UNIVERSITY OF CALIFORNIA SAN DIEGO

Dense Human Surface Reconstruction with Lightweight Multi-view System

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Signal and Image Processing)

by

Kunyao Chen

Committee in charge:

Professor Truong Nguyen, Chair
Professor Cheolhong An
Professor Pamela C. Cosman
Professor Ryan Kastner
Professor Falko Kuester

2022

The dissertation of Kunyao Chen is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

DEDICATION

To my mom, C.C. and lovely Mimi. Hope to see you soon.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Foremost, I would like to thank my mom, Fang Yang. Her constant loving support and encouragement assist me in walking through hurdles in life. She respects my decisions even it means she has to carry more burdens on her own. I have no chance to freely chase my dreams without her.

Thank you, my dear C.C. and Mimi. You are the great source of my happiness.

I would like to express my deepest gratitude towards my advisor, Professor Truong Nguyen who cultivated an environment in which I was offered precious opportunities to explore my own interest and build up my theory. When I struggle with over-ambitious ideas, he guided me in a way that not only corrects my misunderstanding but also shows me a perfect paradigm of research. His diligence and devotion to work and education also influence me in a profound way.

Special credit goes to all the people who helped in my projects. I was so lucky to mentor these intelligent and hardworking junior students. Bang Du helped to realize the overall capture system. Fei Yin was deeply involved in many parts of the algorithms and devoted much of his time. Baichuan Wu paid a lot of effort to craft our demonstration models as well as render the final results to the industrial level.

I also feel privileged that I could work with so many colleagues in the video processing lab, Menglin Zeng, Igor Fedorov, Byeongkeun Kang, Ji Dai, Chen Du, Junkang Zhang, Yiqian Wang, and Shiwei Jin. I got so much inspiration from your work and the seminar presentations.

I would like to pay my regards to Professors Cheolhong An, Pemela C. Cosman, Falko Kuester, Ryan Kastner, who serve as my committee members. I appreciate your advice for my research.

Lastly, I am thankful of all my close friends Bentao Zhang, Keming Cao, Wenchuan Wei, Jun Wang, Xinyuan Wang, Lutong Wang, Xueshi Hou and Yi Liu.

Chapter 2, in part, is a reprint of the material as it appears in the paper: K. Chen, C. An and T. Nguyen, “Screen-space Regularization on Differentiable Rasterization”, *2020 International*

Conference on 3D Vision (3DV), Fukuoka, Japan, 2020 pp. 220-229. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, is a reprint of the material as it appears in the paper: K. Chen, F. Yin, B. Wu, B. Du, and T. Nguyen, "Mesh Completion With Virtual Scans", *IEEE International Conference on Image Processing (ICIP)*, 2021: 3303-3307. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, is a reprint of the material as it appears in the paper: K. Chen, F. Yin, B. Wu, B. Du, and T. Nguyen, "Efficient Registration for Human Surfaces via Isometric Regularization on Embedded Deformation", submitted to *IEEE Transactions on Visualization and Computer Graphics(TVCG)*, 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 5, in part, is a reprint of the material as it appears in the paper: K. Chen, B. Li, Y. Ye, and T. Nguyen, "FaceFusion: Towards Lightweight High-fidelity 4D Faces", submitted to *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. The dissertation author was the primary investigator and author of this paper.

VITA

2014	B. S. in Electronic Engineering, TsingHua University
2014-2016	M. S. in Electrical Engineering (Signal and Image Processing), University of California San Diego
2016-2018	Founder of VR startup, OWLII, Beijing
2019-2022	Ph. D. in Electrical Engineering (Signal and Image Processing), University of California San Diego

PUBLICATIONS

K. Chen, S. Tripathi, Y. Hwang and T.Nguyen, "Moving objects detection using classifying object proposals for driver assistance system", *2016 International SoC Design Conference (ISOCC)*, 2016, pp. 177-178, doi: 10.1109/ISOCC.2016.7799839.

S. Liang, X. Huang, X. Meng, K. Chen, L.G. Shapiro and I. Kemelmacher-Shlizerman, "Video to fully automatic 3d hair model", *ACM Transactions on Graphics (TOG)*, 7(6), 1-14.

Y. Zhao, Y. Lu, Z. Wen, Y. Xu, X. Jiang, P. Zhao, K. Chen, inventors; Beijing Dajia Internet Information Technology Co Ltd, assignee. "Processing 3d video content", *United States patent application*, US 16/630,427. 2020 Jul 2.

Z. Wen, J. Li, Y. Zhao, Y. Lu, Y. Xu, Z. Feng, L. Wang, K. Chen, P. Zhao, X. Jiang, inventors; Beijing Dajia Internet Information Technology Co Ltd, assignee. "Processing holographic videos", *United States patent application*, US 17/061,406. 2021 Apr 8.

K. Chen, C. An and T. Nguyen, "Screen-space Regularization on Differentiable Rasterization", *2020 International Conference on 3D Vision (3DV)*, Fukuoka, Japan, 2020 pp. 220-229.

K. Chen, F. Yin, B. Wu, B. Du, and T. Nguyen, "Mesh Completion With Virtual Scans", *IEEE International Conference on Image Processing (ICIP)*, 2021: 3303-3307.

K. Chen, F. Yin, B. Wu, B. Du, and T. Nguyen, "Efficient Registration for Human Surfaces via Isometric Regularization on Embedded Deformation", *IEEE Transactions on Visualization and Computer Graphics(TVCG)*, 2021, submitted.

K. Chen, B. Li, S. Wang, Y. Ye, and T. Nguyen, "FaceFusion: Towards Lightweight High-fidelity 4D Faces", *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022, submitted.

ABSTRACT OF THE DISSERTATION

Dense Human Surface Reconstruction with Lightweight Multi-view System

by

Kunyao Chen

Doctor of Philosophy in Electrical Engineering (Signal and Image Processing)

University of California San Diego, 2022

Professor Truong Nguyen, Chair

Digital humans play the leading roles in every aspect of the virtual world. In recent years, multi-view reconstruction is emerging as a cutting-edge technique to generate human models. It reveals the potential to restore intricate surfaces for a wide range of applications requiring more than minimalist avatars. Even with many promising results, the related systems are still cumbersome with many requirements on devices and environment. The capture often has to be performed in a large-scale studio with dozens of HD camera rigs in order to preserve the surface details as well as mitigate artifacts introduced by occlusion.

Our research aims to simplify the multi-view system and bring it to wider application scenarios. This goal requires a significant upgrade to the state-of-the-art algorithms in order to

address gap-region surface and large-displacement registration challenges introduced by reducing input coverage and curbing computing capability. We tackle these challenges by leveraging learning-based prior knowledge on 3D human geometries as well as a more robust deformation framework.

Firstly, we explore learning-based 3D reasoning methods. The current end-to-end framework shows the capability to generate arbitrary 3D shapes with even monocular inputs. We propose a novel back-propagation method for differentiable rasterizer that improves the accuracy and decreases the complexity significantly.

With the foundation of reliable surface inference, in the next step, we investigate on completing the gap-region surface. We realize this purpose in an innovative way that transfers the time-consuming 3D procedures into a simple 2D image-domain processing. It can work compatibly with the traditional multi-view depth fusion method.

After that, given a complete key model, we work on fusing the surface information from other frames in the sequence to further enhance the details. Based on the state-of-the-art method, we boost the robustness of the embedded deformation in the large-displacement cases when the models have significant pose and appearance differences. We achieve this with a novel adaptive regularization and an efficient two-step optimization.

