Neural Surfaces

Luca Morreale

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of
University College London.

Computer Science
University College London

April 11, 2024

I, Luca Morreale, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Classic shape representations define 3D models as a set of discrete elements, generally as triangles or quads. This description is widely adopted across graphics pipelines, from rendering, compression, and shape editing, to shape correspondence. Recently, neural shape representations have emerged as effective tools to describe complex geometries and structures. Models are encoded in network weights, that are then queried through a 3D or 2D point. However, most of these representations are task-specific and thus must be converted back to meshes to be used across applications. The need for a flexible representation, adaptable across the different neural pipelines, that lends itself to optimizations, remains.

This thesis explores the idea of neural surface representations as a map. First, we define surfaces as $2D$-$3D$ map encoded into neural network weights, dubbed Neural Surface Map. We adopt it as a building block to define a comprehensive mapping framework. Indeed, by composing multiple maps, we establish and optimize correspondences between shapes. This framework sidesteps the intricacies of optimization encountered by conventional methods while achieving continuous and bijective maps. Then, building on this foundation, we relax constraints within the mapping framework. In particular, we eliminate the need for human supervision by extracting (noisy) labels from pre-trained models. These labels are used to distill inter-surface maps between highly non-isometric shape pairs. We compare to state-of-the-art shape correspondence methods, demonstrating its effectiveness.

Finally, by interpreting a shape as the composition of a coarse structure and detail, we extend the neural representation to enable shape manipulation, compression, and detailed transfer. Intuitively, the structure defines the pose of the model

and details repeating information, such as wrinkles. This representation lends itself to interactive shape manipulation, such as feature enhancement, while compressing detail into the network weights.

Acknowledgements

This document is the culmination of a long and arduous journey. Throughout this adventure, I faced challenges and successes that contributed to my personal growth beyond what I had deemed possible. I would like to take this opportunity to express my gratitude to everyone who played a role in my life during this time.

I thank my supervisor, Niloy J. Mitra, for initiating this journey, guiding my studies, and support during the toughest times. Special thanks to Vladimir G. Kim for his patience and paternal advice. A big thanks to Noam Aigerman for his candid opinions, for teaching me how to scrutinize my own research, and for his infinite optimism.

Thank you to my collaborators at Meta: Herny Howard-Jenkins, Chris Xie, Armen Avetisyan, Samir Aroudj, Suvam Patra, Tsun-Yi Yang, and Vasielios Balntas. I am grateful for your patience, guidance, support, and help during and after the internship, it was an incredible experience working with you.

Thank you to my friends, colleagues, and fellow adventurers Siddhant, Preddy, Wamiq, Mohamed, Thu, David, Filippos, Yu-Shiang, Gizem, Wonbong, Eric, Nels, Mirgahney, Ziwen for the laughs, banter, and advice. In particular, I am extremely grateful to Michael and Sanjeev for their friendship and support during stressful deadlines – you have been invaluable.

Gratitude also goes to my good friends Vittorio, Simone, and Fabrizio. You have been reference points for me, always knowing how to make the most of life and embracing every situation, no matter the hardship. Thank you for your wisdom and for being there during this time.

Thanks to all the loving and amazing people of Camden Martial Arts. I will be

eternally grateful for the laughs, beers, and all the time together. A big thank you to Julian for creating such a wonderful group, and thanks to Arta, Brix, Cass, Darren, Kaz, Liv, Lukas, Mal, Mark, Nes, Ras, Rik, Robin, Roshana, Sto, and Tom.

I am grateful to Jacopo Forloni for your unconditional support during the best and worst times. Thank you to my good friend Ewerton de Oliveira Lopes for your incredible advice on navigating life and the PhD.

Lastly, I am deeply thankful to my mother for always being proud of me and her unwavering support.

Publications

Note that the work presented in this thesis has been published as different conference papers, references are listed below.

- Chapter 3:

  Luca Morreale, Noam Aigerman, Vladimir G. Kim, Niloy J. Mitra. Neural Surface Maps In *Proc. Computer Vision and Pattern Recognition, IEEE*, 2021.

- Chapter 4:

  Luca Morreale, Noam Aigerman, Vladimir G. Kim, Niloy J. Mitra. Neural Semantic Surface Maps In *Eurographics*, 2024.

- Chapter 5:

  Luca Morreale, Noam Aigerman, Paul Guerrero, Vladimir G. Kim, Niloy J. Mitra. Neural Convolutional Surfaces In *Proc. Computer Vision and Pattern Recognition, IEEE*, 2022.

Impact Statement

This thesis presents three works that define neural representations enabling geometry processing tasks. Each work was presented at CVPR or Eurographics. The work in Chapter 3 presents a surface representation that acts as a building block for a mapping framework for subsequent chapters. Furthermore, this representation enables inter-surface mappings, a challenging task in geometry processing. Due to several constraints in its formulation, the mapping framework is then extended in Chapter 4, bringing it closer to real-world applications. Finally, Neural Convolutional Surfaces (Chapter 5) reformulates the initial representation for a wider range of geometry processing tasks. This definition brings the representation closer to an interpretable generative framework for 3D models.

Contents

1 Introduction **17**

1.1 Challenges 17

1.2 Contributions 21

2 Literature review **23**

2.1 Differential geometry 23

2.2 Neural shape representations 25

2.2.1 Atlas-based representations 25

2.2.2 Implicit fields 27

2.3 Shape Matching 28

2.3.1 Spectral methods 29

2.3.2 Classic methods 30

2.4 Image-based Shape Analysis 31

2.5 Geometry manipulation 32

2.5.1 Image-based Surface Editing 32

2.5.2 Neural Surface Manipulation 33

3 Neural Surface Maps **35**

3.1 Introduction 35

3.2 Method 38

3.2.1 Neural Maps 38

3.2.2 Overfitting Neural Surface Maps 39

3.2.3 Surface Map Distortion 40

3.2.4 Geometry-preserving optimization via composition 41
3.2.5 Compositing Neural Maps 42
3.3 Evaluation 44
3.3.1 Neural representation 45
3.3.2 Surface Parametrization 45
3.3.3 Surface-to-surface Maps 47
3.3.4 Composition with Analytical Maps 47
3.3.5 Cycle-consistent Mapping for Collections of Surfaces . . . 48
3.3.6 Comparison 50
3.4 Implementation Details 50
3.5 Limitations 51
3.6 Conclusions 52

4 Neural Semantic Surface Maps **53**
4.1 Introduction 53
4.2 Method 56
4.2.1 Semantic Shape Alignment 57
4.2.2 Distilling Fuzzy 3D Matches via Visual Semantics 57
4.2.3 Computing rendering correspondences. 58
4.2.4 Aggregating the Fuzzy Matches to an Inter-surface map . . 60
4.2.5 Seamless Neural Surface Map. 60
4.2.6 Cones. 62
4.2.7 Seamlessness. 62
4.2.8 Optimization energies. 63
4.2.9 Total energy. 64
4.2.10 Rendering Settings 65
4.3 Evaluation 65
4.3.1 Datasets 65
4.3.2 Metrics 66
4.3.3 Baselines 67
4.3.4 Quantitative Evaluation 67

4.3.5 Qualitative Evaluation . . . 69
4.3.6 Ablation . . . 69
4.4 Implementation details . . . 77
4.5 Limitations . . . 78
4.5.1 Improvement over Neural Surface Maps . . . 78
4.6 Conclusion . . . 79

5 Neural Convolutional Surfaces **80**
5.1 Introduction . . . 80
5.2 Neural Convolutional Surfaces . . . 82
5.2.1 Coarse Model . . . 83
5.2.2 Fine Model . . . 84
5.2.3 Local Reference Frame . . . 85
5.2.4 Training . . . 86
5.3 Experiments . . . 86
5.3.1 Baselines . . . 86
5.3.2 Metrics . . . 87
5.3.3 Comparison . . . 88
5.3.4 Feature enhancement . . . 88
5.3.5 Detail transfer . . . 90
5.3.6 Ablation . . . 90
5.3.7 Interpretability of the kernels . . . 91
5.4 Implementation details . . . 93
5.5 Limitations . . . 94
5.5.1 Improvement over Neural Surface Maps . . . 94
5.6 Conclusions . . . 95

6 Conclusions **97**
6.1 Summary . . . 97
6.2 Limitations and Future Work . . . 98
6.3 Remark . . . 99

Appendices **101**

A Neural Surface Maps **101**

A.1 Approximating Surfaces 101

A.2 Analytical Maps 102

B Neural Semantic Surface Maps **103**

B.1 Computing rendering correspondences 103

B.1.1 Patch generation, feature extraction, and PCA 103

B.2 Comparison Details 104

B.3 Metrics 105

C Neural Convolutional Surfaces **107**

C.1 Comparisons 107

C.2 Expressive power. 107

C.3 Architecture Details 108

Bibliography **110**

List of Figures

1.1 Mesh discreteness 18
1.2 Shape correspondence 20

2.1 Chart in differential geometry 24
2.2 AtlasNet representation architecture 26
2.3 Neural fields 27
2.4 Shape correspondence 29
2.5 Image-based shape analysis 31
2.6 Surface manipulation 33

3.1 Neural surface maps framework 37
3.2 Overfit framework 39
3.3 Surface overfitting 45
3.4 Surface parametrization 46
3.5 Inter-surface maps 47
3.6 Analytical surface maps 48
3.7 Collection map 49
3.8 Comparison classic method 50
3.9 Comparison SOTA 51

4.1 Highly non-isometric automatic map 54
4.2 Neural semantic surface maps pipeline 56
4.3 Co-aligning input surfaces 58
4.4 Fuzzy semantic correspondences 59
4.5 Cutting through cone points 61

4.6 Seamless cut 63
4.7 Comparison with state-of-the-art 68
4.8 Qualitative comparison with ENIGMA 69
4.9 Qualitative maps 70
4.10 Ablation on similarity score 71
4.11 Robustness to misalignment 73
4.12 Pose variation 74
4.13 Qualitative comparison with SMvAT 76
4.14 Scan to SMPL map 77

5.1 Neural Convolutional Surfaces 81
5.2 Architecture 83
5.3 Reconstruction quality 87
5.4 Reconstruction 89
5.5 Feature enhancement 90
5.6 Detail transfer 91
5.7 Ablation 92
5.8 Interpretability 92
5.9 Effect of patches 93
5.10 Limitation 95

A.1 Activation effect 101

B.1 SHREC19 qualitative comparison 104

C.1 Higher genus maps 107
C.2 Qualitative evaluation 109

List of Tables

1 Thesis notation 16

1.1 Pros and cons of different representations 19

4.1 Quantitative evaluation 67
4.2 Dino-ViT rendering ablation 72
4.3 Dino-ViT pose ablation 73

5.1 Quantitative comparison 88
5.2 NCS models 94

C.1 Architecture detail 108

Notation

Table 1: Notation used throughout the thesis.

Symbol		Meaning
$\mathbb{R}$		Set of real numbers
x	$\in \mathbb{R}^2$	Vector representing a 2D point
X	$\in \mathbb{R}^3$	Vector representing a 3D point
$\mathbf{A}$	$\mathbb{R}^3 \to \mathbb{R}^2$	Source mesh
$\mathbf{B}$	$\mathbb{R}^3 \to \mathbb{R}^2$	Target mesh
$f_{\mathbf{A}}$	$\mathbb{R}^2 \to \mathbb{R}^3$	Source overfitted neural surface map
$f_{\mathbf{B}}$	$\mathbb{R}^2 \to \mathbb{R}^3$	Target overfitted neural surface map
h	$\mathbb{R}^2 \to \mathbb{R}^2$	Neural map bridging two surface maps
Ψ	$\mathbb{R}^3 \to \mathbb{R}^3$	Map between source and target shape
$\Omega_{f_{\mathbf{A}}}$		Domain for map $f_{\mathbf{A}}$
J^h	$\in \mathbb{R}^{h_{in} \times h_{out}}$	Jacobian matrix of transformation h
M	$\in \mathbb{R}^{2\times 2}$	First Fundamental Form
$\|\cdot\|_k$	$\in \mathbb{R}$	L-k norm

Chapter 1

Introduction

In the era of digitization, our reliance on 3D data has deepened. The transition from analog to digital 3D models by artists and engineers highlights the need for software capable of parsing and manipulating digital surfaces. Whether creating animated movies or games, artists now utilize 3D graphics software like Blender to operate directly on meshes, encapsulating the complexity of 3D assets.

AI and deep learning have become pivotal in addressing many 3D challenges, pushing researchers to leverage these technologies. However, the use of meshes in this context often necessitates ad-hoc algorithms to handle numerous edge cases. This thesis defines a more adept representation tailored for neural-based downstream 3D tasks, alleviating the challenges associated with intricate mesh structures and fostering a more streamlined integration of 3D data into neural computational frameworks.

1.1 Challenges

Triangle meshes have been the most popular representation across much of geometry processing since its early stages. However this representation bears several shortcomings: it describes surfaces in a piece-wise linear fashion, *i.e.*, discrete and discontinuous, and it does not scale well with the model complexity. Due to these limitations, meshes often require task-specific optimization formulations, that often fail to generalize to surfaces of varying sizes. Generally, geometry processing researches heavily rely on maps as tools to manipulate and act on meshes, however,

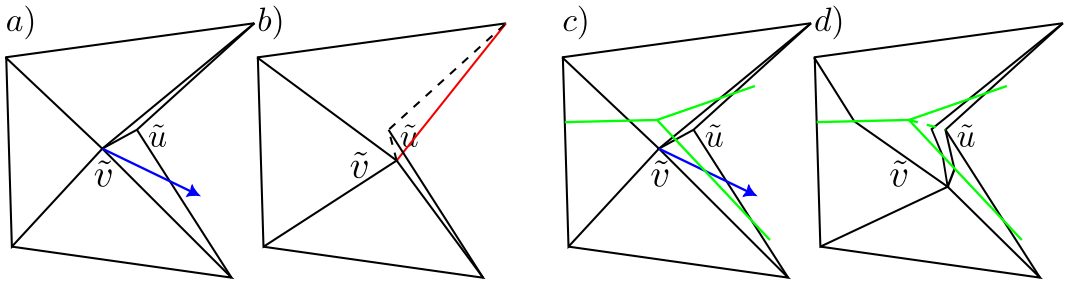


Figure 1.1: Meshes discreteness: optimizing mesh vertices directly, blue arrow in (a), results in foldovers or artefacts, red edge in (b). The optimization is more complicated when we slides one surface over the other, green lines in (c). Edges must follow the geometry of the other mesh while avoiding artefacts (d).

maps assume continuity of the input domain (the surface) while meshes are not. Similarly, neural networks work best with fully discrete data and fixed-size data [10], such as voxel grids, while meshes are partly continuous. Therefore, designing maps or neural tools to effectively process and manipulate meshes is non-trivial, often requiring hacks to handle edge cases as shown in Figure 1.1. We therefore need a new representation that lends itself to various types of processing.

The rising prominence of deep learning has led researchers to investigate ways to represent shapes via neural networks. The immediate use of these techniques is to define a shared latent space to describe shapes. Notable approaches use implicit fields [7, 8], volumes [9, 10], or hybrid representations [6, 11, 12, 13]. Shared latent spaces enforce a strong prior over the network weights, enabling the compression of 3D models in a small vector. Indeed, through the use of a shared template, 3DCoded [13] can effectively morph and model human deformations. Implicit field techniques, like DeepSDF [7], instead can encode a large variety of data independently of their topology, while neural volumetric representations demonstrate excellent results in shape analysis [10]. However, due to a lack of diversity in the training data, these approaches fail to generalize to ”artistic” models and are often inaccurate or imperfect as network weights trade accuracy for diversity.

Alternative methods use a shape-specific set of weights to represent a specific shape instance. These approaches rely on the same techniques as generative models but capture geometric detail efficiently and accurately, thus creating outputs that are on par with existing 3D models. Furthermore, they introduce novel properties

Table 1.1: Representations: meshes are widely adopted across the graphics pipeline. On the other hand, most neural representations are made ad-hoc for specific tasks, such as rendering, or compression. The representations proposed in this thesis span a wider range of applications.

Name	Type	Manipulation	Correspondence	Rendering	Detail transfer	Compression	Geometry processing
Meshes	Explicit	✔	✔	✔	✔	✔	✔
Deep GI [1]	Hybrid	✗	✗	✗	✗	✔	✗
IDF [2]	Implicit	✗	✗	✗	✔	✔	✔
ACORN [3]	Implicit	✗	✗	✗	✗	✔	✔
NGLOD [4]	Implicit	✗	✗	✔	✗	✔	✔
NERF [5]	Implicit	✗	✗	✗	✔	✗	✗
AtalsNet [6]	Hybrid	✗	✗	✗	✔	✔	✗
Chapter 3	Hybrid	✗	✔	✗	✗	✗	✔
Chapter 5	Hybrid	✔	✔	✗	✔	✔	✔

not attainable with surface meshes, such as differentiability. These neural representations for shape instances were demonstrated to be useful in several applications such as rendering [4], shape compression [3], surface parametrization [14], and video consistent shape reconstruction [15]. However, most of these techniques are constrained to the specific task they are designed for, hence limiting their applicability. We highlight the nature and flexibility of different representations in Table 1.1.

Shape correspondence is a notable task in geometry processing for which the representation is fundamental. Intuitively, we expect to map neighboring points from a shape onto close-by points of another, thus preserving the continuity of the original shapes, as shown in Figure 1.2. The most seminal technique attempting to solve this problem is Functional Maps [16]. This method defines a linear map in the spectral domain of the two shapes, thus mapping a vertex to a vertex. Over the years, this technique has been extended to handle partial shape [17, 18], scans [19], and shape collections [20]. However, none of the proposed solutions is continuous or bijective, as it should be. This is because the underlying representation is not continuous and the technique does not enforce or exploit continuity of the representation. In Chapter 3, we explore the idea of a map-based representation and demonstrate that through such change we can optimize a bijective and continuous map. In particular, we introduce the concept of Neural Surface Map (NSM) that encodes surfaces in neural network weights as a map $\mathbb{R}^2 \rightarrow \mathbb{R}^3$. Thanks to the intrinsic

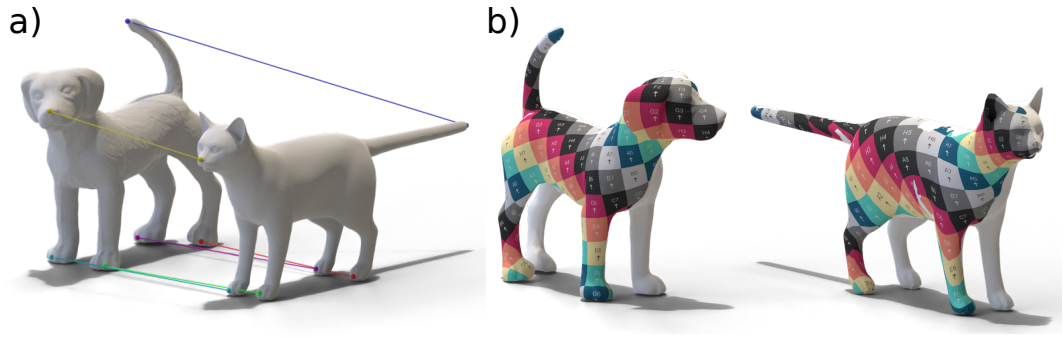


Figure 1.2: Shape correspondence: starting from manual annotations (a), a map is optimized to define correspondences between the two meshes (b), here shown through texture.

continuity of the neural network, this representation lends itself to optimization. We use NSM as a building block to define a framework for the inter-surface map between shape pairs and collections. We later extend this framework in Chapter 4, and reduce the human supervision required. In particular, we replace the human in the loop with noisy supervision from DinoV2 [21], and show we can achieve semantic maps.

More broadly, the choice of shape representation and neural network architecture plays a critical role in how efficiently the network capacity is utilized and for its flexibility. The majority of existing representations use MLPs to model the shape as a function that maps points either from a 2D atlas to the surface [22] or points in a 3D volume to an implicit function such as a distance field [7]. However, these techniques entangle geometric details and overall shape structure and do not have a natural mechanism to reuse the network weights to represent repeating local details, as Convolution Neural Networks (CNNs) achieve on images. Alternatively, instead of a single global MLP, some prior techniques leverage repetitions by breaking the shape into smaller 3D voxels, each represented by an SDF [3], however, these representations do not account for surface details alignment, and thus, are less effective at representing local geometric textures that flow with the shape. These choices constrain the flexibility of the representation, preventing it from being used across multiple manipulation tasks. In Chapter 5, we extend the representation defined in Chapter 3 and define a convolution-based description, dubbed Neural Convolu-

tional Surface (NCS). We show NCS enables shape manipulation, detail transfer, and compression, on top of tasks already possible with NSM.

1.2 Contributions

In line with the challenges emphasized above, *we hypothesize there exists a better representation than meshes that lend themselves to manipulation enacted by neural networks*. This thesis seeks to define surface representations that lend themselves to manipulation enacted by neural networks. Specifically, we focus on single-shape representation that must faithfully capture the intricate features of the underlying surface. This level of fidelity is as important as the representation's flexibility. Thus, we showcase both quality and applicability across various tasks within the realm of computer graphics and geometric processing.

After reviewing the literature in Chapter 2, in Chapter 3 we introduce Neural Surface Map (NSM) as surface representation. *We build NSM on the concept of maps and atlases and use it as a building block for the entire thesis*. This definition can faithfully encode 3D models into network weights. Furthermore, it allows us to define a mapping framework between surfaces of different kinds, thus showcasing the usability of this representation for tasks such as parametrization, shape correspondence, and collection mapping.

In Chapter 4 we extend the mapping framework previously defined to alleviate its constraints. In particular, we show how by adopting seamless maps it is possible to mitigate the need for manual labels in shape correspondence in different tasks. This allows us to *automate the entire shape correspondence pipeline*, bridging the gap to state-of-the-art methods.

Chapter 5 extends the Neural Surface Map representation introducing a convolutional bias, thus defining a Neural Convolutional Surface (NCS). The main contribution of this chapter is *the separation of coarse shape structure and surface details*. We prove that this shape disentanglement along with the convolutional bias can effectively compress the shape while enabling surface feature enhancement and detail transfer.

Finally, in Chapter 6 we summarize the contributions of this thesis, and its limitations, and highlight future works.

Chapter 2

Literature review

In this chapter, we review the literature associated to shape space representations as well as the specific trend of representing a single shape via a neural network. We focus on atlas-based representation, which inspires this thesis, and delve into specific applications of shape analysis, *i.e.*, such as shape correspondence and manipulation, which are used to demonstrate the validity of the presented representations.

2.1 Differential geometry

Atlas-based representations are deeply grounded in differential geometry. Here we briefly review the basic concepts referred to in this thesis.

Generally speaking, surfaces are envisioned as 2D manifolds lying in 3D, *i.e.*, 2D surfaces that live in a higher dimensional space. Thus, locally, from any given point, the surface would appear to be flat. This idea can be leveraged to define a chart as a local mapping of an open subset of the surface S_i onto a plane through a homeomorphic function: $\psi : S \to \mathbb{R}^2$. Intuitively, a chart is a bijective and continuous function mapping a local part of the surface onto a planar domain. Atlases extend this concept to cover the whole surface, namely an atlas is a collection of charts $\mathscr{A} = \{(S_i, \psi_i) : i \in I\}$, such that $S = \bigcup_{i \in I} S_i$. More formally, these charts must agree, such that their composition, or their inverse, is consistent.

Leveraging these concepts, we can define the tangent plane at a point p as the set of vectors tangent to the surface at p. Mathematically, we can define the basis

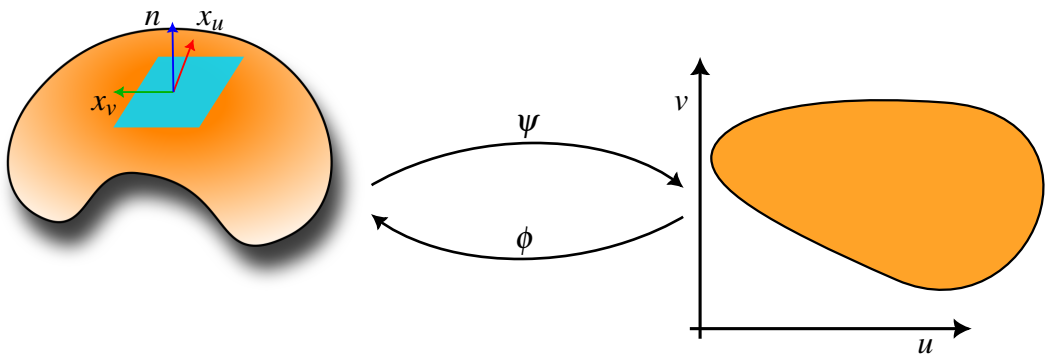


Figure 2.1: Thanks to differential geometry we can define a local map from the surface to the plane. Given this relation, it is possible to estimate a tangent plane to the surface, and its normal.

for such a local plane in terms of the map differential:

$$x_u = \frac{\partial \phi}{\partial u} \quad \text{and} \quad x_v = \frac{\partial \phi}{\partial v}, \tag{2.1}$$

where x_u and x_v are unit vectors, and ϕ is the inverse of the chart as show in Figure 2.1. Notably, the normal at a point p is defined as the cross-product of the basis:

$$n_p = x_u \times x_v. \tag{2.2}$$

Then, we define the First Fundamental Form as:

$$M = \begin{bmatrix} \langle x_u, x_u \rangle & \langle x_u, x_v \rangle \\ \langle x_u, x_v \rangle & \langle x_v, x_v \rangle \end{bmatrix} = \begin{bmatrix} E & F \\ F & G \end{bmatrix}, \tag{2.3}$$

which encodes surface properties, such as curvature.

It is possible to use similar concepts for a general transformation $T : \mathbb{R}^D \to \mathbb{R}^G$. In this case, x_u and x_v are not unit vectors, and we refer to them as Jacobian $J^T = [x_u, x_v] \in \mathbb{R}^{G \times D}$ of the transformation T. In this case, the First Fundamental Form can be rewritten as $M = J^T J$, and it encodes essential properties about the transformation, *e.g.*, we can infer the per-point distortion applied by T and its type. For example, an isometric transformation preserves geodesic distances between points, and its first fundamental form would belong to $M \in SO(2)$. Similarly, a transformation that preserves angles is a rotation up to a scaling factor λ.

These concepts are extensively used in geometry processing when defining maps. For example, SLIM [23] parametrizes meshes to a plane in a single chart such that it minimizes isometric energy called Symmetric Dirichlet. Similarly, ARAP [24] morph shapes between different poses by constraining the triangle-wise transformation to a rotation. Other works define a global conformal mapping [25, 26, 27] to the plane, while more recent works use higher dimensional domains, *e.g.*, spheres [28, 29]. Please refer to [30] for an in-depth discussion on parametrization.

2.2 Neural shape representations

Neural shape representations have emerged as a transformative paradigm in the domain of geometric modeling and computer graphics. Unlike traditional methods that often struggle to scale with the complexity of shapes, neural approaches leverage the power of deep learning to capture and encode complex, and possibly repeating, geometries, easing further processing required for complex tasks. However, the community has not yet reached a consensus on which shape representation to adopt. For example, in 3D shape reconstruction we have a rich palette of representations, ranging from pixel-wise depth maps [31, 32, 33, 34, 35], triangular meshes [32, 36, 37, 38, 39, 40, 41, 42], implicit functions [43, 44, 7, 45, 46], voxels [47, 48, 49], shape primitives [50, 51, 52, 53, 54, 55], atlas-based representations [56, 6, 13, 14, 54, 22], or combinations of some of these [57, 58, 59, 60].

There is no free lunch, each representation has specific benefits and drawbacks, for example in terms of memory requirements, fitting precision, adaptability to different shape topologies, or access to the surface properties. Below we delve into atlas-based representations as they inspire the work discussed in this thesis.

2.2.1 Atlas-based representations

Atlas-based representations are mathematically defined as map $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3$. Generally, this function is parametrized by a neural network, or more specifically an MLP [6, 22], which maps a $2D$ point x to $3D$. Seminal work from Groueix *et al.* [6] parametrizes a surface as the combination of multiple networks and charts.

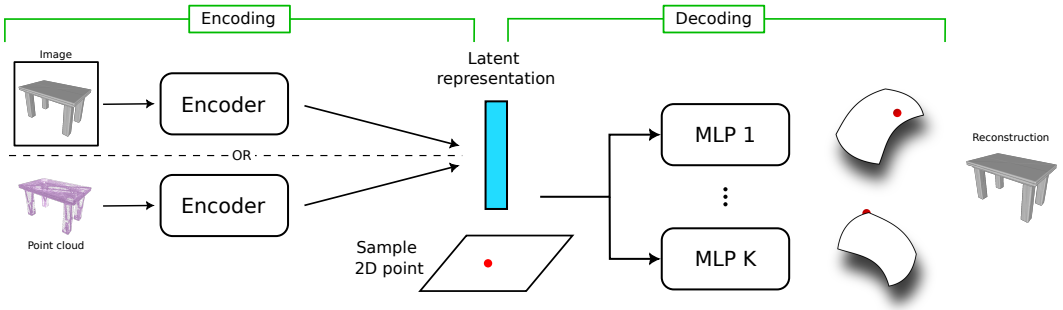
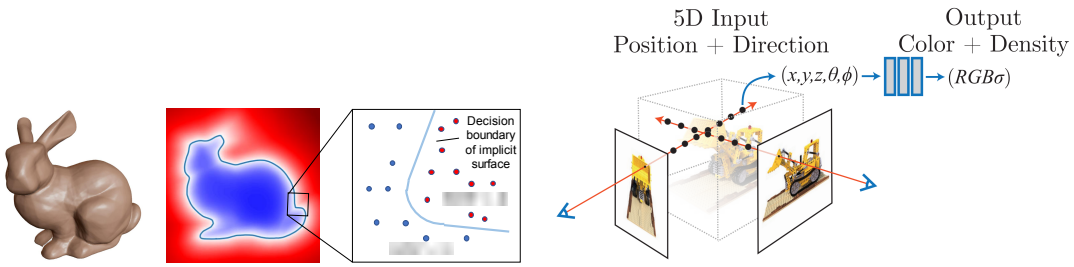


Figure 2.2: AtlasNet [6] encodes an input image, or point cloud (left), then decodes it as composition of several charts. Each chart is decoded independently through an MLP, by sampling a point on a 2D regular grid (right).

Each network is tasked to uplift a chart to 3D, such that the union of charts describes the whole surface. This design can represent surfaces of arbitrary topology in a continuous and differentiable fashion, see Figure 2.2. However, as these functions are defined independently, the overall approach allows inconsistencies across the individual mappings. Bednarik *et al.* [22] incorporated a penalization of inconsistencies while encouraging independent conformal mappings. Concurrently, FoldingNet [12] parametrizes a surface as a single chart. This formulation sidesteps the inconsistencies arising from an atlas, but cannot accurately represent large shape spaces.

Recently, AtlasNet has been extended in [15] to consistently parametrize shapes across a motion sequence, thus demonstrating that given a few corresponding landmarks across the sequence, it is possible to encode motion in network weights. With a similar formulation, PCTMA-Net [61] exploits the inductive bias of transformer networks to better model a latent shape space, thus enhancing the reconstruction quality. Further extension decomposes shapes into deformable parts [54], or primitives, encoded in an atlas and corresponding mappings. Similarly, Badki *et al.* [56] optimize a dictionary of local patches and place them to cover the surface. Orthogonally, TearingNet [62] avoids the need for multiple mappings by learning to tear apart the input domain and fit arbitrary genus objects with a single function.

Several techniques specialize in certain shape categories or topologies. SCALE [63] focuses on the human body with an AtlasNet-like design. Multi-chart Generative Surface Modelling [64] assumes spherical shape topology, but it



(a) DeepSDF **(b)** NeRF

Figure 2.3: Neural fields: Implicit techniques represent the surface through a signed distance function as in DeepSDF [7] (a). Alternatively, rendering focused methods define an occupancy that must be accumulated along a ray, as in NeRF [5] (b). Figures taken from [66].

can produce a globally consistent set of mappings. 3D-Coded [13] optimizes multiple MLPs deforming a (human) template to a target shape, thus implicitly defining correspondences. Similarly, NERS [65] morphs a sphere to the target model while, disjointly, optimizing texture.

