 10D elementary structures we embed the 3D human into a hyper-cube, keeping higher dimensions as zero. The difference in performance is clearer for human reconstruction than for generic object reconstruction, which can

be related both to the fact that humans are complex with articulations and that we use a single elementary structure for all human reconstructions.

3.5 Conclusion

We have presented a method to take a collection of training shapes and learned common elementary structures that can be deformed and composed to consistently reconstruct arbitrary shapes. We learn consistent structures without explicit point supervision between shapes and we demonstrate that using our structures for reconstruction and correspondence tasks results in significant quantitative improvements. When trained on shape categories, these structures are often interpretable. Moreover, our deformation learning approach learns elementary structures as the deformation of continuous surfaces, resulting in output surfaces that can densely sampled and meshed at test time. Our approach opens up possibilities for other applications, such as shape morphing and scan completion.

Acknowledgments. This work was partly supported by ANR project EnHerit ANR-17-CE23-0008, Labex Bézout, and gifts from Adobe to École des Ponts.

CHAPTER 4

Learning Joint Surface Atlases

Abstract

This chapter describes new techniques for learning atlas-like representations of 3D surfaces, i.e. homeomorphic transformations from a 2D domain to surfaces. Compared to prior work, we propose two major contributions. First, instead of mapping a fixed 2D domain, such as a set of square patches, to the surface, we learn a continuous 2D domain with arbitrary topology by optimizing a point sampling distribution represented as a mixture of Gaussians. Second, we learn consistent mappings in both directions: charts, from the 3D surface to 2D domain, and parametrizations, their inverse. We demonstrate that this improves the quality of the learned surface representation, as well as its consistency in a collection of related shapes. It thus leads to improvements for applications such as correspondence estimation, texture transfer, and consistent UV mapping. As an additional technical contribution, we outline that, while incorporating normal consistency has clear benefits, it leads to issues in the optimization, and that these issues can be mitigated using a simple repulsive regularization. We demonstrate that our contributions provide better surface representation than existing baselines.

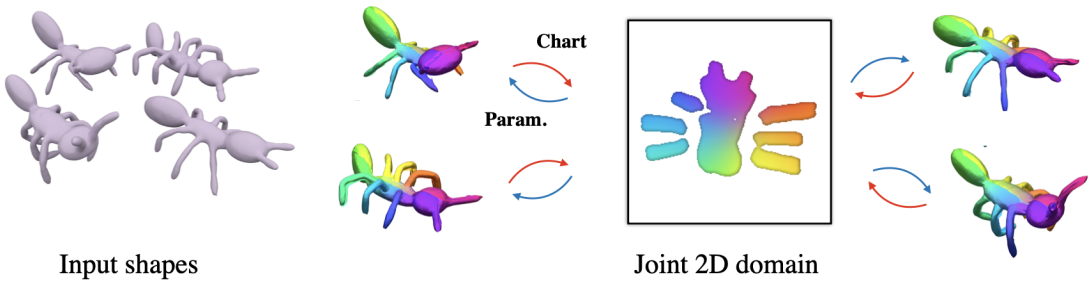


Figure 4.1: From a collection of shapes without annotations, we learn a 2D domain which can be used to parameterize all shapes, parametrizations (blue) and chart-mappings (red).

4.1 Introduction

This chapter is concerned with 3D surfaces and their representation as atlases or UV maps, i.e., their mappings to and from a domain of the 2D Euclidean space. Surfaces and altases are closely related: surfaces are 2-manifolds embedded in $\mathbb{R}^3$ and are defined in topology by the existence of *charts*, homeomorphic mappings from the surface to a 2D Euclidian domain. This relation is also central to many algorithms related to surfaces: on one hand, the computation of UV maps of meshes is a highly-active research topic: on the other hand, deep learning works have successfully used atlas-like representation and learn local parametrizations to represent 3D surfaces [Bednarik et al., 2020, Groueix et al., 2018b]. These last techniques typically learn to map a fixed sets of 2D squares to 3D, which can approximate a 3D surface and makes it possible to use the 2D domain to compute correspondences between *predicted* surfaces [Bednarik et al., 2021] for which parametrizations are learned jointly.

While these parametrization-based methods produce rather-accurate 3D surface reconstructions, they do not lead to a well-defined homeomorphic map between a 2D domain and the predicted surface, which limits the scope of applications of these techniques. For examples, when mapping a set of 2D squares to 3D, AtlasNet [Groueix et al., 2018b] leads to many overlaps, and the chart-mappings

from 3D to 2D are thus not well defined. The predicted maps from 2D to 3D might also include large amount of distortion, thus not yielding a good UV map of the generated 3D surface, and for example preventing using it to define mappings between different input surfaces. Previous works have attempted to address these limitations in various ways. Williams et al. [Williams et al., 2019] performs optimization for a single shape on local neighbourhoods with Earth Mover Distance to obtain homeomorphic parametrizations and consistent transitions maps, but this leads to an heavy optimization, with many local parametrizations, and has no obvious extension to multiple shapes. Rather than enforcing consistency between the mappings of square patches AtlasNet v2 [Deprelle et al., 2019] attempts to learn the 2D domain, e.g., by learning the positions of a fixed set of points, but loses the continuous aspect of the mapping and can still lead to overlapping patches. DSR [Bednarik et al., 2020] keeps the AtlasNet framework and its intrinsic limitations, but uses several regularizations to encourage conformal mappings, to minimize the 3D overlap between the images of the square patches and to prevent patch collapse. However, it is still limited to square patches and we found it difficult to use on complex shapes. We argue that the two tasks of parameterizations and charts prediction are complementary to one another, and that learning the 3D shape(s) reconstruction should go together with learning a relevant 2D domain. We present an architecture for such a joint optimization, where we learn the 2D domain by learning a 2D probability distribution defined as a mixture of Gaussians and from which we sample points for reconstructing the surface. These sampled points are mapped to 3D and compared (via chamfer distance) to the target point cloud; similarly, the point cloud is mapped to 2D and compared to the sampled points. We optimize for cyclic consistency between the two mappings, as well as for geometric losses such as isometric regularization. Through experiments, we show that our method is able to better reconstruct surfaces than existing baselines, in particular leading to more meaningful parametrizations with fewer artefacts and yielding meaningful

correspondences between shapes in a collection. Our code is available on our project webpage[1]

4.2 Related work

4.2.1 Optimizing charts-mappings for UV parametrization.

Identifying chart is a long-standing problem in geometry processing [Sheffer et al., 2007a]. Most prior techniques take a 2-manifold input represented as a mesh and map each point to a 2D domain. These methods typically aim to produce a bijective mapping [Smith and Schaefer, 2015], while also minimizing some distortion metric such as Dirichlet [Pinkall and Polthier, 1993b], ARAP [Liu et al., 2008], LSCM [Lévy et al., 2002], and symmetric Dirichlet [Rabinovich et al., 2017]. Some techniques also aim to predict consistent chart-mappings for a collection of related shapes, so that semantically-similar points on different meshes map to the same point in 2D. Doing so enables many applications such as correspondence estimation [Aigerman et al., 2015], morphing [Kraevoy and Sheffer, 2004], and texture transfer [Praun et al., 2001]. Unlike our method, these techniques do not use deep learning and typically require manual input, such as a sparse set of corresponding points. By using neural networks to represent the UV map we can learn consistent charts without the user input or explicit supervision. We can also co-parameterize point clouds without knowing the underlying mesh. To the best of our knowledge, our work is the first method to use neural networks to learn consistent chart-mappings.

[1] https://imagine.enpc.fr/ deprellt/joint-surface/

4.2.2 Learning surface parameterization for reconstruction.

Previous learning-based techniques for surface parameterization mostly focused on modeling 2D to 3D map for reconstruction. To learn such a mapping from a collection of shape, AtlasNet [Groueix et al., 2018b] and FoldingNet [Yang et al., 2018a] pioneered the idea of using a Multi-Layered Perceptrons (MLP) as a class of continuous parametric function that embed a 2-manifold into 3D. By conditionning the MLP on a latent code, and using a large-scale 3D shape repository [Chang et al., 2015a], AtlasNet [Groueix et al., 2018b] demonstrated the possibility to predict the surface parameterization from an input mesh, point-cloud or even a single image. Perhaps surprisingly, Williams et al. [Williams et al., 2019] showed that parameterization did not necessarily need the regularization brought by learning on a collection of shapes, and could also be optimized on individual shapes.

Several approaches introduced novel losses to improve the parameterization. Deng et al. [Deng et al., 2020] improve the global arrangement of the different parts of the parameterization via a normal-aware reconstruction loss and a stiching loss. In DSR, Bednarik et al. [Bednarik et al., 2020] regularize the smoothness of the reconstructed surface by optimizing a conformal energy, based on the Jacobians of the mappings. Remarkably, having access to analytical jacobians in a higher-order differentiation procedure opens-up exciting applications. For instance, Bednarik et al. [Bednarik et al., 2021] achieve temporally coherent parameterizations for each frame in a video by regularizing the deformation to locally have a constant metric tensor.

Particularly relevant to us is AtlasNet-v2 [Deprelle et al., 2019] which jointly optimize the shape of the 2D manifold and learns the surface parameterization using two proposed strategies. The first one consists in learning a deformation of a fixed template into an elementary structure common to all shapes via an additional AtlasNet-like MLP. This strategy leads to elementary structures that can easily be meshed, but have the same topology as the initial template. The second strategy consists in sampling a fixed set of points on the template and adding them to the

optimizer. After optimization, the sampled points can form a complex elementary structure, without topology constraints, but meshing it is not straight-forward. In this work, we propose a method to combine the best features of both strategies, namely the ability to learn topologically complex structures and the ability to mesh them. We achieve this via a novel differentiable layer to sample points from 2D gaussians with a learnable mean (see Sec. 4.3.3).