Finally, we design a new framework specifically for human head reconstruction. Instead of sculpturing the complicated geometries of faces, we focus on generating HD textures and fusing input sequences with a geometry-aware texture stitching framework. The results preserve high resolutions details and can be executed in real-time.

Chapter 1

Introduction

3D human digitization has been drawing huge attention in recent years. While computer-aided-designed modeling is long applied in the movies and video games industry, the latest advancements in delicacy and fidelity astonish viewers every now and then. With the support of modern rendering devices, some of the work (as in Figure 1.1) has blurred the boundaries between generation and acquisition, achieving the so-called photorealistic phase. Their success evokes the sense that a digital replica of the real world might be just around the corner.

Figure 1.1: In NVIDIA 2021 GTC keynote, whole kitchen background and several frames of the speaker was fabricated with advanced modeling and rendering techniques, which was such a success blindfold that no one noticed until the company revealed the details itself.

When people are ambitiously heading towards metaverse, we would like to reconsider the mature human digitization techniques and argue that the approaches to capture and craft 3D human models are still far from satisfying for future applications. One line of existing solutions focuses on generating high-fidelity models without considering efficiency and flexibility. They adopt very complicated procedures involving numerous capture devices and extensive labor work to model every detail of the 3D characters and then use motion capture systems or AI-based tracking methods to restore human movements. However, it is impossible to apply these industrial solutions to consumer-oriented scenarios in the XR era, where content creators will no longer be limited to professional and skillful artists, and users may not have access to complex procedures and overnight processing for a fine-grained model. On the contrary, the other line of solutions can achieve controllable 3D avatars even with mobile devices. However, these methods are either over-simplified into cartoon styles or based on certain parametric models, which are often limited in shape variations, failing to broadly express personal characteristics, subtle body languages, and some intricate contents such as loose cloth and fancy hairstyles.

Figure 1.2: Off-the-shelf lightweight 3D human digitization solutions. Left: Horizon Avatars from Meta. Right: parametric models from SMPL-X.

In this context, we aim to build an efficient system that is capable of creating temporal-consistent, visual-appealing, and detail-enhanced 3D human geometries. Instead of tracking pose with invasive devices or estimating the parameters for the underlying models, we focus on directly

generating ready-to-use dense surfaces with multi-view reconstruction methods. We constrain the system setup with four consumer-level cameras in combination with a single computing unit, which is an acceptable upgrade to most current VR/AR hardware packages. Based on this lightweight design, we proposed several algorithms to guarantee reasonable and complete model generation with limited resources. To better explain the proposed methods, several basic concepts of multi-view reconstruction are summarized in the following section.

1.1 Multi-view Reconstruction

1.1.1 TSDF Representation

In this dissertation, multi-view reconstruction specifically refers to a branch of methods that integrates multi-source images from multiple viewpoints into a 3D representation. We designate depth images as our main source of input. The general consideration is depth images provide more explicit spatial information compared with conventional RGB images. In practice, they can help us significantly reduce the number of capture devices required for reconstruction. Another noteworthy point is, due to the emergence of low-cost IR sensors, acquiring HD indoor depth is possible and affordable.

Most recent solutions to fuse the depth images from different viewpoints can be traced back to KinectFusion [136]. It adopts a 3D volumetric representation denoting as Truncated Signed Distance Function (TSDF). For each voxel in the space, TSDF value indicates the distance from the voxel to the nearest surface. Inside regions and outside regions are separated with positive and negative signs. Besides, the data are truncated to accommodate the fusion logic. TSDF can be further transferred into mesh with marching cube [82] or directly rendered into image with ray tracing techniques [42].

1.1.2 Warping Field Estimation

The essential step to generate TSDF is to map various co-related 2.5D depth back into a single 3D volume. For static objects, the mapping function (usually specified as a four-by-four matrix in homogeneous coordinates) is uniformed across the whole space given a specific viewpoint. However, for dynamic objects, integrating depth from different frames is far more challenging. In this scenario, the mapping function is not uniformed. A warping field has to be estimated for variation of the geometries.

DynamicFusion [91] calculates a down-sampled version of the 6D warping field based on a canonical model. A series of works [29, 98, 28] depict an innovative telepresence system. Unlike DynamicFusion[91], they abandon non-robust camera pose estimation with several synchronized cameras at fixed positions. Yet with a similar fusion idea, they progressively merge all TSDF data in the past short-time interval into a complete model and improve efficiency and accuracy on warping field estimation via embedded deformation framework [115]. Although the system achieves real-time performance, they are not robust to the hard footages with fast body motion and large shape changes. In addition, the method still cannot provide a complete model if there is a region uncaptured in the whole sequence.

1.1.3 Challenges

The aforementioned telepresence method requires more than eight GPUs and a PC cluster to satisfy the data collection and processing for a complete reconstruction with dozens of surrounding cameras. There is still a huge gap to our purpose.

However, reducing the input coverage and computing power usually introduce significant challenges for the current algorithms. With severe occlusion and unseen areas, the results often lose track of large non-trivial region. It will continuously aggravate the non-robustness of the warping field estimation. Besides, in real-time application, restrained computing capability leads

Figure 1.3: Hardware synchronization board and calibration blocks.

to lower processing rate, which will introduce another large-displacement challenges to the state-of-the-art deformation based estimation method.

1.2 Multi-view System Setup

In our multi-view capture prototype, we leverage Intel Realsense D415s to collect both RGB and IR images with 720p resolution in 30 fps. We arrange three Realsense cameras in an outside-in setup. All of the six cameras are hardware synchronized with less than 1ms latency. We also expand IO bandwidth to support real-time data transmission.

We build a novel calibration tool sticking with ChArUco [36] patterns to simplify the calibration process, as shown in Figure 1.3. The calibration follows the method in [147]. We initially calibrate the cameras pairwisely and then use the bundle adjustment [128] method to globally optimize on all the cameras.

1.3 Overview

The algorithms we proposed to enhance the reconstruction in the lightweight multi-view system will be discussed in the remainder of this dissertation.

In Chapter 2, we investigate an approach to improve the bottleneck module in the unsuper-

vised 3D reasoning framework, i.e. differentiable rasterization. While some recently published global-gradient methods achieve superior results, they sacrifice much efficiency in the original local-gradient methods. In our study, we find one substantial fact that has great influence on the model appearance, the series of 2D supervision in the dataset may not fully represent the underlying 3D object. To directly tackle this problem, we propose a screen-space regularization method. Unlike the common geometric regularization, our method targets the unbalanced deformation due to the limited viewpoints. We show that our method shares some high-level ideas with the global-gradient methods in terms of gradient passing, but our design is more efficient. By applying the regularization on both multi-view deformation and single-view reconstruction tasks, the results illustrate that the proposed method can significantly enhance the visual appearance, i.e. reducing the visual hull redundancy.

In Chapter 3, we address the mesh completion problem by proposing an efficient method to fill the gap regions on surfaces via leveraging the templates inferred from the 3D reasoning method. We first segment both source and template models into corresponding parts. Each part will be aligned with non-rigid deformation. We then modify the gap regions by "virtual" depth maps rendered using the aligned parts from newly selected viewpoints. The algorithm is evaluated through experiments on a synthetic dataset.

In Chapter 4, we propose a novel solution that improves embedded deformation by decoupling regularization from the underlying deformation model via explicitly managing the rigidity of surface vertex clusters. We further design an efficient two-step optimization solution that alternates between isometric deformation and embedded deformation with cluster-based regularization. Our method can easily support region-adaptive regularization with cluster refinement and execute in real-time. Extensive experiments demonstrate the effectiveness of our approach for mesh alignment tasks even under large-scale deformation and imperfect data.