These techniques have been explored mainly for reconstruction and generative tasks. In this thesis, we explore the use of atlas-based representation for a single shape and demonstrate it can be used across several tasks while carrying all the advantages of an explicit surface.

2.2.2 Implicit fields

Implicit fields define shapes as signed distance [7, 3, 67, 68] or occupancy grids [10]. These techniques are extremely popular since they can accurately describe arbitrary topologies with few parameters. High compression rate and accuracy can be obtained as these techniques do not store the surface explicitly, but it is extracted through a non-differentiable marching cubes algorithm.

Seminal work, DeepSDF [7], encodes shapes in a shared latent space through a decoder-only architecture, see Figure 2.3(a). Each 3D Model is decoded by querying the decoder at different points in the volume, whose SDF values then are used to reconstruct the surface via marching cubes. At inference time, a 3D model is encoded through a slow latent code optimization. Many works build on this idea, SIREN [67] focuses on a single shape or image and defines an implicit field with

sinusoidal activation. Sitzmann *et al.* prove the effectiveness of this activation compared to more classic ones. Chan *et al.* [69] extend this idea to generative settings through meta-learning [70, 71]. Davies *et al.* [72] offer an in-depth analysis of this representation in terms of compactness, the impact of alignment, stability, scale, and convergence. Differently, [3] subdivides the space through an integer-linear program to compactly encode a single shape into features, which are later decoded by a network into distance values. Yifan *et al.* [2] propose a two-network shape representation, one defining a coarse structure, while the latter encodes high-frequency details. Recently Yang *et al.* [73] showcased several geometry processing applications exploiting the differentiability of the underlying representation.

Mildenhall *et al.* in [5] cast the surface reconstruction with implicit fields as multi-view volumetric rendering, see Figure 2.3(b). NERF encodes visual appearance and occupancy in two separate networks that are queried along rays to render or retrieve the surface. Several techniques extend this approach incorporating lightning [74], depth-maps [75], few views [76, 77], unbounded scenes [78], compressing [79], estimate camera poses [80, 81], exploiting image features [82, 83], and encoding dynamic scenes [84]. The literature regarding this type of neural field is vast and mostly orthogonal to this thesis. We refer to [66] for an in-depth discussion of these methods. Overall, neural fields effectively encode volumes enabling rendering but are not flexible, *i.e.*, they cannot be adopted across different applications.

2.3 Shape Matching

Shape correspondence is a fundamental problem in geometry processing, Figure 1.2 shows an example. Simple techniques, like Iterative Closest Points (ICP) methods [85, 86], align and match shape pairs through rigid transformations. Learned features can be leveraged to identify good correspondences and define an initial global alignment [87, 88, 89]. Most recent algorithms characterize correspondences in the shape spectral space [16, 17, 17, 18], while mesh-based algorithms define maps between charts or meshes [90, 91, 92, 93]. In the following, we review the shape-matching literature.

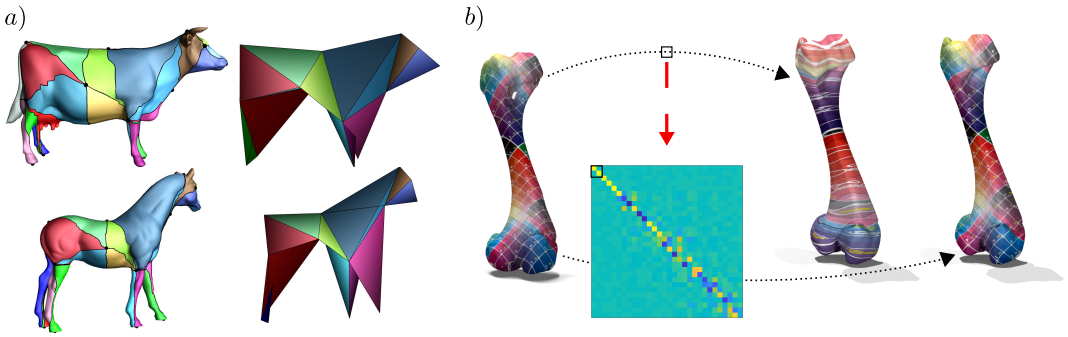


Figure 2.4: Shape correspondence: [93] in (a) defines the inter-surface maps on the mesh directly, optimizing mesh-mesh intersection. On the other hand, ZoomOut [166] in (b), incrementally optimizes a map in the spectral domain of the meshes. Pictures taken from respective papers.

2.3.1 Spectral methods

Most notable shape correspondence methods are based on Functional maps [16]. These approaches compute a fuzzy correspondence by aligning the spectral basis of two shapes with linear transformation [16, 94]. This technique offers elegant machinery to compute shape correspondence and proved to be reliable for near-isometric shape pairs. The key advantage is the capacity to express maps as small matrices, encoded in a reduced basis, which greatly simplifies the optimization problems, see Figure 2.4(b).

A fundamental aspect of functional map techniques is the use of shape descriptors to identify a set of correspondences. Most methods rely on hand-crafted input features, such as SHOT [95], Heat Kernel Signature [96] or Wave Kernel Signature [97]. However, these descriptors generalize poorly across datasets, as the input features can change significantly, or are unreliable for non-isometric shapes. For example, SHOT descriptors [95] are sensitive to the mesh topology, while WKS [97] works poorly in non-isometric shape pairs.

Aware of these issues, researchers integrated norms [98, 99], map recovery approaches [100, 101], and regularizers [102] in the solution. Ezuz *et al.* [101] denoise maps based on a smoothness before formulated as cycle consistency. [103] define the initial map exploring the space of maps with Monte Carlo, this approach helps them disambiguate intrinsic symmetries while yielding a smoother map.

More recent techniques attempt to learn descriptors specifically for shape cor-

respondence through mesh convolution [104, 105, 106], siamese networks [107], refining the input descriptors [108, 19], or completely unsupervised [109]. Unfortunately, these methods are bounded to the dataset's distribution, *e.g.*, humanoid shapes.

While the original formulation of functional maps was restricted to genus 0 shapes, Litany *et al.* [110, 17] extended it to partial shapes by relying on a localized quasi-harmonic basis [110]. More recent works explore the space low-distortion map [111], define maps between non-isometric shape pairs [19, 112], and leverage optimal transport methods [113]. Although effectiveness of these techniques, they struggle to define a continuous and bijective map.

2.3.2 Classic methods

Classic methods act directly on meshes themselves. Seminal work from Schreiner *et al.* [93] divides shape into corresponding parts, then optimizes the surface-to-surface map directly as mesh-mesh intersection, as shown in Figure 2.4(a). This leads to a combinatorial problem due to the surface discreteness, *i.e.*, triangle meshes. Kim *et al.* [114] define a map as a blending of maps, dubbed Blended Intrinsic Maps (BIM). The authors explore the space of low-distortion maps and then define a map by blending and interpolating several conformal maps.

Recent techniques define maps through a common domain such as a plane [115, 91] or a sphere [116, 29]. Aigerman *et al.* [115] first cut each mesh, jointly flatten them, and finally optimize an affine map on the plane. This strategy has then been extended in [90] to account for inconsistent shape cuts through a seamless map.

Schmidt *et al.* [91] optimize a bijective map in the planar domain between shape pairs minimizing isometric energy, while accounting for the mesh discontinuities. Later, the authors define an optimization scheme directly on the mesh estimating geodesics in metrics of constant Gaussian curvature [92], thus being invariant to the mesh topology. In an attempt to overcome shape discretization, Schmidt *et al.* [29] defines an iterative optimization scheme for genus 0 shapes by using spheres as an intermediate domain. The authors incrementally refine the map

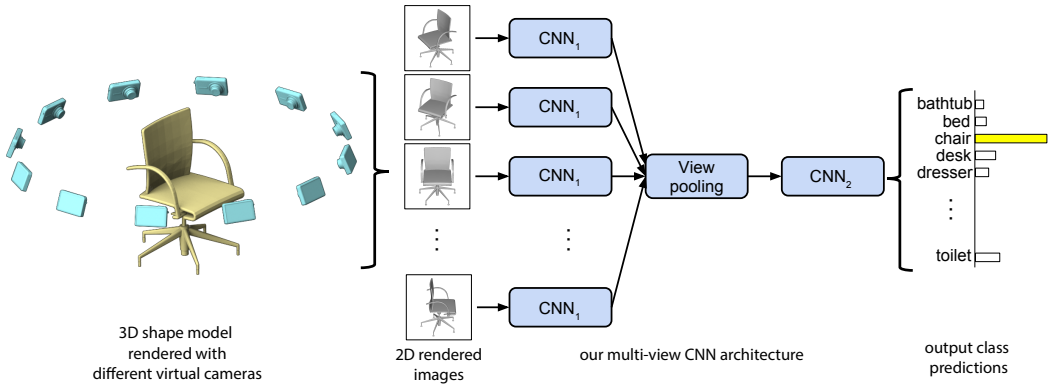


Figure 2.5: Image-based analysis: given a 3D model it is possible to render it from a different point of view and then process these images with a CNN model to extract information, *e.g.*, classify the 3D object. Figure taken from [118].

in a coarse to fine manner while encouraging bijectivity and continuity. Unfortunately, due to the refinement, the original mesh is lost in the process. We refer to [117] for a more comprehensive analysis of shape correspondence methods.

In this thesis, we take a different approach and introduce map-based representation and demonstrate its effectiveness in the challenging task of matching shapes while preserving continuity.

2.4 Image-based Shape Analysis

Many works analyse shapes through images: first models are rendered multiple times and from different viewpoints, then they are analysed by neural networks in the image space, as shown in Figure 2.5. Prominent examples are classification [118], segmentation [119, 120, 121], or matching [122] where these methods rely on, possibly pre-trained, CNNs acting on images. The output may be aggregated on the shape via additional optimization [123] or ray casting. Often these methods require fine-tuning with 3D supervision and thus can only work on categories of shapes with labelled 3D data. For example, Genova et al. [124], train a 3D segmentation technique by using a 2D method to produce pseudo labels. Abdelreheem *et al.* [125] describes a training-free approach for 3D shape semantic segmentation using pre-trained visual transformers, Blip2 [126], and the functional maps framework [127].

Although the use of pre-trained CNN features marked a vital milestone for computer vision tasks, such as object detection and segmentation [128], these network representations encode a wide range of visual information from low-level (statistical) features, (*e.g.*, edges, and auto-correlation matrices), to object parts. Vision Transformers [129], dubbed ViTs, have shown the ability to discover both local and non-local relations. Dino-ViT [130] trains a transformer network through self-distillation and uses its features in multiple tasks, *e.g.*, image retrieval and object segmentation. Several works demonstrated the utility of Dino-ViT internal representation as a black box [131, 132] for tasks such as semantic segmentation [133] and category discovery [134]. Amir *et al.* [135] study these features and use them to solve vision tasks, such as image correspondences, in zero-shot settings. Recently, Oquab *et al.* [21] extended Dino-ViT, introducing DinoV2, showcasing enhanced semantic interpretability compared to the original version, and also exhibiting broader applicability.

Conversely to state-of-the-art methods, we use ViT models to extract semantic information between shape pairs, thus lifting the need for human labeling in shape correspondence tasks.

2.5 Geometry manipulation

Manipulating shapes is a key task for 3D artists. Thus a representation must enable artists to edit the geometry as easily as possible, in multiple ways, *e.g.*, through brushes, procedurally, or through visual guidance. Although in this thesis we tackle the task of transferring and enhancing high-frequency shape details, hereafter we broadly review shape manipulation approaches.

2.5.1 Image-based Surface Editing

Early applications of image-based mesh editing were mainly focused on mesh simplification and accelerated rendering [136]. Most methods reduce polygon count for efficient rendering while preserving perceptual appearance [137, 138, 139, 140]. Recently, Liu *et al.* [141] proposed to utilize rendering for a larger family of surface editing tasks. In particular, the user can apply changes in the image space with a

(a) Image-based [141] **(b)** Stylization [143] **(c)** Subdivision [144]

Figure 2.6: Surface manipulation: image-based techniques change the surface based on appearance obtained from renderings (a), while neural approaches (b-c) edit the model based on data-driven priors captured from meshes. Figures taken from respective papers.

wide range of image processing filters, which are then translated into shape edits as shown in Figure 2.6(a). Similarly, Kato *et al.* [142] apply for image style transfer on renderings by propagating the image gradient to the geometry. The subtle difference between these two recent works lies in the gradients: Paparazzi define them analytically, whereas N3DMR uses approximations.

2.5.2 Neural Surface Manipulation

Recent advances in deep learning enabled the use of learnable components in 3D editing, such as generation [145], stylization [146], subdivision [144], and decomposition [147, 145].

Generation: directly generating a mesh as a set of vertices and faces is infeasible due to the lack of regular structure and combinatorial variability in the output space. Thus most approaches produce only coarse models [148] or deform a template [149, 150, 41], hence constraining the output's topology.

Stylization: stylization is a challenging task in computer graphics involving the manipulation of a 3D model based on an input image or conditioned on a certain style. While the same problem has been extensively studied in the image space, [151] being the prime example, 3D modeling lacked behind. Early methods involved combining different shapes as collage art [152, 153], generating non-realistic shapes, lego-like models [154, 155], or model with manga style [156]. More recently, neural network approaches have shown the ability to transfer texture from any template to a new target shape [143, 158] as in Figure 2.6(b), or specific style [146], or from text [157].

Subdivision: subdivision techniques attempt to retrieve the original mesh from a compressed one. Seminal work [159] inserts a new vertex at each edge's midpoint, subdividing each face in 4. [160] proposes a similar, weighted, approach that inserts new vertices by interpolating over existing ones. However, these classic techniques rely on hand-crafted filters [159, 161, 160], thus failing to retrieve the original geometry. Recent techniques rely on learned priors to upsample 3D models and restore them from a low-poly geometry [144, 162]. Liu *et al.* [144] propose a data generation technique for creating various coarse variants of the same mesh with a low-distortion bijective map, then learning to upsample it, see Figure 2.6(c). Similarly, [162] exploits repeating patterns in 3D models to encode local detail in a shared latent space. Then, a decoder upsamples the coarse mesh conditioned on the local latent codes, restoring the lost features.

In this thesis, we propose a neural representation that compresses a 3D model, enables the transfer of style or detail or detail between meshes, and enhances existing features.

Chapter 3

Neural Surface Maps

3.1 Introduction

Maps are one of the most fundamental concepts in geometry processing: as discussed in Section 2.1, in differential geometry, a surface, *i.e.*, a 2-manifold, is usually (locally) defined as the image of a (non-degenerate) map

$$f : \mathbb{R}^2 \to \mathbb{R}^n.$$

Not surprisingly, maps are also used to define correspondences between different parts of surfaces in an atlas, to evaluate similarity between surface pairs, or across surface collections.

Accordingly, computing maps is central in most geometry processing tasks operating on surfaces. The ubiquitous concept of a UV map [163], mapping a surface into the plane, provides a local coordinate system on surfaces, and hence enables downstream tasks such as texturing, surface correspondence, remeshing, quad-meshing [164] to name a few. Similarly, surface-to-surface maps [93] enable defining correspondences between surfaces, which are at the heart of shape analysis, transfer of properties, deformations, or defining morph sequences. Indeed, almost all shape processing tasks, including parametrization, surface correspondence, remeshing, and deep learning on surfaces, heavily rely on access to such surface maps.

However, many of the tasks related to maps and their computation become ex-

tremely hard to handle when the target domain is a surface, *i.e.*, a 3D mesh ($n = 3$). This is due mostly to the fact that meshes are combinatorial representations, which in turn leads to a combinatorial representation of the surface maps, and taints the optimization task with a combinatorial nature as well. Although elegant solutions in the form of discrete differential geometry [165, 92], meshing invariant spectral analysis [166, 167], functional maps [16, 19] have been proposed to work around the combinatorial representation, the diverse choices and different data representations inhibit easy end-to-end optimization and adaptation outside the specialized geometry processing community.

As an example, consider the problem of computing a mesh-to-mesh mapping in which a continuous map from one surface to the other is computed: one needs to account for the image of each source vertex, which lands on a triangle of the other mesh, and the image of a source edge may span several triangles of the target; this leads to extensive bookkeeping, and any attempt to optimize, *e.g.*, the map's inter-surface distortion leads to combinatorial optimization of the choice of target triangle for each source vertex as in [93, 168]. An alternative is to optimize proxy maps into a common base domain [115, 29] in the hope that the resulting surface-to-surface map will be optimized by proxy. Such an approach, however, does not yield surface maps that are even a local minimizer of the energy they set to minimize. This is particularly problematic when optimizing inter-surface maps across shape collections.

In this work, we consider *neural networks* as a parametric representation of both individual surfaces as well as inter-surface maps. Specifically, we consider networks with parameters θ that receive 2D points as input and output points either in 2D or 3D, $f_{\mathbf{A}} : \mathbb{R}^2 \to \mathbb{R}^n$. While this definition is similar to, *e.g.*, AtlasNet [6], we do not aim to perform any learning task, and our network does nothing more than map 2D points with the aim of performing one task: approximate a single surface map $f_{\mathbf{A}} \sim f : \mathbb{R}^2 \to \mathbb{R}^n$, so we can work with neural networks instead of with, e.g., mappings of triangular meshes.

Specifically, we use a map $f_{\mathbf{A}} \sim f : \mathbb{R}^2 \to \mathbb{R}^3$ to directly characterize a given

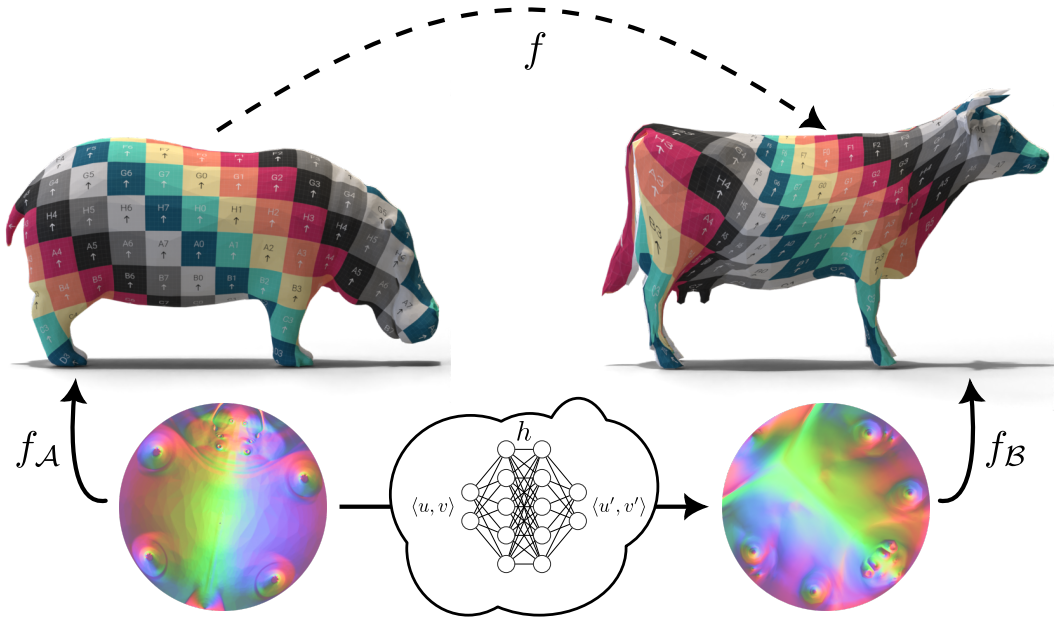


Figure 3.1: Here, we depict an inter-surface map between two non-isometric shape pairs. We optimize such a map by minimizing isometric energy of the composition of several mappings, $f_{\mathbf{A}}$, h, and $f_{\mathbf{B}}$. These mappings are defined as neural networks describing a surface ($f_{\mathbf{A}}$, $f_{\mathbf{B}}$) or a map between planar domains (h). To preserve the surfaces, we only optimize h to cope with the distortion of the surface mappings. This operations is done in a complete differentiable manner through PyTorch.

manifold shape **A** (restricted to surface patches homeomorphic to a disc), and use another map $h : \mathbb{R}^2 \to \mathbb{R}^2$ to update the surface map by restricting movements on the underlying 2-manifold. These neural networks are, by construction, differentiable and composable with one another, hence they lend us a simple model for defining a differentiable algebra of surface maps, enabling us to compose maps with one another and optimize objectives directly over their composition, rather than propose approximations via intermediate proxy domains.

We employ this concept in two ways that build on top of one another: first, we revisit the differential-geometry definition of a surface as a map from 2D to 3D, by overfitting a neural network to a given UV parametrization computed via a standard parametrization algorithm, such as Tutte's embedding [169] or SLIM [23]. Two such maps, $f_{\mathbf{A}}, f_{\mathbf{B}}$, are shown in Figure 3.1. This gives us a parametric, differentiable representation of the surface, from a canonical domain. Second, we compose the overfitted map with other maps, either to optimize the distortion of the map, or

to compute distortion-minimizing maps between two or more surfaces. Figure 3.1 shows an example of a distortion-minimizing map f defined by composing h with $f_{\mathbf{A}}$ and $f_{\mathbf{B}}$.

We evaluate NSM on a variety of triangular meshes with varying complexity and show their efficacy in computation of parametrizations, surface-to-surface distortion-minimizing mapping, and also for mapping across collections of shapes. We also provide comparison to baseline methods. In summary, our main contribution is introducing neural surface map as a novel representation and utilizing it towards addressing a variety of geometry processing applications. We particularly stress the modular nature of the representation that enables harnessing the power of current deep learning frameworks to solve many (classical) shape analysis tasks in a uniform framework.

3.2 Method

We now define neural surface maps and how to compute and optimize them.

3.2.1 Neural Maps

We use the term *neural surface map* (NSM) to refer to any neural network considered as a function $f_{\mathbf{A}} : \mathbb{R}^2 \to \mathbb{R}^3$. This ensures the map's image is always a 3D surface, and, assuming the map is non-singular, also a 2-manifold.

Neural surface maps are an alternative method to describe a surface that holds two main advantages: *differentiability* and ability to be *composed* with other maps. This enables us to easily compose neural maps $f_{\mathbf{A}} \circ f_{\mathbf{B}}$, and define an objective over the composition $O(f_{\mathbf{A}} \circ f_{\mathbf{B}})$ which can be differentiated and optimized via standard (deep learning) libraries and optimizers without the need to write tailor-made code to handle new objective, work with combinatorial mesh representations, or deal with the notoriously-hard map composition problem. Furthermore, thanks to the universal approximation theorem [170], there always exists a network capable of approximating a given surface function.

We obtain and manipulate neural surface maps via two processes – overfitting and optimization, which we detail next.

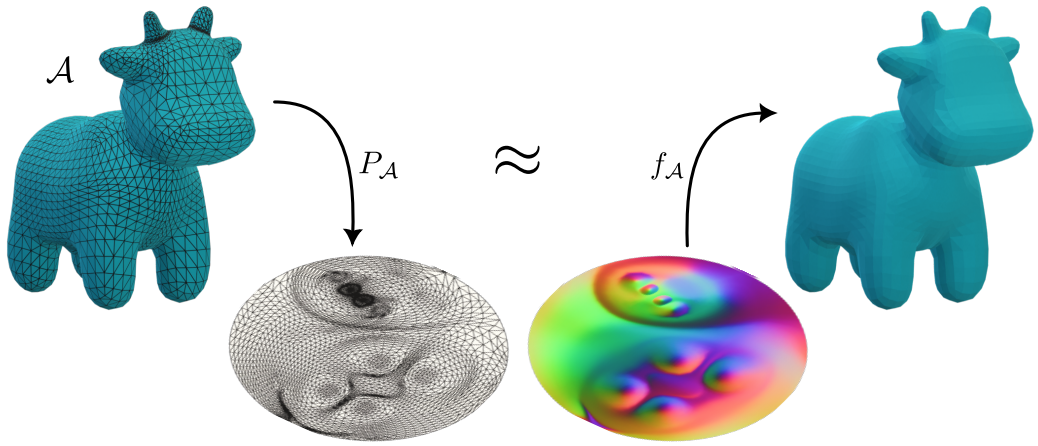


Figure 3.2: Starting from a 3D mesh, $\mathbf{A}$, we parametrize to the 2D disk. Thus, obtaining a piece-wise linear map $P_{\mathbf{A}}$. We overfit an MLP to the inverse of such a map $f_{\mathbf{A}} \approx P_{\mathbf{A}}^{-1}$.

3.2.2 Overfitting Neural Surface Maps

Let $\Omega \subset \mathbb{R}^2$ be the unit circle. All our neural maps will make use of Ω as a canonical domain. Given any map $P : \Omega_P \to \mathbb{R}^n$, we can approximate it via a neural surface map f by using black-box methods to train the neural network and *overfit* it to replicate P. In the case of surfaces, this ground-truth map can be the inverse of a parametrization, $P_{\mathbf{A}} : \mathbb{R}^3 \to \mathbb{R}^2$, of a surface $\mathbf{A}$. This would be a continuous piecewise-linear map mapping triangles to triangles.

To approximate $P_{\mathbf{A}}^{-1}$ we minimize the least-square deviation of $f_{\mathbf{A}}$ from $P_{\mathbf{A}}^{-1}$ and the surface normal error as follows:

$$\begin{aligned}\mathscr{L}_{\text{overfit}} = \int_{p\in\Omega_f} \left\| P_A^{-1}(p) - f_{\mathbf{A}}(p) \right\|^2 + \\ \lambda_n \int_{p\in\Omega_f} \left\| n_p^{f_{\mathbf{A}}} - n_p^{P_A} \right\|^2,\end{aligned} \tag{3.1}$$

where $n_p^{f_{\mathbf{A}}}$ is the estimated normal at p, and $n_p^{P_A}$ is the ground-truth normal. In practice, we optimize this objective by approximating the integral in Monte-Carlo fashion by summing the integrand over a random set of sample points.

Namely, to use neural surface maps to represent surfaces, we first compute a ground truth map $P_{\mathbf{A}}$ by overfiting to a UV parametrization of the mesh $\mathbf{A}$ into 2D, computed via any bijective parametrization algorithm of our choosing – in this

thesis, we show results with SLIM [23], by which we achieve an injective map of the mesh into $\Omega \subset \mathbb{R}^2$. We consider the inverse of this map, which maps Ω back into the 3D mesh $\mathbf{A}$, as our input $f : \Omega \to \mathbf{A}$, and overfit $f_{\mathbf{A}}$ to it by minimizing Equation 3.1, as shown in Figure 3.2. Thus, we obtain a neural representation of the surface. More specifically, this is a *mapping* into the surface, endowed with specific UV coordinates, with point $f_{\mathbf{A}}(x,y)$ having UV coordinates x,y. Figure 3.3 shows several examples of such overfitted neural maps and their faithfulness to the original geometry. NSM can faithfully represent smooth shapes as well as those having sharp edges. Note that we assume that the objects are or have been cut open to be homeomorphic to a disc.

Before progressing to discussing how can we compose maps and optimize them, we define the distortion measures we wish to optimize.

3.2.3 Surface Map Distortion

We wish to optimize several energies related to neural surface maps. Similarly to [22], for a neural map $f_{\mathbf{A}} : \Omega_{f_{\mathbf{A}}} \to \mathbb{R}^3$, we denote by $J_p^{f_{\mathbf{A}}} \in \mathbb{R}^{3\times 2}$ the matrix of partial derivatives at point $p \in \Omega_{f_{\mathbf{A}}}$, called the *Jacobian* of $f_{\mathbf{A}}$. The Jacobian essentially quantifies the local deformation at a point. Letting $M_p = J_p^T J_p$, we subsequently can quantify the symmetric Dirichlet energy [23]:

$$D_{\text{iso}} = \int_{\Omega} \text{trace}\left(M_p\right) + \text{trace}\left(\left(M_p + \varepsilon I\right)^{-1}\right) \tag{3.2}$$

where I is the identity matrix, added with a small constant ε, set to 0.01, to regularize the inverse.

Likewise, we can define a measure of conformal distortion via

$$D_{\text{conf}} = \int_{\Omega} \left\| \frac{\text{trace}\left(M_p\right)}{\left\|M_p\right\|^2} M_p - I \right\|^2. \tag{3.3}$$

We evaluate the integrals by random sampling of the function in the domain.

Next, we show how to define surface-to-surface maps via various compositions of the maps and optimize their distortion, in the pairwise and in the shape collection

setting.

3.2.4 Geometry-preserving optimization via composition

Our basic representation of 3D geometries is, as discussed above, via an *overfitted* neural surface map $f_{\mathbf{A}} : \Omega_{f_{\mathbf{A}}} \to \mathbb{R}^3$ that approximates a given map $P_{\mathbf{A}}$. We now treat $f_{\mathbf{A}}$ as our de-facto representation of the geometry. Our goal is to optimize various properties relating to the surface map, without affecting the geometry. However, optimization of the map is not trivial since it will immediately change the 3D geometry. We propose a solution to completely avoid this issue, next.

Assume we are given a neural surface map representing some surface $f_{\mathbf{A}} : \Omega_{f_{\mathbf{A}}} \to \mathbf{A}$; we wish to optimize the distortion $D(f_{\mathbf{A}})$ of the map. It is immediate to optimize $f_{\mathbf{A}}$ *itself* with respect to our differentiable notion of distortion, however that will cause the map to change, and thus its image, the 3D surface, will change and could, for instance, flatten to the plane. To overcome this, we suggest introducing another neural surface map $h : \Omega \to \Omega_{f_{\mathbf{A}}}$. We can now define a new map, $f^h_{\mathbf{A}} = f_{\mathbf{A}} \circ h$. As long as we solely optimize h and ensure it is onto $\Omega_{f_{\mathbf{A}}}$, we are guaranteed that the image of $f^h_{\mathbf{A}}$ is still the original image of $f_{\mathbf{A}}$, *i.e.*, respects the original surface.

We can now optimize the distortion of $f^h_{\mathbf{A}}$, by optimizing h and keeping $f_{\mathbf{A}}$ fixed, thereby finding a map from Ω to $\mathbf{A}$ which is (at least a local) minimizer of the distortion measure of our choice:

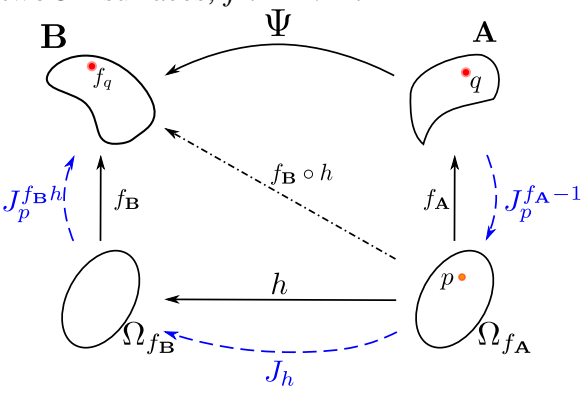
$$\min_h D\left(f^h_{\mathbf{A}}\right).$$

The distortion is a differentiable property of the map and hence is readily available, *e.g.*, via automatic differentiation. In fact, composition, and minimization of distortion can be achieved in a mere few lines of code in Pytorch.

We can now consider composing more than two of these maps, to enable maps into more intricate domains.

3.2.5 Compositing Neural Maps

One of the many advantages of NSM's composability is to enable representing maps between a pair of surfaces, using the classic method of a common domain, as depicted in Figure 3.1: we posses two overfitted neural maps, $f_{\mathbf{A}}, f_{\mathbf{B}} : \Omega \to \mathbb{R}^3$, respectively representing two surfaces $\mathbf{A}, \mathbf{B}$, and we wish to define and optimize an inter-surface mapping between these two 3D surfaces, $f : \mathbf{A} \to \mathbf{B}$.



To address the above problem, we define as before a map $h : \Omega_{f_{\mathbf{A}}} \to \Omega_{f_{\mathbf{B}}}$ and the composition $f_{\mathbf{B}}^h = f_{\mathbf{B}} \circ h$. At first glance, it would seem that in this case, to map a point from $\mathbf{A}$ to $\mathbf{B}$, we will need to consider the map $f_{\mathbf{B}} \circ h \circ f_{\mathbf{A}}^{-1}$, which includes an inverse of the *entire* map, that is of course not readily tractable.