Our approach is different from AtlasNet [Groueix et al., 2018b] and its variants [Badki et al., 2020, Bednarik et al., 2021, Bednarik et al., 2020, Deng et al., 2020, Deprelle et al., 2019, Pang et al., 2021, Williams et al., 2019] in that: (i) we aim to jointly learn the surface parametrizations and their inverse functions, the charts-mappings; (ii) we learn a 2D domain relevant to a family of shapes by optimizing a probability density function in the 2D Euclidean plane. We also differ in two novel losses that correctly orient the normals of the reconstructed surface and fix point-collapse in 3D which is a common artifact of AtlasNet-type of approaches.

4.3 Method

4.3.1 Overview.

Given a collection of shapes, our goal is to learn for all shapes surface parameterizations with their inverse chart-mappings and a join 2D domain on which the parametrizations are defined. For simplicity, we first present our approach in the case of a single shape $\mathcal{S}$: in Section 4.3.2 we explain how to model surface parameterization and chart-mapping, and introduce our main architectural blocks; in Section 4.3.3 we explain how we learn the 2D UV domain as a probability distribution; in section 4.3.4 we discuss our losses. Our pipeline for a single shape is illustrated in Figure 4.2. Finally, in Section 4.3.5 we explain how to train our approach jointly on a family of shapes.

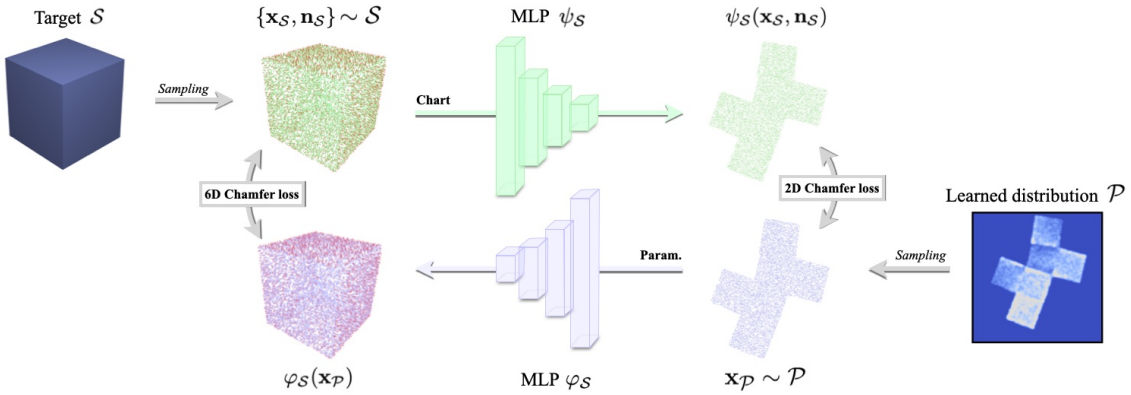


Figure 4.2: **Single shape architecture.** Given a shape $\mathcal{S}$, we optimize jointly a 2D probability distribution $\mathcal{P}$, a parametrization network, $\varphi_\mathcal{S}$ and chart-mapping network $\psi_\mathcal{S}$. The parameterisation network $\varphi_\mathcal{S}$ takes as input a point $\mathbf{x}_\mathcal{P} \in \mathbb{R}^2$ randomly sampled according to the density probability function $\mathcal{P}$ to produce the 6D vector (3D point and normal coordinates) $\varphi_\mathcal{S}(\mathbf{x}_\mathcal{P})$. The chart-mapping network $\psi_\mathcal{S}$ takes as input a point $\mathbf{x}_\mathcal{S}$ sampled on the shape $\mathcal{S}$, its normal $\mathbf{n}_\mathcal{S}$ and outputs a 2D points $\psi_\mathcal{S}(\mathbf{x}_\mathcal{S}, \mathbf{n}_\mathcal{S})$.

Notations. We use the following notations:

- $\mathcal{S}$: 3D shape of interest
- $\mathcal{P}$: learned probability distribution in 2D
- $\phi_\mathcal{S} : \mathbb{R}^2 \to \mathbb{R}^3$: surface parameterization
- $\varphi_\mathcal{S} : \mathbb{R}^2 \to \mathbb{R}^6$: surface parameterization with normals
- $\psi_\mathcal{S} : \mathbb{R}^6 \to \mathbb{R}^2$: chart-mapping

We slightly abuse the brackets notation to indicate sampling a set of M points, e.g.:

- $\{\mathbf{x}_\mathcal{S}\}$ is a set of M points sampled on $\mathcal{S}$ using a uniform probability distribution
- $\{\mathbf{x}_\mathcal{P}\}$ is a set of M points sampled in $\mathbb{R}^2$ according the probability distribution $\mathcal{P}$

4.3.2 Parametrization and chart-mapping

We explain the two main components of our architecture for a given shape $\mathcal{S}$, a chart-mapping $\psi_{\mathcal{S}}$ and a surface parameterization network $\varphi_{\mathcal{S}}$.

Chart-mapping. We learn a single chart-mapping from the shape $\mathcal{S}$ to $\mathbb{R}^2$, using a Multi-Layer Perceptron (MLP) with both point coordinates and normals as input. We observe that naively learning an $\mathbb{R}^3$ to $\mathbb{R}^2$ mapping leads to the collapse of thin surfaces after their mapping to the 2D domain. Indeed, coordinate-based MLPs are continuous functions and Tancik et. al. [Tancik et al., 2020] showed that they have a prior to learn smooth functions in the absence of positional encoding. Hence, two 3D points with very close spatial coordinates but opposite normals tend to be mapped closely in 2D. For chart-mappings, such smoothness is generally a desirable feature that should be maintained, but distant normals should be a strong cue to indicate that points are intrinsically far. To handle thin surfaces, we propose to learn a mapping $\psi_{\mathcal{S}}$ from $\mathbb{R}^6$ to $\mathbb{R}^2$ that takes as input a point $\mathbf{x}_{\mathcal{S}}$ and its associated normal $\mathbf{n}_{\mathcal{S}}$ scaled to have norm α. The parameter α is a hyperparameter of our approach which controls how much normals contribute to distances compared to the 3D positions of the points.We set it to 0.01 in all our experiments.

Surface parameterization. For a shape $\mathcal{S}$, we seek to learn the inverse function of the chart-mapping, $\psi_{\mathcal{S}}^{-1} : \mathbb{R}^2 \to \mathbb{R}^6$. We first use an MLP $\phi_{\mathcal{S}} : \mathbb{R}^2 \to \mathbb{R}^3$ to parameterize the surface $\mathcal{S}$. The function $\phi_{\mathcal{S}}$ maps a point $\mathbf{x} = (u, v)$ in $\mathbb{R}^2$ into a point $\phi_{\mathcal{S}}(\mathbf{x})$ in $\mathbb{R}^3$. The Jacobian $\mathbf{J}_{\mathbf{x}}$ of the mapping $\phi_{\mathcal{S}}$ is defined at every point $\mathbf{x}$ by:

$$\mathbf{J}_{\mathbf{x}} = \left[\mathbf{J}_{\mathbf{x},u}, \mathbf{J}_{\mathbf{x},v}\right] = \left[\frac{\partial \phi_{\mathcal{S}}}{\partial u}(\mathbf{x}), \frac{\partial \phi_{\mathcal{S}}}{\partial v}(\mathbf{x})\right] . \tag{4.1}$$

The two partial derivatives are trivial to compute with Pytorch auto-differentiation [Paszke et al., 2019]. The normal $\mathbf{n}$ to the parametrizedsurface at $\phi_{\mathcal{S}}(\mathbf{x})$ can be computed as the normalised cross product of $\mathbf{J}_{\mathbf{x},u}$ and $\mathbf{J}_{\mathbf{x},v}$ and scaled by the same hyperparameter

α used for the chart-mapping $\psi_{\mathcal{S}}$:

$$\mathbf{n} = \alpha \frac{\mathbf{J}_{\mathbf{x},u} \times \mathbf{J}_{\mathbf{x},v}}{\|\mathbf{J}_{\mathbf{x},u} \times \mathbf{J}_{\mathbf{x},v}\|}. \tag{4.2}$$

We define $\varphi_{\mathcal{S}} : \mathbb{R}^2 \to \mathbb{R}^6$ as the concatenation of an output point and its scaled normal:

$$\varphi_{\mathcal{S}}(\mathbf{x}) = [\phi(\mathbf{x}), \mathbf{n}] \ . \tag{4.3}$$

To summarize, the mapping $\varphi_{\mathcal{S}}$ is designed to represent the inverse of the chart-mapping $\psi_{\mathcal{S}}$. However, it is not defined for every point in $\mathbb{R}^2$, and in the next section we focus on identifying the 2D domain for which it is defined.

4.3.3 Learning a 2D domain as a sampling probability distribution

To parametrize a shape $\mathcal{S}$, we want to define a domain in $\mathbb{R}^2$ such that $\phi_{\mathcal{S}}$ defines a bijection from this 2D domain to $\mathcal{S}$. In practice, during training, we want to learn this domain, sample points inside it, and map them using $\varphi_{\mathcal{S}}$. Instead of handling explicitly the 2D domain geometry, e.g., points or primitives similar to [Deprelle et al., 2019], we take a probabilistic approach and learn the parameters of a probability distribution $\mathcal{P}$ from which to sample points. This enables us to easily deal with topological changes. We now detail how we represent this probability distribution, learn it, and use it to define a 2D domain.