In Chapter 5, we explore a pipeline to generate temporal-consistent and high-fidelity 3D face models. Firstly, we utilize the state-of-the-art real-time single image face reconstruction

network to estimate the parameters of a 3DMM model, thus obtaining a rough face mesh. Secondly, we generate texture with a simple projection-based method in order to preserve the input resolution to a larger extent and keep the overall procedure efficient. After that, the accumulated texture of previous frames and texture of the current frame are fed into a stitching network that is capable of fusing two incomplete textures and mitigating alignment artifacts. In the meanwhile, the source image and the textured mesh are passed to another geometry-refinement network, adjusting the surface at vertex level for better photo-consistency. In the end, by progressively blending images from different viewing angles, we obtain a refined surface with a complete texture map containing high-quality details in all regions.

Chapter 6 provides conclusion and discussion on future work.

Chapter 2

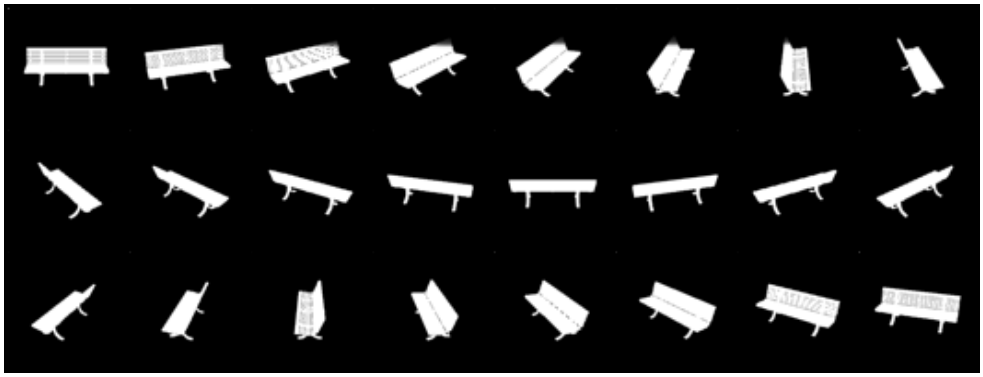
Screen-space Regularization on Differentiable Rasterization

2.1 Introduction

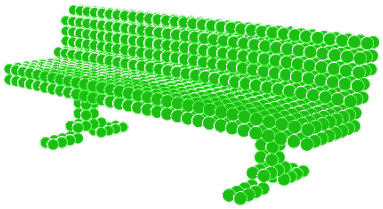
Establishing the mapping from the 2D images to its underlying 3D scene is the essential problem in computer vision, as our brains can easily transfer visual information to perceive the rich 3D environment[135] but machines can not. Moreover, our brains process many vision tasks in a combined manner with both 2D and 3D properties instead of purely 2D features[44].

In recent years, with great success in both 2D and 3D problems separately[103, 102], researchers apply deep learning [64] to those 2D-3D mixed tasks [105, 57, 149, 148, 19, 20, 32, 46]. Most of the work adopt the unsupervised methods, as the supervised methods require large amount of highly accurate 3D ground truth data. Consequently, these works are limited to synthetic dataset[17, 138, 112] consisting of computer-aided design (CAD) models.

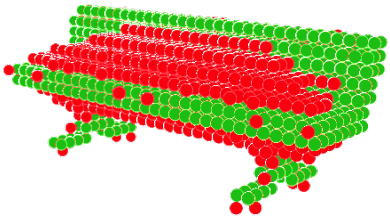
The basic idea of the unsupervised methods is rendering 3D output back to 2D images and matching with the input. Hence, the key approach is designing a differentiable renderer [19, 26, 39, 58, 61, 77, 81, 92, 129, 100] and applying them to one of the two 3D representation:



(a) Silhouettes in 24 views



(b) Ground truth voxels



(c) Reconstructed voxels

Figure 2.1: Example showing that images in the dataset are not enough to fully represent the 3D model. The error is commonly demonstrated as the visual hull redundancy on the top or bottom of the 3D shape. As shown in (c), green points denote the ground truth voxels, red points denote the redundant voxels in the reconstructed result.

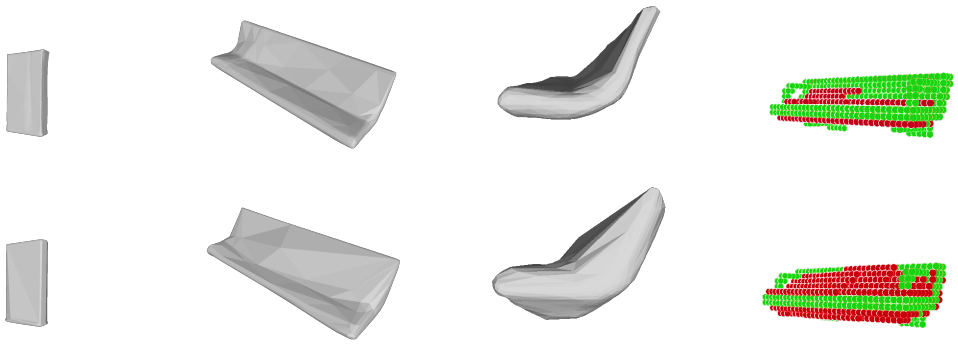
mesh or voxels.

We focus on the mesh representation, despite the fact that mesh-based methods[81, 39, 58, 77, 19, 48] are not as flexible as volume-based methods due to a fixed template. Comparing to the current emerging implicit representation[87, 95], the mesh-based method has an advantage: it allows more straightforward regularization on the geometry.

The previous work on mesh-based differentiable renderer (rasterizer) can be divided into two categories depending on the gradient approximation approach in the back-propagation step. For the local-gradient methods[81, 39, 48], the pixels within the close neighbourhood are calculated to approximate the gradient of a vertex projected on the image. On the other hand, the global-gradient methods[58, 77, 19] use farther pixels, even all of the pixels in the image into

computation. Although the global-gradient methods achieve more satisfying results, they largely increase temporal and spatial complexity.

In fact, one issue existing in the image-based 3D reconstruction but yet seldom analysed is that in the current dataset there is a gap between input 2D images and the ground truth 3D models. Specifically, input images cannot fully represent the shapes of the 3D models. As shown in Figure 2.1, the surface generated with traditional multi-view reconstruction method shape form silhouette method [1, 118] using 24 views is subject to visual hull redundancy on the top and bottom. There are large amount of ambiguous regions, any shapes reconstructed in the region are all satisfied with the 2D inputs. This issue can be addressed with higher resolution images from extra views. However, in realistic application, without 3D data, we are not guaranteed to gather all the 2D information we need.



(a) Concave surface (b) Viewpoint0 (c) Viewpoint1 (d) Voxels

Figure 2.2: Comparison of results using different types of methods; Upper row demonstrates the reconstruction with the global-gradient method, lower row demonstrates the reconstruction with the local-gradient method; (d) shows the results in voxels, green points represent the ground truth, red points represent the redundancy.

In our experiment, we find that this space ambiguity has great influence on local-gradient methods but has less impact on the global-gradient methods. As shown in Figure 2.2, for the learning-based single-view reconstruction task, both types of methods generate good results on the simple concave surface, while the local-gradient method produces significant number of

bumps on the complicated convex surface. Moreover, for those convex surfaces, although the inferred results are acceptable projecting from the specific 24 viewpoints trained during iterations, they have unnatural look from other arbitrary viewpoints. In the following sections, we will show that these unnatural visual hull redundancies come from the unbalanced deformation around the silhouette boundary. The global-gradient methods suffer less from this issue as they deform the vertices on the boundary diversely, and they also deform vertices inside the silhouette along with the vertices on the boundary.