However, we can define the inter-surface map Ψ via the following simple definition: for any point $p \in \Omega_{f_{\mathbf{A}}}$, Ψ is implicitly defined as the map satisfying $\Psi \circ f_{\mathbf{A}} \triangleq f_{\mathbf{B}} \circ h$, or in simple words: for any point $p \in \Omega_{f_{\mathbf{A}}}$, Ψ matches the image of p under $f_{\mathbf{B}}^h$ with the image of p, mapped through h and then through $f_{\mathbf{B}}$ (refer to Figure 3.1 for an illustration). This definition is known as the *common domain* definition of a map and has been used in many works [171, 93, 168, 172, 115, 173, 25, 174]. It can be verified that this definition is identical to the one using the inverse, as long as the inverse exists, and can still provide a bijective map between the surfaces even in cases where it does not exist (cf., [173, 115]).

3.2.5.1 Computing distortion in the common domain

Even though Ψ itself is not tangible for optimization, as it is implicitly defined by h, luckily the only differential quantity we need from Ψ to compute the distortion, is the Jacobian of f, denoted J_q^{Ψ} at point $q = f_{\mathbf{A}}(p)$. Using basic differential calculus arithmetic, J_q^{Ψ} can be derived to be exactly

$$J_q^{\Psi} = J_p^{f_{\mathbf{B}} \circ h} \left(J_p^{f_{\mathbf{A}}} \right)^{-1}, \tag{3.4}$$

which is composed of the Jacobian of $f_{\mathbf{B}}^{h}$ and the inverted Jacobian of $f_{\mathbf{A}}$ at point p, both readily available. Hence, to optimize the distortion of Ψ, we can take Equation 3.4, and plug it as the Jacobian used to define M in one of the distortion measures Equation 3.2, Equation 3.3, which we denote as $D(\Psi)$.

3.2.5.2 Optimizing h for bijectivity

For h to be a well-define surface map, it needs to map exactly bijectively (*i.e.*, 1-to-1 and onto) to the source domain of $f_{\mathbf{B}}$, which is $\Omega_{f_{\mathbf{B}}}$. To ensure that, we only need to ensure that h has a positive-determinant Jacobian everywhere, and maps to the target boundary injectively. We optimize h to map the boundary onto itself, via the energy

$$B(h) = \int_{p \in \partial\Omega_{f_{\mathbf{A}}}} \sigma(h(p)), \tag{3.5}$$

where σ is the signed distance function to the boundary of $\Omega_{f_{\mathbf{B}}}$. Note that the boundary map is free to slide along the boundary of Ω during optimization, enabling the boundary map to change. This is true for all points on the boundary, except those mapped to the four corners which are fixed to place and are essentially keypoint constraints between the two models.

Further, we also optimize h to encourage its Jacobian's determinant to be positive, via:

$$G = \lambda_{\text{inv}} \int \max\left(-sign\left(\left|J^{h}\right|\right) exp\left(-\left|J^{h}\right|\right), 0\right). \tag{3.6}$$

3.2.5.3 Keypoint constraints

Lastly a sparse set of corresponding key points on the two surfaces are given, and it is required that the surface map Ψ maps those points to one another. Given keypoints on $\mathbf{A}$, we can, in a preprocess before optimization, find their preimages in $\Omega_{f_{\mathbf{A}}}$, to get a set of points $Q^{\mathbf{A}}$ s.t. $f_{\mathbf{A}}\left(Q_i^{\mathbf{A}}\right)$ maps to the i-th keypoint. We likewise can find the preimages of the keypoints from $\mathbf{B}$ and their preimages $Q^{\mathbf{B}}$ under $f_{\mathbf{B}}$. If these key points are required to be mapped to one another between the two surfaces by Ψ, we can achieve that by requiring $h\left(Q_i^{\mathbf{A}}\right) = Q_i^{\mathbf{B}}$, which guarantees the induced Ψ maps the points correctly. We optimize for that equality by reducing its

least-squares error:

$$C(h) = \lambda_C \sum_i \left\| h\left(Q_i^{\mathbf{A}}\right) - Q_i^{\mathbf{B}} \right\|_2^2. \tag{3.7}$$

To facilitate the optimization, we apply a rotation, R, to the input of h. R is pre-computed from the landmarks.

3.2.5.4 Optimization for surface-to-surface maps

To compute the surface map, we optimize the distortion of Ψ with respect to h, while ensuring h respects the mapping constraints

$$\min_h D(\Psi) + C(h) + B(h) + G(h). \tag{3.8}$$

This yields a map h that maps onto the domain, and represents a distortion-minimizing surface map Ψ that maps the given sets of corresponding keypoints correctly, as shown for instance in Figure 3.1.

3.2.5.5 Cycle-consistent surface mapping

We also extend our method to discover inter-surface mapping among a collection of k surfaces $\mathbf{A}^1, \mathbf{A}^2, ..., \mathbf{A}^k$ represented respectively via neural maps $f_{\mathbf{A}}^1, f_{\mathbf{A}}^2, ..., f_{\mathbf{A}}^k$, we can define a *cycle consistent* [175, 176] set of surface maps by considering k additional neural maps, $h_i : \Omega_{f_{\mathbf{A}}^1} \to \Omega_{f_{\mathbf{A}}^i}$, define the composition $f_{\mathbf{A}}^i \circ h_i$, and then define the surface-to-surface maps $\Psi_{i\to j} : \mathbf{A}^i \to \mathbf{A}^j$ via $\Psi_{i\to j} \circ f_{\mathbf{A}}^i \circ h_i \triangleq f_{\mathbf{A}}^j \circ h_j$. This naturally allows extracting a set of mutually consistent maps while additionally optimizing for (all pairs) surface-to-surface maps, see Figure 3.7. Note that achieving similar qualities via classic methods is significantly challenging, while previous techniques could compute cycle consistency, none could optimize for true surface-surface distortion minimization over the entire collection.

3.3 Evaluation

We evaluate the discussed neural-mapping representation in the context of various mapping problems, such as surface parametrization, inter-surface mapping, and mapping a collection of shapes.

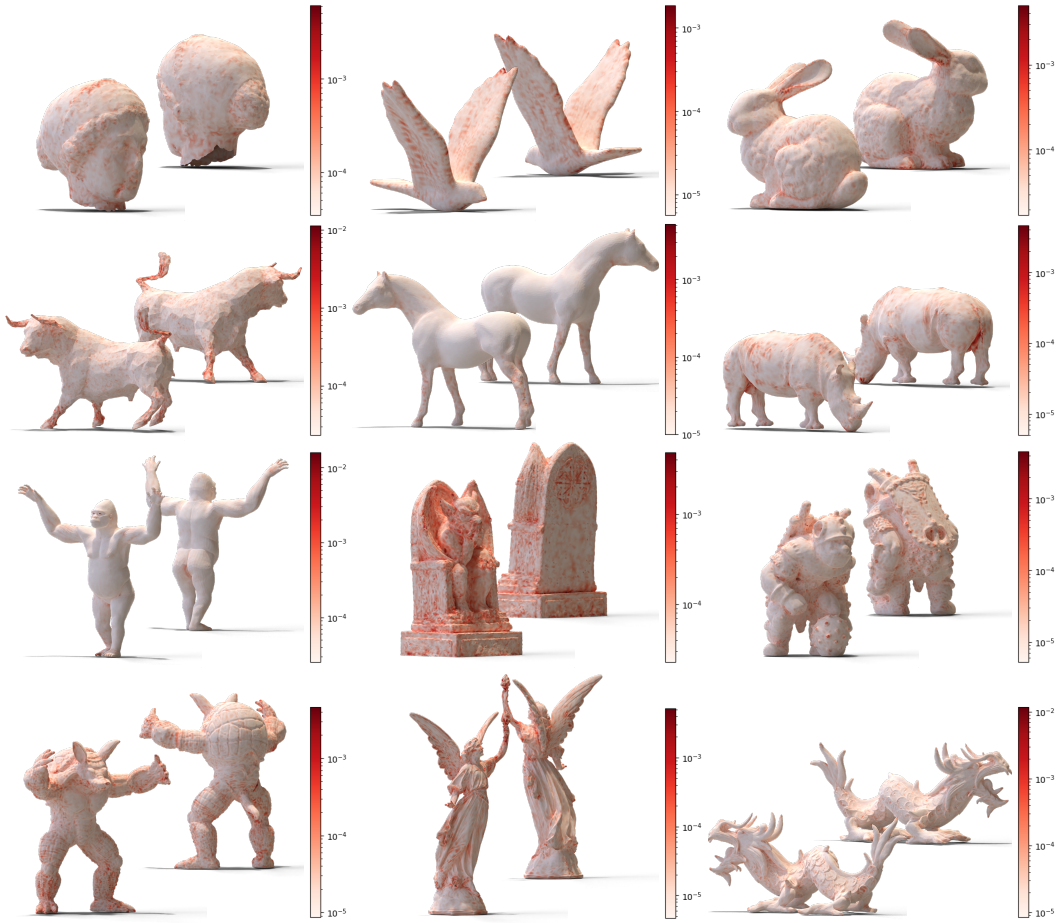


Figure 3.3: Neural surface maps faithfully represent the original surface, no matter the complexity. Here, we depict the point-wise error (L2 norm) for each vertex with respect to the ground truth as colour map.

3.3.1 Neural representation

Our overfitting procedure is able to capture even very detailed features of the original shape with a high fidelity. Figure 3.3 illustrates the difference between our reconstruction and the input mesh (highlighted in red). There are minor discrepancies between the models in regions exhibiting complex patterns like the Armadillo's legs. We observe that our reconstructions tend to be slightly smoother than the original shapes due to the use of Softplus activation.

3.3.2 Surface Parametrization

The main advantage of neural mapping is not in representing the surfaces, but in representing the mapping. We now take a map $f_{\mathbf{A}} : \Omega_{f_{\mathbf{A}}} \to \mathbb{R}^3$ from Figure 3.3, and

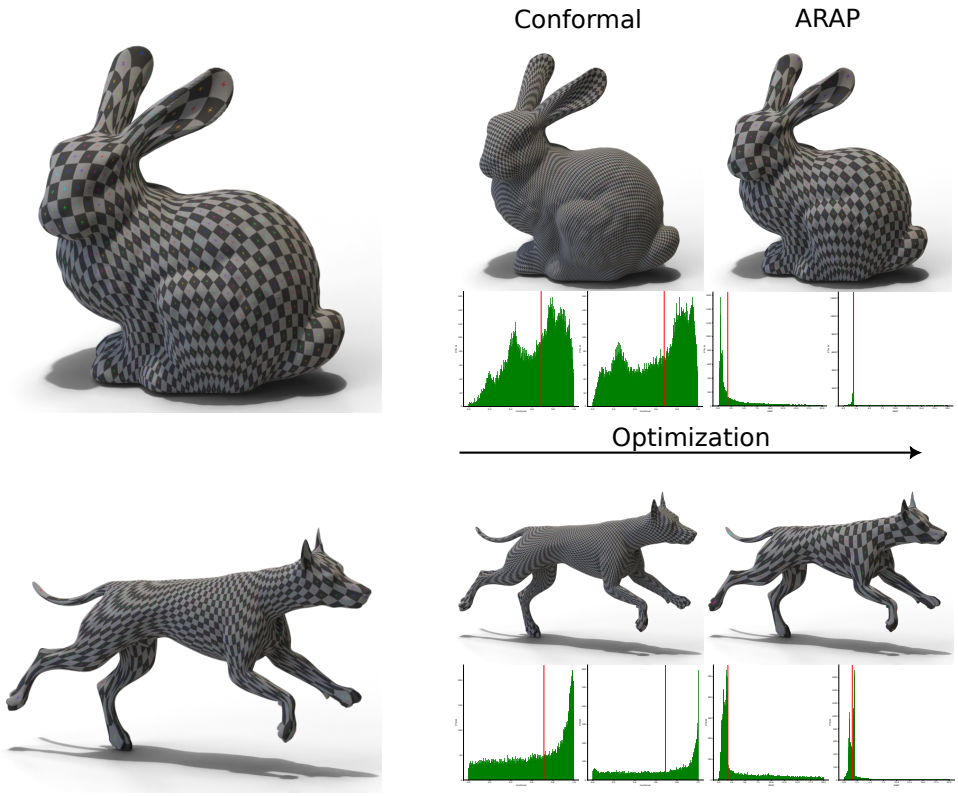


Figure 3.4: Starting from a Neural Surface Map (left), it is possible to optimize a neural map h minimizing conformal energy (centre) or isometric energy (right) with free-boundary. Independently to the model's size, the neural maps can parametrize the input mesh, with very few parameters. Adding a constraint over the boundary shape is as simple as regularize the mesh boundary.

introduce another map $h : \Omega_{f_\mathbf{A}} \to \mathbb{R}^2$, where we do not constrain its output domain. Similarly to the discussion in Section 3.2.5, we can define the map Ψ from the 3D model implicitly via as $\Psi(f_\mathbf{A}(p)) = h(p)$ for all $p \in \Omega_{f_\mathbf{A}}$. We then minimize the isometry distortion of Ψ (Equation 3.2), using the method to extract the Jacobian discussed in Section 3.2.5. Note that this objective is different from the one that was used to produce $f_\mathbf{A}$, hence we undo the original parametrization's distortion by compositing the neural map with a newly optimized map in Figure 3.4.

In contrast to UV parametrizations of meshes, the complexity of our optimization for this composition is completely independent of the resolution of the geometry.

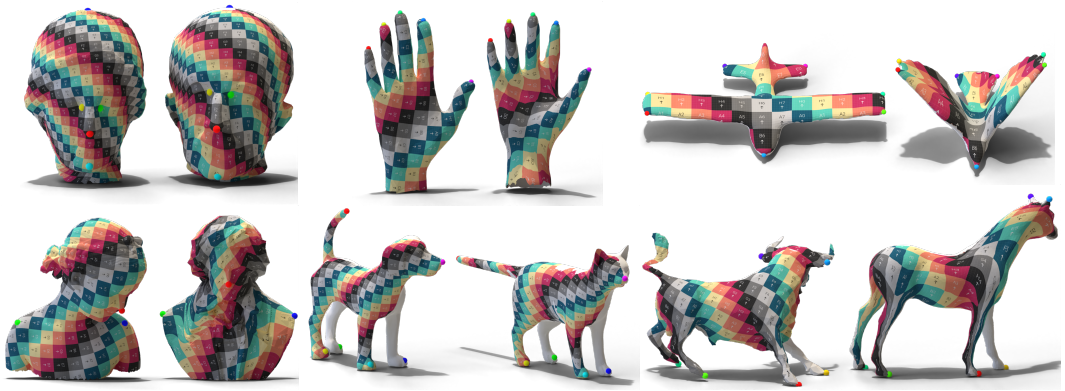


Figure 3.5: Here we showcase example of inter-surface maps. In each case, we first convert the models into Neural Surfaces, then we optimize a neural map h that minimizes the distortion of the map while respecting landmarks, bijectivity and continuity. We observe no inversion of flips in the maps. Corresponding landmarks are depicted with sphere of the same colour.

3.3.3 Surface-to-surface Maps

We can obtain a surface-to-surface map by compositing neural maps with a map between two atlases, as discussed in Section 3.2.5. In Figure 3.5, we depict several maps by mapping texture from one model onto the other. Spheres depict landmarks used to guide the map. Generally, our inter-surface maps exhibit low isometric distortion while corresponding regions are mapped correctly, see the wings of the plane and the wings of the bird. Furthermore, the proposed framework can map non-isometric shape pairs such as bull to horse. Note how despite significant geometric differences between surfaces, the result is a bijective, low-distortion mapping.

In all cases, we optimize only a neural map h initialized with map identity. Experimentally, we observe this speed-up the optimization, while leading to more stable results.

3.3.4 Composition with Analytical Maps

Our method can optimize an inter-surface map Ψ from $f_{\mathbf{A}}, f_{\mathbf{B}}$ just as well when $f_{\mathbf{B}}$ is not a neural map, but rather an analytical mapping defining some surface. Indeed, only h itself is required to be neural in our formulation of surface-to-surface maps. In Figure 3.6, we show mappings of Bimba and David into four such analytical surfaces. In this case, we optimize the conformal distortion Equation 3.3 of Ψ. Please refer to the supplementary for equations of the analytic surfaces.

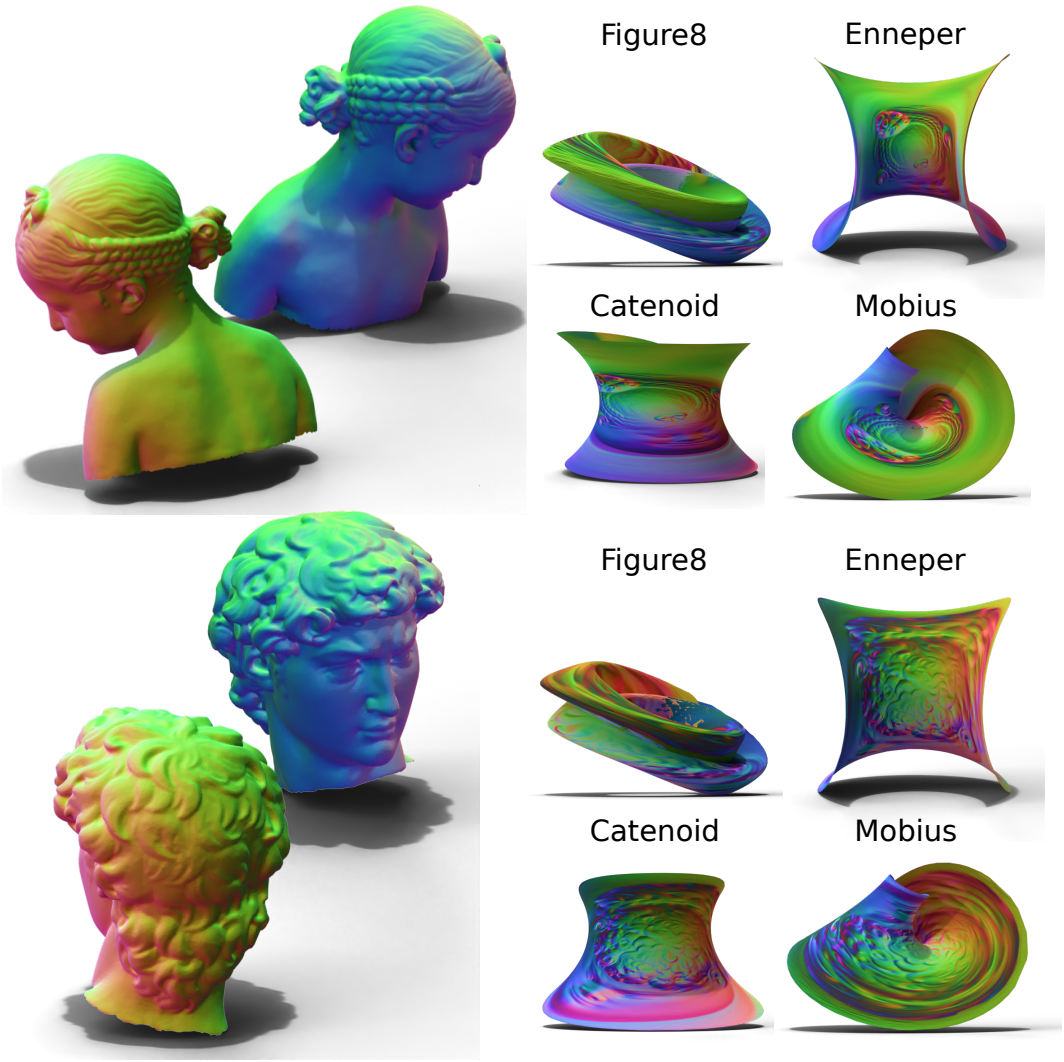


Figure 3.6: Inter-surface map between 3D models (left) and several analytical surfaces (right). Colours are based on the source model's normals.

3.3.5 Cycle-consistent Mapping for Collections of Surfaces

Finally, we show that thanks to the compose-ability of neural surface maps, our method can be efficiently applied to cycle-consistent mappings for a collection of shapes. Furthermore, since we use a common domain, the maps are guaranteed to be cycle-consistent, as in [175, 176]. We minimize the isometric distortion of the surface-to-surface maps between all pairs of surfaces in a collection of three models, following the method discussed in Section 3.2.5. Figure 3.7 illustrates that we were able to obtain cycle-consistent low-distortion maps between all shapes in the collection. We used keypoints, as shown by the sphere, to ensure correct

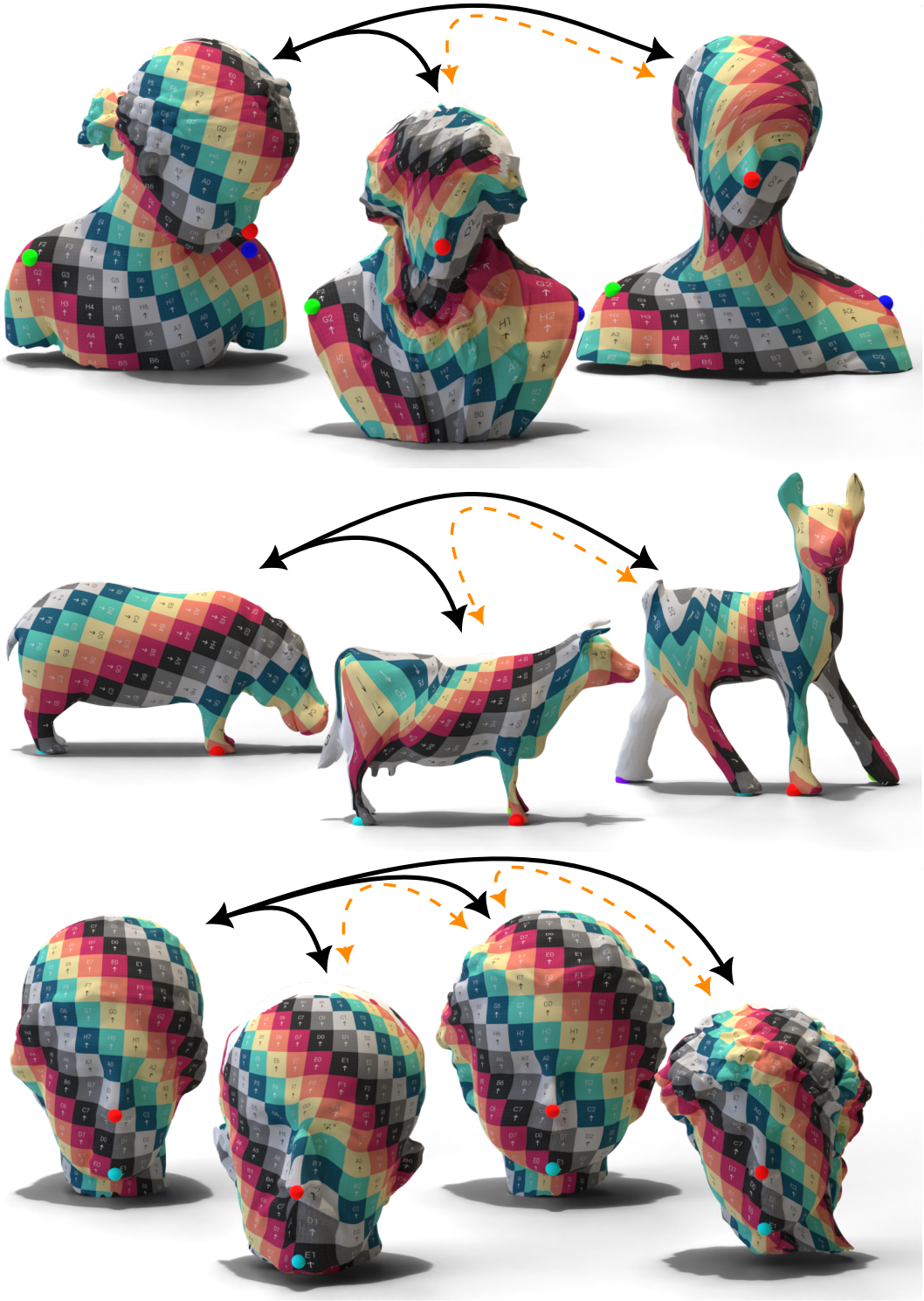
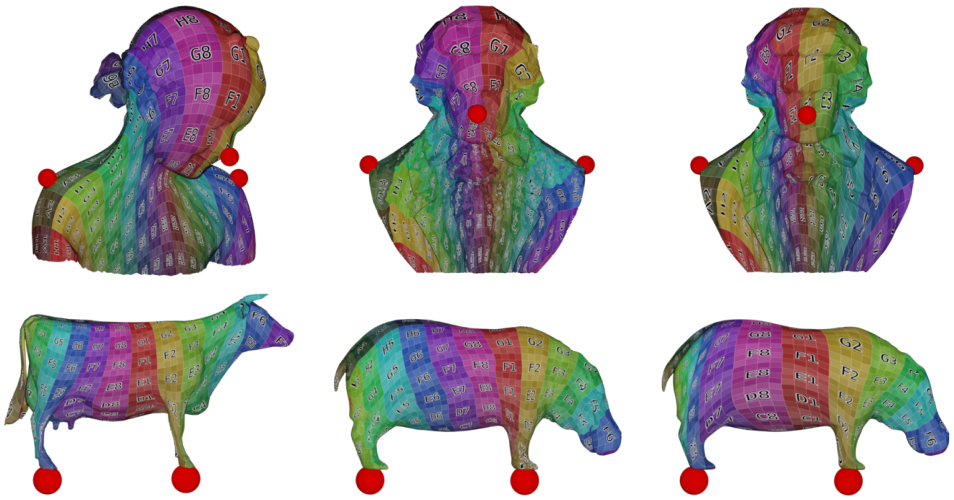


Figure 3.7: We define neural maps from one model (hub) to all other through neural maps, **`black arrows`**. Then, we minimize the distortion between *all pairs*, maps among all other nodes are implicitly defined by composing existing maps, **`orange arrows`**. This formulation ensures cycle consistency by construction. Spheres of corresponding colours depicts corresponding landmarks used during the optimization.



(a) source **(b)** [93] **(c)** ours

Figure 3.8: Comparison with inter-surface mapping [93].

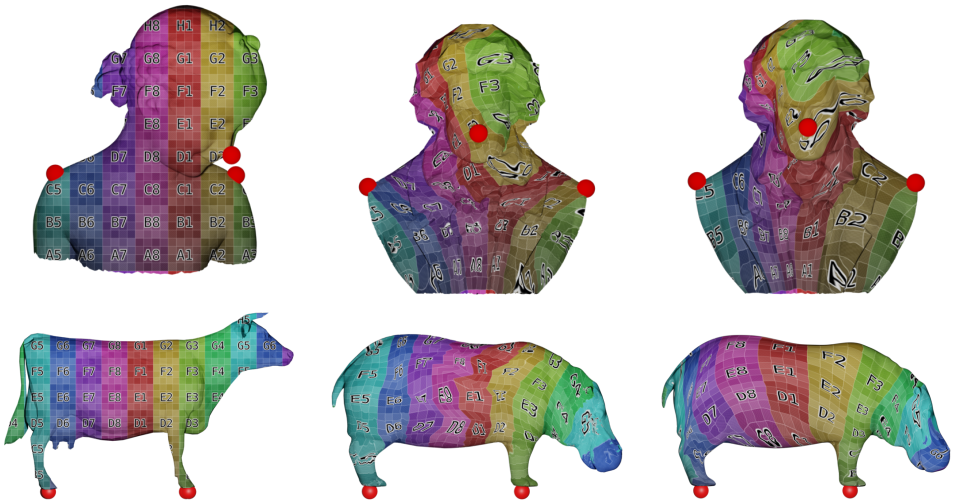
alignment.

3.3.6 Comparison

To validate neural surface maps, we offer visual comparisons with the classic inter-surface method [93] and Mandad *et al.* [167]. Schreiner *et al.* fails to produce smooth maps while matching landmarks: respectively for the bust and animal shown in Figure 3.8, [93] presents 8.58% and 8.54% triangles flips with a median $D_{iso} = 4.90$ and $D_{iso} = 7.00$. Similarly, Mandad *et al.* achieve a median $D_{iso} = 7146$, with 49.91% of flips, and $D_{iso} = 10669$ with 49.86% of flips, see Figure 3.9. Note, [167] introduces discontinuities in the map, resulting in large distortion and misalignment. On the other hand, our method offer a continuous, properly aligned, map. Numerically, our map for busts exhibit $D_{iso} = 7.00$ with no triangle flips, $D_{iso} = 8.56$ and 0.03% flips for animal.

3.4 Implementation Details

In all our experiments, we use a neural network consisting of ten-layer residual fully-connected network, with 256 hidden units per layer, with a Softplus activation function. We use $\lambda_n = 0.01$, $\lambda_B = 10^6$, $\lambda_{inv} = 10^2$, $\lambda_C = 10^3$ in all experiments. We sample the initial mesh uniformly with 500k points. Since our goal is to fully-



(a) source **(b)** [167] **(c)** ours

Figure 3.9: Comparison with state of art shape correspondence [167].

optimize the networks, they are trained until the gradient's norm drops below a threshold of 0.1. In all cases, we optimize the network with and RMSProp, and initialize the optimization procedure with a learning rate of 10^{-4} and momentum 0.9, the step size is modulated with [177]. Similarly, maps used for surface mapping are four-layer fully-connected network of 128 hidden units, with Softplus. In general, overfitted networks converge in 1h, while, surface-map and collection-map optimization take around 1.5h to reach a stable configuration.

3.5 Limitations

The representation and mapping framework that stems from it has several limitations. For one, it can only represent disk-topology surfaces, other topologies can be approached with cuts. Furthermore, we assume the cuts to be in correspondence, which it might not be a realistic assumption.

Regarding inter-surface maps, we assumed h being bijective and mapping the keypoints correctly; in theory, we cannot guarantee that this requirement is upheld, however, in our experiments, it is rare for this condition to be violated.

The overfitting and map optimization are computationally expensive, requiring a matter of hours for the networks to be trained.

3.6 Conclusions

In this chapter, we introduced Neural Surface Maps as a core representation for surfaces that is easily differentiable and composeable. Using the common domain approach, we can optimize for different properties of parametrization or inter-surface maps. Overfit to individual meshes allows encoding shapes as network weights, and subsequently optimize maps while keeping the surface approximation quality fixed. We demonstrated the universality of neural maps addressing challenging classical tasks including parametrization, surface-to-surface distortion minimization, and extracting maps across a collection of shapes.

Future works

We see many immediate uses to the differentiability and composability of our representation, such as applying differential geometry operators to the models as well as solving PDEs on them. Resorting to neural network generalization capabilities can bring large high-resolution dataset within our reach, exposing neural surface maps to applications like segmentation and classification.

Chapter 4

Neural Semantic Surface Maps

4.1 Introduction

In Chapter 3 we introduced a mapping framework which allows us to compute inter-surface maps between shapes given manual annotations. In this chapter, we extend NSM and propose an *automatic* method to compute a continuous correspondence between two genus-zero surfaces. The core contribution is an approach for computing *semantic* maps that matches semantically corresponding points to one another (e.g., nose to nose, arm to arm, etc.).

As discussed earlier, computing correspondences between domains is a fundamental problem spanning a wide array of domains such as text snippets [178], audio [179], images [180], or general graphs [181]. In the context of 3D surfaces, establishing such correspondences enables texture or deformation transfer [182, 183], shape analysis [184, 185, 186, 187], and shape space exploration [188, 189, 190].

Continuous surfaces (2-manifolds), encoded as triangular meshes, remain the most natural and common representation of 3D shapes in graphics and discrete differential geometry. Correspondence between two such surfaces is typically required to be a map that is continuous, one-to-one, onto, and with a continuous inverse, *i.e.*, a *homeomorphism*. Decades of research (see surveys [117, 191]) have been dedicated to tackle the task of mapping between surface pairs. These previous works, being geometric, cannot extract (semantic) maps over the space of homeomorphisms; instead, they have focused on surrogate optimization tasks that minimize some geo-

Figure 4.1: Here, we show the extracted map between two non-isometric shapes, `tiger` and `iguana`. Although the shapes are highly non-isometric, *e.g.*, lengths of the tail and legs, our method successfully associated these regions between shapes, thus yielding a semantically correct map. This map is seamless by construction, and optimized with no supervision thanks to pre-trained ViT models which can identify semantically corresponding points across shape renderings.

metric notion of "distortion" of the map, *e.g.*, preserving geodesic distances as best as possible. Such distortion-minimizing geometrically-guided maps are, of course, not necessarily semantically meaningful. Thus, a human-in-the-loop approach is usually taken to manually indicate landmark correspondences, which are then used to optimize a map.