Modelling 2D sampling probability as a mixture of Gaussians. We sample 2D points according to a probability distribution $\mathcal{P}$ which we model as a mixture of K 2D Gaussians with means $\mu_i \in \mathbb{R}^2$ for $i = 1, ..., K$, a fixed standard deviation $\sigma \in \mathbb{R}$ and fixed mixing coefficients equal to $1/K$:

$$\mathcal{P}(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}(\mathbf{x}|\mu_i, \sigma) \quad \text{with} \quad \sigma = \frac{1}{\sqrt{K}} \ . \tag{4.4}$$

During training, at every iteration, we sample N 2D points $\{\mathbf{x}_\mathcal{P}\}$ from $\mathcal{P}$, which are both the input of the parameterisation network $\varphi_\mathcal{S}$ and the target of the chart-mapping network $\psi_\mathcal{S}$.

Learning the Gaussian means. To allow learning the 2D domain, we make the means μ_i learnable parameters of the method. To do so, we need to see the sampled points as differentiable with respect to the μ_i. We achieve this with the pathwise gradient estimator from [Kingma and Welling, 2013], also called the reparameterization trick. This consist in expressing a parameterized random variable via a parameterized deterministic function of a parameter-free random variable. For Gaussian Mixture Models (GMM), this simply amounts to sampling a GMM with zero means and adding the means to the sampled points, i.e., defining each point $\mathbf{x}_\mathcal{P}$ as the result of the following process: first selecting the id i of a mixture component using a uniform distribution; then sampling a 2D point $\mathbf{x}_\mathcal{N}$ from a Gaussian distribution of mean 0 and standard deviation sigma $\mathbf{x}_\mathcal{N} \sim \mathcal{N}(0, \sigma)$; finally, defining the point $\mathbf{x}_\mathcal{P}$ as $\mathbf{x}_\mathcal{P} = \mathbf{x}_\mathcal{N} + \mu_i$, which is trivially differentiable with respect to μ_i. Please see Figure 4.5 for examples of learned probability distributions.

From probability distribution to 2D continuous domain. Once the distribution has been optimized, we simply threshold the probability distribution function $\mathcal{P}$ to obtain a 2D domain. We can then compute a 2D triangulation of the domain, and use the parameterization network ϕ to obtain a 3D mesh (as can be seen in Figure 4.3).

4.3.4 Training losses

We now explain the loss function we optimize. We write our loss for a single shape:

$$\mathcal{L}_{single}(\varphi_\mathcal{S}, \psi_\mathcal{S}, \mu) = \lambda_{\text{6D}}\mathcal{L}_{\text{6D}} + \lambda_{\text{2D}}\mathcal{L}_{\text{2D}} + \lambda_{\text{cycle}}\mathcal{L}_{\text{cycle}} + \lambda_{\text{iso}}\mathcal{L}_{\text{iso}} + \lambda_{\text{rep}}\mathcal{L}_{\text{rep}} \ , \quad (4.5)$$

where $\mu = (\mu_1, ..., \mu_K)$, the λ are scalar hyper-parameters and the different loss terms are detailed bellow. To compute distances between two sets of points $\mathcal{X}$ and $\mathcal{Y}$, we base our losses on the Chamfer distance defined as:

$$\mathcal{L}_{\text{chamfer}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{y} \in \mathcal{Y}} \|\mathbf{x} - \mathbf{y}\|^2 + \frac{1}{|\mathcal{Y}|} \sum_{\mathbf{y} \in \mathcal{Y}} \min_{\mathbf{x} \in \mathcal{X}} \|\mathbf{y} - \mathbf{x}\|^2 \ , \tag{4.6}$$

where $|\mathcal{X}|$ and $|\mathcal{Y}|$ are the number of points in $\mathcal{X}$ and $\mathcal{Y}$ respectively. Our set of losses is designed to enforce two objectives : (i) ensuring an accurate surface parameterization $\varphi_{\mathcal{S}}$ with low distortion, (ii) ensuring that $\varphi_{\mathcal{S}}$ and $\psi_{\mathcal{S}}$ are indeed inverse of each other.

Surface parameterization reconstruction loss. The surface parameterization $\varphi_{\mathcal{S}}$ takes as input a set of 2D points $\{\mathbf{x}_{\mathcal{P}}\}$ sampled from the probability distribution $\mathcal{P}$ and outputs a set of 6D points (3D points with scaled normals). We minimize the 6D Chamfer distance between the set of generated points $\{\varphi_{\mathcal{S}}(\mathbf{x}_{\mathcal{P}})\}$ and a set of points $\{\mathbf{x}_{\mathcal{S}}\}$ associated to normals $\{\mathbf{n}_{\mathcal{S}}\}$ sampled on the target shape $\mathcal{S}$:

$$\mathcal{L}_{\text{6D}}(\varphi, \mu) = \mathcal{L}_{\text{chamfer}}(\{\varphi_{\mathcal{S}}(\mathbf{x}_{\mathcal{P}})\}, \{\mathbf{x}_{\mathcal{S}}, \mathbf{n}_{\mathcal{S}}\}) \ . \tag{4.7}$$

Chart-mapping reconstruction loss. We encourage the overall 2D projection of the shape under $\psi_{\mathcal{S}}$ and the probability distribution $\mathcal{P}$ to be the same. Recall that $\psi_{\mathcal{S}}$ takes as input $\{\mathbf{x}_{\mathcal{S}}, \mathbf{n}_{\mathcal{S}}\}$ (3D points uniformely sanpled on $\mathcal{S}$ with scaled normals) and outputs a set of 2D points $\{\psi_{\mathcal{S}}(\mathbf{x}_{\mathcal{S}}, \mathbf{n}_{\mathcal{S}})\}$. We simply minimize the Chamfer distance between the generated 2D points and a set of points $\{\mathbf{x}_{\mathcal{P}}\}$ sampled from $\mathcal{P}$:

$$\mathcal{L}_{\text{2D}}(\psi, \mu) = \mathcal{L}_{\text{chamfer}}(\{\psi_{\mathcal{S}}(\mathbf{x}_{\mathcal{S}}, \mathbf{n}_{\mathcal{S}})\}, \{\mathbf{x}_{\mathcal{P}}\}) \ . \tag{4.8}$$

Cycle-consistency loss. We want the two mappings φ and ψ to be inverse of one another. We encourage this using a cycle-consistency loss on 2D points sampled

from $\mathcal{P}$ and on 3D points sampled uniformly on $\mathcal{S}$ with their associated normals:

$$\mathcal{L}_{\text{cycle}}(\varphi, \psi, \mu) = \frac{1}{M} \sum_{\mathbf{x} \in \{\mathbf{x}_{\mathcal{P}}\}} \|\mathbf{x} - \psi \circ \varphi(\mathbf{x})\|^2 + \frac{1}{M} \sum_{(\mathbf{x},\mathbf{n}) \sim \{\mathbf{x}_{\mathcal{S}}, \mathbf{n}_{\mathcal{S}}\}} \|\mathbf{x} - \varphi \circ \psi(\mathbf{x}, \mathbf{n})\|^2 \ . \quad (4.9)$$

Distortion regularization loss. We limit distortion in the parameterization with an isometric regularization. Given the Jacobian $\mathbf{J}_{\mathbf{x}}$ of the transformation ϕ at point $\mathbf{x}$ (Equation 4.1) and $\mathbf{I}$ the identity matrix, the isometric loss can be written:

$$\mathcal{L}_{\text{iso}}(\varphi_{\mathcal{S}}, \mu) = \frac{1}{M} \sum_{\mathbf{x} \in \{\mathbf{x}_{\mathcal{P}}\}} \|\mathbf{J}_{\mathbf{x}} \mathbf{J}_{\mathbf{x}}^T - \mathbf{I}\| \ , \quad (4.10)$$

where the sum is over points sampled according to $\mathcal{P}$. As already observed in [Bednarik et al., 2021, Groueix et al., 2018a], this type of regularization has the additional benefit of making the parameterizations more consistent across shapes.

Probability distribution regularization loss. Non-uniform density is a known failure mode of the Chamfer distance, as shown in Figure 4.5 and also observed in [Bednarik et al., 2020]. In theory, a loss based on optimal transport like the Earth Mover distance would be ideal to fix this problem. However, in practice, we ran into optimisation, training time and parameter tuning issues when using EMD. On the contrary the Chamfer loss is simple to use and fast to compute. We thus use the Chamfer loss and introduce a repulsive loss between the Gaussian means defining the probability distribution $\mathcal{P}$ as a regularization:

$$\mathcal{L}_{\text{rep}}(\mu) = \frac{1}{K^2} \sum_{i,j \in [0,K]} \exp(-\frac{\|\mu_i - \mu_j\|}{\sigma}) \ , \quad (4.11)$$

where σ is the Gaussian standard deviations in the definition of $\mathcal{P}$. We found that this simple loss lead to much more uniform distributions of points both in the 2D plane and in the reconstructed shapes.