In this chapter, we address the unbalanced deformation problem of the local gradient method by exploring the density inequality of the sparsity map in the 2D space. We propose a screen-space regularization, and show that it significantly enhances the visual appearance of the local-gradient method and removes the visual hull redundancy. The proposed method is effective in both multi-view deformation task and learning-based single-view reconstruction task. Comparing with the state-of-the-art global-gradient methods, our approach achieves better subjective and slightly better objective results. Since the proposed approach is based on the local-gradient method, it has lower temporal and spatial complexity, as well as better compatibility to the modern rendering pipeline.

2.2 Related Work

Differentiable Rasterization. Inversing the rendering process has been investigated for many years [101]. In the early days, as limited by the hardware, most of the approaches [143, 101, 41, 86] only involve optimizing on a few specific parameters of the model.

OpenDR [81] is the first general differentiable rasterizer in the sense that it provides derivatives for most of the parameters in the imaging system and allows optimization towards image evidences. For derivatives approximation, it applies first-order Taylor expansion, and only computes within a close neighbourhood in the image space. Later on, many researchers further

extend the idea by adding additional complicated physics or lighting model, making it faster and compatible with modern rendering pipeline. Recently, as deep learning emerges, the works in [39, 48, 123, 26, 92, 39] applied the idea to the deep learning framework. Although the systems are more complicated nowadays, the fundamental gradient approximation method is unchanged.

Neural 3D Mesh Renderer(NMR) [58] is the first proposed differentiable rasterizer for the learning-based single-view reconstruction task, which shows promising result. Instead of using local-gradient approximation, NMR accumulates all the gradients from pixels along X and Y axis separately. SoftRas [77] embeds an aggregation function in the forward step which connects each pixel value to all of the faces on the mesh. It achieves top performance.

One obvious shortcoming of global-gradient method is that it is not compatible with common rendering pipeline and also requires higher temporal and spatial complexity. SoftRas [77] has $O(MN)$ temporal complexity in both forward and backward steps, (M denotes the number of pixels in the image, and N denotes the nunber of faces on the mesh.) which is larger than $O(CN)$ of a common rasterizer, where C denotes the number of pixels around each face (on average, $C \ll M$ [96]). In addition, global-gradient methods dramatically change the position of the vertices that are already inside the silhouette. There is no guarantee on the robustness of this inner deformation. For the case in which a 3D surface has already matched with the 2D silhouette, adding extra pixels noise outside the boundary will introduce the ill-deformation globally on the whole surface. In comparison, the impact is only limited to the local region, for local-gradient method.

Note that in this chapter, we only focus on the most challenging part, i.e., the surface reconstruction from 2D binary silhouettes, and other lighting and textures estimation can be easily added to our framework.

Geometric Regularization. Many approaches have been proposed to improve the visual appearance of a surface. Traditional Laplacian term [119, 25] is widely used to smooth and denoise the mesh. There are some variants using cotangent formula [130] or filtering on higher-

order features [30]. Recently, more researches [79, 45, 55] focus on regularization on the implicit volumetric representation.

Unlike all of these geometric regularization which are based on the topology priors, the proposed screen-space regularization targets at the specific issue on image-based deformation process. It regularizes on the unbalanced deformation in the 3D space due to the limited input viewpoints.

2.3 Proposed Method

2.3.1 Screen-space Sparsity

The key insight comes from the observation that in the natural-shape mesh reconstructed with the global-gradient method, faces are relatively equally distributed in the projected image. On the contrary, in the ill-shape mesh generated by the local-gradient method, a large number of faces are squeezed near the silhouette boundary, as shown in Figure 2.3. We quantify the screen-space sparsity as a variable of each pixel accumulating the number of faces within distance d_{sp} (unit in pixel). This variable can be easily calculated along with rasterization process, and does not add more complexity to the current rendering pipeline.

The extreme unequal distribution of the results can be explained with unbalanced deformation on the 3D surface due to the limited viewpoints. During the deformation, the unsupervised 2D loss (specifically defined in the equation (2.7)) drags the vertices lying outside the silhouette towards the boundary. However, this force disappears once the vertices reach the boundary. In 2D perspective, vertices pile up on the silhouette boundary if there is no fix from the other viewpoints. In 3D perspective, this issue will result in the visual hull redundancy.

In addition, if the surface cannot completely fill the silhouette, then the 2D loss will push the vertices outward. In this case, as in the local-gradient method 2D loss only impacts on the boundary pixels, a few faces are involved in the reconstruction. In 3D perspective, this may result

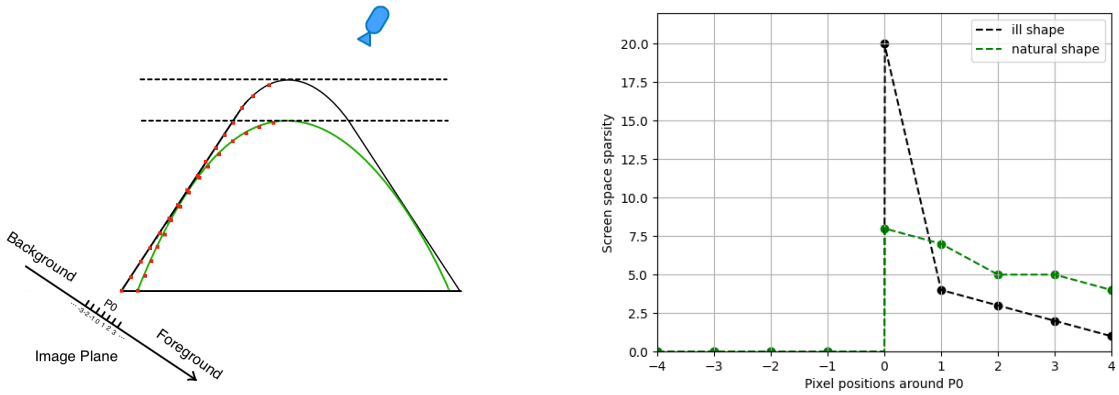


Figure 2.3: A demonstration on how screen-space sparsity regularizes the shape. The left part shows both ill shape (black) and natural shape (green). Red dots represent the vertices on this projection view. The right part shows the screen-space sparsity value around the boundary pixel of both ill shape (black) and natural shape (green).

in losing high curvature geometry details.