Computing semantic homeomorphism faces two main challenges. First, the lack of annotated 3D data inhibits learning high-level semantic priors. Second, most 3D representations either hinder or – completely – prevent the computation of bijective inter-surface maps from semantic priors. In contrast, recent works [131, 133, 134, 135, 21] demonstrate that the features of a pre-trained vision transformer (ViT) are often semantically meaningful and can be used reliably across multiple vision tasks, even on out-of-training image data in a zero-shot setting. We aim to bridge the semantic matching capabilities of the image domain with the computation of inter-surface maps from potentially noisy correspondences, encouraging continuity and bijectivity. Our core observation is that suitable renderings of the surfaces, without access to surface texture, are already sufficient for image transformers (*i.e.*, ViT) to produce 2D matches that can subsequently be used as fuzzy (*i.e.*, partial and non-injective) maps between the surfaces. Then, we formulate an optimization to aggregate multiple such fuzzy matches obtained from

multiview renderings to produce a surface map that best conforms to these fuzzy matches, thereby distilling their semantic priors, see Figure 4.1.

Specifically, given the fuzzy matches, we utilize Neural Surface Maps (NSM) defined in Chapter 3 to optimize a map between two surfaces. The original NSM framework encodes surfaces using dedicated neural functions, offering a differentiable backbone and avoiding complexities arising from topological changes (i.e., mesh connectivity for different triangulations). However, it has two limitations: it expects the individual surfaces to be cut into disc topologies with the two respective boundaries *already* in correspondence, and requires a set of exact landmark correspondences. We address the first problem by proposing a seamless Neural Surface Maps (sNSM) framework, which relaxes the requirement from exact boundary correspondences to only cone-point matchings. We address the second problem by optimizing a custom objective that encourages the image of a specific point to best accommodate the fuzzy (semantic) matches while identifying and disregarding outliers (see Figure 4.4). The resultant optimization problem is solved using gradient descent, simply through PyTorch's SGD optimizer.

Through quantitative and qualitative experiments, we evaluate our ability to match upright object pairs with varying levels of isometry for objects from the same semantic class and across different ones. We also compare ours to competing surface map extraction algorithms. In summary, our main contributions are:

- proposing a fully automatic algorithm for extracting semantic maps between upright shape pairs;
- sampling and integrating a set of image-based correspondences to form fuzzy object space correspondence maps;
- extending the Neural Surface Maps framework proposed in Chapter 3 to seamless maps that can work with fuzzy, and potentially noisy guidance, to distil semantic maps; and
- demonstrating, via extensive evaluation and comparisons, that the algorithm yields semantically valid maps for both isometrically and non-isometrically

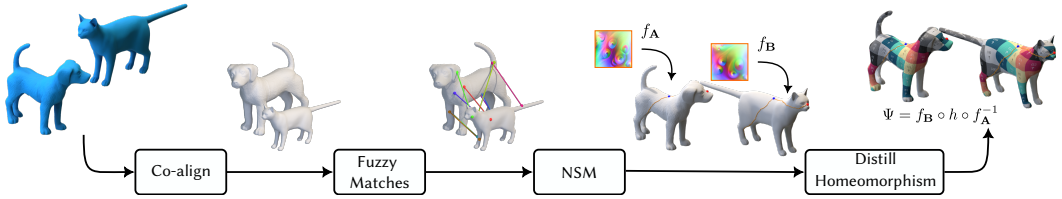


Figure 4.2: Overview. Starting from a pair of upright genus-zero surfaces, we automatically distil an inter-surface map from a set of fuzzy matches. First, we align the input shapes, then extract a set of fuzzy matches through DinoV2 [21] semantic visual features. We use these features to independently cut the two meshes and then optimize a (seamless) map between them.

related shape pairs.

4.2 Method

We now detail our framework (see Figure 4.2) for an automatic inter-surface map. We assume to be given two upright 3D surfaces, **A** and **B**, in arbitrary relative poses. The majority of meshes from online repositories, such as the 3D Warehouse, TurboSquid, or Sketchfab, satisfy this requirement, alternatively, methods like [192, 193] can be used as pre-processing. We assume both shapes to have zero genus, although the method can be extended to higher genus surfaces. We aim to compute an inter-surface map $\Psi : \mathbf{A} \leftrightarrow \mathbf{B}$ guided by visual semantics. Our framework proceeds in three stages:

1. Given two shapes, **A** and **B**, that are assumed to be oriented upright, we automatically align them using semantic matches.

2. We aggregate fuzzy matches (i.e., general matching of pairs of points which is neither 1-to-1, onto, nor maps all points on the source surface) between the surfaces by applying 2D matching techniques to renderings, over multiple views.

3. Optimize a surface map that best agrees with the fuzzy semantic matches while handling outliers.

We provide pseudocode for our semantic homeomorphic map extraction framework in Algorithm 1.

Algorithm 1: Semantic Surface Homeomorphism

Data: source **A**, target **B**
$\mathbf{R} \leftarrow \text{COALIGN}(\text{DinoViT}(), \mathbf{A}, \mathbf{B})$;
`fuzzyMatches` $\leftarrow \text{COMPUTEMATCHES}(\text{DinoViT}(), \mathbf{A}, \mathbf{B}, \mathbf{R})$;
$A_{disk}, B_{disk} \leftarrow \text{ASYNCCUT}(\mathbf{A}, \mathbf{B}, \texttt{fuzzyMatches})$;
$A_{NSM} \leftarrow \text{OVERFITNSM}(A_{disk})$;
$B_{NSM} \leftarrow \text{OVERFITNSM}(B_{disk})$;
`map` $\leftarrow \text{DISTILMAP}(A_{NSM}, B_{NSM}, \texttt{fuzzyMatches})$;
return `map`

4.2.1 Semantic Shape Alignment

Given two upright shapes, **A** and **B**, we first align them to have the same orientations. We achieve this by casting this problem as (semantic) circular string matching between shape renderings: given two "strings" – sets of renderings – of the same length, we find the global rotation **R**, about the upaxis, to best align one string with the other. Intuitively, we order one sequence to convey semantic information in the same order as the other, see Figure 4.3 for an overview.

First, we render each mesh from 12 viewpoints around it, $R_i^{\mathbf{A}}$ and $R_i^{\mathbf{B}}$ (see Section 4.2.10 for a discussion on rendering). These images constitute the two strings $s^{\mathbf{A}} := \{R_i^{\mathbf{A}}\}$ and $s^{\mathbf{B}} := \{R_i^{\mathbf{B}}\}$. Then, we extract a set of DinoV2 [21] features for each image. Finally, we compute the alignment score for the 12 possible rotations as the total number of "Best Buddy matches" [194] between the two strings of features. We pick the (relative) rotation with the highest score as the rotation and use it to co-align the shapes.

4.2.2 Distilling Fuzzy 3D Matches via Visual Semantics

Next, we extract fuzzy matches from renderings, taken from different viewpoints, of the aligned surfaces. Each such viewpoint V results in a pair of rendering that we use to define a fuzzy correspondence $\phi^V := \left(p_i^V, q_i^V\right)_{i=1}^n$ with $p \in \mathbf{A}, q \in \mathbf{B}$, which consists of pairs of corresponding points on **A** and **B**.

Although correspondences are imprecise or inaccurate, we assume that these imprecisions balance out, leading to approximately correct matches. Embracing this assumption, we leverage it as a guiding principle during map optimization.

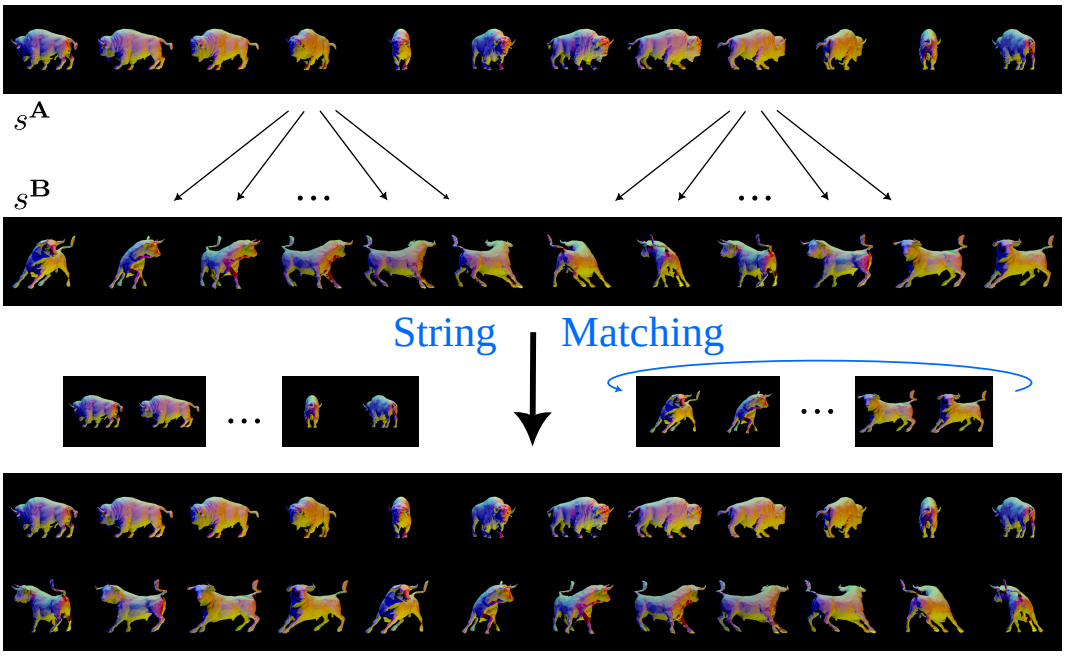


Figure 4.3: Co-aligning input surfaces: Starting from a pair of upright meshes (`bison` and `bull` in this example), we render 12 views around them ($s^{\mathbf{A}}$ and $s^{\mathbf{B}}$). Then, we extract DinoV2 features from each rendering independently and match these features as a string-matching problem. Specifically, we optimize over a cyclic shift of the rendered views (i.e., one degree of freedom) to maximize agreement of image-based semantic correspondences.

4.2.3 Computing rendering correspondences.

Given a viewpoint V, we render the two untextured surfaces from that viewpoint to get two renderings, $R_V^{\mathbf{A}}$ and $R_V^{\mathbf{B}}$. To extract correspondences, we take inspiration from recent methods that leverage deep image features from [21] for matching 2D images and design a method for extracting dense visual correspondences. Specifically for each image patch processed by DinoV2, we extract a feature vector with $\lambda_i^{\mathbf{A}}$ and $\lambda_i^{\mathbf{B}}$ being the features of rendering of $R_V^{\mathbf{A}}$ and $R_V^{\mathbf{B}}$, respectively. Then, we segment foreground/background through PCA and compute the cosine similarity between all pairs of source and target patch foreground features, as score

$$S_{ij} = \left\langle \lambda_i^{\mathbf{A}}, \lambda_j^{\mathbf{B}} \right\rangle. \tag{4.1}$$

Finally, we define the match of patch $i \in R_V^{\mathbf{A}}$ as the patch $j \in R_V^{\mathbf{B}}$ with the highest cosine similarity, and vice versa, the match of patch $j \in R_V^{\mathbf{B}}$ as the patch $i \in R_V^{\mathbf{A}}$ with the highest cosine similarity. In summary, the pair $(i, j), i \in R_V^{\mathbf{A}}, j \in R_V^{\mathbf{B}}$ is a match,

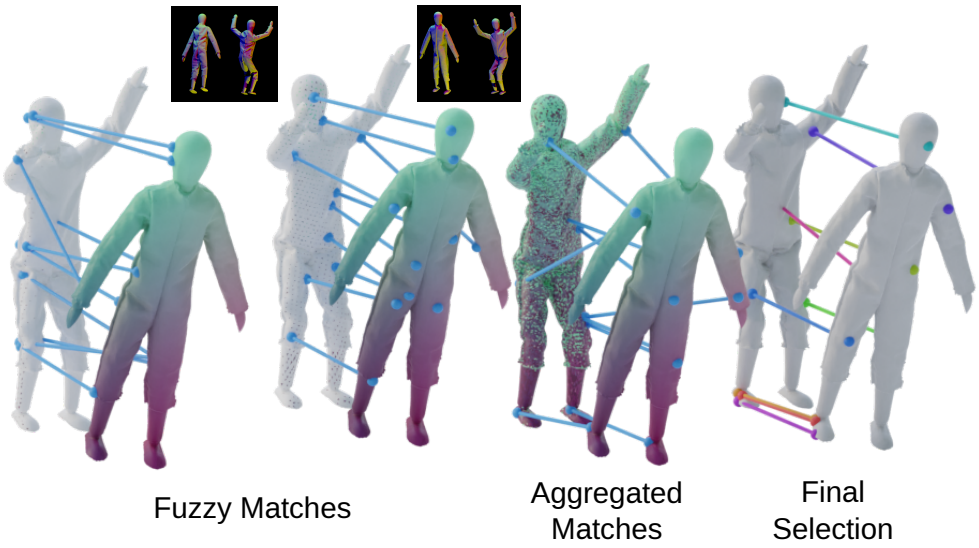


Figure 4.4: Fuzzy semantic correspondences. (Left) We lift 2D image-based correspondences, obtained using a pre-trained vision-transformer [21] on rendered image source/target pairs from sampled views, to obtain fuzzy and spurious 3D (semantic) correspondences. We collect correspondence, shown with colouring and a random set highlighted with lines, from each of the sampled views and aggregate them across views to get aggregated fuzzy matches (middle), which contain erroneous matching, e.g., thigh getting mapped to the arm. (Right) We propose an optimization to distil these fuzzy matches into an inter-surface map, here depicting a subset of matches closer than a given threshold ($d < 0.1$) wrt the optimized map.

if

$$S_{ij} = \max_k S_{ik} \text{ or } S_{ij} = \max_l S_{lj}. \tag{4.2}$$

We transform a match from patch level to pixel level, as the patch size is known. In contrast to ours, [135] selects only "Best Buddy" matches [194], augments features with binning, and does not segment foreground/background through PCA features. Although [135] produces a more expressive set of features and possibly a more reliable set of fuzzy matches, we found it time-consuming (2hrs in our settings), and our experiments did not provide sufficient justification for such a design choice.

Given dense 2D correspondences in an image, we lift (unproject) each pixel to the 3D mesh by performing ray intersection between that pixel's corresponding ray from viewpoint V and the 3D mesh $\mathbf{T}$, thereby associating every 2D pixel with a point on the surface, represented as barycentric coordinates at the triangle the ray intersects. The fuzzy matches are thus pairs of matching 3D points (represented as

barycentric coordinates on triangles): $(\phi^V = p_i, q_i)_{i=1}^n$.

We repeat this process from multiple viewpoints and obtain a collection $\{\phi^i\}_{i=1}^k$ of fuzzy matches. Our final task is to distil them to produce an automatic map.

4.2.4 Aggregating the Fuzzy Matches to an Inter-surface map

Given the fuzzy matches, we wish to optimize a continuous map Ψ between $\mathbf{A}$ and $\mathbf{B}$ using a differentiable loss that encourages agreement with the fuzzy matches.

Our final goal is thus to devise an optimization scheme that will lead to a map $\Psi : \mathbf{A} \leftrightarrow \mathbf{B}$ which balances smoothness with the number of respected matches. We compare each point's image with its designated matches from ϕ^i to achieve this goal. We then use an L_1 norm over the error of different points, which is known to be robust to outliers and encourage sparsity (*i.e.*, agree with as many matches of the fuzzy matches as possible). In practice, we select a subset (N) of all possible correspondences and minimize the discrepancy between the predicted point and its ground truth, thus averaging across redundant noisy matches:

$$\mathscr{L}_{\text{Matches}} = \frac{1}{N} \sum_{j=1}^{N} \|\Psi(p_j) - q_j\|_1, \tag{4.3}$$

where (p_j, q_j) is a randomly selected correspondence. To evaluate $\mathscr{L}_{\text{Matches}}$ we first extract the barycentric coordinates of p_j and convert it to a point in the square, $p_j^{2D} \in \mathbb{R}^2$. Then, this point is mapped forward through $f^{\mathbf{B}} \circ h$ and used to compute the error as $\|f^{\mathbf{B}}(h(p_j)) - q_j\|_1$ for each match. To optimize Equation 4.3, we adopt a recent method for optimization of the surface map, Neural Surface Maps (NSM) Chapter 3 as described next.

4.2.5 Seamless Neural Surface Map.

We follow NSM's paradigm: we first parametrize each one of the two cut surfaces via SLIM [23] into a *square* $D \in \mathbb{R}^2$ to get two bijective seamless parametrizations, $P_{\mathbf{A}} : \mathbf{A} \leftrightarrow D, P_{\mathbf{B}} : \mathbf{B} \leftrightarrow D$. Then, we fit a neural network to each of the two parametrizations' inverse, $f_{\mathbf{A}} \approx P_{\mathbf{A}}^{-1}$, $f_{\mathbf{B}} \approx P_{\mathbf{B}}^{-1}$. Finally, using another neural network that maps the square to itself, h, we can define the inter-surface map

$\Psi = f_{\mathbf{B}} \circ h \circ f_{\mathbf{A}}^{-1}$. By optimizing solely the parameters of h while maintaining its bijectivity, and holding the overfitted networks $f_{\mathbf{A}}, f_{\mathbf{B}}$ fixed, NSM enables optimization over the space of maps between the two surfaces.

As we cannot guarantee corresponding cuts between genus 0 meshes, see cut examples in Figure 4.5, we relax the boundary-matching constraint in the original NSM and extend it to support seamless maps. Intuitively, a borderless, or seamless, parametrization is a $2D$-$3D$ mapping that is independent of the choice of cut path, given a set of K boundary points. In other words, the map emerging from the parametrization has several equivalent maps with different boundaries, see Figure 4.6(c). Only the K points, referred to as cones, remain constant and must have the same mapping across all equivalent maps. Mathematically, a seamless parametrization is a mapping equipped with homotopic cuts (i.e., the cuts can be changed homotopically but the produced mapping will stay the same). In particular, for three cones on a sphere, all cuts are homotopic, and thus the embedding is independent of the cut choice. Please refer to [90] for more details.

Furthermore, the class of seamless parametrization requires a specific type of cut such that triangles, or points for that matter, can be mapped to the other side of the cut by a family of transformations R. In terms of NSM, a seamless map requires matching corresponding cones while the boundary is allowed to move. Thus, the

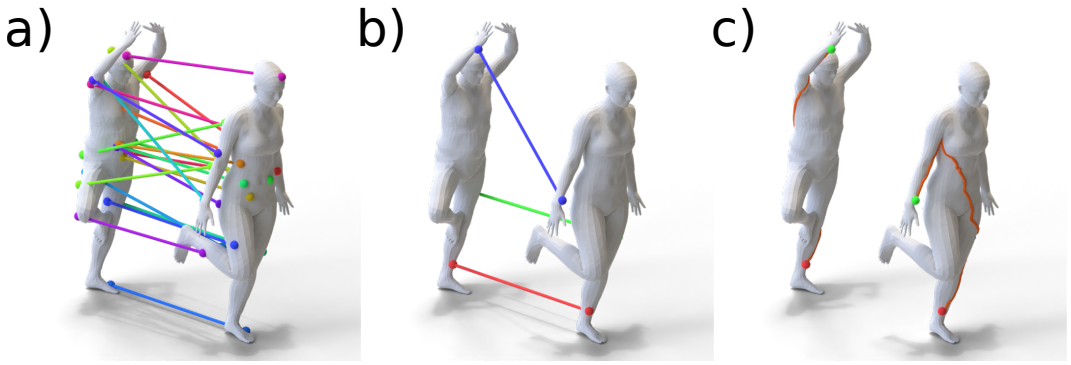


Figure 4.5: Cutting through cone points. We collect a set of spurious and noisy matches (a). Then, we select the most reliable $K = 3$ correspondences (b). Finally, using these matches as cut endpoints, or cones, we cut the two meshes independently (c). Note how the cut differs in the two shapes: the man is cut through the back, while the woman is cut through the front. Refer to Sec. 4.2.6 for details.

accuracy required to define the cut path, and hence the $2D$ boundary, through fuzzy matches is reduced, *e.g.*, see Figure 4.5(c) for cut paths. Below, we first detail how we extract corresponding cones, then describe a seamless map.

4.2.6 Cones.

To identify cones, we first aggregate the fuzzy matches by counting for each triangle $F_i^{\mathbf{A}} \in \mathbf{A}$, how many fuzzy matches associate it with triangle $F_j^{\mathbf{B}} \in \mathbf{B}$, yielding a large sparse matrix M such that its (i, j) entry is the total count for matches of $F_i^{\mathbf{A}}$ to $F_j^{\mathbf{B}}$,

$$M_{ij} = \sum_k \left| \left\{ (p,q) \text{ s.t. } p \in F_i^{\mathbf{A}}, q \in F_j^{\mathbf{B}}, (p,q) \in \phi^k \right\} \right|, \tag{4.4}$$

where $|\cdot|$ stands for the cardinality of the set. Next, we consider M as the adjacency matrix of an edge-weighted graph, with M_{ij} being the weight on edge (i, j). Then, through bipartite graph matching [195] we obtain a matching, *i.e.*, a list of pairs (i_k, j_k), s.t. $i, j = \underset{i,j}{\operatorname{argmax}} M_{i,j}$. We select the $K = 3$ correspondences (i, j) with the highest M_{ij} values, such that the geodesic distance – averaged between the two shapes – between all K points is at least $\tau = 0.3$. Finally, we use these landmarks as the cut's endpoints and the midpoint.

4.2.7 Seamlessness.

Since we cannot rely on the cut quality, we reformulate the neural map h, which we optimize to define the map Ψ, to support seamlessness. This constrains the map to work on shape pairs with the same genus. Furthermore, the definition of the seamless map changes based on the genus. For a sphere, h changes to $\tilde{h}$:

$$\tilde{h} = \left\{ x \to T \cdot h(x) + \eta \,|\, T = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \in \mathbb{R}^{2\times 2}, \eta \in \mathbb{R}^{2\times 1} \right\} \tag{4.5}$$

for all points mapped outside the domain Ω, which rotates around the cone c_i (η) of a rotation $R(T)$.

To achieve seamlessness h must perfectly match cones c_i to their ground truth $\tilde{c}_i$. Therefore, we formulate such a constraint by penalizing the deviation of mapped

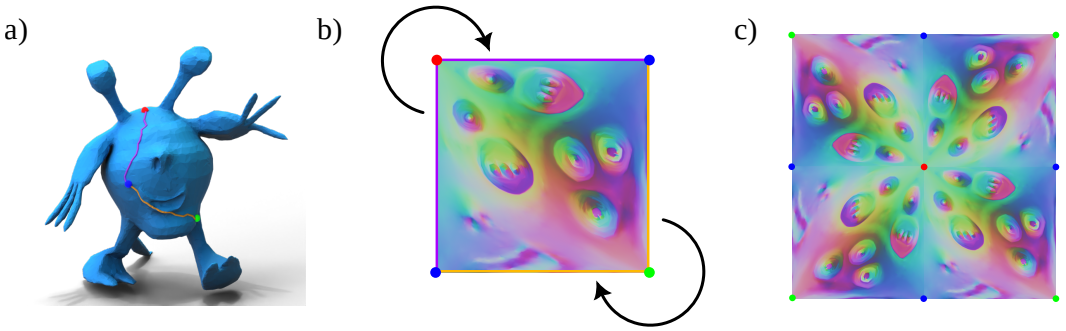


Figure 4.6: Seamless cuts. To parametrize a genus-zero mesh (a) we cut and map it to a disc topology, cut visualized as in (b). The two corresponding sides of the cut match perfectly, i.e., when we connect the two parts, the map remains continuous across the cut (c).

cones, $h(c_i)$, to their ground truth position:

$$\mathscr{L}_{\text{Cones}} = \|h(c_i) - \tilde{c}_i\| \,. \tag{4.6}$$

In the case of spheres, we have 3 cones, of which one is duplicated, thus one for each vertex of the square in the square domain Ω. In the case of torus, a single point is duplicated 4 times, corresponding to all 4 square vertices.

A second condition for seamlessness concerns the duplicated points on the boundary. In the case of spheres, each point on the boundary p_1 has a corresponding point p_2 which is a rotation of 90° with respect to one of the cones. For the case of a sphere, we formulate the constraint as the following energy:

$$\mathscr{L}_{\text{Seamless}} = \|h(p_1) - R \cdot (h(p_2) - c_i) + c_i\| \,, \tag{4.7}$$

where c_i is the cones wrt p_2 undergoes a rotation R to be a clone of p_1. Note, the rotation can either be $\pi/2$ or $-\pi/2$. In the case of a torus, p_2 is on the opposite side of the boundary of p_1, i.e., the transformation being a translation along x or y.

4.2.8 Optimization energies.

We follow Section 3.2.5 and encourage the map to be bijective through a loss term that prevents the map h's Jacobian J_p at every point $p \in \Omega$ from having a negative

determinant:

$$\mathscr{L}_{\mathrm{J}} = \int_{\Omega} \max\left(-sign\left(|J_p|\right) e^{-|J_p|}, 0\right). \tag{4.8}$$

Thus, encouraging, but not guaranteeing, continuity and bijectivity of the map.

To cope with the sparsity of fuzzy matches and obtain a well-defined map in undefined regions, we use an energy term that encourages smoothness and prevents large distortion:

$$\mathscr{L}_{\mathrm{Smooth}} = \int_{\Omega} \left\| J_p^{\Psi} - J_{p^{\varepsilon}}^{\Psi} \right\|, \tag{4.9}$$

where J_p^{Ψ} is the Jacobian at a point p of the map Ψ. While p^{ε} is the point p perturbed by $\varepsilon \sim \mathscr{N}(0, 0.1)$ through barycentric coordinates. Intuitively, we want the Jacobian of the map to change slowly. Note, in Chapter 3, the authors used Symmetric Dirichlet [23] in a similar context, however, this energy promotes isometric maps rather than smooth ones. Such behaviour can actively damage the map optimization and force it to ignore certain – correct – matches, while we aim to attend to unregularized areas.

4.2.9 Total energy.

Our total loss is expressed as:

$$\mathscr{L} = \alpha_1 \mathscr{L}_{\mathrm{matches}} + \alpha_2 \mathscr{L}_{\mathrm{J}} + \alpha_3 \mathscr{L}_{\mathrm{Cones}} + \alpha_4 \mathscr{L}_{\mathrm{Seamless}} + \alpha_5 \mathscr{L}_{\mathrm{Smooth}}, \tag{4.10}$$

where $\alpha_1 = 10^4$, $\alpha_2 = 10^6$, $\alpha_3 = 10^6$, $\alpha_4 = 10^6$, and $\alpha_5 = 10^{-1}$ in all experiments. These hyper-parameters were selected experimentally. We optimize network weights h using this loss, and to alleviate the impact of incorrect matches; we incrementally drop those that strongly disagree with the current map, *i.e.*, 20% of matches with the highest Euclidean distance. Experimentally, this explicit filtering reduces the impact of incorrect matches, thus preventing the network from getting stuck in incorrect energy minima due to the presence of inaccurate matches.

In summary, the proposed pipeline optimizes for an inter-surface map *automatically* between a pair of upright shapes. First, the input mesh pair is automatically aligned, then through pre-trained ViT [21] we extract a large set of semantic fuzzy

matches between them. Finally, we distil an inter-surface map; this step is fundamental to filter out any incorrect matches, enhancing the overall accuracy and reliability of the resultant map. Next, we quantitatively and qualitatively evaluate the quality of the distilled map.

4.2.10 Rendering Settings

We render shape pairs and use these images with DinoV2 [21]; this model is known to be forgiving in cases of image variation. As the shape alignment is unknown, we render an object-centric scene with a fixed perspective camera and 5 point lights aimed at the shape. Different points of view are obtained by rotating *only* the shape by fixed increments, while the rest of the scene (*i.e.*, camera and lights) stays fixed. We set up the scene to ensure the entire object is visible by the camera's field of view.

To boost the matching capabilities of Dino-ViT and aid it in distinguishing left from right, top from bottom, while enhancing scene details, we strategically position coloured lights around the object in a half-dome fashion. Specifically, we employ five coloured point lights (red, blue, green, yellow, and white) for this purpose. As depicted in Figure 4.3, corresponding regions in the images exhibit similar colours; for instance, the right part of the images tends to appear reddish due to illumination from a red light source. In cases involving textured meshes, we replace the coloured lights with white ones.

4.3 Evaluation

We evaluated our method on various datasets for inter-surface mapping and compared it against multiple baselines that focus on obtaining surface-to-surface maps.

4.3.1 Datasets

We assess maps' quality on available benchmarks comprising isometric and non-isometric shape pairs. (i) We randomly select 30 pairs from FAUST [186], containing isometric deformations and pose variations of human shapes. (ii) We choose 30 random same-category shape pairs from SHREC07 [198], containing non-isometric deformations across multiple categories of shapes. (iii) We also extract 30 random

shape pairs among the listed test set of SHREC19 from Dyke *et al.* [199], containing a mix of isometric and non-isometric deformations.

To ablate the effect of pose variation, we use FAUST [186], SCAPE [200], and TOSCA [201]. To ablate the effects of rendering settings and rotation, we use FAUST [186]; 3DBiCar [202], which comprise a variety of textured shapes; and SHREC15 [203], which contain significant non-isometric-variations, with manually-annotated sparse correspondences. Furthermore, we present additional ablations highlighting the crucial role of initial alignment, the method's robustness to mesh holes and noise, and discuss DinoV2 features. To summarize, significant misalignment negatively impacts matching quality; feature similarity does not reflect their matching accuracy; finally, the method effectively maps meshes with holes, *e.g.*, scans.

All meshes used in our experiment are watertight and genus zero, and range from 11K to 90K faces. The shape pairs include a mix of some isometric and mostly non-isometric cases.

4.3.2 Metrics

We assess map quality (see also [127]) based on their accuracy, bijectivity, and inversion as:

- **Accuracy** (*Acc* $\downarrow$): measures the ability of the algorithms to respect ground-truth correspondences. We measure it as the geodesic distance normalized, as defined in [114], for each landmark.

- **Bijectivity** (*Bij* $\downarrow$): measures the geodesic distance of all vertices mapped forward, and then back to the source mesh wrt their original position. A zero value indicates perfect bijectivity.

- **Inversion** (*Inv* $\downarrow$): measures how often the map flips the surface as the percentage of inverted triangles; we compute it as the agreement of the normal of the mapped triangles wrt the faces on which the triangle vertices are mapped.

Table 4.1: Quantitative evaluation. We compute each map's accuracy (i.e., average geodesic error) and averaged them over 30 shape pairs for each dataset.

	FAUST			SHREC07			SHREC19		
	Inv ↓	Bij ↓	Acc ↓	Inv ↓	Bij ↓	Acc ↓	Inv ↓	Bij ↓	Acc ↓
ICP	0.06	0.17	0.25	0.09	0.65	**0.23**	0.07	0.75	0.15
BIM	0.09	0.03	0.04	0.49	0.48	**0.23**	0.05	0.82	0.04
ZoomOut	0.33	0.23	0.15	0.25	0.65	0.54	0.29	0.76	0.32
Smooth-shells	0.01	**0.00**	**0.01**	0.03	0.72	0.26	0.01	0.83	**0.01**
Ours	**0.00**	**0.00**	0.13	**0.00**	**0.00**	**0.23**	**0.00**	**0.00**	0.11

4.3.3 Baselines

We compare with three other techniques that focus on extracting maps between given surfaces: (i) BIM [114], (ii) ZoomOut [166], and (iii) Smooth-shells [103]. We also include (iv) ICP, which uses the closest points as correspondence, as a straw-man approach that performs well in case of negligible pose variation. Results are presented in Table 4.1, and selection of the pairs shown in Figure 4.7.

We cast ICP as nearest neighbour search after rigid alignment. Specifically, we use our pipeline first to align each shape pair and then compute nearest neighbour correspondences for each point on the source to the target mesh. This approach may perform well for shapes in similar poses with low isometric deformations.