4.3.5 Joint learning on a family of shapes.

We now explain how we learn jointly atlases on a collection of N shapes $\mathcal{S}_1, ..., \mathcal{S}_N$. Since we want to share the 2D domain between the different shapes, we do not condition the probability distribution $\mathcal{P}$ on the shape, and learn a single one for all shapes. On the contrary, the parametrization and chart-mappings are expected to depend on the shape. Rather than learning them completely independently for each shape, we use the auto-decoder framework [Park et al., 2019], where we optimize for each shape $\mathcal{S}_i$ feature vectors $\mathbf{f}_{\psi,i}$ and $\mathbf{f}_{\varphi,i}$ which we use to define respectively $\varphi_{\mathcal{S}_i}$ and $\psi_{\mathcal{S}_i}$. More precisely, we learn jointly for all shapes networks φ and ψ, and define for each shape $\mathcal{S}_i$ for all $\mathbf{x} \in \mathbb{R}^2$, $\varphi_{\mathcal{S}_i}(\mathbf{x}) = \varphi(\mathbf{x}, \mathbf{f}_{\psi,i})$ and for all $\mathbf{x} \in \mathbb{R}^6$, $\psi_{\mathcal{S}_i}(\mathbf{x}) = \psi(\mathbf{x}, \mathbf{f}_{\psi,i})$. We then optimize ψ, φ and μ by minimizing the loss:

$$\mathcal{L}_{\text{full}}(\varphi, \psi, \mu, \mathbf{f}_\psi, \mathbf{f}_\varphi) = \sum_{i=1}^{N} \mathcal{L}_{\text{single}}(\varphi_{\mathcal{S}_i}, \psi_{\mathcal{S}_i}, \mu) \ , \tag{4.12}$$

where $\mathbf{f}_\psi = (\mathbf{f}_{\psi,1}, ..., \mathbf{f}_{\psi,N})$ and $\mathbf{f}_\varphi = (\mathbf{f}_{\varphi,1}, ..., \mathbf{f}_{\varphi,N})$.

Implementation details. We use latent codes $\mathbf{f}_{\varphi,i}$ and $\mathbf{f}_{\psi,i}$ of dimension 256. The architectures we use for φ and ψ are MLP with 5 hidden layers of size 256 and ReLU activations. We do not use batch normalization layers. $\mathcal{P}$ is defined using $K = 10,000$ mixture components. We sample $M = 10^4$ points both on the shapes and according to $\mathcal{P}$ at every training iteration. We train with $\lambda_{\text{6D}} = 1$, $\lambda_{\text{2D}} = 10^{-2}$, $\lambda_{\text{cycle}} = 1/M$, $\lambda_{\text{iso}} = 10^{-4}/M$ and $\lambda_{\text{rep}} = 1$.

4.4 Experiments

Datasets. We use individually the shapes of Williams et al. [Williams et al., 2019], which come from five high-resolution scans with over a million points with associated normals. We generate the manifold meshes from this data using screened Poisson Surface Reconstruction. The resulting meshes have a variety of geometric

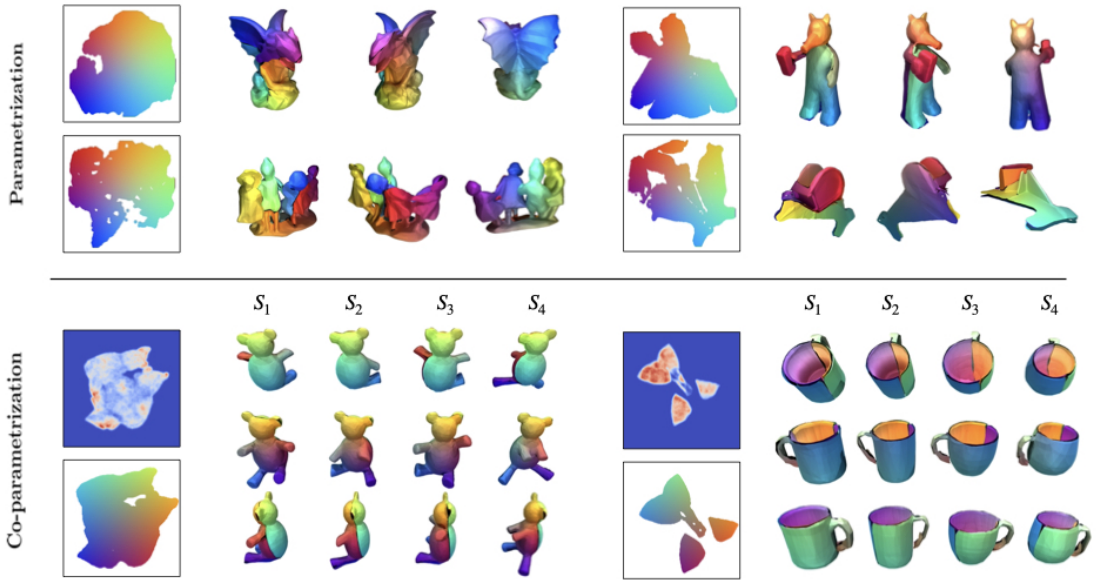


Figure 4.3: **Top: single shape parametrization.** On the shapes of [Williams et al., 2019], we transfer a colored mesh of the learned 2D domain (left) to 3D, which enables visualizing correspondence and cuts (right). **Bottom: co-parameterization.** For 'teddy' and 'cup' shapes of SHREC [Giorgi et al., 2007], we show the joint sampling probabilities and 2D domain, and the reconstructions of 4 shapes with 3 different viewpoints and consistent coloring. Note the quality of the parametrization compared to patch-based methods [Bednarik et al., 2020, Groueix et al., 2018b]

details and different topologies, providing interesting challenges for atlas-based representations. The SHREC dataset [Giorgi et al., 2007] contains 400 manifold meshes with sparse correspondence annotations, which enables us to quantitatively evaluate the consistency of our joint atlases. For our experiments we selected categories, *ant*, *teddy*, *cups* and *armadillo*, aiming for topological and geometric diversity, and four shapes in each.

Reconstruction Metrics. To evaluate how well our representation matches the input shape, we report two commonly used metrics: 3D Chamfer Distance and the Earth Mover's Distance (EMD). Given two sets of points $\mathcal{X}$ and $\mathcal{Y}$ with $M' = 2000$

randomly-sampled points each, we approximate the EMD as:

$$\mathcal{L}_{\text{EMD}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{M'} \sum_{\mathbf{x}_i \in \mathcal{X}} \sum_{\mathbf{y}_j \in \mathcal{Y}} C_{i,j} \|\mathbf{x}_i - \mathbf{y}_j\|^2 \ , \tag{4.13}$$

where the association matrix C is such that $C_{i,j} = 1$ if point $\mathbf{x}_i$ is mapped to point $\mathbf{y}_j$ and $C_{i,j} = 1$, and is computed to minimize $\mathcal{L}_{\text{EMD}}$ via Hungarian Algorithm [Kuhn, 1955]. To evaluate the reconstructions of the normals of the surfaces, we also report the distances of the normals using the associations given either by the chamfer distance or the EMD using the spatial coordinates.

Correspondence Metric. Given a pair of shapes $\mathcal{S}_1, \mathcal{S}_2$ and a set of keypoints $\mathbf{p}_i \in \mathcal{S}_1$, we find the corresponding points $\mathbf{q_i} \in \mathcal{S}_2$ by compositing the chart-mapping and parameterization networks: $\mathbf{q}_i = \varphi_{\mathcal{S}_2} \circ \psi_{\mathcal{S}_1}(\mathbf{p}_i)$. For the baselines that do not have a chart-mapping network, we obtain correspondences for each point $\mathbf{p}_i$ using the following process: we sample points according to $\mathcal{P}$, map them to $\mathcal{S}_1$, select the one which image by $\varphi_{\mathcal{S}_1}$ is closest to $\mathbf{p}_i$, and map it to $\mathcal{S}_2$ using $\varphi_{\mathcal{S}_2}$. Given M'' ground truth correspondences $\mathbf{q}_i^{\text{gt}}$ for each $\mathbf{p_i}$, we measure the L2 loss on the spatial coordinates as $\frac{1}{M''} \sum_i |\mathbf{q_i} - \mathbf{q_i^{\text{gt}}}|$.

Results and analysis. Qualitative results for individual shapes of Williams et al. [Williams et al., 2019] are shown in Figure 4.3 (top). Joint atlases for the SHREC dataset [Giorgi et al., 2007], can be seen for the 'ants' are in the teaser Figure 4.1 and for 'cup' and 'teddy' in Figure 4.3 (bottom). Note how we manage to learn continuous 2D domains with complex topology, which we can mesh to obtain a high quality 3D mesh for the shapes. Also note that compared to the patch-based approaches we obtain few and meaningful cuts in the parametrization, without any overlap. Finally, note the consistency in the reconstruction of the different shapes, which can be visualized by consistent colors transfered from the joint 2D domain to all shapes.