2.3.2 Screen-space Regularization

We regularize on the screen-space sparsity $S(\boldsymbol{p};\boldsymbol{V})$ defined in the previous section, where $\boldsymbol{p} \in \mathbb{R}^2$ denotes the pixel position, and $\boldsymbol{V}$ represents the set of all the vertices positions. The regularization loss can be formulated with:

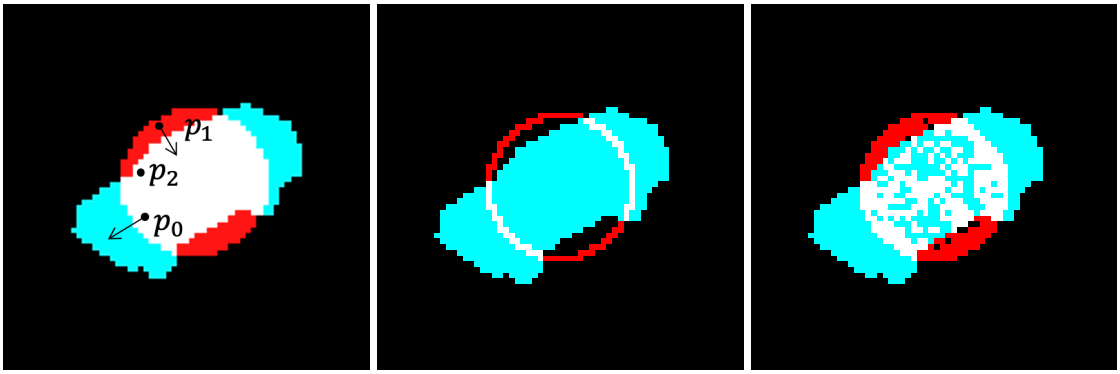
$$L_{sp} = \sum_{(i,j)\in\mathcal{N}} K(S(\boldsymbol{p_i};\boldsymbol{V}), S(\boldsymbol{p_j};\boldsymbol{V})) \tag{2.1}$$

$\mathcal{N}$ is the set of all the neighbouring pixel pairs. The kernel function K is defined as:

$$K(x,y) = \begin{cases} |x-y|, & c_0 \le |x-y| \le c_1, \\ 0, & otherwise \end{cases} \tag{2.2}$$

where c_0 and c_1 are the hyper-parameters to control the regularization.

As shown in Figure 2.4, in the local-gradient method, the unsupervised 2D loss only forces vertices projected on the boundary pixels $\boldsymbol{p_0}$ and $\boldsymbol{p_1}$ to move towards the silhouette. However, the inner vertices without connection to boundary vertices (for example the one projected on $\boldsymbol{p_2}$) cannot contribute to reduce the loss, and it remains static in the position. With the regularization, the deformation of $\boldsymbol{p_0}$ will decrease the surrounding sparsity while the deformation of $\boldsymbol{p_1}$ will increase the surrounding sparsity. To increase sparsity uniformity, $\boldsymbol{p_2}$ is forced to deform according to the movement of $\boldsymbol{p_0}$ and $\boldsymbol{p_1}$.



(a) (b) (c)

Figure 2.4: Demonstration on the effect of our proposed regularization. The target silhouette is rendered in cyan. In (a), red channel shows the projected silhouette of the surface; In (b) and (c), red channel only consists of the pixels need to be deformed. (b) represents the local-gradient method and (c) represents the local-gradient method with the regularization.

Note that the proposed method does not make the sparsity map entirely equal but to robustly deform the inner vertices. Therefore, in the kernel function, we design the lower cut-off value c_0 which allows a certain flexibility and reduce the regularization sensitivity to the minor changes on the boundary. Moreover, the upper cut-off value c_1 is designed for the case where two noncontinuous 3D surfaces are projected close to each other.

As the derivative of sparsity function $S(\boldsymbol{p_i}; \boldsymbol{V})$ with respect to each vertex position $\boldsymbol{v_i}$ is not straightforward, we simplify the gradient computation as:

$$\frac{\partial S(\boldsymbol{p_i};\boldsymbol{V})}{\partial \boldsymbol{v_i}} \approx -\frac{\partial S(\boldsymbol{p_i}|\boldsymbol{V})}{\partial \boldsymbol{p_i}} \cdot \frac{\partial \boldsymbol{p_i}}{\partial \boldsymbol{v_i}} \tag{2.3}$$

where $S(\boldsymbol{p_i}|\boldsymbol{V})$ denotes the static sparsity map where all vertices' positions are fixed. The negative sign before the approximation is used to indicate that we deform in the inverse direction of the gradients of the sparsity map $\frac{\partial S(\boldsymbol{p_i}|\boldsymbol{V})}{\partial \boldsymbol{p_i}}$. Moreover, the term $\frac{\partial \boldsymbol{p_i}}{\partial \boldsymbol{v_i}}$ is also required when computing the gradient of the 2D photo-consistency loss. It can be obtained with the local-gradient method [81].

2.3.3 Comparison with the global-gradient method

Intuitively, the regularization works as an internal stress to the deformed model in the 2D space. Vertices lie around the denser zone of the sparsity map will face against the force from inner vertices to reduce the further movement, while relatively the ones around the sparser zone will deform more.

In the state-of-the-art global-gradient method [77, 19], pixel value of the silhouette image I_i is computed with the aggregation function:

$$I_i = 1 - \prod_j (1 - D_j^i) \tag{2.4}$$

D_j^i a sigmoid-like function of the shortest distance d from the pixel p_i to one edge of the face f_j. It ranges within $[0, 1]$.

The partial derivative of pixel intensity I_i with respect to the distance of a specific face f_{j^*} equals to:

$$\frac{\partial I_i}{\partial d_{j^*}} = \prod_{j \neq j^*} (1 - D_j^i) \cdot \frac{\partial D_{j^*}^i}{\partial d_{j^*}} \tag{2.5}$$

For most of the pixels far away from the face f_{j^*}, $d \to \infty, D_j^i \to 0$. Therefore, equation (2.5) can be approximated with only the faces closed to pixel p_i. Let N be the number of these faces, where

N in most cases is more vital than the exact value of D^i_j. If we assume $\tau = 1 - D^i_j$ is a constant number, then equation (2.5) can be simplified as:

$$\frac{\partial I_i}{\partial d_{j^*}} \approx \tau^N \cdot \frac{\partial D^i_{j^*}}{\partial d_{j^*}} \tag{2.6}$$

The term τ^N is similar to our definition on screen-space sparsity. Comparing with equation (2.6) which uses the sparsity as the weight to control the gradient flow, we explicitly use it as the regularization in equation (2.1).

In addition, with the regularization, vertices lie inside target silhouette will deform along with the vertices on the boundary. Unlike the global-gradient method in which inner vertices directly respond to the 2D loss, our proposed regularization deforms those vertices in a passive way. It's more robust and can generate smoother surfaces.

2.4 Experiments

To validate the effectiveness of our approach, we apply our regularization method to both multi-view deformation task and learning-based single-view reconstruction task. Two tasks share the same rasterization framework, loss combination and dataset. In multi-view deformation task, we focus more on the accuracy and efficiency of the different frameworks, while in the learning-based task, we focus more on the generality of the frameworks among a variety of shapes.

2.4.1 Local-gradient Method Implementation

In our local-gradient differentiable rasterization, we adopt gradient approximation on the close neighbourhood similar to [81]. Unlike the original work, we build all the operations on GPU and follow the logic of the modern rendering pipeline. Besides, we propagate the gradient to

all of the fragments, the fundamental elements in the fragment shader, instead of just propagating to the vertices on the surface.

2.4.2 Losses

Following the previous work [58, 77, 19], we adopt the intersection over union (IoU) as 2D loss to evaluate the silhouette prediction. For our regularization method, we additionally apply the screen-space sparsity regularization loss defined in (2.1). Moreover, to generate smooth and appealing surfaces, we impose geometric Laplacian loss and flattening loss.

IoU Loss

$$L_{IoU} = 1 - \frac{||S \odot \widetilde{S}||_1}{||S \oplus \widetilde{S} - S \odot \widetilde{S}||_1} \tag{2.7}$$

where S and $\widetilde{S}$ denotes the binary silhouette mask of input and output; $\odot$ is element-wise product; $\oplus$ is element-wise addition.