Additionally, we compare qualitatively to Enigma [197] that uses genetic algorithms along with a combinatorial search to find a set of good sparse correspondence, which are then interpolated to a dense low-distortion map. While this method produces smoother and more semantic maps than other baselines, it still suffers from large and uneven distortions, see Figure 4.8.

4.3.4 Quantitative Evaluation

We report quantitative errors using the metrics discussed earlier. In particular, for accuracy, we follow the standard practice and measure the mean geodesic distance to ground truth correspondence on a unit-area mesh.

Although not guaranteed by construction, we empirically found that ours consistently offers more bijective and continuous maps, see Table 4.1 "*Bij*" and "*Inv*", while others can fail to perfectly achieve these properties in both isometric and non-isometric cases. Our technique shows comparable quality in the maps in non-

Figure 4.7: Comparisons. Left-to-right: Source model, results using BIM [114], ICP, Smooth-shells [103], ZoomOut [166], and Ours. Although geometric methods produce good maps, they often yield discontinuous maps, e.g., see the wings of planes. Ours explicitly encourages continuity and bijectivity. Coloured landmarks and paths show *automatically* selected cones and cuts by our method. Note that our maps are continuous across the cut seams. No explicit energy term is used to encourage isometric maps.

isometric cases (SHREC07) compared to state-of-the-art methods, Table 4.1 "*Acc*", while it performs worse in isometric cases (FAUST and SHREC19). In general, our method suffers in these cases as it does not exploit geometric cues and does not have an explicit isometric energy term, thus producing less accurate maps than competing methods.

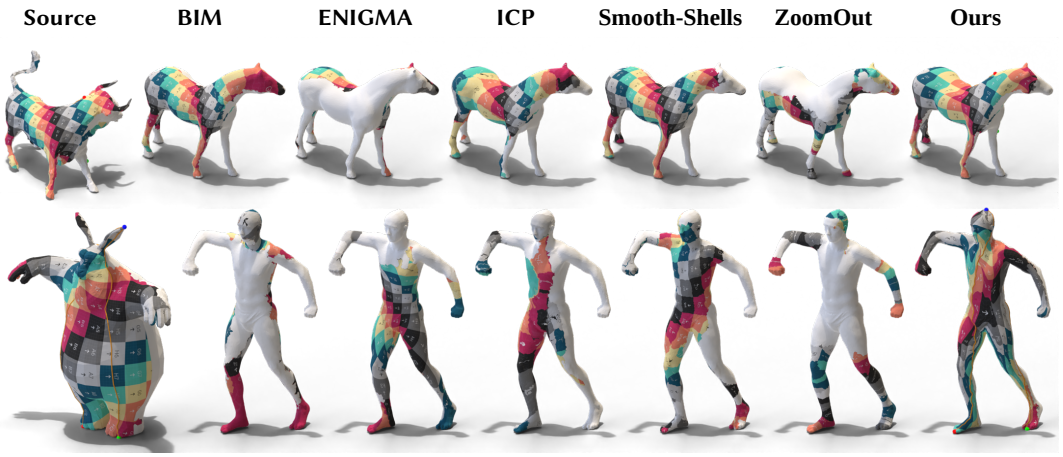


Figure 4.8: Qualitative comparison. ENIGMA [197] fails to produce correct mappings, in cases of extreme deformations. Similarly, other state-of-the-art methods may lack bijectivity or correct correspondence. Ours can better handle these cases, see Table 4.1 for quantitative comparison. Coloured landmarks and paths show *automatically* selected cones and cuts by our method.

4.3.5 Qualitative Evaluation

Figure 4.9 shows Neural Semantic maps extracted using our fully automatic approach. The produced maps accurately match semantic features despite the fuzzy aggregated correspondences being erroneous and confused by symmetries (*e.g.*, mapping incorrect limbs). Ours also work well across dissimilar shapes. These non-isometric cases require introducing significant local stretching to preserve semantic correspondence. The extracted maps still exhibit low isometric distortion, where possible, while adhering to semantically meaningful correspondences. Yet, artefacts may arise (see Armadillo's leg in Figure 4.7) when the smoothing energy is not sufficient to balance the noisiness of matches. State-of-the-art methods, such as ENIGMA [197] or Smooth-shells [103], suffer from self-symmetry ambiguities, *e.g.*, see `bull-horse` in Figure 4.8.

4.3.6 Ablation

4.3.6.1 On DinoV2 features

As aforementioned, we deem a match if the cosine similarity S_{ij} between patch features – $\lambda_i^{\mathbf{A}}$ and $\lambda_i^{\mathbf{B}}$ – is the highest. While this is a common similarity measure, it is important to acknowledge its inherent limitations. Specifically, one notable

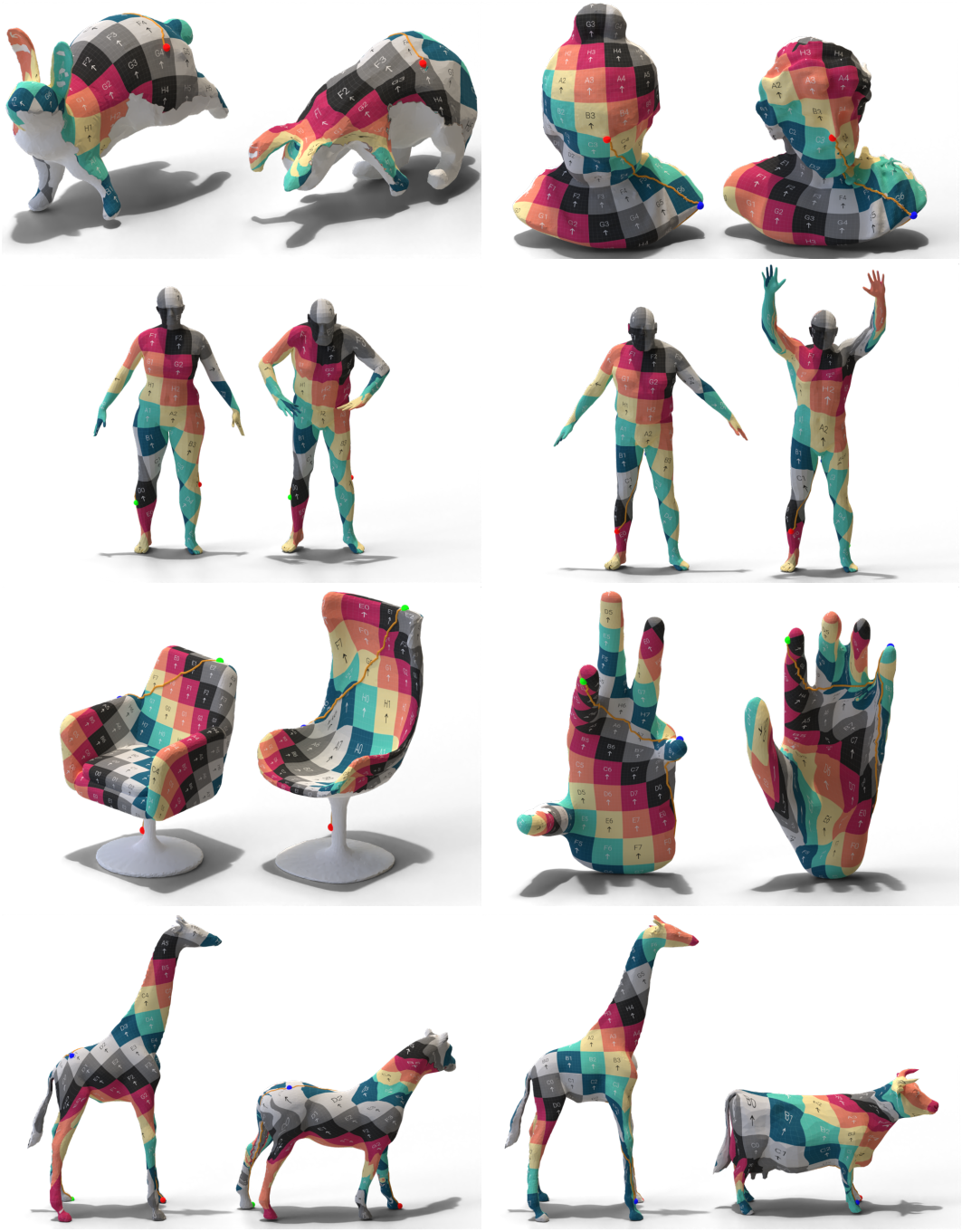


Figure 4.9: Results. Automatic maps extracted by the optimization on various surface pairs using aggregated fuzzy correspondences. Coloured landmarks and paths show *automatically* selected cones and cuts by our method. The rabbit, hands, humans, and heads examples represent near isometric pairs with pose variations; the chairs, giraffe-horse, giraffe-cow examples produce non-isometric mappings with spatially varying distortions. Note the semantic nature of the extracted maps. No explicit energy term was used to encourage the maps to be isometric.

challenge is that similarity scores derived from different images may not be directly comparable. For example, for two matches with scores 0.9 and 0.8, the former match pair is not necessarily better than the latter. In essence, features extracted from one view may be extremely dissimilar to those extracted from another view, even for the same shape. This arises from the inherent variation in image structure across different views and how features are generated from them. This inherent vari-

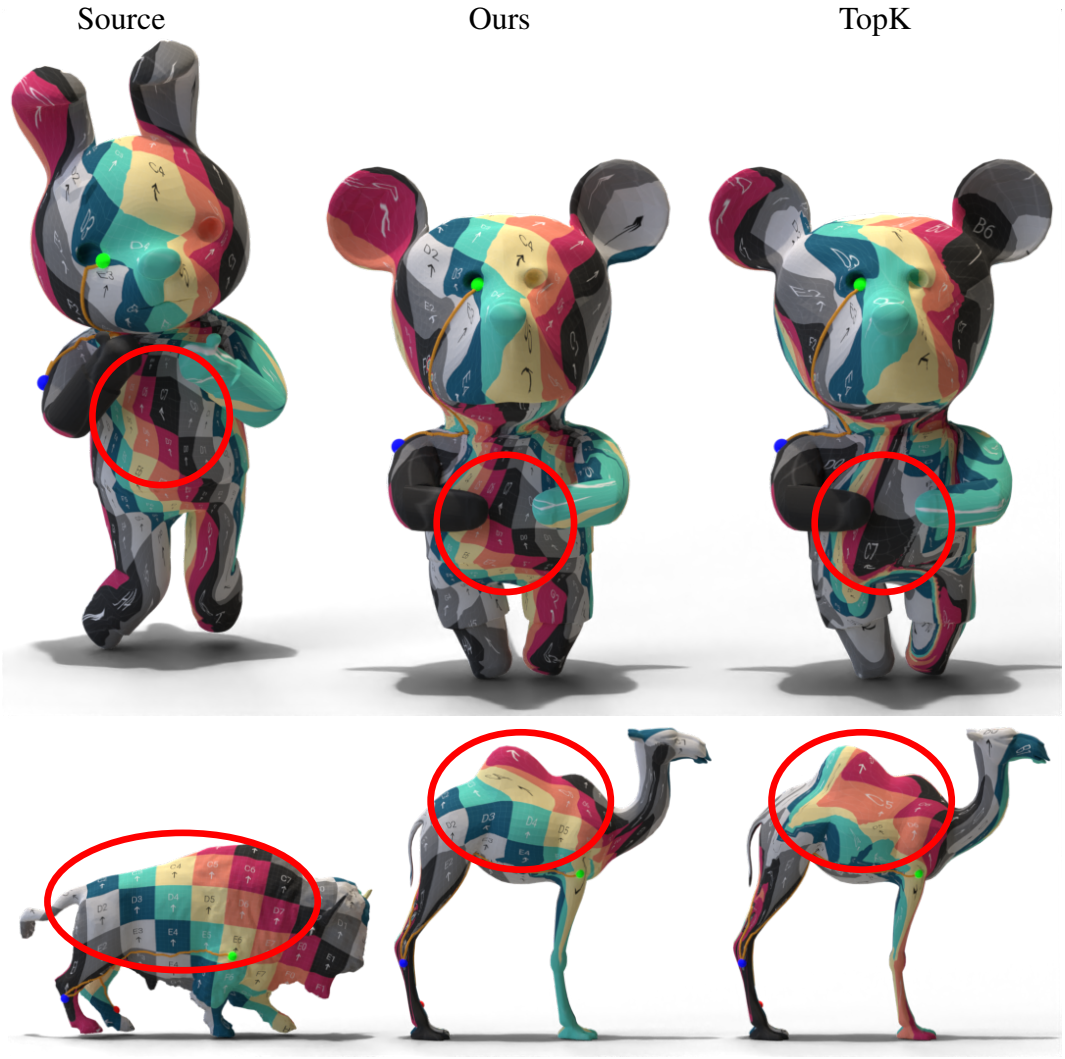


Figure 4.10: Ablation on similarity scores. Left: source mesh. **Middle**: map optimized using all correspondences. **Right**: a map optimized with the top $k = 100$ correspondences based on the similarity score. The map optimized with matches with the highest similarity score matches shows several incorrectness, highlighted with a red circle. This is the result of several incorrect matches which bias the map towards an incorrect energy minimum. Differently, using all correspondences prevents this behaviour, as the optimization process automatically filters out wrong matches.

ability hinders consistency in cross-image feature comparisons. Consequently, the process of aggregating features across different views can potentially yield unexpected outcomes, leading to either incorrect matching or highly inaccurate results.

Experimentally, sampling the top $k = 100$ correspondences based on the similarity produced far worse results than uniform sampling or uniform weighting, see Figure 4.10 for qualitative comparison. In both cases, we optimize maps following the proposed algorithm: *Ours* uses all matches, while *TopK* is limited to $k = 100$ correspondences with the highest similarity score. Visibly, some of these correspondences are incorrect and bias the map towards incorrect minima, thus their similarity score is not representative of their quality. Indeed, the use of all matches prevents the map from falling into such a degenerate solution, as the majority of matches are reasonably correct.

4.3.6.2 Tuning Dino-Vit Matches

We ablate the quality of matches based on Dino-ViT's degrees of freedom – layer features – in different contexts: pose variation, presence of texture, lights, and misalignment. We conduct our analysis on three distinct datasets: **FAUST** [186], **3DBiCar** [202], and **SHREC15** [203] each with **dense** or **sparse** ground truth.

We select 12 shape pairs, 4 for each dataset, to ablate texture and misalign concerning the choice of Dino-ViT feature layer, as discussed in [135]. Similarly, we assess the effect of pose variation for the same model with a single instance of

Table 4.2: Dino-ViT ablation: DinoV2 [21] works better than its predecessor [130], with no significant difference between features from L9 and L11. The use of coloured lights (rows DinoV1 and DinoV2) offers better visual cues to extract matches than white lights. Although counter-intuitive, the use of *simple* texture reduces the visual cues available to Dino ViT.

	FAUST		SHREC15		3DBiCar	
Layer	9	11	9	11	9	11
DinoV1	0.10	0.12	0.32	0.32	0.36	0.49
DinoV2	0.11	0.11	0.24	0.24	0.33	0.33
w/ white lights (V1)	0.20	0.18	0.27	0.35	0.38	0.38
w/ white lights (V2)	0.11	0.11	0.24	0.24	0.30	0.31
w/ texture (V1)	-	-	-	-	0.26	0.26
w/ texture (V2)	-	-	-	-	0.29	0.29

Table 4.3: Dino-ViT pose ablation: DinoV2 [21] matches are significantly more accurate than DinoV1 [130] in case of pose variation, with no significant difference between features from L9 and L11.

	FAUST		SCAPE		TOSCA	
Layer	9	11	9	11	9	11
DinoV1	0.16	0.16	0.38	0.40	0.27	0.29
DinoV2	0.09	0.09	0.18	0.18	0.27	0.25

FAUST, SCAPE, and TOSCA mapped onto all the other provided poses. We report the quantitative results in Table 4.2 and Table 4.3 and show shape pairs examples and qualitative optimization results in Figure 4.12.

We assess the quality of the aggregated matches in terms of the normalized average geodesic distance [114]. We aggregate the fuzzy matches, thus, obtaining a face-wise map M from one mesh onto the other. Finally, the geodesic distance is computed on the target mesh between the centroid of the mapped face to the centroid of the ground truth target face.

In general, DinoV2 [21] outperforms its predecessor V1 [130] (Dino-ViT), offering more accurate and robust matches. The depth at which features are extracted (9 vs 11) does not impact the matches of DinoV2, while it plays a significant role for Dino-ViT, as discussed in [135]. The presence of texture is beneficial to Dino-ViT, while it only offers a minor improvement for DinoV2. This is reassuring as our

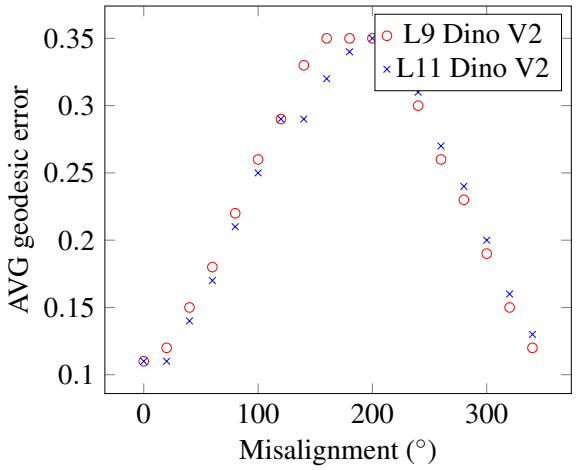


Figure 4.11: Robustness to misalignment: the quality of matches depends on the quality of alignment. In the case of severe misalignment (60° or more), we observe poor correspondence.

method can only assume access to untextured models. The choice of coloured lights offers additional shading and visual features for Dino-ViT, but it is less relevant for DinoV2 as white lights perform equally with the base case.

4.3.6.3 Effect of Initial Alignment

We ablate the effect and robustness to misalignment for match quality, see Figure 4.11. We start from a correct alignment with 12 shape pairs and incrementally misalign one shape – step of 20° around the up axis. We report the quality of matches

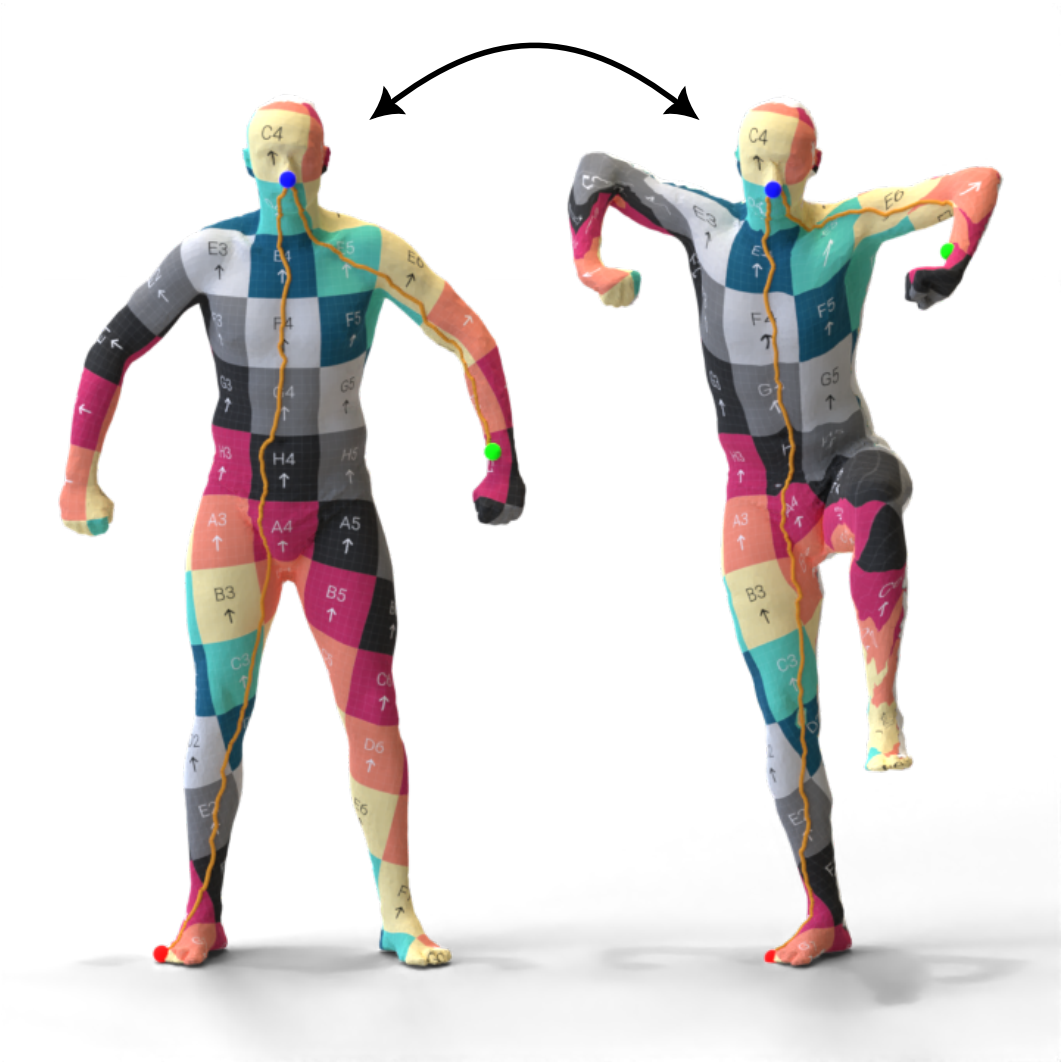


Figure 4.12: Pose variation: we assess the ability of DinoV2 [21] to establish matches between shapes in different poses, as those in the figure. Experimentally, DinoV2 yields correspondences able to guide our pipeline to a proper solution. Coloured landmarks and paths show *automatically* selected cones and cuts by our method.

in terms of geodesic error, i.e., accuracy. The quality sensibly decreases with severe misalignment – more than 40° – reaching a peak with opposite orientation – 180°. We additionally compare the quality of matches for the last two layers of Dino-ViT and show that, for such a case, a deeper level (L11) seems to encode slightly better semantic information than the previous layer (L9).

4.3.6.4 Aggregation

To assess the importance of the map distillation module, we present a qualitative comparison in Figure 4.13 with the method proposed by Surface Maps via Adaptive Triangulations (SMAT) [29], where we replace manual correspondences with automatically extracted ones. As the original approach requires a set of bijective correspondences, we randomly subsample a set of $N = 64$ matches from the automatically extracted ones to ensure consistency, *i.e.*, no vertex appears twice. Then, we optimize for a bijective map that respects these landmarks. We refer to it as DinoV2+SMAT. Note SMAT [29] optimize for isometric energy (Dirichlet), while we optimize only for smoothness, see Equation 4.9.

As SMAT does not account for inaccurate nor imprecise correspondences, it is unable to filter out wrong correspondences. In our observations, optimizing a map with the original hyperparameters leads to visible inversions. This issue arises from SMAT's attempt to preserve all landmarks, resulting in maps with extreme stretches, a phenomenon intensified by the discrete nature of meshes. Adaptive remeshing struggles to handle these extreme stretches effectively, leading to visibly distorted maps. To mitigate this effect, we trade landmark precision for map continuity and quality. As shown in Figure 4.13, although both maps appear continuous, [29] is unable to filter out inaccurate correspondences and yield a reasonable map. Note that this experiment mainly assesses the importance of our correspondence distillation step, and DinoV2+SMAT is not the mode SMAT was originally designed for. Ours, without any explicit isometric or conformal energy term, still can produce smooth and semantics-respecting bijective maps.

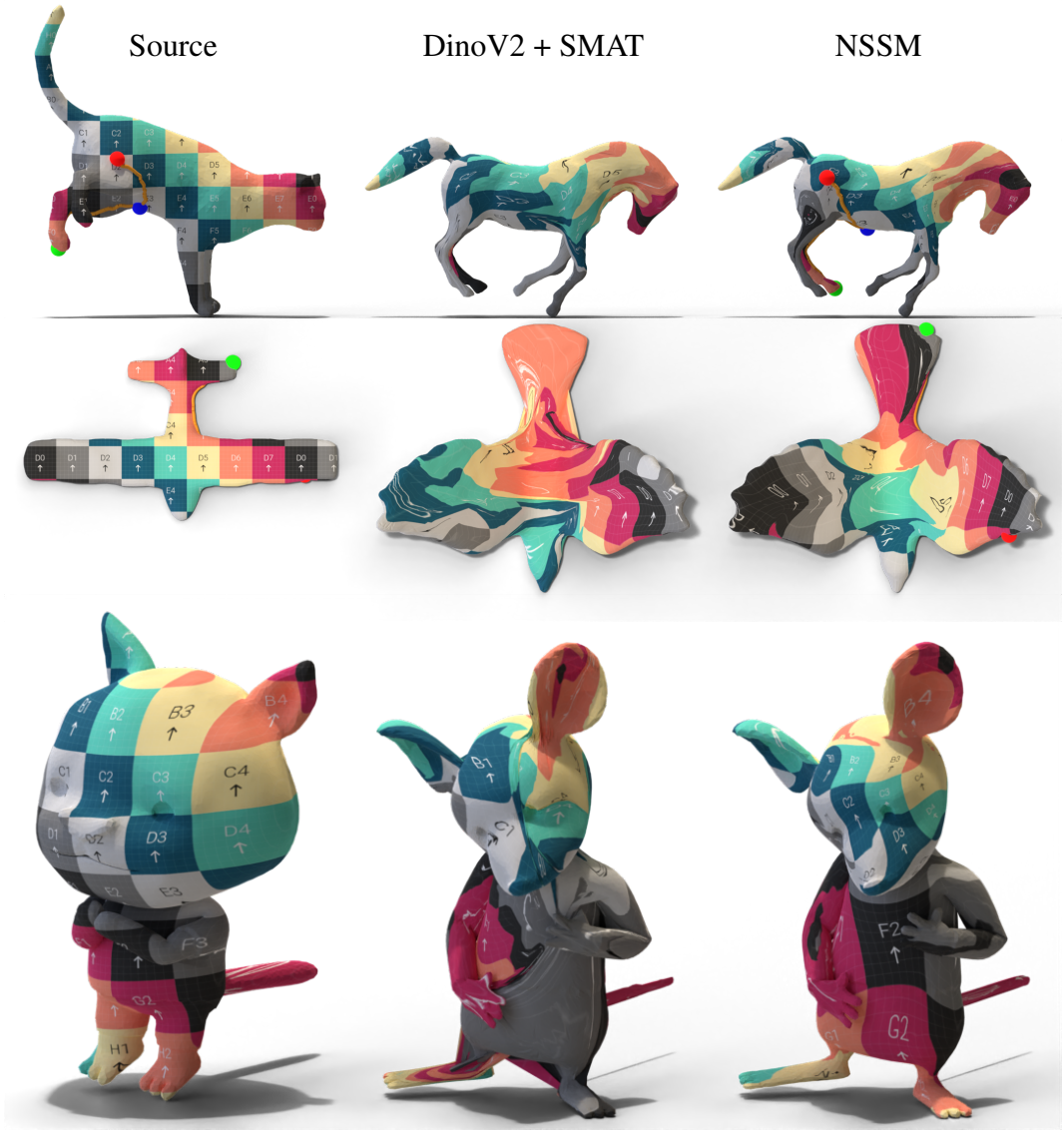


Figure 4.13: Qualitative comparison. Surface Maps via Adaptive Triangulations (SMAT) [29] optimize for bijective and continuous maps, relying on manual annotations. We pair it with DinoV2 – DinoV2+SMAT – by replacing these manual annotations with $k = 64$ automatically extracted ones, $\{\phi^i\}$ with $i = 1 : k$, then we optimize the inter-surface map to construct an automatic inter-surface map. While DinoV2+SMAT attempts to satisfy all matches together with bijectivity, ours automatically filters out incorrect matches, yielding a more continuous and semantically correct map.

4.3.6.5 Handling Noise and Holes

Raw scans present noise or holes, thus inhibiting the applicability of our method since it assumes watertight genus zero meshes. Intuitively the presence of large holes, and missing limbs such as arms, may severely mislead DinoV2 and thus our

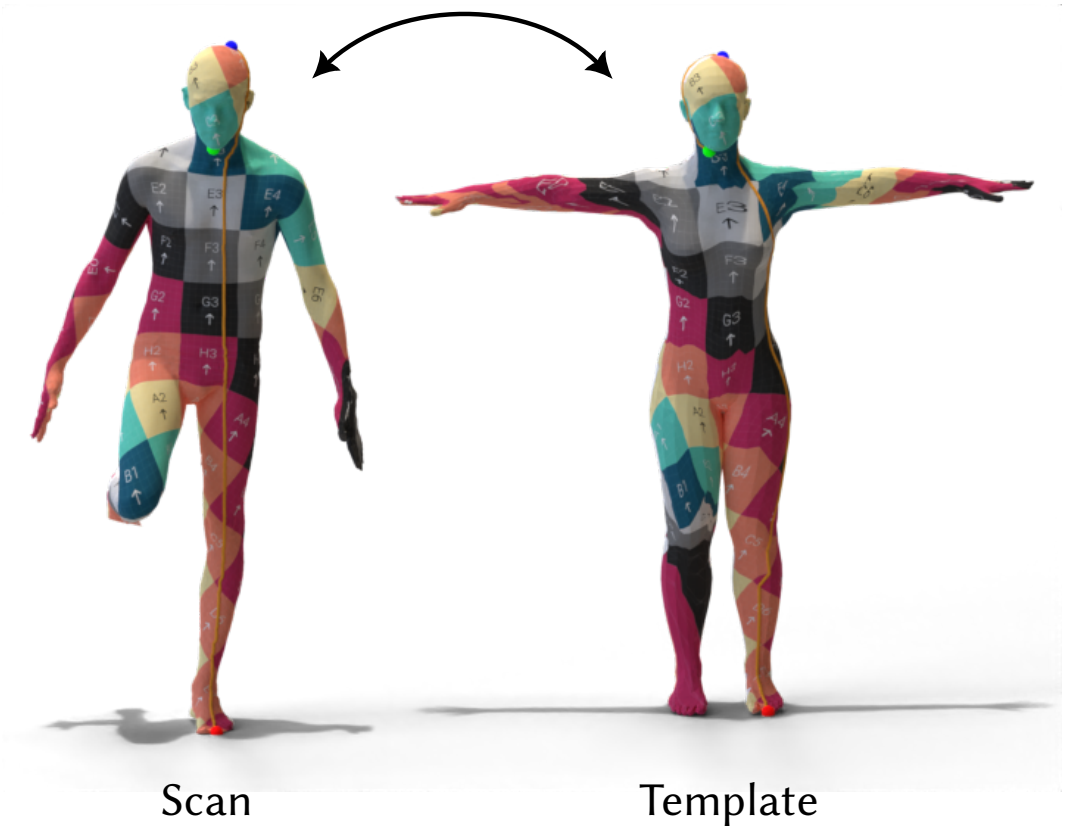


Figure 4.14: Scan to SMPL: we first close holes in the raw scan (left) with Meshlab[205], then we map it onto the template SMPL model[204] and mask out the surfaced introduced to fill holes. Coloured landmarks and paths show *automatically* selected cones and cuts by our method.

pipeline. On the other hand, small holes can be dealt with by applying a simple hole-filling approach. In Figure 4.14, we use our method to map a raw scan to the SMPL template [204]. We prefill small holes with Meshlab [205] and then apply our pipeline.

4.4 Implementation details

We realize the neural map (h) as a 4-layer residual MLP of 128 neurons each, while neural surfaces (f_*) are always 8 layers residual MLP with 256 neurons. We sample 128 correspondences, 1024 points to enforce injectivity and smoothness, and 128 points for the boundary in each optimization iteration.

4.4.0.1 Rendering Details

In all cases, we render images of the same size, i.e., 1344×1344 with Mitsuba [196] using $spp = 150$ and a path integrator. When extracting semantic matches, we limit to rotations around the up-axis (y) – 20 steps between $[0, 2\pi)$ – and forward-axis (z) – 10 steps between $[\frac{-\pi}{2}, \frac{\pi}{2})$ – obtaining 200 images for each shape. Similarly, to align shapes, we rotate around the up-axis – 12 steps – with fixed increments. To uplift 2D pixels to 3D for the matches, we use ray-triangle intersection. On average, we get 328 matches per view, totalling 65k matches across the 200 views.