		[Williams et al., 2019] Spatial Ch. ↓	Spatial Emd ↓	Normal Ch. ↓	Normal Emd ↓	[Giorgi et al., 2007] Spatial Ch. ↓	Spatial Emd ↓	Normal Ch. ↓	Normal Emd ↓	Corresp. L2 ↓
Baselines										
(1)	ANv1-1	2.3±0.0	**5.0**±0.1	1.2±0.2	1.2±0.2	2.2±0.0	**5.4**±0.5	1.1±0.2	1.1±0.2	2.6±0.8
(2)	ANv2-1	**2.2**±0.0	11.1±2.1	1.2±0.1	1.3±0.1	**2.0**±0.0	7.1±1.4	1.3±0.2	1.3±0.2	1.3±0.1
(3)	ANv1-10	2.4±0.0	11.4±0.9	1.1±0.0	1.1±0.0	2.3±0.1	11.1±0.4	1.1±0.1	1.1±0.1	3.5±1.4
(4)	ANv2-10	2.2±0.0	11.1±2.1	1.2±0.1	1.3±0.1	2.1±0.0	13.4±0.5	1.0±0.1	1.1±0.1	2.8±1.6
(5)	DSR-10	4.3±0.8	18.2±0.8	1.4±0.0	1.2±0.0	3.1±1.1	10.0±2.8	1.3±0.1	1.3±0.1	2.6±1.2
Ours										
(6)	ϕ	**2.2**±0.0	15.2±5.9	1.0±0.2	1.1±0.2	**2.0**±0.0	6.8±0.9	1.1±0.2	1.1±0.2	2.0±0.8
(7)	ϕ, n	2.4±0.1	20.3±3.3	0.3±0.0	0.7±0.0	2.1±0.3	13.1±8.0	**0.2**±0.0	0.5±0.0	1.4±1.1
(8)	ϕ, n, r	**2.2**±0.1	12.9±4.4	**0.2**±0.1	0.6±0.1	**2.0**±0.1	9.6±6.6	**0.2**±0.0	**0.4**±0.1	2.8±0.9
(9)	ϕ, n, r, p	2.3±0.2	13.2±4.0	0.3±0.1	0.6±0.1	**2.0**±0.0	8.2±2.1	**0.2**±0.0	**0.4**±0.0	1.5±1.0
(10)	ϕ, ψ, n, r, p	2.6±0.0	**5.1**±0.1	**0.2**±0.0	**0.5**±0.0	2.2±0.0	**5.4**±0.2	**0.2**±0.0	**0.4**±0.0	1.0±1.0
(11)	ϕ, ψ, n, r, p, i	2.6±0.2	5.5±1.3	**0.2**±0.3	**0.5**±0.2	2.3±0.1	5.9±0.6	**0.2**±0.0	**0.4**±0.0	**0.8**±1.0

Notations n: normal - r: repulsion loss - p: Probability distribution function $\mathcal{P}$ - i: Isometric regularisation of φ}

Table 4.1: **Comparison and ablation study.** We compare our method, successively adding its different componenets, to AtlasNet [Groueix et al., 2018b], AtlasNetV2 [Deprelle et al., 2019] and DSR [Bednarik et al., 2020]. We compute association between target and reconstructed 3D points using Chamfer distance and EMD, then report for each association the average distance between the points coordinates ('Spatial') and their normals ('Normals'). We also evaluate the quality of the correspondences when available. All the values are scaled by a factor of 10^{-2}. Please see text for details.

We report quantitative results in Table **??**. To account for randomness in initialization and optimization, we re-run each method four times and report mean and standard deviation. We report results for three baselines: vanilla AtlasNet [Groueix et al., 2018b] (rows 1 and 3) which only learns the parameterization network, AtlasNet v2-points [Deprelle et al., 2019] (rows 2 and 4) which learns elementary point structures in the 2D domain, and DSR [Bednarik et al., 2020] (row 5) which uses several regularisation losses including an isometry loss. We try the first two methods with 1 and 10 UV patches. We found that methods using a single patch typically lead to similar Chamfer distances but much lower EMD. We believe that this is because using a single patch encourages points to be more uniformly distributed on the surface. We also found that DSR [Bednarik et al., 2020] provided quantitativelty slightly worse results, which is consistent with what was reported in this chapter that mainly aims at visual quality.

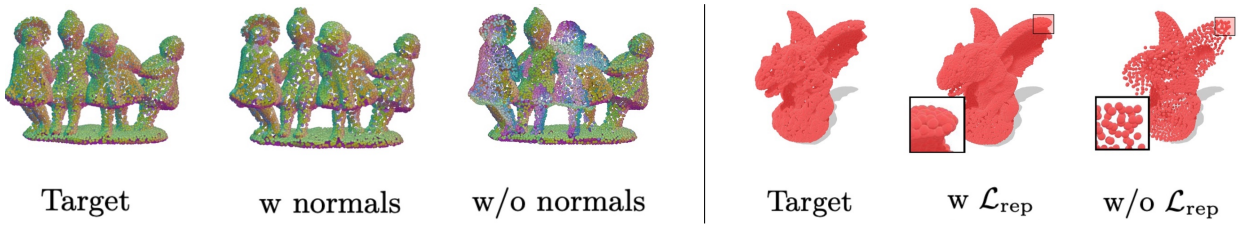


Figure 4.4: **Impact of the normals and the repulsion loss.** Note the variation of the color-coded normal directions (left) and the variations in the point density (right).

We evaluate our method by successively adding our different components. We start (row 6) by only using our parametrization network, learning a fixed set of input point positions with only 3D Chamfer distance as supervision, similar to AtlasNet v2 with 1 patch and obtained similar results.

We then add normals to the Chamfer loss (row 7), which unsurprisingly gives a very strong boost to the normal metrics. Qualitatively, the effect can be seen in the left part of Figure 4.4 where the orientation of the normals is color-coded for each point: without the normals in the loss, a significant part of the normals are back-facing. Adding the normals in the loss however has a second undesired effect and significantly increases the EMD. This can be understood qualitatively by looking at the results, where we see greatly varying density of points over the reconstructed shapes and in the 2D space. We interpret this as the fact that adding normals in the loss complexifies the loss landscape and adds bad local minima.

To avoid this effect, we add our repulsive regularization to the loss (row 8). We can see this improves the EMD results without degrading the Chamfer results and normal consistency. The effect on the points density is also striking qualitatively, as visualized in the right part of Figure 4.4.

At this point, the parametrization is still only optimized on a set of points. To recover an interpretation of the parametrization as a continuous mapping of a 2D domain, we introduce our sampling probability distribution (row 9) which has little

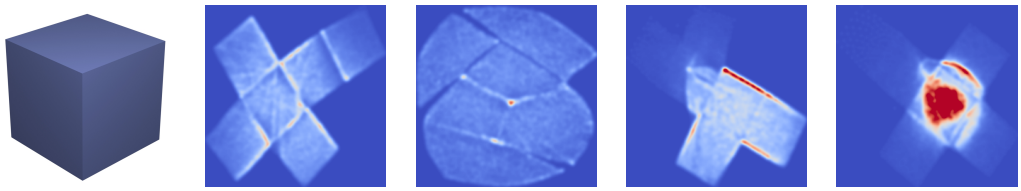


Figure 4.5: **Qualitative ablation results.** Given a target cube shape, we show the probability distribution $\mathcal{P}$ learned by our method and three key ablations: not using the isometric loss, not using the normals, and not using the repulsive loss.

quantitative effect but is crucial to define a continuous 2D mapping.

We can then introduce our chart-mapping, together with the 2D reconstruction and cycle losses (row 10). It can be seen that beside its theoretical interest and benefits, it has a clear quantitative impact: both the EMD and the correspondence metrics are clearly boosted.

Finally, adding the isometric regularization (row 11) provides a small additional boost in term of correspondence quality, i.e., consistency between different jointly learned atlases.

Note that all of our regularization losses remain necessary to the success of our method, as can be visualized qualitatively in Figure 4.5, where we use our architecture to learn the reconstruction of a simple cube and visualize the learned 2D sampling distribution. Without the normal loss, we observe a concentration of the density on edges, an equivalent of point collapse described in [Bednarik et al., 2020]. Without the repulsion loss, the density is concentrated on a single face. Without the isometry loss, the shape of the cube is highly distorded. FWith our full method the net of the cube is clear and the point density relatively uniform.

4.5 Conclusion

We propose a novel technique for representing a shape or a collection of shapes, with two key differences with respect to prior work on atlas-like representations. First, we joinlty learn two maps, a parameterization and a chart-mapping. Second, we learn the 2D domain on which the parametrization is defined. This makes our representation much closer than previous works to be a proper atlas, i.e., one that defines an homeomorphism between 3D shape and a 2D domain. It also enables co-parameterization, where homeomorphisms are learned between several shapes and a single 2D domain, which can have applications for consistent texturing and correspondence estimation. We further offer other technical contributions, such as learning the 2D domain by optimizing a sampling probability distribution, analyzing the effect of incorporating normals in the optimization, and introducing a repulsive loss to have a more even point distribution. We believe our work is an important step towards learning consistent atlases and it will inspire future work on further improving quality of atlases, such as modeling transition maps, minimizing seams, and distortion.

Acknowledgments Thanks to F. Darmon, R. Loiseau, E. Vincent for their feedbacks on the manuscript; E. Shechtman , D. Picard, Y. Siglidis and G. Ponimatkin for inspiring discussions; and B. George for code suggestions; This work was partly supported by ANR project EnHerit ANR-17-CE23-0008, Labex Bézout, gifts from Adobe to École des Ponts and HPC resources from GENCI-IDRIS (2021-AD011011937R1).

CHAPTER 5

Conclusion

5.1 Summary of the contributions

We proposed several approaches using deformable elements for surface representations and used it to develop more interpretable model of shape generation.

- **In chapter 3** we introduced *elementary structures* that are learned on a collection of shape. We proposed two models of elementary structures : **(i)** *patch deformation module* and **(ii)** *point learning module*. The first learns parametric surfaces that enable to generate continuous surfacic representation of the collection of shapes. The second learns a parametric set of points which are not constrained by topology hence allowing us to generate more complex elementary structures. Our approaches improved the quality of the reconstruction of the ShapeNet dataset [Chang et al., 2015a] and pushed futher the state of the art of shape correspondences on the Faust benchmark [Bogo et al., 2014]. We developed several methods as well, to adjust the elementary structures together in order to reconstruct the target shape surface. Such

adjustment can be seen as continuous mapping functions from the elementary structure space to $\mathbb{R}^3$. They can be continuous non linear mapping defined by a set of MLPs or linear function whose coefficients and intercepts are predicted using a different set of MLPs. While the non linear mapping yields overall a better reconstruction quality, they lack interpretability. On the opposite, the linear mappings produce more interpretable results with a lower reconstruction quality.