Laplacian Loss is a penalty on the distance of a vertices v_i to the center of mass of its 1-ring neighbours $\mathcal{N}_i$:

$$L_{lap} = \sum ||\delta_i||_2^2 \tag{2.8}$$

where $\delta_i = v_i - \frac{1}{||\mathcal{N}_i||} \sum_{j \in \mathcal{N}_i} v_j$

Flatten Loss is a penalty on angles θ_i between two adjacent faces:

$$L_{fl} = \sum (\cos \theta_i + 1)^2 \tag{2.9}$$

The overall loss combines all the three particular losses with one regularization loss only

for our proposed method:

$$L = L_{IoU} + \lambda_{lap} L_{lap} + \lambda_{fl} L_{fl} + \lambda_{sp} L_{sp} \tag{2.10}$$

2.4.3 Datasets

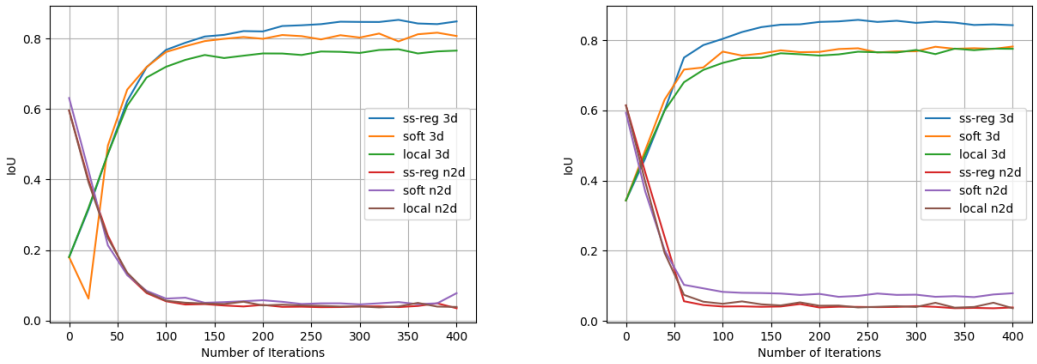
We use the models from ShapeNet Core dataset [17] which consists of 13 object categories. We take the same train/test split and render the object in the same 24 views with 64×64 resolution as in [77] except that we turn off the anti-aliasing function during the rendering process.

2.4.4 Settings

In our experiments, we set $\lambda_{sp} = 0.1$, $\lambda_{lap} = 1 \times 10^{-5}$ and $\lambda_{fl} = 1 \times 10^{-6}$. We optimize with Adam optimizer [59] with $\alpha = 0.0001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The batch size is 64. A good setting of hyper-parameters in our regularization method is $d_{sp} = 1$, $c_0 = 7$ and $c_1 = 30$. We will discuss the parameters in the ablation study section.

2.5 Multi-view Deformation

In the multi-view deformation task, we directly pass the gradient from different viewpoints to deform vertices of the template. We fix the template as a standard sphere. However, we deliberately choose the radius of the sphere to perform two subtasks: shrinkage and extension. In the realistic application, the mesh is deformed in the manner mixed with the two types. Here, we separate the task for clearer demonstration. In the shrinkage subtask, we aim at the performance on reducing the visual hull redundancy. In the extension subtask, we aim at the performance on the robustness of the deformation, as the erroneous deformation can be fixed and hidden during shrinkage.



(a) Shrinkage (b) Extension

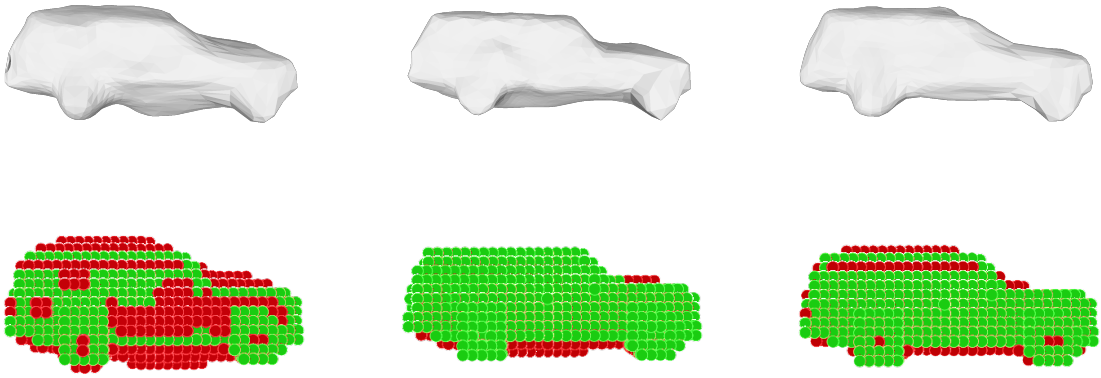
Figure 2.5: Optimization curve: 3D and negative 2D IoU v.s. number of iterations.

2.5.1 Shrinkage Task

As shown in Figure 2.5, we compare three methods: local-gradient method, global-gradient method (SoftRas [77]) and the proposed method: local-gradient method plus the screen-space regularization (refer to as SS-Reg) with IoU metrics in both 2D and 3D cases. For the shrinkage subtask shown on the left, as we directly optimize on the 2D IoU, all of the three methods converge at same level (around 95.5%). However, the results on 3D IoU diverge: local-gradient method converges at 77.2%, SoftRas converges at 81.7% while SS-Reg can reach 84.3%.

SS-Reg also achieves better visual appearance. In Figure 2.6, the result from local-gradient method has visual hull redundancy all around the surface. The result from SoftRas still leaves redundancy at the bottom. The difference between the reconstructed volume from SS-Reg and the ground truth volume is only one layer which is inevitably caused by the limited input resolution.

Figure 2.7 shows the heatmap from different viewpoints after convergence. The heatmap of SS-Reg is smoother comparing to the other two methods. It shows the effectiveness of the proposed regularization.



(a) Local-gradient (b) SoftRas (c) SS-Reg

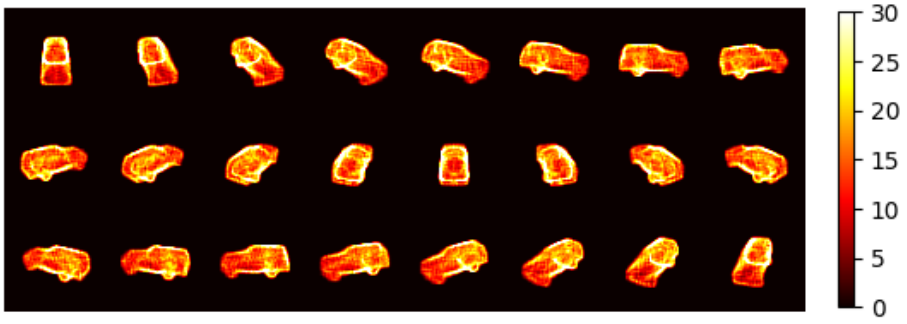
Figure 2.6: Mesh and voxels reconstructed with three methods.

2.5.2 Extension Task

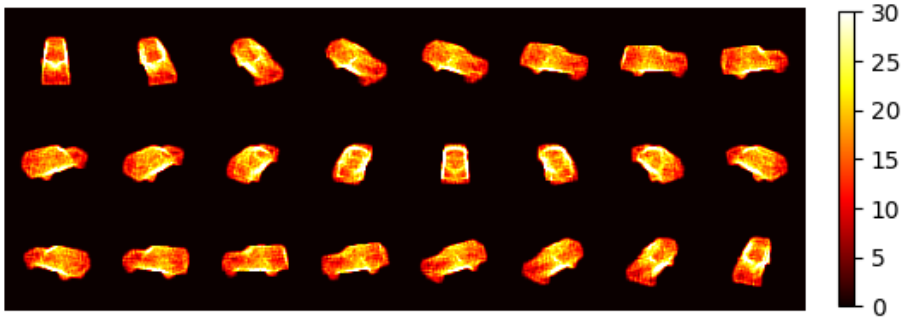
For the extension subtask shown on the left side of Figure 2.5, the performances of local-gradient method and SS-Reg are quite stable. Local-gradient method converges at 77.6% and SS-Reg converges at 84.2%. However, the result of SoftRas slips down to 78.2% (2D IoU is also lower as ill-shape surface prevents it from fully convergence). As mentioned in the previous discussion, global-gradient method is subject to erroneous deformation of vertices inside the silhouette. Without correction from the other views, global-gradient method cannot generate satisfying results, as shown in Figure 2.8. Note that SS-Reg method does not have this issue.