4.5 Limitations

The approach discussed in this chapter has several limitations. For starter, it still requires to parametrize surfaces to a plane – square in this case. Thus, thin shapes, *e.g.*, lamps, remain challenging as they require complex cuts.

Similarly to NSM Chapter 3, a key limitation is the long running time. The map optimization takes on average 1.5 hours, converting the meshes into their neural representation which requires about 1 hour, and extracting all Dino-ViT matches takes 21 minutes. We plan to investigate approaches, such as Meta-Learning [206, 207], better sampling [208], and better caching, to speed up this process.

Finally, the presence of self-occlusion in shape pairs prevents Dino-ViT from correctly mapping regions across shapes, thus consistently making mistakes. We believe incorporating other priors, or an advanced rendering pipeline (e.g., layered rendering) may help cope with this issue.

4.5.1 Improvement over Neural Surface Maps

Compared to the framework described in Chapter 3, here, we relax the constraint over the cut: only 3 points are required to be in correspondence, while the rest of the boundary can move. Furthermore, as we rely on seamless maps, the boundary and domain terms in Equation 3.8 are not needed, thus easing the optimization.

Thanks to the use of DinoV2 [21] to extract noisy matches, manual labels are no longer required. Finally, we can handle more non-isometric shapes, since we replaced the isometric prior, Equation 3.2, with a softer, smoothing one, Equation

4.9.

4.6 Conclusion

In this chapter, we have presented a method that produces a semantic surface-to-surface map guided by visual semantic priors, by computing it from a set of candidate non-injective and discontinuous partial maps extracted by matchings over renderings of untextured $3D$ surfaces. This method has many potential practical applications, ranging from matching scans of human faces and bodies to clothes, anatomical scans, and archaeological findings. These depend on the quality of the matchings achieved over the renderings of objects from these categories, which we aim to explore.

Future work

Even after introducing seamlessness in the NSM framework, we require surfaces to be cut which makes our method more prone to error. We aim to improve the existing pipeline to avoid cutting altogether by replacing the 2D disks with 3D spheres [209], as successfully used in [29]. Our optimization cannot guarantee achieving a global optimum nor that the global optimum defines the ”most-meaningful” semantic map, and we mark extending our method to directly *learn* to produce maps from a dataset as an important future direction. Our method can create such a dataset, augmented with manual input to score the goodness of any extracted semantic map. We believe this work is only a step in producing semantic-driven maps. Candidate fuzzy maps extracted from other means can be considered. For instance, methods to predict fuzzy geometric correspondences directly over 3D surfaces trained for specific tasks can alternatively produce fuzzy maps and can be used in conjunction with semantic and/or visual cues.

Chapter 5

Neural Convolutional Surfaces

5.1 Introduction

Triangle meshes have been the most popular representation across much of geometry processing since its early stages, however research has been devoted to devising novel representations of geometry to circumvent many of the shortcoming of triangular meshes. Lately, the rising prominence of deep learning has lead researchers to investigate ways to represent shapes via neural networks. While the immediate use of neural networks in this context is to represent entire shape spaces by using the same set of weights to decode any shape from a shared latent space, other methods use a shape-specific set of weights to represent a specific instance. This approach captures geometric detail efficiently and accurately and creates outputs that are on par with existing $3D$ models, while holding novel properties not attainable with surface meshes, such as differentiability. These neural representations for shape instances were demonstrated to be useful in geometry processing applications such as efficient rendering [4], level of details [3], surface parametrization, and inter-surface mapping as we discussed in Chapter 3.

The choice of the shape representation, and of the neural network's architecture, plays a critical role in how efficiently the capacity of the network is utilized. Existing representations usually use MLPs to model the shape as a function that maps points either from a $2D$ atlas to the surface Chapter 3 or points in a $3D$ volume to an implicit function such as a distance field [7]. The disadvantage of these

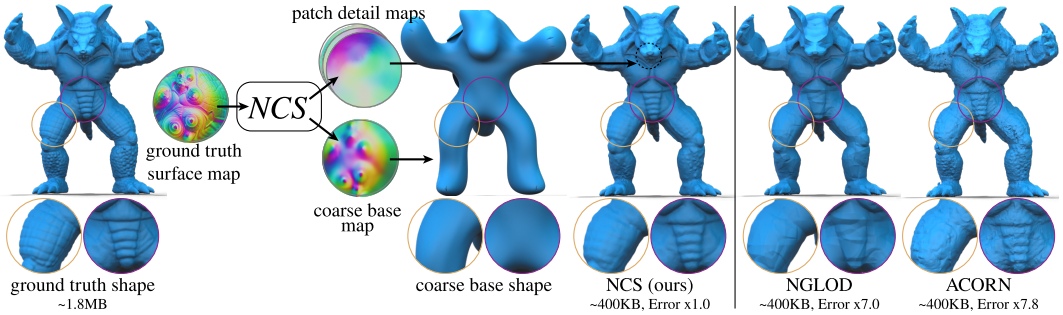


Figure 5.1: Neural Convolutional Surfaces (NCS) can faithfully represent a given ground-truth shape while disentangling coarse geometry from fine details, leading to a highly accurate representation of the shape. Compared to other state-of-the-art methods neuralLOD [4] and Acorn [3], NCS achieves significantly more accurate results for the same memory footprint.

architectures is that they entangle geometric details and overall shape structure, and do not have a natural mechanism to reuse the network weights to represent repeating local details, as Convolution Neural Networks (CNNs) achieve on images. Some methods indeed opt to use $2D$ images to represent geometry [1], however those exhibit finite resolution and hence cannot model surfaces with details in sub-pixel resolution. Alternatively, instead of a single global MLP, some prior techniques leverage repetitions by breaking the shape into smaller $3D$ voxels, each represented by an SDF function [3], however, these representations do not account for the fact that surface details are usually aligned with the surface, and thus, are less effective at representing local geometric textures that flow with the shape.

In this chapter, we set to define a representation that achieves separation of local geometric details (”texture”) from the global coarse geometry of the model, and thus leads to the reuse of network weights for repeating patterns that change their orientation with the surface. We achieve this by extending the atlas-based representation discussed in Chapter 3. Here, we encode a surface as combination of a *coarse surface*, defining the general, coarse structure of the shape, represented via an MLP, along with an associated fine *detail map*, which adds geometric texture on top, represented via a CNN, which defines a continuous map of offsets. The geometric details are added to the coarse geometry either along its normal directions, or as general displacement vectors. Since the local displacement details are expressed

with convolutional kernels, they can effectively be reused across similar regions of the surface. We call this hybrid representation *Neural Convolutional Surfaces*.

This architecture enables the network to disentangle the fine CNN representation from the coarse MLP representation, in a completely *unsupervised* manner, i.e., without the need to supervise the split explicitly during fitting. We show that the inductive bias in our designed architecture leads to automatic separation of the shapes into coarse base shapes and reusable convolutional details, see Figure 5.1.

We evaluate NCS on a range of complex surfaces and explore the associated trade-off between representation quality and model complexity. We compare against a set of state-of-the-art alternatives (e.g., NeuralLod [4], ACORN [3], Neural Surface Maps Chapter 3) and demonstrate the achieves better accuracy at a fraction of the model-complexity – between 1% to 10% parameters. Additionally, we demonstrate that the convolutional aspect of the representation makes it *interpretable*, leading to applications including detail modification within individual shapes and details transfer across different models – see Figure 5.8.

5.2 Neural Convolutional Surfaces

Our goal is to represent a $3D$ surface $S \subset \mathbb{R}^3$ with a neural network g_θ, so that the parameters θ compactly encode the surface. We assume to have computed a bijective parametrization of the surface into the unit circle in $2D$ (we use SLIM [23] in all our experiments). We consider this parametrization as a map from the unit disk to the surface $D : \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\| \leq 1\} \rightarrow S$, mapping a $2D$ point q to a $3D$ point $p \in S$ on the surface. Following Chapter 3, we fit the network g_θ to approximate this function: $g_\theta(q) \approx s(q)$.

We propose *Neural Convolutional Surfaces* (NCS) as an accurate and compact neural approach to model s. The NCS g_θ consists of two modules: (i) a coarse module, g^c_ϕ, based on a standard MLP-based model, which aims to approximate the coarse shape of the surface; and (ii) a fine module, g^d_ψ, which adds detailed displacements to the coarse surface. Internally, this fine module is comprised of a CNN component generating a grid of codes, which are interpolated and fed to an

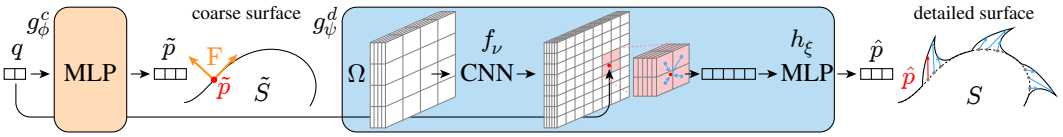


Figure 5.2: Overview of Neural Convolutional Surfaces. Surfaces S are represented by two models, a coarse model g^c_ϕ that encodes a coarse version $\tilde{S}$ of the surface and allows computing a local reference frame $\mathbf{F}$, and fine model g^d_ψ that encodes geometric detail as offsets $\hat{p}$ from the coarse surface, in coordinates of the local reference frame.

MLP in order to get the final fine offset vector. These two modules are then added to get the final map from $2D$ to $3D$,

$$g_\theta(q) := g^c_\phi(q) \; + \; \mathbf{F}_{g^c_\phi}(q) \; g^d_\psi(q), \tag{5.1}$$

where $\mathbf{F}_{g^c_\phi}$ is a rotation to the local reference frame at a coarse surface location and $\theta = (\phi, \psi)$. Please refer to Figure 5.2 for an overview. These two models are trained *jointly*, with only the target mapping s as supervision. Intuitively, the coarse-fine separation allows the fine model g^d_ψ to expend capacity only on the high-frequency geometric texture of the surface. Repeating structures in this geometric texture can be modelled efficiently by the shared weights of the convolution kernels.

We describe the coarse model next in Section 5.2.1, the fine model in Section 5.2.2, the local reference frame in Section 5.2.3, and the overall training setup in Section 5.2.4.

5.2.1 Coarse Model

The coarse model represents a smooth, coarse version of the ground truth surface S, which serves as a basis, by providing a well defined local coordinate frame at any surface point, for applying the detailed geometric texture predicted by the fine model.

Our coarse model approximates a coarse version of s with a low-capacity MLP, which takes as input a $2D$ coordinate q in the unit disk and outputs the corresponding point $\tilde{p}$ on the coarse surface: $\tilde{p} = g^c_\phi(q)$. The limited capacity of the MLP ensures that only a coarse approximation of the surface is modelled, while the MLP's archi-

tecture enforces smoothness of the coarse surface, thereby delegating reconstruction of sharp, fine, and repeating structures to the fine model. Note that we do *not* use intermediate supervision for the coarse surface $\tilde{p}$.

5.2.2 Fine Model

The fine model represents a detailed geometric texture that is applied to the coarse surface g^c_ϕ. The fine model is designed to first generate a high-resolution $2D$ grid of codes, and then interpolate the codes and map the interpolated code to $3D$ displacement vectors via a small MLP. This is facilitated by three components: (i) we keep a low-resolution input grid of learned features (i.e., a low-resolution $2D$ image with multiple channels) $\Omega \in \mathbb{R}^{D_0 \times H_0 \times W_0}$; (ii) a CNN f_ν transforms and up-samples(2x per layer) the feature map into a high-resolution $2D$ grid of codes; and (iii) for a given $2D$ query point q that falls into a grid cell, the 4 grid codes at that cell's corners are interpolated, and a small 2-layer MLP h_ξ finally maps the interpolated code into a local displacement vector:

$$g^d_\psi(q) = h_\xi\left(f_\nu(\Omega)|_q\right), \tag{5.2}$$

where $X|_q$ denotes bilinear interpolation of the image X at q (assuming pixel coordinates in $[-1,1]^2$) and $\psi = (\Omega, \nu, \xi)$. Intuitively, the feature map Ω stores coarse information about the geometric surface details that is refined by the CNN f_ν, introducing learned priors stored in the shared CNN kernels. The interpolation for a given sample q is performed in feature space as opposed to $3D$ space, and followed by a small MLP to allow for complex non-linear interpolating surfaces between the pixels of the CNN output. For details of the model architectures, please refer to the supplementary material.

5.2.2.1 Patches

Directly discretizing the full parameter domain of s on a grid has two drawbacks: (i) The resolution of the pixel grid processed by the CNN would need to be very large to accurately model small geometric detail; and (ii) the initial mapping s may exhibit significant area distortion, i.e., there can be a large difference between scale

factors in different regions of the mapping, making a single global resolution inefficient.

To avoid these problems, we split the surface S into small overlapping patches $R_0, \ldots, R_m$ with each having a separate local parametrization $r_i : [-1,1]^2 \to R_i$. Since each patch only covers a small region of the surface, the distortion within each individual mapping is small, and the resolution of the pixel grid in each patch can be lower, without compromising geometric detail. Further, since we assume the fine model is CNN-based, we can learn and reuse the same CNN kernels for each one of our patches without harming our goal of training the fine model to represent repeated geometry, now simply split into different patches.

5.2.2.2 Patch-based model

Once we decompose the input into multiple patches, Equation 5.2 can be generalized as:

$$g^d_\psi(q) = \frac{1}{\sum_i w_i(q)} \sum_i w_i(q)\; h_\xi\big(f_\nu(\Omega_i)|_{l_i(q)}\big), \tag{5.3}$$
$$\text{with } w_i(q) = \max\big(0, -d^b_i\big(l_i(q)\big)\big),$$

where $l_i(q)$ maps the global parameters q to the local patch parametrization r_i, and the contribution of overlapping patches to a point q is weighed as a function of the signed distance d^b_i from the boundary of the patch to $l_i(q)$ in the parameter domain of the patch. We train the same MLP and CNN (with shared weights ν, ξ) over all patches, thereby encouraging the CNN to reuse filters across patches; the only different parameter between different patches is the coarse input feature grid map Ω_i that is assigned to each patch.

5.2.3 Local Reference Frame

The output of the fine model $g^d_\psi(q)$ is a displacement vector $\hat{p}$ of the coarse surface at $\tilde{p} = g^c_\phi(q)$. Naively adding $\tilde{p} + \hat{p}$ would render the displacements sensitive to the local orientation of the coarse surface. Hence, to encourage consistency between the displacements, we define them in a local coordinate frame $\mathbf{F}_{g^c_\phi}$ aligned to the

tangent space of the coarse surface, measured via the Jacobian of the coarse surface mapping, as:

$$\mathbf{J}^c := [\mathbf{J}^c_u, \mathbf{J}^c_v] = \left[\frac{\partial g^c_\phi}{\partial q_u}, \frac{\partial g^c_\phi}{\partial q_v} \right], \tag{5.4}$$

where q_u and q_v are the two coordinates of the global parametrization The local coordinate frame as a function of q is then defined as:

$$\mathbf{F}_{g^c_\phi} := [n,\ \mathbf{J}^c_u,\ n \wedge \mathbf{J}^c_u] \text{ with } n = \mathbf{J}^c_u \wedge \mathbf{J}^c_v, \tag{5.5}$$

where n returns the normal of the coarse surface and $\wedge$ denotes the cross product. Note that, although not shown in the expressions above, each of the axis vectors are normalized to be of unit length.

5.2.4 Training

As discussed earlier, the coarse and fine modules together define a neural network mapping $2D$ points to $3D$. We fit this combined map to the ground truth surface mapping s via an L2 loss:

$$\mathscr{L}_{\text{joint}} = \int_{Q_S} \| g_\theta(q) - s(q) \|_2^2 \, dq, \tag{5.6}$$

where Q_S is the subregion of the global parameter domain Q that maps to the surface S.

5.3 Experiments

We now detail the various experiments we performed. We compare NCS's reconstruction quality to other methods, move on to ablations, and finish with a additional applications enabled by our representation.

5.3.1 Baselines

We compare NCS against three state-of-the-art methods for neural shape representations: ACORN [3], NGLOD [4], and Neural Surface Maps (NSM) (Chapter 3). We compare the methods on a variety of shapes with different amount and type of geometric details (see Table 5.2). In this section we provide comparisons on

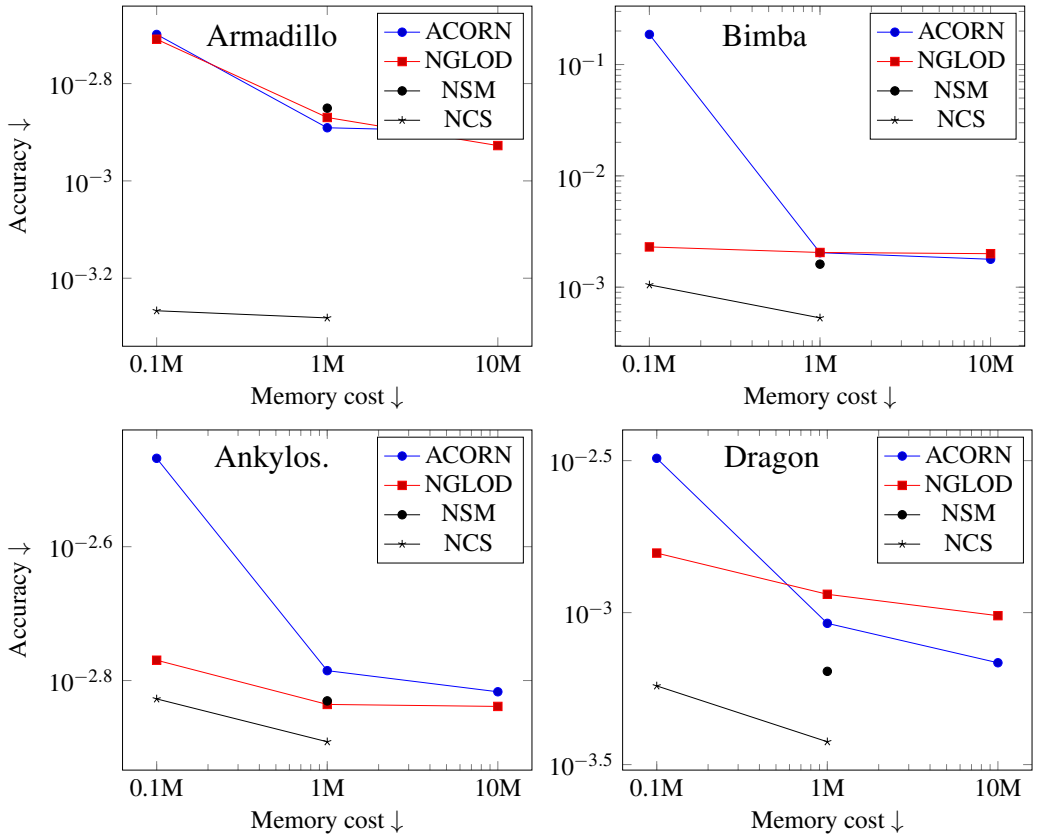


Figure 5.3: Accuracy ↓ versus Memory cost ↓ for different models. NCS achieves better reconstruction quality with significantly lower memory footprint. The values are reported in log scale. Bottom row shows our results. Given our reconstructions, we do not use more than 1M parameters.

different shapes. Note that all models are scaled to a unit sphere.

5.3.2 Metrics

All models are evaluated along two main axes:

- **Accuracy** ↓: measured by the Bidirectional Chamfer distance between output and ground truth surfaces, assess the faithfulness of the representation.

- **Memory cost** ↓: measured as the number of parameters required by a representation, determines the compression of the model. Typically parameters are represented as 32 bit floats, so multiplying by 4 gives the number of bytes.

Table 5.1: Comparison of shape representations with 100K parameters wrt Chamfer distance ↓. Numbers are multiplied by 10^3.

	NGLOD [4]	(Sparse) NGLOD [4]	IDF [2]	NCS
Armadillo	1.95	1.34	1.06	**0.54**
Bimba	2.30	2.07	2.09	**1.04**
Dino	1.70	1.55	2.55	**1.48**
Dragon	1.57	1.12	0.62	**0.57**
Grog	2.06	1.06	**0.81**	1.28
Seahorse	1.26	1.15	-	**0.44**
Elephant	4.06	**2.24**	3.93	2.49
Gargoyl	6.30	-	8.51	**2.29**

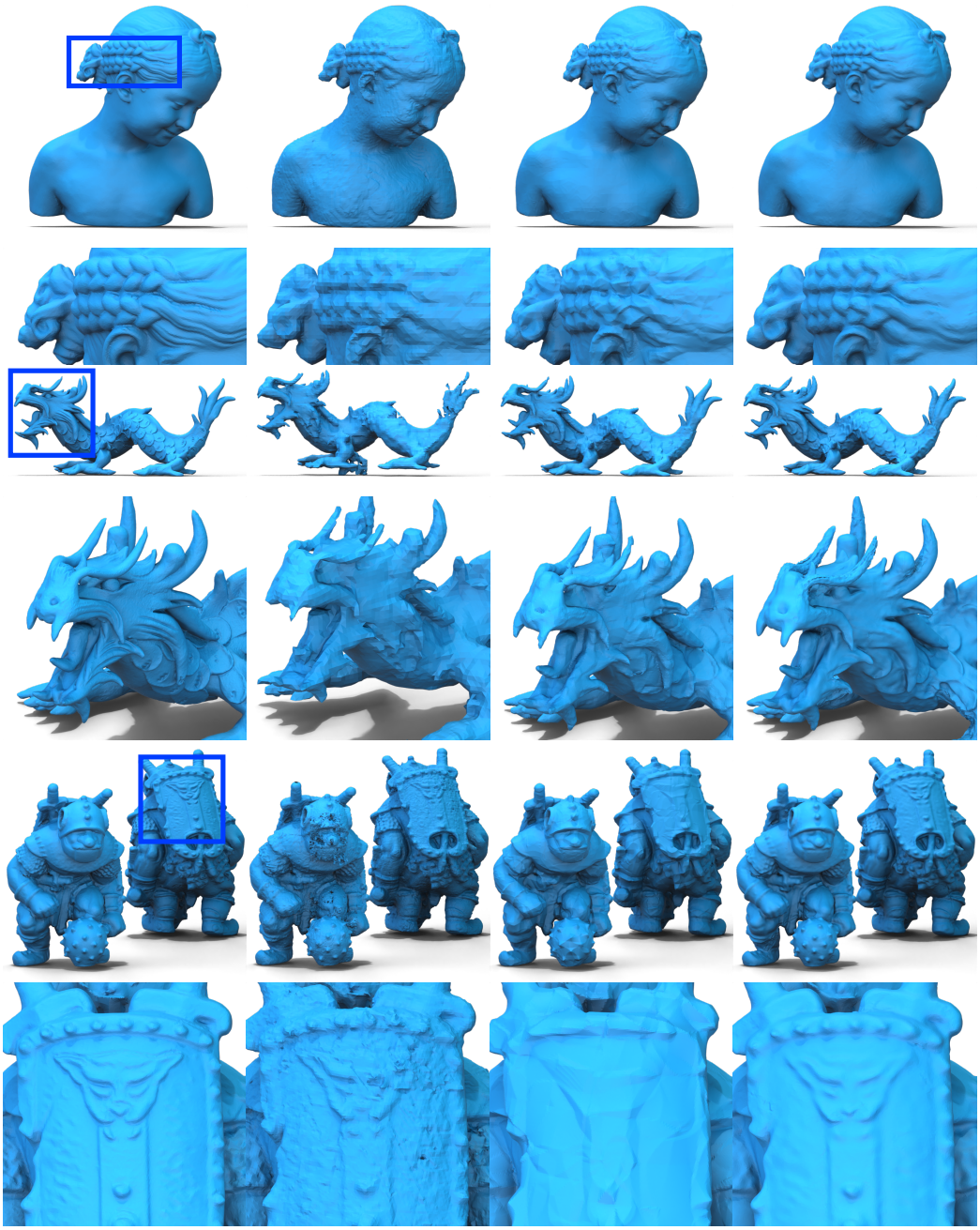
5.3.3 Comparison

As can be seen in Figure 5.3 and Table 5.1, for the same number of parameters, our method achieves significantly higher accuracy than the state of the art. Furthermore, in all cases but one (Lucy), our method's accuracy exceeds all others methods even when using 10 times less parameters. As shown in Figure 5.4, our method preserves detail such as the dragon's scales and Bimba's braids much more accurately than both ACORN and NGLOD. Furthermore, both competing methods exhibit discretization artefacts or noise, while our method provides a smooth, artefact-free surface.

We offer additional comparisons against the concurrent method IDF [2] in Table 5.1. IDF [2] fails to correctly describe the iso-surface of Dino and Elephant. Note that iso-surfaces produced by IDF include excess geometry that wraps around the shape. To offer a numerical comparison, we manually removed the additional excess surface. IDF completely fails to represent the Seahorse and hence we leave a '-' in the table.

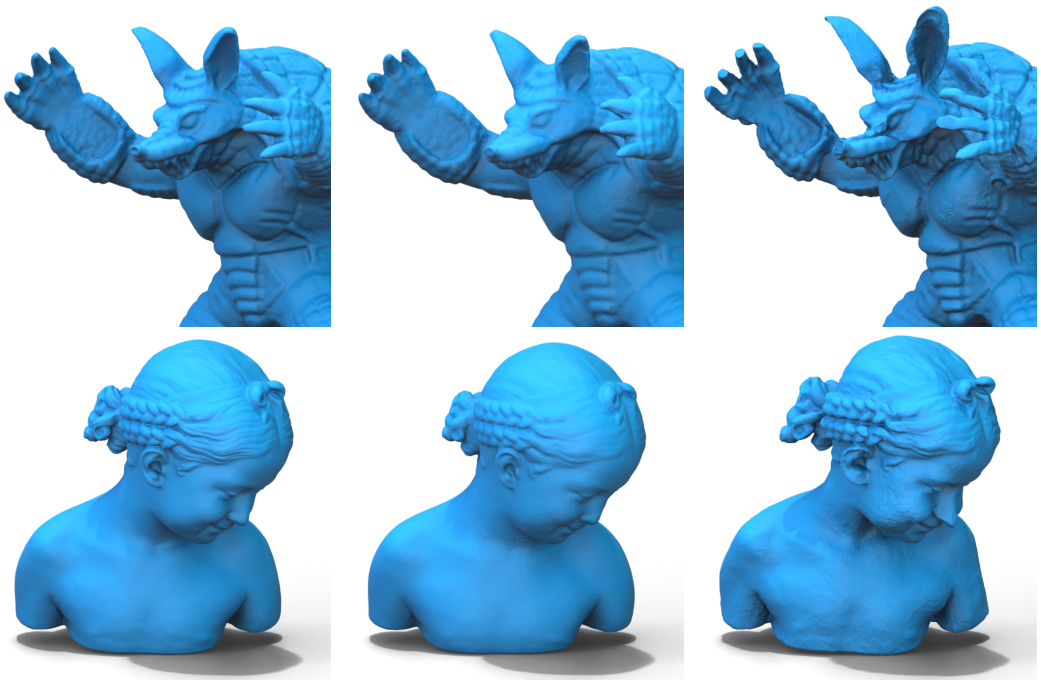
5.3.4 Feature enhancement

Since the CNN encodes local geometric detail, we can edit geometric detail by manipulating the CNN's feature maps. In Figure 5.5 we perform feature enhancement on a few models by scaling up the CNN output, before feeding it into the MLP of the fine model. Similarly, we can perform smoothing, by scaling down the same features.



(a) GT **(b)** ACORN **(c)** NGLOD **(d)** NCS

Figure 5.4: Surface representation: The reconstruction quality of our method, compared with ACORN [3] and NeuralLOD [4] for two models, using the same number of network parameters on each method model size (100K parameters in this example). Our result exhibits higher accuracy and reconstruction of fine details, while not exhibiting artefacts such as artificial edges or aliasing.



(a) Original **(b)** Smoothing **(c)** Sharpening

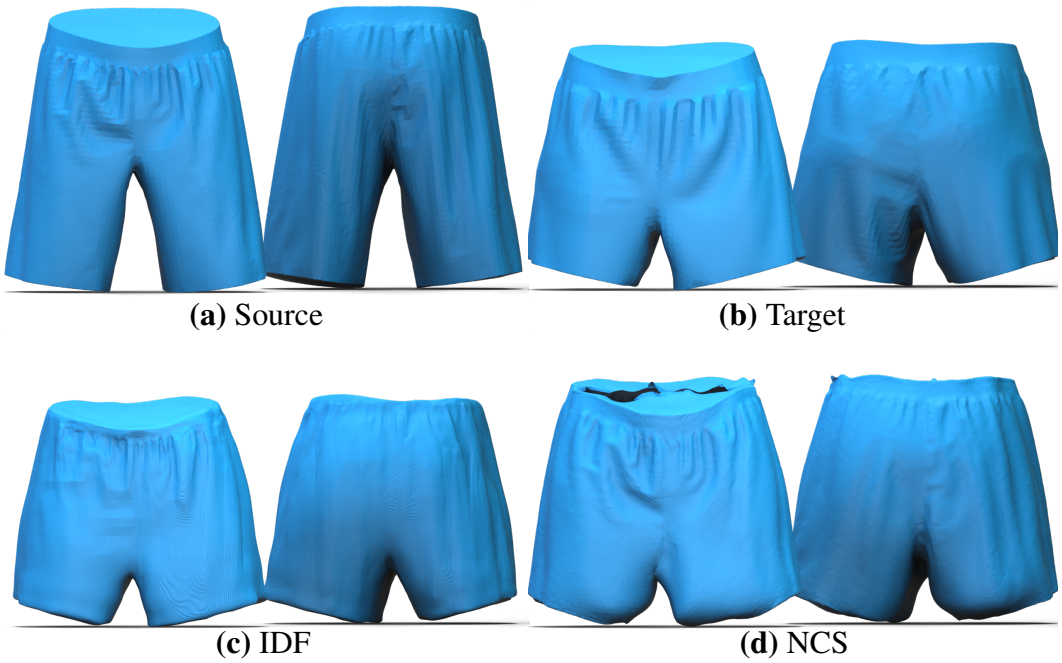
Figure 5.5: Sharpening and smoothing: Our NCS naturally decomposes shapes into coarse shapes and fine details. Boosting or suppressing the fine details and reconstructing the shapes, naturally results in exaggeration or smoothing of surface features.

5.3.5 Detail transfer

Our disentanglement of coarse and fine detail enables us to perform detail transfer, similarly to IDF [2]. Figure 5.6 show results of our method, transferring creases from one model to the other. Similarly to IDF, we achieve this by training one coarse network for the source shape and one for the target shape. The fine network is trained to accurately fit the source shape. We then transfer the details to the target shape with a forward pass using the source global and local parametrization

5.3.6 Ablation

Figure 5.7 evaluates the necessity of various components in our framework: (i) *scalar displacements*: we restrict the fine network to only apply scalar displacements along normal directions, instead of displacement vectors as used in the full model. This significantly hinders the ability of the fine model to add details on top of the coarse model, leading to an oversmoothed result that resembles the coarse



(a) Source **(b)** Target

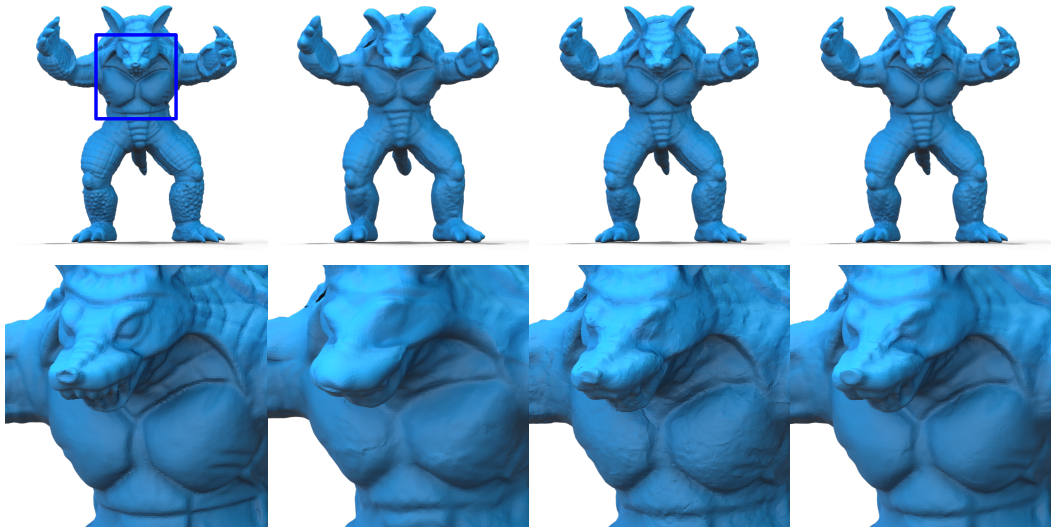
(c) IDF **(d)** NCS

Figure 5.6: Detail transfer: our architecture intrinsically decomposes shapes into coarse base models and associated geometric details, which allows us to transfer learned details from one model to another base shape. In this case, from one pair of shorts (a), to another pair of shorts (b). The fitter coarse model for each of the two pairs is shown at the top. Here we compare our detail transfer results with those of IDF [2]. To transfer details we replace the source coarse model with the target coarse one, and reconstruct the shape. Note, this is possible because the global geometry images, source and target, are aligned. In case of misalignment, an inter-surface map between the coarse models could be computed using, e.g., Section 3.2.

model; (ii) *PCA only*: we remove the coarse model and use only the fine model. For the local reference frame **F** required by the fine model, we pre-compute and store the PCA frame of the ground truth patch. This causes the fine model to spend its capacity on re-creating the coarse geometry, and as a result, artefacts and ripples appear in the reconstruction.