- **In chapter 4** we proposed to learn 2D domain for the parameterization by representing it it as a probability distribution function. We saw that this novel approach allows to learn a surfacic representation free of topological constraints. We modeled the method after atlases and its defining elements, namely a 2D domain, a parameteriziation and a chart and we propose to learn them all together. The parameteriziation and chart functions are defined by continuous non linear functions defined as MLPs. We showed that learning the chart improves the parameterization by reducing the number of overlaps and by improving the consistency. This method enables to push further the quality of the parameterization techniques and atlas-net approaches [Groueix et al., 2018b, Williams et al., 2019, Bednarik et al., 2020].

5.2 Future work

On elementary structures In Chapter 3 we detailed a method to learn elementary structures to generate accurate, meaningful, and interpretable shape surface reconstruction. They were introduced to replace the hard coded 2D domain representation of AtlasNet [Groueix et al., 2018b] pipeline. Yet, several other elements are still hard coded that could be replaced and learned instead. In particular, the fact that we use a fixed set of elementary structures and that we cannot use one of them multiple times is a limitation of the approach. Having a model that can learn a set of

interpretable elementary structures and pick from them, potentially multiple times, to reconstruct a given shape is an interesting direction for future work. Monier et al. developed this idea [Monnier et al., 2021] on an image generation task but it is yet to be done on a 3D pipeline.

On joint surface parameterization The pipeline we described in Chapter 4 was initially thought of as a UV-map generation technique where a collection of meshes were to be parameterized by a single 2D domain. For joint texturing purposes or any other parameterization-related task, the learned parameterization function needs to be as close as possible to a bijection. In practice, this was not the case, and we observed several issues regarding cuts, triangle overlap, and overall distortion that did not meet the quality standard of classical approaches. Many of the problems we faced came from the fact that MLPs are by definition continuous and that we need a discreet linear mapping to represent a mesh triangulation of a shaped surface in 2D. For instance, two vertices with similar 3D coordinates that are separated by a cut will be mapped far away on the 2D domain. Such mapping is particularly hard to learn with a neural network. In our case, it introduced many overlaps and flipped faces. We understand that representing a chart function using an MLP or any continuous function from 3D to 2D is difficult. While in theory learning a model to predict a chart and a UV-map of a given mesh is interesting, it is not clear that the quality of such a parameterization technique will ever match one of the methods that parameterize a single shape surface. After this thesis, the question remains unanswered and we believe it could be another interesting research topic.

Bibliography

[Ico, 2022] (2022). Iconem project. `https://iconem.com`.

[Kat, 2022] (2022). Katsukagi textures. `https://3dtextures.me`.

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2015). Tensorflow: Large-scale machine learning on heterogeneous systems.

[Aigerman et al., 2015] Aigerman, N., Poranne, R., and Lipman, Y. (2015). Seamless surface mappings. *ACM SIGGRAPH*.

[Allen et al., 2002] Allen, B., Curless, B., and Popovic, Z. (2002). Articulated body deformation from range scan data. *SIGGRAPH*.

[Allen et al., 2003a] Allen, B., Curless, B., and Popović, Z. (2003a). The space of human body shapes: reconstruction and parameterization from range scans. *ACM transactions on graphics (TOG)*, 22(3):587–594.

[Allen et al., 2003b] Allen, B., Curless, B., and Popovic, Z. (2003b). The space of human body shapes: reconstruction and parameterization from range scans. *SIGGRAPH*.

[Alliez et al., 2008] Alliez, P., Ucelli, G., Gotsman, C., and Attene, M. (2008). Recent advances in remeshing of surfaces. *Shape analysis and structuring*, pages 53–82.

[Anguelov et al., 2005] Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., and Davis, J. (2005). Scape: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, pages 408–416.

[Ashburner and Friston, 2000] Ashburner, J. and Friston, K. J. (2000). Voxel-based morphometry—the methods. *Neuroimage*, 11(6):805–821.

[Ashburner and Friston, 2005] Ashburner, J. and Friston, K. J. (2005). Unified segmentation. *Neuroimage*, 26(3):839–851.

[Badki et al., 2020] Badki, A., Gallo, O., Kautz, J., and Sen, P. (2020). Meshlet priors for 3d mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2849–2858.

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

[Bednarik et al., 2021] Bednarik, J., Kim, V. G., Chaudhuri, S., Parashar, S., Salzmann, M., Fua, P., and Aigerman, N. (2021). Temporally-coherent surface reconstruction via metric-consistent atlases. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10458–10467.

[Bednarik et al., 2020] Bednarik, J., Parashar, S., Gundogdu, E., Salzmann, M., and Fua, P. (2020). Shape reconstruction by learning differentiable surface representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4716–4725.

[Bennis et al., 1991] Bennis, C., Vézien, J.-M., and Iglésias, G. (1991). Piecewise surface flattening for non-distorted texture mapping. *ACM SIGGRAPH computer graphics*, 25(4):237–246.

[Biederman, 1987] Biederman, I. (1987). Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115.

[Biermann et al., 2002] Biermann, H., Martin, I., Bernardini, F., and Zorin, D. (2002). Cut-and-paste editing of multiresolution surfaces. *ACM Transactions on Graphics (TOG)*, 21(3):312–321.

[Binford, 1971] Binford, I. (1971). Visual perception by computer. In *IEEE Conference of Systems and Control*.

[Blinn, 1978] Blinn, J. F. (1978). Simulation of wrinkled surfaces. *ACM SIGGRAPH computer graphics*, 12(3):286–292.

[Bogo et al., 2014] Bogo, F., Romero, J., Loper, M., and Black, M. J. (2014). Faust: Dataset and evaluation for 3d mesh registration. In *CVPR*.

[Chang et al., 2015a] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. (2015a). Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*.

[Chang et al., 2015b] Chang, A. X., Funkhouser, T. A., Guibas, L. J., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015b). Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012.

[Chum and Matas, 2005] Chum, O. and Matas, J. (2005). Matching with prosac-progressive sample consensus. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 220–226. IEEE.

[Cohen-Steiner et al., 2004] Cohen-Steiner, D., Alliez, P., and Desbrun, M. (2004). Variational shape approximation. In *ACM SIGGRAPH 2004 Papers*, pages 905–914.

[Collins and Stephenson, 2003] Collins, C. R. and Stephenson, K. (2003). A circle packing algorithm. *Computational Geometry*, 25(3):233–256.

[Commons, 2005] Commons, W. (2005). Exemple d'utilisation du normal mapping.

[Dai et al., 2017] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[Deng et al., 2020] Deng, Z., Bednařík, J., Salzmann, M., and Fua, P. (2020). Better patch stitching for parametric surface reconstruction. In *2020 International Conference on 3D Vision (3DV)*, pages 593–602. IEEE.

[Deprelle et al., 2019] Deprelle, T., Groueix, T., Fisher, M., Kim, V., Russell, B., and Aubry, M. (2019). Learning elementary structures for 3d shape generation and matching. In *Advances in Neural Information Processing Systems*, pages 7433–7443.

[Desbrun et al., 2002] Desbrun, M., Meyer, M., and Alliez, P. (2002). Intrinsic parameterizations of surface meshes. In *Computer graphics forum*, volume 21, pages 209–218. Wiley Online Library.

[Do Carmo, 2016] Do Carmo, M. P. (2016). *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications.

[Eck et al., 1995] Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. (1995). Multiresolution analysis of arbitrary meshes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 173–182.

[Fan et al., 2017] Fan, H., Su, H., and Guibas, L. J. (2017). A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613.

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

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[Floater, 1997] Floater, M. S. (1997). Parametrization and smooth approximation of surface triangulations. *Computer aided geometric design*, 14(3):231–250.

[Floater, 2003] Floater, M. S. (2003). Mean value coordinates. *Computer aided geometric design*, 20(1):19–27.

[Floater and Hormann, 2005] Floater, M. S. and Hormann, K. (2005). Surface parameterization: a tutorial and survey. *Advances in multiresolution for geometric modelling*, pages 157–186.

[Fukunaga and Hostetler, 1975] Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40.

[Fukushima, 1988] Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130.

[Garland et al., 2001] Garland, M., Willmott, A., and Heckbert, P. S. (2001). Hierarchical face clustering on polygonal surfaces. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 49–58.

[Genova et al., 2019] Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W. T., and Funkhouser, T. A. (2019). Learning shape templates with structured implicit functions. *CoRR*, abs/1904.06447.

[Giesecke et al., 2016] Giesecke, F. E., Mitchell, A., Spencer, H. C., Hill, I. L., Dygdon, J. T., Novak, J. E., Loving, R., Lockhart, S., and Johnson, C. (2016). *Technical drawing with engineering graphics*. Peachpit Press.

[Giorgi et al., 2007] Giorgi, D., Biasotti, S., and Paraboschi, L. (2007). Shape retrieval contest 2007: Watertight models track. *SHREC competition*, 8:7.

[Golovinskiy and Funkhouser, 2009] Golovinskiy, A. and Funkhouser, T. (2009). Learning Consistent Segmentation of 3D Models. *Computers and Graphics (Shape Modeling International)*.

[Groueix et al., 2018a] Groueix, T., Fisher, M., Kim, V. G., Russell, B., and Aubry, M. (2018a). 3d-coded : 3d correspondences by deep deformation. In *ECCV*.