2.5.3 General Deformation

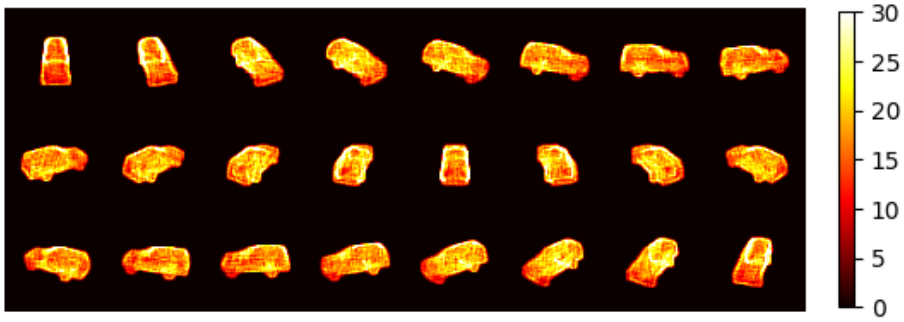
We further test the algorithms on more shapes. We randomly select 500 samples from five sub-classes in ShapeNet Core dataset [17], 100 samples each. The remaining categories are either too simple, (such as phone and display), or too complex involving large topology changes which can hardly be transformed with deformation method (such as bench and table). We use middle-size sphere as the template, therefore the deformation consists of both shrinkage and



(a) Local-gradient



(b) SoftRas



(c) SS-Reg

Figure 2.7: Heatmap of the sparsity value from different viewpoints. Truncated in $[0, 30]$ for comparison

(a) Local-gradient (b) SoftRas (c) SS-Reg

Figure 2.8: Meshes reconstructed in the extension subtask.

extension.

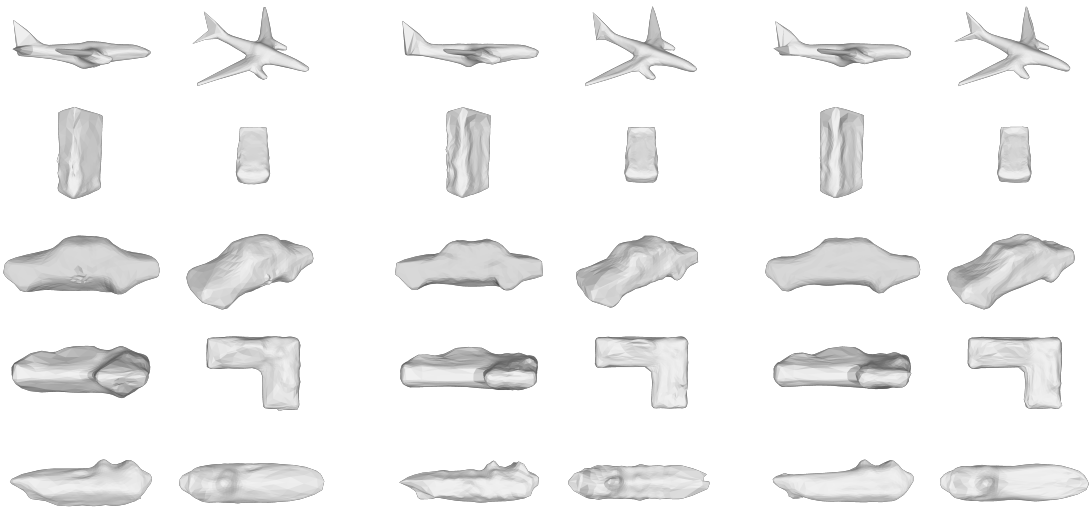
Table 2.1: Muti-views deformation results on five categories. Number shows in percentage of 3D IoU. Bold-faced numbers indicate the better results in SS-Reg.

Category	Plane	Cabinet	Car	Sofa	Boat	Mean
Local	69.2	68.9	65.2	62.2	63.9	65.9
SoftRas	70.6	72.1	70.9	70.5	67.7	70.3
SS-Reg	**71.4**	**72.7**	**73.5**	69.2	**69.1**	**71.2**

As shown in Table 2.1, the proposed SS-Reg significantly improves the results of local-gradient method. Comparing to the state-of-the-art method, SS-Reg achieves better numerical results in 4 out of 5 classes. Moreover, SS-Reg successfully removes the visual hull redundancy shown on the meshes in Figure 2.9.

2.5.4 Complexity

Testing with 1080TI GPU, SoftRas takes 0.56s on average for each iteration comparing to 0.15s for SS-Reg. In other words, SS-Reg is approximately four times faster than SoftRas in the deformation task. With higher resolution input 128×128, SoftRas takes 2.07s for each iteration while SS-Reg only need 0.32s, a factor of 6 in speed improvement.

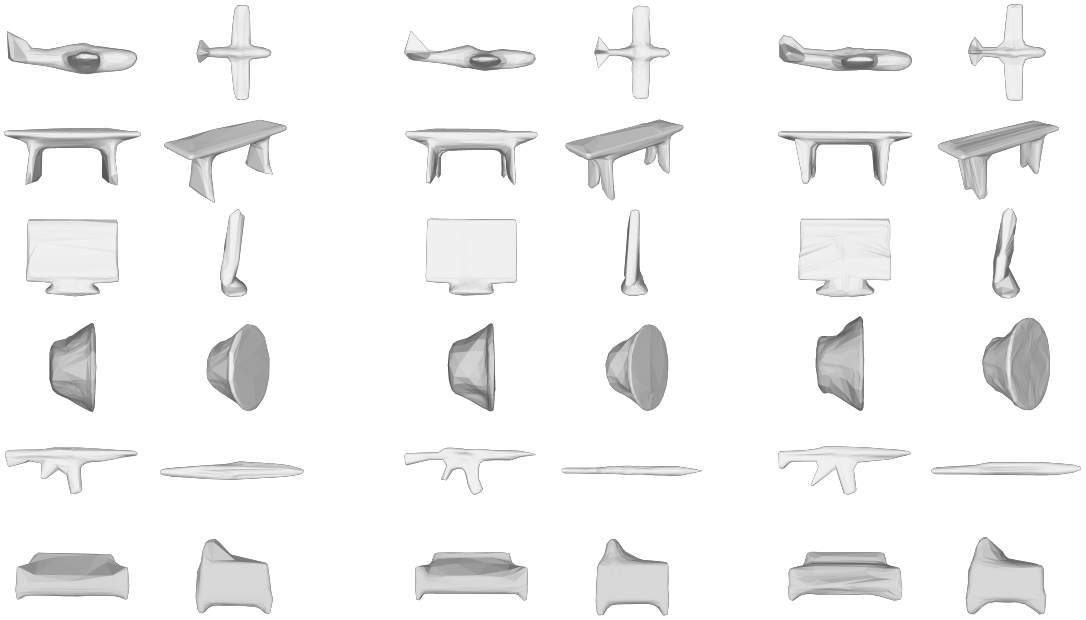


(a) Local-gradient (b) SoftRas (c) SS-Reg

Figure 2.9: Meshes of five categories reconstructed with multi-view deformation rendered in two different viewpoints

Table 2.2: Single-view reconstruction results on all categories. Number shows in percentage of 3D IoU. Bold-faced numbers indicate the better results in SS-Reg.