5.3.7 Interpretability of the kernels

The use of a CNN in the fine branch of our network's architecture leads to interpretable kernels, i.e., specific kernels react to specific details of the geometry. In Figure 5.8 we show an example in which we select a region on the model, find features that are activated strongly in that region, and then highlight other areas



(a) GT **(b)** normal **(c)** PCA **(d)** NCS

Figure 5.7: Ablation study: a) the ground truth model; b) adding scalar displacement along the normal direction (to a learned base coarse model) yields smoothed-out results; c) adding displacement vectors to a per-patch canonical coordinate frame (established using patch's PCA axes) yields artefacts and surface ripples; d) our reconstruction is sharp and does not exhibit artefacts.

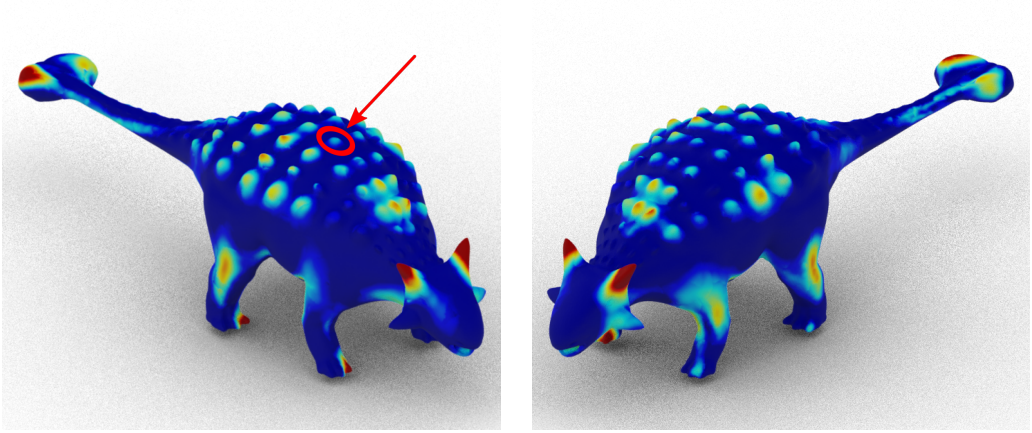
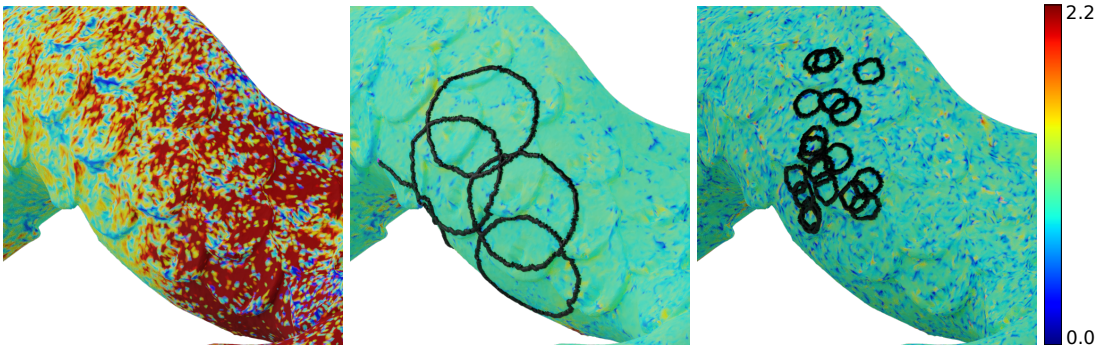


Figure 5.8: Our method yields interpretable convolutional kernels: we select one spike (highlighted) on the dino, identify the CNN features that are strongly active in its region, and then identify other regions where the same features are active. High correlation (hotter colours) implies regions with similar geometric details.

in which those features are activated. As we can see, the features associated with one of the Dino's spikes also affect all other spikes. This shows that our kernels are reused across the model, explaining our network's ability to represent detailed models with a smaller number of network parameters than previous methods. This may also lead to future work where we use the same kernels to a larger collection



(a) 1 patch **(b)** 700 patches **(c)** >2000 patches

Figure 5.9: Effect of number of patches. Representing a shape with a single patch, results in high distortion in the underlying parametrization (red in (a)). Decomposing the base domain into many small patches (c) leads to much lower per-patch distortion. However, this comes at the cost of a much increased memory budget to represent the shape. Medium size patches (b) strikes a balance by reducing the per-patch distortion while still being light in the final memory requirement.

of surface, to learn more specific and robust features.

5.4 Implementation details

Patches are found by randomly sampling patch centres c_i on S and selecting all points within a geodesic radius ρ: $R_i = \{p \mid d^{\text{geo}}(p, c_i) \leq \rho\}$. We use an iterative approach: after creating a patch, all points inside the patch are marked as forbidden for the following patch centres with a probability η, which controls the amount of overlap between patches. In our experiments, we set $\eta = 0.5$ and $\rho = 0.04$ times the maximum extent of S along any coordinate axis. Figure 5.9 shows the effect of the choice of number of patches.

In terms of training performance, we observed that we can achieve better results with a training schedule that starts by warming up the coarse model before slowly ramping up training of the fine model:

$$\mathscr{L} = (1-\lambda)\mathscr{L}_{\text{joint}} + \lambda\mathscr{L}_{\text{reg}} \tag{5.7}$$
$$\text{with } \mathscr{L}_{\text{reg}} = \int_{Q_S} \|g^c_\phi(q) - s(q)\|_2^2 \, dq.$$

We start with $\lambda = 1$ and progressively decrease $\lambda \to 0$ over the warm-up phase, which lasts for 100K iterations. At the same time, we increase the learning rate of

Table 5.2: Evaluation and distribution of the network parameters between the different modules in our architecture, for various models tested in the paper, of size 100K. The latent codes use the majority of parameters, while the other components are self-contained.

			Coarse	Fine			TOT.
	#V	#F	MLP	Code	CNN	MLP	params
Armadillo	172K	346K	13K	93K	6K	467	113K
Bimba	50K	100K	9K	115K	6K	467	130K
Lucy	877K	1753K	13K	96K	4K	435	114K
Dino	26K	51K	4K	120K	6K	467	130K
Dragon	451K	902K	13K	99K	6K	467	119K

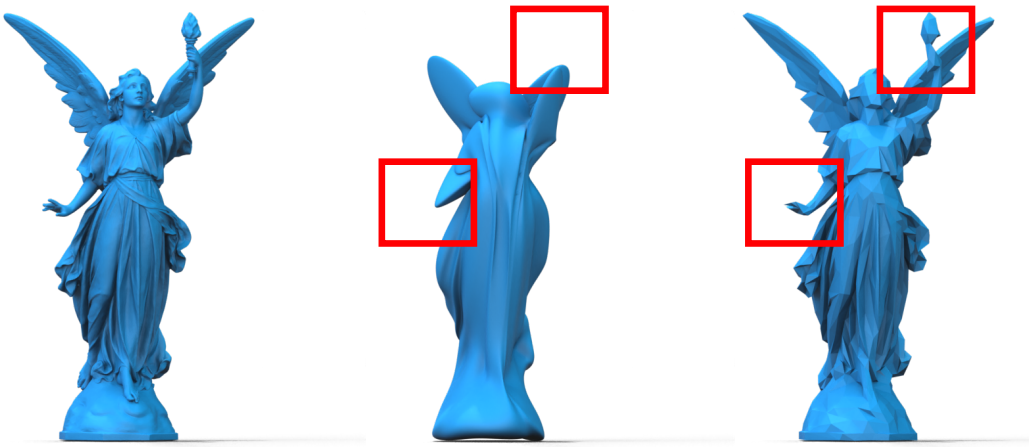
the fine model and decrease the learning rate of the coarse model over the warm-up phase: the learning rate of the coarse model follows a cosine annealing schedule [177], down to a minimum learning rate of 0 at the end of the warm-up phase, while the learning rate of the fine model is set to $1e-4$ minus the coarse learning rate. The total number of iteration varies based on the complexity of the model, *e.g.*, between 800K to 1.4M iterations, using the RMSProp optimizer.

5.5 Limitations

Although the CNN-based architecture leads to significant compression by reusing the kernels across object-centric local coordinate frames, the kernels themselves are still regular, $2D$ Euclidean image kernels, and hence are not rotationally invariant, as they ideally should be to handle geometry. This hinders perfect reuse of kernels across the shape, *e.g.*, in cases of asymmetric features that are reoriented on the shape (rotated on the local tangent space), for example, the scales on the dragon. Furthermore, the kernels cannot be reused to capture local *deformations* of the underlying geometric details. Lastly, we note that in some cases our pipeline, in absence of intermediate supervision, may associate coarse structures as fine, *e.g.*, Lucy's (the angel) torch in Figure 5.10 is reconstructed mainly by the fine module of our networks, and as a result is reduced in size when the details are smoothed.

5.5.1 Improvement over Neural Surface Maps

In this chapter we extended the neural representation discussed in Chapter 3. While the underlying framework remains the same, *i.e.*, encoding a shape as $2D \rightarrow 3D$



(a) Original **(b)** NCS coarse **(c)** Simplified

Figure 5.10: Limitation: NCS extract a coarse representation in an unsupervised manner (b) from a 3D mesh (a). However, this technique may categorize important structures, *e.g.*, the torch or hand, as details instead of part of the structure. Conversely, classic simplification approaches offer more control (c).

map, here we decouple the shape structure from its details. This formulation, along with the convolutional inductive bias, reduces the number of parameters required to represent a $3D$ model.

Nevertheless, the discussed representation is limited to shapes topologically equivalent to disks, as NSM. Furthermore, the overall optimization still requires significant amount of time, matters of hours.

5.6 Conclusions

Neural Convolutional Surfaces enable faithfully representing a given surface via a neural network, with higher accuracy and a smaller network capacity (e.g., 10-80x) compared to multiple state-of-the-art alternatives. Key to our method is an inductive bias in the network architecture that results in a split representation, with an MLP producing a coarse abstraction of the shape, and a fine detail CNN-like layer that adds geometric displacements based on the local reference frame the local UV charts.

We demonstrate that this coarse-fine disentanglement emerges naturally, without any intermediate supervision, and leads to the fine module reusing its convolutional kernels, which in turn enable meaningful geometric operations like mesh

smoothing and feature exaggeration.

Future Work

While we focused on faithfully representing individual shapes for the scope of this work, we intend to follow-up the next goal, of capturing *distributions* of shapes. We observe that geometric details is often reused across shapes and hence we can aspire to learn a universal dictionary of CNN detail kernels, that can then be applied across a diverse set of shapes, where the global structures are captured by shape-specific coarse abstractions. Such a universal dictionary of local geometric details will be a close analogue of low level features learnt on images (e.g., VGG features learnt using ImageNet), which, in turn, will enable both manipulation or transfer of details, as well as compression of shapes with a fixed universal shape-dictionary.

Chapter 6

Conclusions

6.1 Summary

In this thesis, we discussed neural surface representations, introduced a neural mapping framework that enables inter-surface map optimization, and we relaxed the required constraints of this framework by removing the human in the loop. Finally, we introduced a convolutional shape representation that enables surface manipulation and detail transfer. Here we summarize our contributions in each chapter.

In Chapter 3 we defined a surface representation as a map. A surface is represented by an MLP which maps a $2D$ point to $3D$, we dubbed it a neural surface map. Such description is continuous and thus can be composed with other maps. In particular, we show that by composing it with a neural map, acting as a bridge between two neural surfaces, it is possible to optimize an inter-surface map. We compare such mapping with classic method [93] and state of the art [167] showing that NSM provides better continuous and bijective maps.

Despite the mathematical advantages the framework offers, it presents several drawbacks. Thus, in Chapter 4, we extended it by describing a seamless neural surface (sNSM) that relaxes the constraints on the boundary. By combining this representation with the foundational model, DinoV2 [21], we obtain an automatic inter-surface map between two genus-0 shapes. These maps have been compared with state-of-the-art methods [166, 103] and classic approaches [114], demonstrating comparable results while offering better continuity and bijectivity.

Finally, in Chapter 5 we introduce a CNN based map-representation. In this context, the shape is decoupled between coarse structure and detail, where the former is encoded by a shallow MLP and the latter by a CNN. Intuitively, details repeat across the shape and thus can be compressed and stored efficiently with a CNN, while the pose or structure is global information. This representation, NCS, demonstrates better compression capacity than state-of-the-art methods [3, 4] while being extremely faithful. Furthermore, the representation can be used to transfer details between shapes, such as wrinkles, or to exaggerate features.

6.2 Limitations and Future Work

The broad adoption of a surface representation involves investigating many challenges. In this section, we elaborate on some future avenues for the proposed representation.

Speed: while the proposed representation lends itself to multiple geometric tasks, it is still far from being used in real-world applications due to the computational time required for training. In the future, we plan to investigate meta-learning solutions to speed up the optimization of such networks.

Appearance: meshes decouple geometry and appearance, and indeed, it is possible to change texture coordinates or vertices colours without affecting the geometry. Currently, the representations defined in this thesis can only store geometry. We wish to investigate dynamic appearance encoding that is decoupled from the underlying geometry.

Rendering: classic graphics pipelines require a triangle mesh to visualize models. Although proposed representations can be converted to meshes, this would entail the loss of the continuity property. We plan to investigate alternative methods that enable rendering models without converting triangular meshes, thus rendering the continuous surface encoded in the network.

Extension to any genus: current representations require cutting the shape open. While this can be manually done for simple meshes, it is extremely constraining for complex models with any genus. We plan to extend neural surfaces to the same

genus maps. For example, by using [209] it is possible to overfit a sphere-to-shape map, sidestepping the need for a cut.

Point clouds : following Chapter 3, shapes are encoded point-wise, thus NSM may encode point clouds. Although in such cases the initial parametrization is not available, or not straightforward to define, several works [210, 211] learn to morph a sphere to a point cloud through a Neural-ODE [212]. Unfortunately, all these methods rely on priors encapsulated into a large collection of shapes, thus for single-shape representation, we must impose geometric priors.

Dynamic compression: although the NCS can significantly compress shapes by reusing the kernels across object-centric local coordinate frames, the kernels themselves are still regular, 2D Euclidean image kernels. Further improvement can be obtained by optimally placing these kernels, with a dynamic size, where planar regions do not require to encode much information.

Dataset representation: geometric details are often reused across several shapes, we aspire to learn a universal dictionary that can be applied across a diverse set of shapes, while the global structures are captured by shape-specific coarse abstractions. This universal dictionary of local geometric details will be a close analogue of low-level features learned on images (e.g., VGG features [213], Dino-ViT [130], or CLIP[214]). A prominent example is human skin which exhibits pores, or micro wrinkles, such structure repeats on the entire surface and across different models. Thus a global dictionary would effectively compress these details and their variation.

Interpretability: the neural convolutional surface formulation has the potential to scale across multiple models and enable interpretable shape modelling, and analysis. Our initial experiments demonstrate a weak correlation between kernels and detail. We intend to pursue this as a future avenue.

6.3 Remark

In this thesis, we discussed two neural surface representations that enable shape correspondence, surface editing, compression, and detail transfer. To evaluate them, we

compared them with state-of-the-art neural representations, highlighting the flexibility, fidelity, and continuity these descriptions provide on artistic data. Furthermore, we described several future research directions. We expect the exploration of these questions to benefit augmented reality, shape modelling, cinematic industry, and graphics applications more broadly.

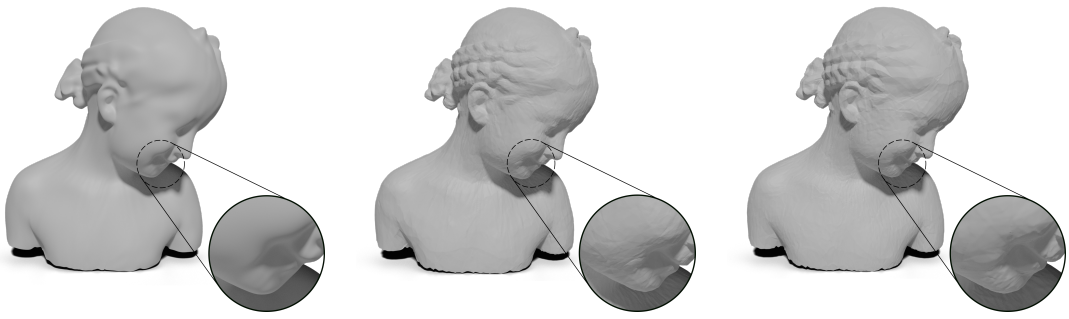
Appendix A

Neural Surface Maps

A.1 Approximating Surfaces

A *neural surface map* $f_{\mathbf{A}}$ can approximate a given surface $\mathbf{A}$ by overfitting the atlas $f_{\mathbf{A}} : \mathbb{R}^2 \to \mathbf{A}$ accurately to the inverse of the parametrization $P_{\mathbf{A}}^{-1}$. Each neural surface map consists of a ten-layer residual fully-connected network with Softplus activation.

In Figure A.1 we compare different activation functions such as LeakyRelu and Relu. These non-smooth functions introduce artefacts, such as unwanted wrinkles Figure A.1 (b and c). Overlooking this behaviour might bear negative effects in map composition as these introduced details can be amplified or inject distortion in the final map. The use Softplus alleviates these artefacts, but biases the map towards smooth surfaces will have minor discrepancies from the ground truth, as some areas



(a) **(b)** **(c)**

Figure A.1: A MLP-based neural surface map with different activation functions. Softplus (a) maps are smooth. LeakyReLU (b) and ReLU (c) are more oscillatory.

will be smoothed out.

A.2 Analytical Maps

As discussed in main body of this thesis, we can directly compose a neural map with analytical surfaces. Below we list the functions used.

Figure8:

$$f_{\mathbf{B}}(t,s) = \begin{cases} x = & (3+cos(\frac{t}{2})\cdot sin(s) - sin(\frac{t}{2})\cdot sin(2s))\cdot cos(t) \\ y = & (3+cos(\frac{t}{2})\cdot sin(s) - sin(\frac{t}{2})\cdot sin(2s))\cdot sin(t) \\ z = & sin(\frac{t}{2})\cdot sin(s) + cos(\frac{t}{2})\cdot sin(2s) \end{cases}, \tag{A.1}$$

where $t = \pi(1+u)$ and $s = \pi(1+v)$.

Enneper:

$$f_{\mathbf{B}}(u,v) = \begin{cases} x = & u\cdot(1-\frac{u^2}{3}+v^2)/3 \\ y = & -v\cdot(1-\frac{v^2}{3}+u^2)/3 \\ z = & (u^2-v^2)/3 \end{cases}. \tag{A.2}$$

Catenoid:

$$f_{\mathbf{B}}(u,v) = \begin{cases} x = & cosh(v)\cdot cos(u) \\ y = & cosh(v)\cdot sin(u) \\ z = & v \end{cases}. \tag{A.3}$$

Mobius:

$$f_{\mathbf{B}}(u,v) = \begin{cases} x = & (1+\frac{3v}{2}cos(\frac{3u}{2}))\cdot cos(3u) \\ y = & (1+\frac{3v}{2}cos(\frac{3u}{2}))\cdot sin(3u) \\ z = & \frac{3v}{2}\cdot sin(\frac{3u}{2}) \end{cases}. \tag{A.4}$$

Appendix B

Neural Semantic Surface Maps

B.1 Computing rendering correspondences

As discussed in the main manuscript, we render the two surfaces from a given viewpoint to get two renderings, $R_V^{\mathbf{A}}$ and $R_V^{\mathbf{B}}$. We leverage DinoV2 [21] to extract semantic features in the image space, thus obtaining $\lambda_i^{\mathbf{A}}$ and $\lambda_i^{\mathbf{B}}$ as features of rendering of $R_V^{\mathbf{A}}$ and $R_V^{\mathbf{B}}$, respectively. Then, to segment foreground/background we rely on PCA's first component of these features as it naturally groups them in opposite half-spaces.

Finally, we match features with the cosine similarity between all feature pairs from the same viewpoint, as score S_{ij}. We define the match of patch $i \in R_V^{\mathbf{A}}$ as the patch $j \in R_V^{\mathbf{B}}$ with the highest cosine similarity, and vice versa, the match of patch $j \in R_V^{\mathbf{B}}$ as the patch $i \in R_V^{\mathbf{A}}$ with the highest cosine similarity. In summary, the pair $(i, j), i \in R_V^{\mathbf{A}}, j \in R_V^{\mathbf{B}}$ is a match, if

$$S_{ij} = \max_k S_{ik} \text{ or } S_{ij} = \max_l S_{lj}. \tag{B.1}$$

B.1.1 Patch generation, feature extraction, and PCA

Images are split into (non-overlapping) patches of 14 pixels. Then, DinoV2[21] embeds these patches in a forward pass. Following [135], we use *keys* as feature vectors.

To segment foreground/background we rely on PCA's first component of the

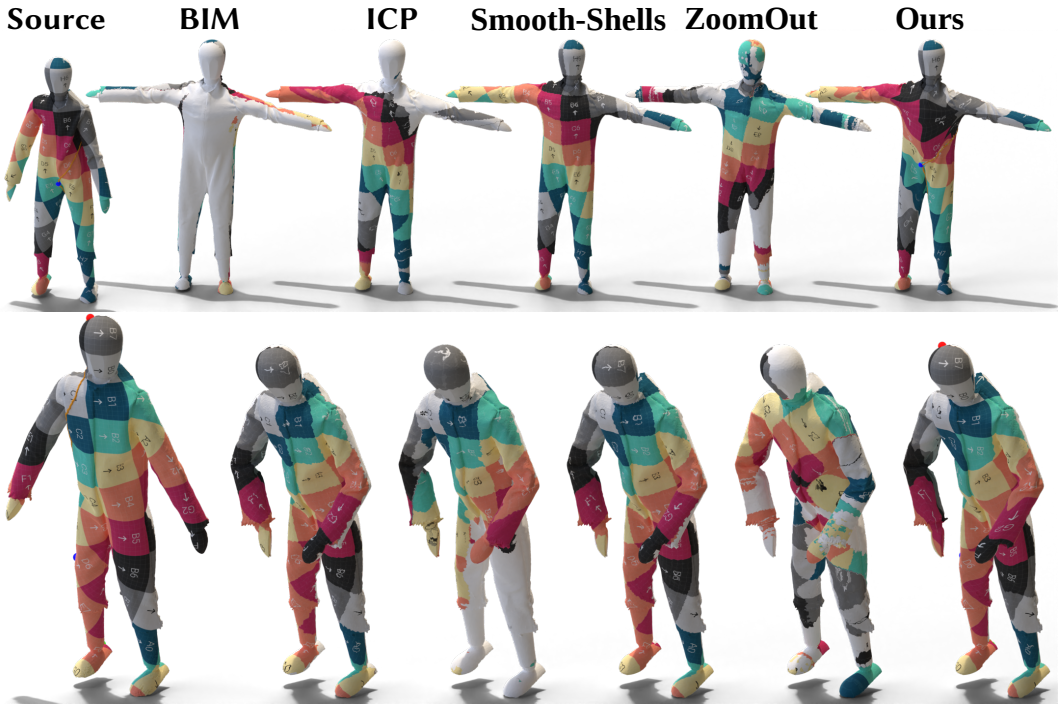


Figure B.1: Qualitative comparison SHREC19: Functional maps-based methods produce good maps, although often being discontinuous. Ours explicitly encourages continuity and bijectivity.

features. As discussed in [21], the features' sign naturally groups them in opposite half-spaces. As the sign is appointed randomly, we use the attention mask from the last layer to select the correct half-space: we average the first PCA component of the features and take the half-space which agrees with the positive attention mask. Matches are estimated only between foreground patches.

Finally, to unproject a match to $3D$, we first translate a patch to a pixel using the known patch size, and then identify the $3D$ point on each shape (ray casting).

B.2 Comparison Details

We discuss the main considerations for/against the competing algorithms we compare against.

Blended Intrinsic Maps (BIM) [114] is a classic method that uses geometric priors without any learning component. Namely, it picks a subset of self-consistent and low-distortion conformal maps and then blends them using weighted averages. Individual conformal maps can handle very non-isometric surfaces, however, they

can produce high isometric distortion even in near-isometric cases. Note also that the resulting blended map is not a homeomorphism nor even continuous.

ZoomOut [166] and Smooth-shells [103] are both functional maps-based methods. ZoomOut starts with a small functional correspondence matrix and iterative upsamples it in the spectral domain. Smooth-shells follow a similar coarse-to-fine scheme, relying on shells as a proxy for functional basis. To handle self-symmetries, Eisenberger et al. [103] incorporate MCMC to evaluate multiple possible functional maps.

We initialize ZoomOut's map (C_{21}) as an identity of size 4 as by official implementation. Then, we refine it until it contains 50 eigenvectors. Similarly, for Smooth-shells we follow the official implementation and use MCMC to bootstrap the map using $K_{min} = 6$ and $K_{max} = 20$. and evaluating $N_{prop} = 500$ proposal. In both cases, no landmarks are used. Finally, for ICP we first align the two input shapes as described in Sec. 3.1, and then estimate the nearest neighbour for each vertex.

We depict maps for the different methods on SHREC19 in Figure B.1. State-of-the-art methods work well as they exploit geometric cues, although they are susceptible to self-symmetries (see BIM[114] first row). Conversely, "Ours" relies purely on visual cues, with no isometric regularization, thus being less accurate on average.

B.3 Metrics

In all experiments, all shapes are automatically normalized and centred.

Bijectivity We estimate the map's bijectivity of the shape vertices for all baselines. For ICP, BIM, ZoomOut, and Smooth-shells we map all vertices forward ($A \rightarrow B$) and then backward ($A \leftarrow B$), using the forward and backward map respectively. Then, we compute the geodesic distance between the starting vertex and its forward-backward map.

Similarly, for consistency we evaluate "Ours" bijectivity only for the shape vertices. In particular, we map a vertex in A onto B's $2D$ domain through h, and

then, we use the piecewise linear map for 2D-3D. For B to A, we pull-back vertices through barycentric coordinates after mapping forward all A's triangles. Empirically, for "Ours" we never observe flips; while for baselines, correspondences are always given, thus, no ambiguity arises.

Appendix C

Neural Convolutional Surfaces

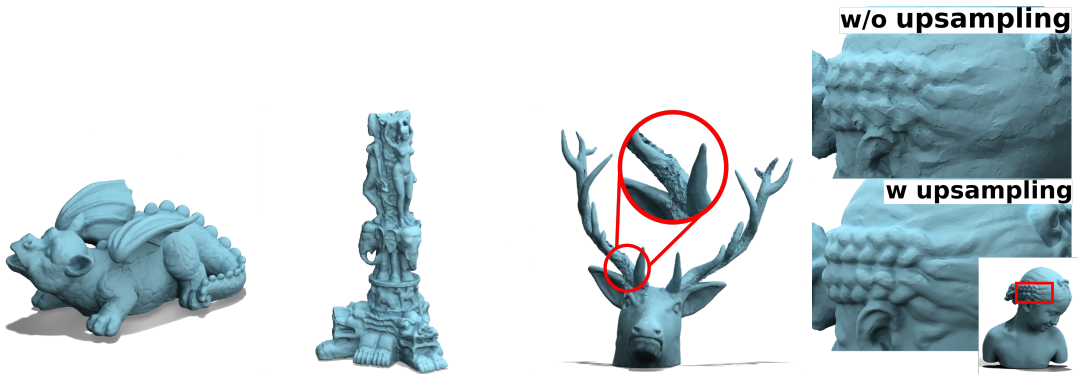
C.1 Comparisons

For all baselines, [3, 4, 2], we used authors' implementations. See Figure C.2 for qualitative a evaluation of NCS, with (b), and without details (c).

C.2 Expressive power.

Our construction is readily applicable to any genus, by cutting the mesh to a disk it is possible to reconstruct any surface. See Figure C.1(a,b) for reconstruction examples with different genus. However, Neural Convolutional Surfaces struggle to represent accurately thin structures. See Figure C.1(c) for such a thin structure our framework is able to reproduce.

Finally, the upsampling is fundamental design choice for the CNN. Without



(a) Genus 2 **(b)** Genus 3 **(c)** ThinStructure **(d)** NoUpsampling

Figure C.1: Results on high genus, thin structures, w/o upsampling.

Table C.1: Architecture details used for each shape presented in the paper.

	Coarse MLP g^c_ϕ	Per-patch Code Ω_i	CNN f_ν channels	Fine MLP h_ξ
Armadillo-100K	128-64-64	$8 \times 4 \times 4$	8	16-16
Bimba-100K	128-64	$8 \times 4 \times 4$	8	16-16
Dino-100K	64-64	$8 \times 8 \times 8$	8	16-16
Dragon-100K	128-64-64	$6 \times 6 \times 6$	8	16-16
Gargoyle-100K	64-64-64-64	$6 \times 4 \times 4$	6	16-16
Grog-100K	128-64-64	$8 \times 4 \times 4$	8	16-16
Seahorse-100K	128-64-64	$8 \times 8 \times 8$	8	16-16
Elephant-100K	128-64-64	$8 \times 6 \times 6$	8	16-16
Armadillo-1M	128-64-64	$64 \times 4 \times 4$	64	16-16
Bimba-1M	128-64	$64 \times 4 \times 4$	64	16-16
Dino-1M	64-64	$64 \times 8 \times 8$	64	16-16
Dragon-1M	128-64-64	$66 \times 6 \times 6$	64	16-16

upsampling the model is unable to capture details, see Figure C.1(d).

C.3 Architecture Details

For the model f_ν, we used a 5-layer residual CNN with ReLU non-linearities. The fine MLP h_ξ uses a ReLU non-linearity after each layer except the last, and the coarse MLP g^c_ϕ uses Softplus activations. Please refer to Table C.1 for complete architecture details of each model.