[Groueix et al., 2018b] Groueix, T., Fisher, M., Kim, V. G., Russell, B., and Aubry, M. (2018b). AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Groueix et al., 2018c] Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. (2018c). 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 230–246.

[Gu, 2003] Gu, X. (2003). *Parametrization for surfaces with arbitrary topologies.* Harvard University.

[Gu et al., 2002] Gu, X., Gortler, S. J., and Hoppe, H. (2002). Geometry images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 355–361.

[Guskov, 2004] Guskov, I. (2004). An anisotropic mesh parameterization scheme. *Engineering with Computers*, 20(2):129–135.

[Guskov et al., 2000] Guskov, I., Vidimče, K., Sweldens, W., and Schröder, P. (2000). Normal meshes. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 95–102.

[Hackel et al., 2017] Hackel, T., Savinov, N., Ladicky, L., Wegner, J. D., Schindler, K., and Pollefeys, M. (2017). Semantic3d. net: A new large-scale point cloud classification benchmark. *arXiv preprint arXiv:1704.03847*.

[Häne et al., 2017] Häne, C., Tulsiani, S., and Malik, J. (2017). Hierarchical surface prediction for 3d object reconstruction. In *2017 International Conference on 3D Vision (3DV)*, pages 412–420. IEEE.

[Hormann and Greiner, 2000] Hormann, K. and Greiner, G. (2000). Mips: An efficient global parametrization method. Technical report, ERLANGEN-NUERNBERG UNIV (GERMANY) COMPUTER GRAPHICS GROUP.

[Hormann et al., 1999] Hormann, K., Greiner, G., and Campagna, S. (1999). Hierarchical parametrization of triangulated surfaces. In Girod, B., Niemann, H., and Seidel, H.-P., editors, *Proceedings of Vision, Modeling, and Visualization 1999*, pages 219–226, Erlangen, Germany. Infix.

[Hough, 1962] Hough, P. V. (1962). Method and means for recognizing complex patterns. *US patent*, 3(6).

[Huang et al., 2021] Huang, J., Zhang, Y., and Sun, M. (2021). Primitivenet: Primitive instance segmentation with local primitive embedding under adversarial metric. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15343–15353.

[Huang et al., 2011] Huang, Q., Koltun, V., and Guibas, L. (2011). Joint-Shape Segmentation with Linear Programming. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*.

[Julius et al., 2005] Julius, D., Kraevoy, V., and Sheffer, A. (2005). D-charts: Quasi-developable mesh segmentation. In *Computer Graphics Forum*, volume 24, pages 581–590. Citeseer.

[Kaiser et al., 2018] Kaiser, A., Ybanez Zepeda, J. A., and Boubekeur, T. (2018). A survey of simple geometric primitives detection methods for captured 3d data. In *Computer Graphics Forum*. Wiley Online Library.

[Kanazawa et al., 2018] Kanazawa, A., Tulsiani, S., Efros, A. A., and Malik, J. (2018). Learning category-specific mesh reconstruction from image collections. In *ECCV*.

[Karras et al., 2017] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

[Kelley, 1960] Kelley, H. J. (1960). Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954.

[Kharevych et al., 2006] Kharevych, L., Springborn, B., and Schröder, P. (2006). Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics (TOG)*, 25(2):412–438.

[Khodakovsky et al., 2003] Khodakovsky, A., Litke, N., and Schröder, P. (2003). Globally smooth parameterizations with low distortion. *ACM transactions on graphics (TOG)*, 22(3):350–357.

[Kim et al., 2013] Kim, V. G., Li, W., Mitra, N. J., Chaudhuri, S., DiVerdi, S., and Funkhouser, T. (2013). Learning Part-based Templates from Large Collections of 3D Shapes. *Transactions on Graphics (Proc. of SIGGRAPH)*, 32(4).

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[Kiryati et al., 1991] Kiryati, N., Eldar, Y., and Bruckstein, A. M. (1991). A probabilistic hough transform. *Pattern recognition*, 24(4):303–316.

[Koch et al., 2019] Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., and Panozzo, D. (2019). Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9611.

[Kós and Várady, 2003] Kós, G. and Várady, T. (2003). Parameterizing complex triangular meshes. *Curve and surface design*, pages 265–274.

[Kraevoy and Sheffer, 2004] Kraevoy, V. and Sheffer, A. (2004). Cross-parameterization and compatible remeshing of 3d models. *ACM Transactions on Graphics (ToG)*, 23:861–869.

[Kraevoy and Sheffer, 2005] Kraevoy, V. and Sheffer, A. (2005). Template-based mesh completion. In *Symposium on Geometry Processing*, volume 385, pages 13–22. Citeseer.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[Kuhn, 1955] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2:83–97.

[Landrieu and Simonovsky, 2018] Landrieu, L. and Simonovsky, M. (2018). Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4558–4567.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Lee et al., 2000] Lee, A., Moreton, H., and Hoppe, H. (2000). Displaced subdivision surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 85–94.

[Lee et al., 2002] Lee, Y., Kim, H. S., and Lee, S. (2002). Mesh parameterization with a virtual boundary. *Computers & Graphics*, 26(5):677–686.

[Lévy, 2001] Lévy, B. (2001). Constrained texture mapping for polygonal meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 417–424.

[Lévy, 2003] Lévy, B. (2003). Dual domain extrapolation. *ACM Transactions on Graphics (TOG)*, 22(3):364–369.

[Lévy and Mallet, 1998] Lévy, B. and Mallet, J.-L. (1998). Non-distorted texture mapping for sheared triangulated meshes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 343–352.

[Lévy et al., 2002] Lévy, B., Petitjean, S., Ray, N., and Maillot, J. (2002). Least squares conformal maps for automatic texture atlas generation. *ACM transactions on graphics (TOG)*, 21(3):362–371.

[Lévy et al., 2002] Lévy, B., Petitjean, S., Ray, N., and Maillot, J. (2002). Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH*.

[Li et al., 2017] Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., and Guibas, L. (2017). Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 36(4):52.

[Li et al., 2018] Li, L., Sung, M., Dubrovina, A., Yi, L., and Guibas, L. (2018). Supervised fitting of geometric primitives to 3d point clouds. *arXiv preprint arXiv:1811.08988*.

[Li et al., 2019] Li, L., Sung, M., Dubrovina, A., Yi, L., and Guibas, L. J. (2019). Supervised fitting of geometric primitives to 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2652–2660.

[Li et al., 2011] Li, Y., Wu, X., Chrysanthou, Y., Sharf, A., Cohen-Or, D., and Mitra, N. J. (2011). Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics*, 30(4):52:1–52:12.

[Liu et al., 2008] Liu, L., Zhang, L., Xu, Y., Gotsman, C., and Gortler, S. J. (2008). A local/global approach to mesh parameterization. In *Computer Graphics Forum*, volume 27, pages 1495–1504. Wiley Online Library.

[Liu et al., 2020] Liu, M., Sheng, L., Yang, S., Shao, J., and Hu, S.-M. (2020). Morphing and sampling network for dense point cloud completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11596–11603.

[Lloyd, 1982] Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

[Loper et al., 2015] Loper, M., Mahmood, N., Romero, J., Pons-Moll, and G., Black, M. J. (2015). Smpl: A skinned multi-person linear model. *SIGGRAPH Asia*.

[Luo et al., 2015] Luo, H., Wang, C., Wen, C., Cai, Z., Chen, Z., Wang, H., Yu, Y., and Li, J. (2015). Patch-based semantic labeling of road scene using colorized mobile lidar point clouds. *IEEE Transactions on Intelligent Transportation Systems*, 17(5):1286–1297.

[Maillot et al., 1993] Maillot, J., Yahia, H., and Verroust, A. (1993). Interactive texture mapping. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 27–34.

[Matas and Chum, 2004] Matas, J. and Chum, O. (2004). Randomized ransac with td, d test. *Image and vision computing*, 22(10):837–842.

[Maturana and Scherer, 2015] Maturana, D. and Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE.

[McCartney et al., 1999] McCartney, J., Hinds, B., and Seow, B. (1999). The flattening of triangulated surfaces incorporating darts and gussets. *Computer-Aided Design*, 31(4):249–260.

[Meagher, 1982] Meagher, D. (1982). Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129–147.

[Mescheder et al., 2019] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function

space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470.

[Mildenhall et al., 2020] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer.

[Mo et al., 2019] Mo, K., Zhu, S., Chang, A. X., Yi, L., Tripathi, S., Guibas, L. J., and Su, H. (2019). Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Monnier et al., 2021] Monnier, T., Vincent, E., Ponce, J., and Aubry, M. (2021). Unsupervised layered image decomposition into object prototypes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8640–8650.

[Norman et al., 2006] Norman, K. A., Polyn, S. M., Detre, G. J., and Haxby, J. V. (2006). Beyond mind-reading: multi-voxel pattern analysis of fmri data. *Trends in cognitive sciences*, 10(9):424–430.

[Pang et al., 2021] Pang, J., Li, D., and Tian, D. (2021). Tearingnet: Point cloud autoencoder to learn topology-friendly representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7453–7462.

[Park et al., 2019] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174.

[Paschalidou et al., 2019a] Paschalidou, D., Ulusoy, A. O., and Geiger, A. (2019a). Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proceedings*

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10344–10353.

[Paschalidou et al., 2019b] Paschalidou, D., Ulusoy, A. O., and Geiger, A. (2019b). Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

[Pedersen, 1995] Pedersen, H. K. (1995). Decorating implicit surfaces. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 291–300.

[Peng et al., 2020] Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., and Geiger, A. (2020). Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer.