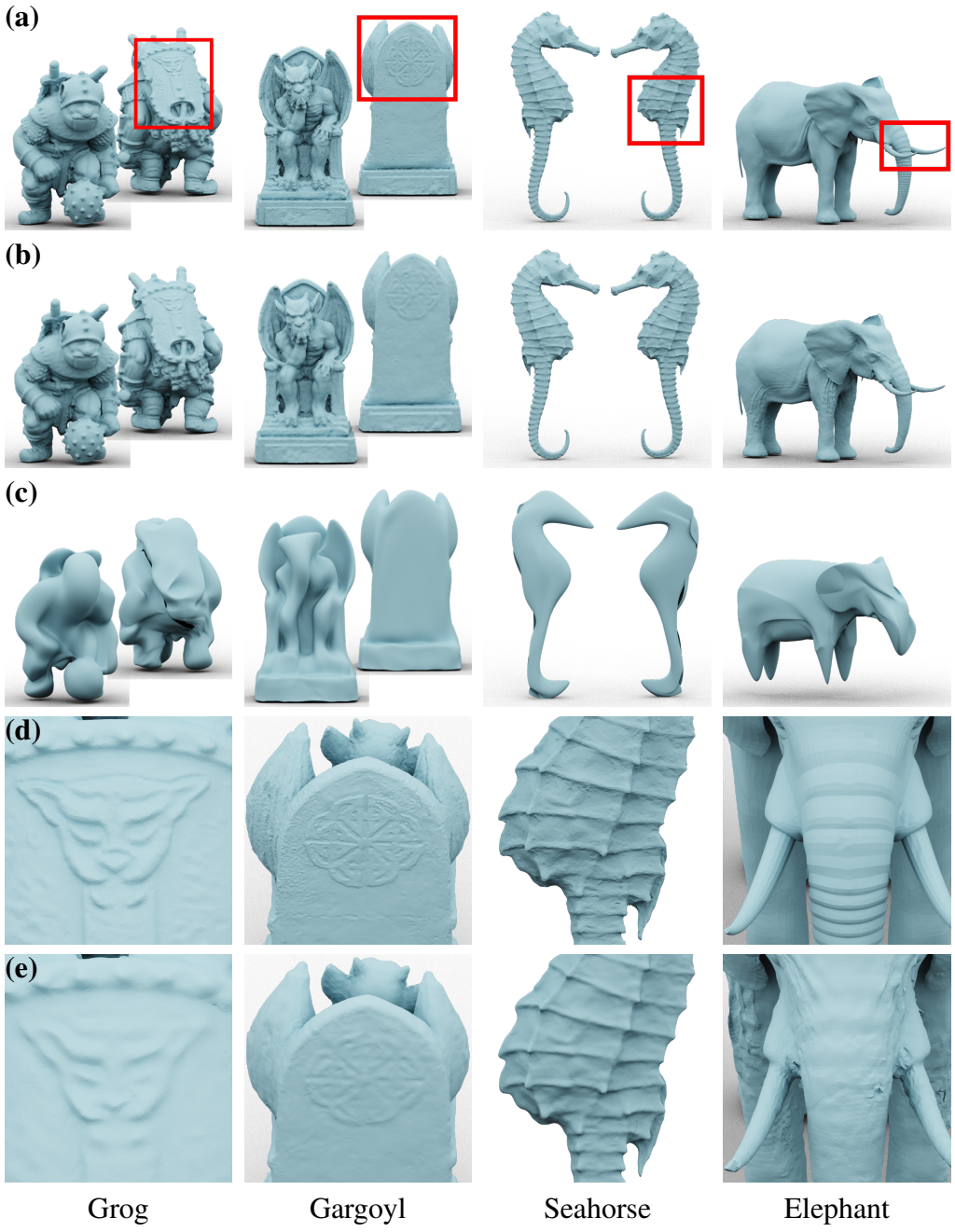


Figure C.2: Representation quality – with (b) and without (c) details – of our method, compared with the ground truth model (a). We limit Neural Convolutional Surfaces to 100K parameters. We show inset zooms (e) of our reconstruction for further assessment, with corresponding inset zooms (d) for ground truth.

Bibliography

[1] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3d shape surfaces using geometry images. In *Proceedings of the European Conference on Computer Vision*, 2016.

[2] Wang Yifan, Lukas Rahmann, and Olga Sorkine-Hornung. Geometry-consistent neural shape representation with implicit displacement fields, 2021.

[3] Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. *ACM Transactions on Graphics (TOG)*, 2021.

[4] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021.

[5] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[6] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 216–224, 2018.

[7] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 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, 2019.

[8] Zhiqin Chen. Im-net: Learning implicit fields for generative shape modeling. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[9] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Conference on Neural Information Processing Systems*, 33:15651–15663, 2020.

[10] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017.

[11] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9054–9063, 2021.

[12] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018.

[13] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the European Conference on Computer Vision*, pages 230–246, 2018.

[14] Theo Deprelle, Thibault Groueix, Noam Aigerman, Vladimir G Kim, and Mathieu Aubry. Learning joint surface atlases. *Proceedings of the European Conference on Computer Vision Workshops*, 2022.

[15] Jan Bednarik, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, Shaifali Parashar, Mathieu Salzmann, and Pascal Fua. Temporally-consistent surface reconstruction using metrically-consistent atlases. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[16] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)*, 31(4):1–11, 2012.

[17] Or Litany, Emanuele Rodola, Alexander M Bronstein, Michael M Bronstein, and Daniel Cremers. Non-rigid puzzles. In *Computer Graphics Forum*, volume 35, pages 135–143. Wiley Online Library, 2016.

[18] Souhaib Attaiki, Gautam Pai, and Maks Ovsjanikov. Dpfm: Deep partial functional maps. In *International Conference on 3D Vision*, pages 175–185. IEEE, 2021.

[19] Or Litany, Tal Remez, Emanuele Rodola, Alex Bronstein, and Michael Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5659–5667, 2017.

[20] Qixing Huang, Fan Wang, and Leonidas Guibas. Functional map networks for analyzing and exploring large shape collections. *ACM Transactions on Graphics (TOG)*, 33(4):1–11, 2014.

[21] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[22] Jan Bednarik, Shaifali Parashar, Erhan Gundogdu, Mathieu Salzmann, and Pascal Fua. Shape reconstruction by learning differentiable surface representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4716–4725, 2020.

[23] Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. Scalable locally injective mappings. *ACM Transactions on Graphics (TOG)*, 36(4):1, 2017.

[24] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing*, volume 4, pages 109–116, 2007.

[25] Noam Aigerman and Yaron Lipman. Orbifold tutte embeddings. *ACM Transactions on Graphics (TOG)*, 34(6):1–12, 2015.

[26] Yousuf Soliman, Dejan Slep*v*cev, and Keenan Crane. Optimal cone singularities for conformal flattening. *ACM Transactions on Graphics (TOG)*, 37(4):1–17, 2018.

[27] Rohan Sawhney and Keenan Crane. Boundary first flattening. *ACM Transactions on Graphics (TOG)*, 37(1):5:1–5:14, December 2017.

[28] Alex Baden, Keenan Crane, and Misha Kazhdan. Möbius Registration. *Computer Graphics Forum (SGP)*, 37(5), 2018.

[29] Patrick Schmidt, Dörte Pieper, and Leif Kobbelt. Surface Maps via Adaptive Triangulations. *Computer Graphics Forum*, 2023.

[30] Michael S Floater and Kai Hormann. Surface parameterization: a tutorial and survey. *Advances in multiresolution for geometric modelling*, pages 157–186, 2005.

[31] Aayush Bansal, Bryan Russell, and Abhinav Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5965–5974, 2016.

[32] Jan Bednarik, Pascal Fua, and Mathieu Salzmann. Learning to reconstruct texture-less deformable surfaces from a single view. In *International Conference on 3D Vision*, pages 606–615. IEEE, 2018.

[33] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Conference on Neural Information Processing Systems*, 27, 2014.

[34] Aggeliki Tsoli, Antonis Argyros, et al. Patch-based reconstruction of a textureless deformable 3d surface from a single rgb image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[35] Jin Zeng, Yanfeng Tong, Yunmu Huang, Qiong Yan, Wenxiu Sun, Jing Chen, and Yongtian Wang. Deep surface normal estimation with hierarchical rgb-d fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6153–6162, 2019.

[36] R Dan*vev*rek, Endri Dibra, Cengiz Öztireli, Remo Ziegler, and Markus Gross. Deepgarment: 3d garment shape estimation from a single image. In *Computer Graphics Forum*, volume 36, pages 269–280. Wiley Online Library, 2017.

[37] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9785–9795, 2019.

[38] Erhan Gundogdu, Victor Constantin, Shaifali Parashar, Amrollah Seifoddini, Minh Dang, Mathieu Salzmann, and Pascal Fua. Garnet++: Improving fast and accurate static 3d cloth draping by curvature loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):181–195, 2020.

[39] Albert Pumarola, Antonio Agudo, Lorenzo Porzi, Alberto Sanfeliu, Vincent Lepetit, and Francesc Moreno-Noguer. Geometry-aware network for non-rigid shape prediction from a single view. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, pages 4681–4690, 2018.

[40] Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Meshsdf: Differentiable iso-surface extraction. *Conference on Neural Information Processing Systems*, 33:22468–22478, 2020.

[41] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision*, pages 52–67, 2018.

[42] Tim Winkler, Jens Drieseberg, Marc Alexa, and Kai Hormann. Multi-scale geometry interpolation. In *Computer Graphics Forum*, volume 29, pages 309–318. Wiley Online Library, 2010.

[43] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020.

[44] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[45] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[46] Mohamed Sayed, John Gibson, Jamie Watson, Victor Prisacariu, Michael Firman, and Clément Godard. Simplerecon: 3d reconstruction without 3d convolutions. In *Proceedings of the European Conference on Computer Vision*, pages 1–19. Springer, 2022.

[47] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision*, pages 628–644. Springer, 2016.

[48] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *International Conference on 3D Vision*, pages 412–420. IEEE, 2017.

[49] Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D Kulkarni, and Joshua B Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1511–1519, 2017.

[50] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 45–54, 2020.

[51] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 31–44, 2020.

[52] Despoina Paschalidou, Luc Van Gool, and Andreas Geiger. Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1060–1070, 2020.

[53] Dmitriy Smirnov, Matthew Fisher, Vladimir G Kim, Richard Zhang, and Justin Solomon. Deep parametric shape predictions using distance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 561–570, 2020.

[54] Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Learning elementary structures for 3d shape generation and matching. *Conference on Neural Information Processing Systems*, 2019.

[55] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[56] Abhishek Badki, Orazio Gallo, Jan Kautz, and Pradeep Sen. Meshlet priors for 3d mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2849–2858, 2020.

[57] Marko Mihajlovic, Silvan Weder, Marc Pollefeys, and Martin R Oswald. Deepsurfels: Learning online appearance fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14524–14535, 2021.

[58] Sanjeev Muralikrishnan, Vladimir G Kim, Matthew Fisher, and Siddhartha Chaudhuri. Shape unicode: A unified shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3790–3799, 2019.

[59] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. *Conference on Neural Information Processing Systems*, 34:13032–13044, 2021.

[60] Omid Poursaeed, Matthew Fisher, Noam Aigerman, and Vladimir G Kim. Coupling explicit and implicit surface representations for generative 3d modeling. In *Proceedings of the European Conference on Computer Vision*, pages 667–683. Springer, 2020.

[61] Jianjie Lin, Markus Rickert, Alexander Perzylo, and Alois Knoll. Pctma-net: Point cloud transformer with morphing atlas-based point generation network

for dense point cloud completion. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5657–5663. IEEE, 2021.

[62] Jiahao Pang, Duanshun Li, and Dong Tian. Tearingnet: Point cloud autoencoder to learn topology-friendly representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[63] Qianli Ma, Shunsuke Saito, Jinlong Yang, Siyu Tang, and Michael J. Black. SCALE: Modeling clothed humans with a surface codec of articulated local elements. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16082–16093, 2021.

[64] Heli Ben-Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[65] Jason Zhang, Gengshan Yang, Shubham Tulsiani, and Deva Ramanan. Ners: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild. *Conference on Neural Information Processing Systems*, 34:29835–29847, 2021.

[66] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022.

[67] Vincent Sitzmann, Julien NP Martel, Alexander W Bergman, David B Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Conference on Neural Information Processing Systems*, 2020.

[68] Vincent Sitzmann, Eric R. Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. MetaSDF: Meta-learning signed distance functions. *Conference on Neural Information Processing Systems*, 2020.

[69] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5799–5809, 2021.

[70] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1440–1448, 2015.

[71] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6202–6211, 2019.

[72] Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. Overfit neural networks as a compact shape representation, 2020.

[73] Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. Geometry processing with neural fields. *Conference on Neural Information Processing Systems*, 34:22483–22497, 2021.

[74] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021.

[75] Arnab Dey, Yassine Ahmine, and Andrew I Comport. Mip-nerf rgb-d: Depth assisted fast neural radiance fields. *arXiv preprint arXiv:2205.09351*, 2022.

[76] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894, 2021.

[77] Di Chen, Yu Liu, Lianghua Huang, Bin Wang, and Pan Pan. Geoaug: Data augmentation for few-shot nerf with geometry constraints. In *Proceedings*

of the European Conference on Computer Vision, pages 322–337. Springer, 2022.

[78] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[79] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[80] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[81] Yang Fu, Ishan Misra, and Xiaolong Wang. Mononerf: learning generalizable nerfs from monocular videos without camera poses. In *International Conference on Machine Learning*, pages 10392–10404. PMLR, 2023.

[82] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021.

[83] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d representation and rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15182–15192, 2021.

[84] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.

[85] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001.

[86] Timo Zinßer, Jochen Schmidt, and Heinrich Niemann. A refined icp algorithm for robust 3-d correspondence estimation. In *Proceedings 2003 international conference on image processing (Cat. No. 03CH37429)*, volume 2, pages II–695. IEEE, 2003.

[87] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5545–5554, 2019.

[88] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 195–205, 2018.

[89] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3523–3532, 2019.

[90] Noam Aigerman, Roi Poranne, and Yaron Lipman. Seamless surface mappings. *ACM Transactions on Graphics (TOG)*, 34(4):1–13, 2015.

[91] Patrick Schmidt, Janis Born, Marcel Campen, and Leif Kobbelt. Distortion-minimizing injective maps between surfaces. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019.

[92] Patrick Schmidt, Marcel Campen, Janis Born, and Leif Kobbelt. Inter-surface maps via constant-curvature metrics. *ACM Transactions on Graphics (TOG)*, 39(4):119–1, 2020.

[93] John Schreiner, Arul Asirvatham, Emil Praun, and Hugues Hoppe. Inter-surface mapping. *ACM Transactions on Graphics (TOG)*, pages 870–877, 2004.

[94] Etienne Corman, Justin Solomon, Mirela Ben-Chen, Leonidas Guibas, and Maks Ovsjanikov. Functional characterization of intrinsic and extrinsic geometry. *ACM Transactions on Graphics (TOG)*, 36(2), mar 2017.

[95] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of the European Conference on Computer Vision*, pages 356–369. Springer, 2010.

[96] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer Graphics Forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009.

[97] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 1626–1633. IEEE, 2011.

[98] Artiom Kovnatsky, Michael M Bronstein, Alexander M Bronstein, Klaus Glashoff, and Ron Kimmel. Coupled quasi-harmonic bases. In *Computer Graphics Forum*, volume 32, pages 439–448. Wiley Online Library, 2013.

[99] Artiom Kovnatsky, Michael M Bronstein, Xavier Bresson, and Pierre Vandergheynst. Functional correspondence by matrix completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 905–914, 2015.

[100] Emanuele Rodolà, Michael Moeller, and Daniel Cremers. Point-wise Map Recovery and Refinement from Functional Correspondence. In David Bommes, Tobias Ritschel, and Thomas Schultz, editors, *Vision, Modeling and Visualization*. The Eurographics Association, 2015.

[101] Danielle Ezuz and Mirela Ben-Chen. Deblurring and denoising of maps between shapes. In *Symposium on Geometry Processing*, 2017.

[102] Jing Ren, Mikhail Panine, Peter Wonka, and Maks Ovsjanikov. Structured regularization of functional map computations. In *Computer Graphics Forum*, volume 38, pages 39–53. Wiley Online Library, 2019.

[103] Marvin Eisenberger, Zorah Lahner, and Daniel Cremers. Smooth shells: Multi-scale shape registration with functional maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12265–12274, 2020.

[104] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 37–45, 2015.

[105] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. *Conference on Neural Information Processing Systems*, 29, 2016.

[106] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[107] Zhiyu Sun, Yusen He, Andrey Gritsenko, Amaury Lendasse, and Stephen Baek. Embedded spectral descriptors: learning the point-wise correspondence metric via siamese neural networks. *Journal of Computational Design and Engineering*, 7(1):18–29, 2020.

[108] Etienne Corman, Maks Ovsjanikov, and Antonin Chambolle. Supervised descriptor learning for non-rigid shape matching. In *Proceedings of the Euro-*

pean Conference on Computer Vision Workshops, pages 283–298. Springer, 2015.

[109] Oshri Halimi, Or Litany, Emanuele Rodola, Alex M Bronstein, and Ron Kimmel. Unsupervised learning of dense shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4370–4379, 2019.

[110] Or Litany, Emanuele Rodola, Alex Bronstein, Michael Bronstein, and Daniel Cremers. Partial single-and multishape dense correspondence using functional maps. *Handbook of Numerical Analysis*, 2018.

[111] Jing Ren, Simone Melzi, Maks Ovsjanikov, and Peter Wonka. Maptree: recovering multiple solutions in the space of maps. *ACM Transactions on Graphics (TOG)*, 39(6):264–1, 2020.

[112] Nicolas Donati, Abhishek Sharma, and Maks Ovsjanikov. Deep geometric functional maps: Robust feature learning for shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8592–8601, 2020.

[113] Gautam Pai, Jing Ren, Simone Melzi, Peter Wonka, and Maks Ovsjanikov. Fast sinkhorn filters: Using matrix scaling for non-rigid shape correspondence with functional maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 384–393, 2021.

[114] Vladimir G. Kim, Yaron Lipman, and Thomas Funkhouser. Blended intrinsic maps. *ACM Transactions on Graphics (TOG)*, 30(4), 2011.

[115] Noam Aigerman, Roi Poranne, and Yaron Lipman. Lifted bijections for low distortion surface mappings. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014.

[116] Arul Asirvatham, Emil Praun, and Hugues Hoppe. Consistent spherical parameterization. In *Computer Graphics and Geometric Modeling (CGGM) 2005 Workshop*, 2005.

[117] Yusuf Sahillioğlu. Recent advances in shape correspondence. *The Visual Computer*, 36(8):1705–1721, 2020.

[118] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2015.

[119] Gopal Sharma, Kangxue Yin, Subhransu Maji, Evangelos Kalogerakis, Or Litany, and Sanja Fidler. Mvdecor: Multi-view dense correspondence learning for fine-grained 3d segmentation. In *Proceedings of the European Conference on Computer Vision*, 2022.

[120] Abhijit Kundu, Xiaoqi Yin, Alireza Fathi, David Ross, Brian Brewington, Thomas Funkhouser, and Caroline Pantofaru. Virtual multi-view fusion for 3d semantic segmentation. *Proceedings of the European Conference on Computer Vision*, 2020.

[121] Dale Decatur, Itai Lang, and Rana Hanocka. 3d highlighter: Localizing regions on 3d shapes via text descriptions. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[122] Haibin Huang, Evangelos Kalogerakis, Siddhartha Chaudhuri, Duygu Ceylan, Vladimir G. Kim, and Ersin Yumer. Learning local shape descriptors from part correspondences with multiview convolutional networks. *ACM Transactions on Graphics (TOG)*, 37(1), 2017.

[123] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3D shape segmentation with projective convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

[124] Kyle Genova, Xiaoqi Yin, Abhijit Kundu, Caroline Pantofaru, Forrester Cole, Avneesh Sud, Brian Brewington, Brian Shucker, and Thomas Funkhouser. Learning 3d semantic segmentation with only 2d image supervision. *International Conference on 3D Vision*, 2021.

[125] Ahmed Abdelreheem, Ivan Skorokhodov, Maks Ovsjanikov, and Peter Wonka. Satr: Zero-shot semantic segmentation of 3d shapes. *ACM Transactions on Graphics (TOG)*, 2023.

[126] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *International Conference on Machine Learning*, 2023.

[127] Jing Ren, Adrien Poulenard, Peter Wonka, and Maks Ovsjanikov. Continuous and orientation-preserving correspondences via functional maps. *ACM Transactions on Graphics (TOG)*, 37(6):1–16, 2018.

[128] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[129] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. dino-vit-features.github.io. *International Conference on Learning Representations*, 2021.

[130] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021.

[131] Oriane Siméoni, Gilles Puy, Huy V Vo, Simon Roburin, Spyros Gidaris, Andrei Bursuc, Patrick Pérez, Renaud Marlet, and Jean Ponce. Localizing ob-

jects with self-supervised transformers and no labels. *British Machine Vision Conference*, 2021.

[132] Yangtao Wang, Xi Shen, Shell Xu Hu, Yuan Yuan, James L Crowley, and Dominique Vaufreydaz. Self-supervised transformers for unsupervised object discovery using normalized cut. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14543–14553, 2022.

[133] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan, Noah Snavely, and William T Freeman. Unsupervised semantic segmentation by distilling feature correspondences. *International Conference on Learning Representations*, 2022.

[134] Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Generalized category discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7492–7501, 2022.

[135] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *Proceedings of the European Conference on Computer Vision Workshops*, 2022.

[136] Martin Weier, Michael Stengel, Thorsten Roth, Piotr Didyk, Elmar Eisemann, Martin Eisemann, Steve Grogorick, André Hinkenjann, Ernst Kruijff, Marcus Magnor, et al. Perception-driven accelerated rendering. In *Computer Graphics Forum*, volume 36, pages 611–643. Wiley Online Library, 2017.

[137] Hugues Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of the 24th annual conference on Computer Graphics and Interactive Techniques*, pages 189–198, 1997.

[138] Jihad El-Sana, Elvir Azanli, and Amitabh Varshney. Skip strips: maintaining triangle strips for view-dependent rendering. In *Proceedings Visualization'99*, pages 131–518. IEEE, 1999.

[139] Peter Lindstrom and Greg Turk. Image-driven simplification. *ACM Transactions on Graphics (TOG)*, 19(3):204–241, 2000.

[140] Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. Appearance-driven automatic 3d model simplification. In *EGSR (DL)*, pages 85–97, 2021.

[141] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. Paparazzi: surface editing by way of multi-view image processing. *ACM Transactions on Graphics (TOG)*, 37(6):221–1, 2018.

[142] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

[143] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Deep geometric texture synthesis. *ACM Transactions on Graphics (TOG)*, 39(4):108–1, 2020.

[144] Hsueh-Ti Derek Liu, Vladimir G Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. Neural subdivision. *ACM Transactions on Graphics (TOG)*, 2020.

[145] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5868–5877, 2017.

[146] Hsueh-Ti Derek Liu and Alec Jacobson. Cubic stylization. *ACM Transactions on Graphics (TOG)*, 2019.

[147] Xingguang Yan, Liqiang Lin, Niloy J Mitra, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Shapeformer: Transformer-based shape completion via sparse representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6239–6249, 2022.

[148] Angela Dai and Matthias Nießner. Scan2mesh: From unstructured range scans to 3d meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5574–5583, 2019.

[149] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5841–5850, 2018.

[150] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision*, pages 704–720, 2018.

[151] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[152] Ran Gal, Olga Sorkine, Tiberiu Popa, Alla Sheffer, and Daniel Cohen-Or. 3d collage: expressive non-realistic modeling. In *Proceedings of the 5th international symposium on Non-photorealistic Animation and Rendering*, pages 7–14, 2007.

[153] Christian Theobalt, Christian Rössl, Edilson de Aguiar, and Hans-Peter Seidel. Animation collage. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer Animation*, pages 271–280. Citeseer, 2007.

[154] Romain Pierre Testuz, Yuliy Schwartzburg, and Mark Pauly. Automatic generation of constructable brick sculptures. Technical report, 2013.

[155] Sheng-Jie Luo, Yonghao Yue, Chun-Kai Huang, Yu-Huan Chung, Sei Imai, Tomoyuki Nishita, and Bing-Yu Chen. Legolization: Optimizing lego designs. *ACM Transactions on Graphics (TOG)*, 34(6):1–12, 2015.

[156] Liang-Tsen Shen, Sheng-Jie Luo, Chun-Kai Huang, and Bing-Yu Chen. Sd models: Super-deformed character models. In *Computer Graphics Forum*, volume 31, pages 2067–2075. Wiley Online Library, 2012.

[157] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13492–13502, 2022.

[158] Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 3dstylenet: Creating 3d shapes with geometric and texture style variations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12456–12465, 2021.

[159] Charles Loop. Smooth subdivision surfaces based on triangles. *Master's thesis, University of Utah, Department of Mathematics*, 1987.

[160] Denis Zorin, Peter Schröder, and Wim Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of the 23rd annual conference on Computer Graphics and Interactive Techniques*, pages 189–192, 1996.

[161] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of the 21st annual conference on Computer Graphics and Interactive Techniques*, pages 295–302, 1994.

[162] Yun-Chun Chen, Vladimir Kim, Noam Aigerman, and Alec Jacobson. Neural progressive meshes. In *ACM SIGGRAPH Conference Proceedings*, pages 1–9, 2023.

[163] Alla Sheffer, Emil Praun, Kenneth Rose, et al. Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision*, 2(2):105–171, 2007.

[164] David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. Quad-mesh generation and processing: A survey. *Computer Graphics Forum*, 32(6):51–76, 2013.

[165] Emil Praun, Wim Sweldens, and Peter Schröder. Consistent mesh parameterizations. In *Proceedings of the 28th annual conference on Computer Graphics and Interactive Techniques*, pages 179–184. ACM, 2001.

[166] Simone Melzi, Jing Ren, Emanuele Rodola, Abhishek Sharma, Peter Wonka, and Maks Ovsjanikov. Zoomout: Spectral upsampling for efficient shape correspondence. *ACM Transactions on Graphics (TOG)*, 2019.

[167] Manish Mandad, David Cohen-Steiner, Leif Kobbelt, Pierre Alliez, and Mathieu Desbrun. Variance-minimizing transport plans for inter-surface mapping. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.

[168] Vladislav Kraevoy and Alla Sheffer. Cross-parameterization and compatible remeshing of 3d models. *ACM Transactions on Graphics (TOG)*, 23(3):861–869, August 2004.

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

[170] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[171] Aaron W. F. Lee, David Dobkin, Wim Sweldens, and Peter Schröder. Multiresolution mesh morphing. In *Proceedings of the 26th annual conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 343–350, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[172] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. *ACM Transactions on Graphics (TOG)*, 27(3):99:1–99:9, August 2008.

[173] Ofir Weber and Denis Zorin. Locally injective parametrization with arbitrary fixed boundaries. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014.

[174] Noam Aigerman and Yaron Lipman. Hyperbolic orbifold tutte embeddings. *ACM Transactions on Graphics (TOG)*, 35(6):217–1, 2016.

[175] Andy Nguyen, Mirela Ben-Chen, Katarzyna Welnicka, Yinyu Ye, and Leonidas Guibas. An optimization approach to improving collections of shape maps. In *Computer Graphics Forum*, volume 30, pages 1481–1491. Wiley Online Library, 2011.

[176] Qi-Xing Huang and Leonidas Guibas. Consistent shape maps via semidefinite programming. In *Computer Graphics Forum*, volume 32, pages 177–186. Wiley Online Library, 2013.

[177] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

[178] Weiwei Hu, Anhong Dang, and Ying Tan. A survey of state-of-the-art short text matching algorithms. In *International Conference on Data Mining and Big Data*, 2019.

[179] Hao Zhu, Man-Di Luo, Rui Wang, Ai-Hua Zheng, and Ran He. Deep audio-visual learning: A survey. *International Journal of Automation and Computing*, 18:351–376, 2021.

[180] Jiayi Ma, Xingyu Jiang, Aoxiang Fan, Junjun Jiang, and Junchi Yan. Image matching from handcrafted to deep features: A survey. *Int. J. Comput. Vision*, 129(1):23–79, 2021.

[181] Hui Sun, Wenju Zhou, and Minrui Fei. A survey on graph matching in computer vision. In *Intn. Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 225–230, 2020.

[182] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics (TOG)*, 23(3):399–405, 2004.

[183] Ilya Baran, Daniel Vlasic, Eitan Grinspun, and Jovan Popović. Semantic deformation transfer. *ACM Transactions on Graphics (TOG)*, pages 1–6, 2009.

[184] Matthew Turk and Alex Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 01 1991.

[185] Michael Black, Javier Romero, Gerard Pons-Moll, Naureen Mahmood, and Federica Bogo. Learning human body shapes in motion. In *SIGGRAPH ASIA 2016 Courses*, SIGGRAPH '16, 2016.

[186] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3794–3801, 2014.

[187] Leonid Pishchulin, Stefanie Wuhrer, Thomas Helten, Christian Theobalt, and Bernt Schiele. Building statistical shape spaces for 3d human modeling. *CoRR*, abs/1503.05860, 2015.

[188] Qixing Huang, Martin Wicke, Bart Adams, and Leonidas Guibas. Shape decomposition using modal analysis. In *Computer Graphics Forum*, volume 28, pages 407–416, Munchen, Germany, April 2009.

[189] Sanjeev Muralikrishnan, Siddhartha Chaudhuri, Noam Aigerman, Vladimir Kim, Matthew Fisher, and Niloy Mitra. Glass: Geometric latent augmentation for shape spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[190] Yong-Liang Yang, Yi-Jun Yang, Helmut Pottmann, and Niloy J. Mitra. Shape space exploration of constrained meshes. *ACM Transactions on Graphics*, 30(6), 2011.

[191] Bernhard Egger, William A. P. Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhöfer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, Christian Theobalt, Volker Blanz, and Thomas Vetter. 3d morphable face models - past, present and future. *CoRR*, abs/1909.01815, 2019.

[192] Xufang Pang, Feng Li, Ning Ding, and Xiaopin Zhong. Upright-net: Learning upright orientation for 3d point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14911–14919, 2022.

[193] Omid Poursaeed, Tianxing Jiang, Han Qiao, Nayun Xu, and Vladimir G Kim. Self-supervised learning of point clouds via orientation estimation. In *International Conference on 3D Vision*, pages 1018–1028. IEEE, 2020.

[194] Tali Dekel, Shaul Oron, Michael Rubinstein, Shai Avidan, and William T Freeman. Best-buddies similarity for robust template matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2021–2029, 2015.

[195] John E Hopcroft and Richard M Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[196] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. Mitsuba 3 renderer, 2022. https://mitsuba-renderer.org.

[197] Michal Edelstein, Danielle Ezuz, and Mirela Ben-Chen. Enigma: Evolutionary non-isometric geometry matching. In *ACM Transactions on Graphics (TOG)*, 2020.

[198] Daniela Giorgi, Silvia Biasotti, and Laura Paraboschi. Shrec: shape retrieval contest: Watertight models track. *Online]: http://watertight. ge. imati. cnr. it*, 7, 2007.

[199] Roberto Dyke, Caleb Stride, Yukun Lai, and Paul Rosin. Shrec-19: shape correspondence with isometric and non-isometric deformations. *Proceedings of the European Conference on Computer Vision Workshops*, 2019.

[200] Dragomir Anguelov, Praveen Srinivasan, Hoi-Cheung Pang, Daphne Koller, Sebastian Thrun, and James Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. *Conference on Neural Information Processing Systems*, 17, 2004.

[201] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[202] Zhongjin Luo, Shengcai Cai, Jinguo Dong, Ruibo Ming, Liangdong Qiu, Xiaohang Zhan, and Xiaoguang Han. Rabit: Parametric modeling of 3d biped cartoon characters with a topological-consistent dataset. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[203] Zz Lıan, Z Zhang, H El Elnaghy, J El Sana, T Furuya, A Gıachettı, Alp Güler, L Laı, C Lı, H Lı, et al. Shrec 15 track non rigid 3d shape retrieval. 2015.

[204] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 34(6):1–16, 2015.

[205] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008.

[206] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[207] Michael Fischer and Tobias Ritschel. Metappearance: Meta-learning for visual appearance reproduction. *ACM Transactions on Graphics (TOG)*, 41(6):1–13, 2022.

[208] Chen Liu, Michael Fischer, and Tobias Ritschel. Learning to learn and sample brdfs. In *Computer Graphics Forum*, volume 42, pages 201–211. Wiley Online Library, 2023.

[209] Craig Gotsman, Xianfeng Gu, and Alla Sheffer. Fundamentals of spherical parameterization for 3d meshes. *ACM Transactions on Graphics (TOG)*, pages 358–363, 2003.

[210] Chiyu Jiang, Jingwei Huang, Andrea Tagliasacchi, and Leonidas J Guibas. Shapeflow: Learnable deformation flows among 3d shapes. *Conference on Neural Information Processing Systems*, 33:9745–9757, 2020.

[211] Kunal Gupta. *Neural mesh flow: 3d manifold mesh generation via diffeomorphic flows*. University of California, San Diego, 2020.

[212] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Conference on Neural Information Processing Systems*, 31, 2018.

[213] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

[214] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[215] Luca Morreale, Noam Aigerman, Vladimir G Kim, and Niloy J Mitra. Neural surface maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4639–4648, 2021.