[Pinkall and Polthier, 1993a] Pinkall, U. and Polthier, K. (1993a). Computing discrete minimal surfaces and their conjugates. *Experimental mathematics*, 2(1):15–36.

[Pinkall and Polthier, 1993b] Pinkall, U. and Polthier, K. (1993b). Computing discrete minimal surfaces and their conjugates. *EXPERIMENTAL MATHEMATICS*, 2:15–36.

[Piponi and Borshukov, 2000] Piponi, D. and Borshukov, G. (2000). Seamless texture mapping of subdivision surfaces by model pelting and texture blending. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 471–478.

[Potamias et al., 2022] Potamias, R. A., Ploumpis, S., and Zafeiriou, S. (2022). Neural mesh simplification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18583–18592.

[Praun et al., 2000] Praun, E., Finkelstein, A., and Hoppe, H. (2000). Lapped textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 465–470.

[Praun et al., 2001] Praun, E., Sweldens, W., and Schröder, P. (2001). Consistent mesh parameterizations. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 179–184.

[Qi et al., 2017a] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660.

[Qi et al., 2017b] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30.

[Rabbani and Van Den Heuvel, 2005] Rabbani, T. and Van Den Heuvel, F. (2005). Efficient hough transform for automatic detection of cylinders in point clouds. *Isprs Wg Iii/3, Iii/4*, 3:60–65.

[Rabinovich et al., 2017] Rabinovich, M., Poranne, R., Panozzo, D., and Sorkine-Hornung, O. (2017). Scalable locally injective mappings. *ACM Transactions on Graphics*, 36:16:1–16:16.

[Riegler et al., 2017] Riegler, G., Osman Ulusoy, A., and Geiger, A. (2017). Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

[Roberts, 1963] Roberts, L. G. (1963). *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology.

[Rodin and Sullivan, 1987] Rodin, B. and Sullivan, D. (1987). The convergence of circle packings to the riemann mapping. *Journal of Differential Geometry*, 26(2):349–360.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[Rousseeuw, 1984] Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880.

[Sander et al., 2001] Sander, P. V., Snyder, J., Gortler, S. J., and Hoppe, H. (2001). Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 409–416.

[Sander et al., 2003] Sander, P. V., Wood, Z. J., Gortler, S., Snyder, J., and Hoppe, H. (2003). Multi-chart geometry images.

[Schnabel et al., 2009] Schnabel, R., Degener, P., and Klein, R. (2009). Completion and reconstruction with primitive shapes. *Computer Graphics Forum (Proc. of Eurographics)*, 28(2):503–512.

[Schnabel et al., 2007] Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library.

[Serafin and Grisetti, 2015] Serafin, J. and Grisetti, G. (2015). Nicp: Dense normal based point cloud registration. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 742–749. IEEE.

[Sharma et al., 2018] Sharma, G., Goyal, R., Liu, D., Kalogerakis, E., and Maji, S. (2018). Csgnet: Neural shape parser for constructive solid geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5523.

[Sheffer, 2002] Sheffer, A. (2002). Spanning tree seams for reducing parameterization distortion of triangulated surfaces. In *Proceedings SMI. Shape Modeling International 2002*, pages 61–272. IEEE.

[Sheffer and De Sturler, 2000] Sheffer, A. and De Sturler, E. (2000). Surface parameterization for meshing by triangulation flattening. In *Proc. 9th International Meshing Roundtable*. Citeseer.

[Sheffer and de Sturler, 2001] Sheffer, A. and de Sturler, E. (2001). Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with computers*, 17(3):326–337.

[Sheffer and Hart, 2002] Sheffer, A. and Hart, J. C. (2002). Seamster: inconspicuous low-distortion texture seam layout. In *IEEE Visualization, 2002. VIS 2002.*, pages 291–298. IEEE.

[Sheffer et al., 2007a] Sheffer, A., Hormann, K., Levy, B., Desbrun, M., and Zhou, K. (2007a). Nnwarp: Neural network-based nonlinear deformation. *SIGGRAPH 2007 Course Notes*.

[Sheffer et al., 2005] Sheffer, A., Lévy, B., Mogilnitsky, M., and Bogomyakov, A. (2005). Abf++: fast and robust angle based flattening. *ACM Transactions on Graphics (TOG)*, 24(2):311–330.

[Sheffer et al., 2007b] Sheffer, A., Praun, E., Rose, K., et al. (2007b). Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision*, 2(2):105–171.

[Sidi et al., 2011] Sidi, O., van Kaick, O., Kleiman, Y., Zhang, H., and Cohen-Or, D. (2011). Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. In *Transactions on Graphics (Proc. of SIGGRAPH Asia)*, pages 126:1–126:10.

[Smith and Schaefer, 2015] Smith, J. and Schaefer, S. (2015). Bijective parameterization with free boundaries. *ACM Trans. Graph.*, 34.

[Soler et al., 2002] Soler, C., Cani, M.-P., and Angelidis, A. (2002). Hierarchical pattern mapping. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 673–680.

[Song et al., 2017] Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., and Funkhouser, T. (2017). Semantic scene completion from a single depth image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1746–1754.

[Sorkine and Alexa, 2007] Sorkine, O. and Alexa, M. (2007). As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116.

[Sorkine et al., 2002] Sorkine, O., Cohen-Or, D., Goldenthal, R., and Lischinski, D. (2002). Bounded-distortion piecewise mesh parameterization. In *IEEE Visualization, 2002. VIS 2002.*, pages 355–362. IEEE.

[Tancik et al., 2020] Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547.

[Tatarchenko et al., 2017] Tatarchenko, M., Dosovitskiy, A., and Brox, T. (2017). Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE international conference on computer vision*, pages 2088–2096.

[Tieleman et al., 2012] Tieleman, T., Hinton, G., et al. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

[Torr and Zisserman, 2000] Torr, P. H. and Zisserman, A. (2000). Mlesac: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding*, 78(1):138–156.

[Tulsiani et al., 2017a] Tulsiani, S., Su, H., Guibas, L. J., Efros, A. A., and Malik, J. (2017a). Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2643.

[Tulsiani et al., 2017b] Tulsiani, S., Su, H., Guibas, L. J., Efros, A. A., and Malik, J. (2017b). Learning shape abstractions by assembling volumetric primitives. In *Computer Vision and Pattern Recognition (CVPR)*.

[Turk, 2001] Turk, G. (2001). Texture synthesis on surfaces. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 347–354.

[Tutte, 1963] Tutte, W. T. (1963). How to draw a graph. *Proceedings of the London Mathematical Society*, 3(1):743–767.

[Varol et al., 2017] Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., and Schmid, C. (2017). Learning from synthetic humans. In *CVPR*.

[Wang et al., 2017] Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y., and Tong, X. (2017). O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11.

[Wang et al., 2019] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12.

[Wei and Levoy, 2001] Wei, L.-Y. and Levoy, M. (2001). Texture synthesis over arbitrary manifold surfaces. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 355–360.

[Williams et al., 2019] Williams, F., Schneider, T., Silva, C., Zorin, D., Bruna, J., and Panozzo, D. (2019). Deep geometric prior for surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10130–10139.

[Woodford et al., 2012] Woodford, O. J., Pham, M.-T., Maki, A., Gherardi, R., Perbet, F., and Stenger, B. (2012). Contraction moves for geometric model fitting. In *European Conference on Computer Vision*, pages 181–194. Springer.

[Xu et al., 1990] Xu, L., Oja, E., and Kultanen, P. (1990). A new curve detection method: randomized hough transform (rht). *Pattern recognition letters*, 11(5):331–338.

[Yan et al., 2012] Yan, D.-M., Wang, W., Liu, Y., and Yang, Z. (2012). Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design*, 44(11):1072–1082.

[Yan et al., 2021] Yan, S., Yang, Z., Ma, C., Huang, H., Vouga, E., and Huang, Q. (2021). Hpnet: Deep primitive segmentation using hybrid representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2753–2762.

[Yang et al., 2018a] Yang, Y., Feng, C., Shen, Y., and Tian, D. (2018a). Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 206–215.

[Yang et al., 2018b] Yang, Y., Feng, C., Shen, Y., and Tian, D. (2018b). Foldingnet: Point cloud auto-encoder via deep grid deformation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[Ying et al., 2001] Ying, L., Hertzmann, A., Biermann, H., and Zorin, D. (2001). Texture and shape synthesis on surfaces. In *Eurographics Workshop on Rendering Techniques*, pages 301–312. Springer.

[Zayer et al., 2005] Zayer, R., Rössl, C., and Seidel, H.-P. (2005). Variations on angle based flattening. In *Advances in Multiresolution for Geometric Modelling*, pages 187–199. Springer.

[Zhang et al., 2005] Zhang, E., Mischaikow, K., and Turk, G. (2005). Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics (TOG)*, 24(1):1–27.

[Zhang et al., 2020] Zhang, Z., Sun, B., Yang, H., and Huang, Q. (2020). H3dnet: 3d object detection using hybrid geometric primitives. In *European Conference on Computer Vision*, pages 311–329. Springer.

[Zhou et al., 2004] Zhou, K., Synder, J., Guo, B., and Shum, H.-Y. (2004). Isocharts: stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 45–54.

[Zigelman et al., 2002] Zigelman, G., Kimmel, R., and Kiryati, N. (2002). Texture mapping using surface flattening via multidimensional scaling. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):198–207.

[Zou et al., 2017] Zou, C., Yumer, E., Yang, J., Ceylan, D., and Hoiem, D. (2017). 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 900–909.

[Zuffi and Black., 2015] Zuffi, S. and Black., M. J. (2015). The stitched puppet: A graphical model of 3d human shape and pose. *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).*