Category	Plane	Bench	Cabinet	Car	Chair
Local	60.4	42.1	66.7	62.3	46.9
SoftRas	62.2	47.1	68.9	71.0	51.8
SS-Reg	61.5	45.8	**69.2**	**73.3**	**52.3**
Category	Display	Lamp	Speaker	Rifle	Sofa
Local	59.7	43.2	63.6	59.6	61.8
SoftRas	61.0	44.7	65.0	65.9	68.2
SS-Reg	60.3	**45.8**	64.6	64.0	67.9
Category	Table	Phone	Boat		Mean
Local	39.4	74.0	56.5		56.6
SoftRas	45.1	79.7	60.2		60.8
SS-Reg	**47.3**	**81.1**	**61.5**		**61.1**



(a) Local-gradient (b) SoftRas (c) SS-Reg

Figure 2.10: Meshes of six less accurate categories reconstructed with single-view reconstruction rendered in two different viewpoints

2.6 Single-view Reconstruction

We also apply our regularization method to learning-based single-view reconstruction task. As optimized with all of the models together, it is a good test on the method's flexibility and robustness. For a fair comparison, we employ the identical neural network structure as in the previous work [58, 77]. The network infers the 3D vertices displacement from a single RGB image. The surface is reconstructed by adding the vertices displacement to the corresponding vertices of the template sphere mesh. As in multi-view deformation task, we render the deformed mesh with three different methods and train in unsupervised way with the loss formulated in equation (2.10).

As shown on Table 2.2, the proposed SS-Reg is also effective in single-view reconstruction task. We greatly improve the result of basic local-gradient method, and out-perform the state-of-the-art method in 7 out of 13 categories. Figure 2.10 shows the reconstructed meshes in the other

6 categories. The visual appearance is appealing, although it may not precisely match the ground truth.

2.7 Ablation Study

SS-Reg has three hyper-parameters: d_{sp}, c_0 and c_1. We fix $d_{sp} = 1$ for 64×64 input resolution in all of our experiments as we find that larger value of d_{sp} reduces the regularization effect. c_1 is the upper cut-off threshold for the kernel function of regularization function to avoid the incorrect regularization across uncontinuous surfaces. Figure 2.11 left shows the failure case of double wings if c_1 is too large. The results are not so sensitive to the exact value of c_1, and from our experience, any value within $(15, 30)$ is effective. c_0 is the lower cut-off threshold. Smaller c_0 value results in over-regularization shown as the depression on the surface (Figure 2.11 right). The ideal value lies within the range of $(5, 10)$ for the dataset.

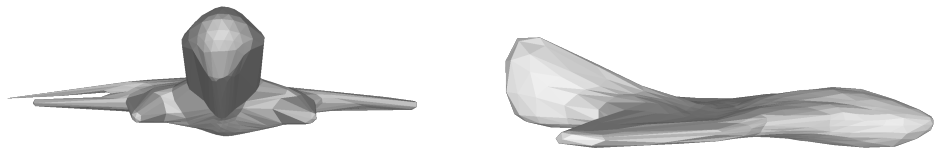


Figure 2.11: Failure cases with bad hyper-parameters.

2.8 Conclusion

We present our work on screen-space regularization. Unlike the common geometric regularization, the proposed method addresses the unequal deformation issue in local-gradient differentiable rasterization. The result shows that the proposed regularization is effective and

significantly enhances the performance. It also achieves better results comparing to the state-of-the-art global-gradient method. Moreover, the proposed method has lower complexity.

There are a few improvements that can be made for the proposed approach. Firstly, there could be a more adaptive way to handle the hyper-parameter, especially c_0. Since we approximate the gradient of sparsity during computation, it can only guarantee the correct deformation direction but not accurate value. Moreover, it is unclear if there is better form to regularize the sparsity, or better definition on the sparsity itself. We plan to investigate these research questions in our future research.

Chapter 2, in part, is a reprint of the material as it appears in the paper: K. Chen, C. An and T. Nguyen, "Screen-space Regularization on Differentiable Rasterization", *2020 International Conference on 3D Vision (3DV)*, Fukuoka, Japan, 2020 pp. 220-229. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Mesh Completion With Virtual Scans

3.1 Introduction

The emergence of consumer-level range scanners and depth cameras enables artists to create virtual contents at lower cost. Many commercial software is available to merge scans from multiple views into 3D models through image-based reconstruction algorithms. Unfortunately, these methods are inevitably subject to occlusion appearing as complex holes on the reconstructed surface (see Fig.3.1). The problem is more severe in the applications requiring fewer camera coverage such as holoportation [98] and fast 3D portrait [71].

Traditional mesh completion methods [72, 66] often fail in the above scenario as they simply connect the gaps using basic shapes without considering the real contents. Advanced template-based completion methods [4, 60] attempt to tackle this issue by replacing the hole regions with the equivalent parts in some standard models. However, to make the results appear natural, templates and the corresponding parts have to be chosen carefully, which often involves extensive manual work. Moreover, these methods are often time-consuming.

Some of the state-of-the-art incremental reconstruction methods [29, 28] make use of temporal information, fusing all volumetric data in the frame sequence into a complete model.

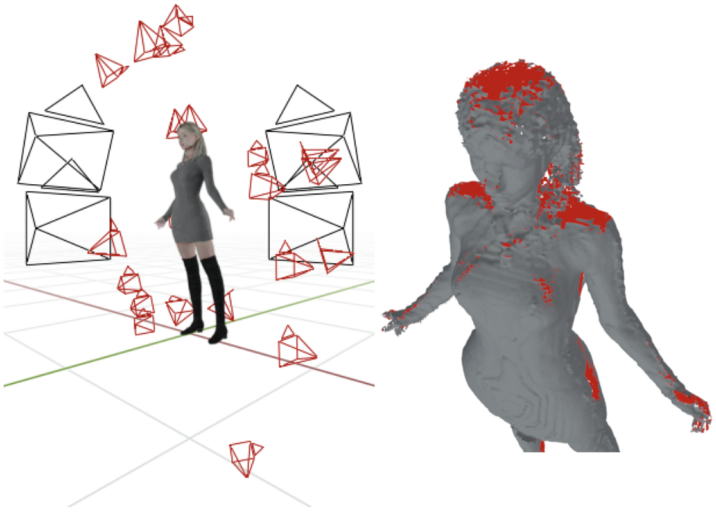


Figure 3.1: Camera rig of the capture system (black). The reconstructed mesh has complex holes, which are shown in red on the surface. We set up virtual cameras (red square pyramids) to fill in the depth information of the gap regions thereby completing the surface.

With the help of modern GPU, these methods can speed up to the real-time level. However, they cannot fill the space uncaptured in the whole sequence.

In recent years, learning-based reconstruction methods [2, 88] achieved promising breakthroughs. They are capable of inferring complete models with data from fewer viewpoints. In the state-of-the-art works [107, 106], through implicit function learning, pixel-aligned surface details can be inferred and reconstructed. The algorithms provide high-quality templates resembling the scanned models.

In this chapter, we propose an effective method to complete scanned meshes by fusing the templates generated from the learning-based method. We follow the basic idea in the incremental volumetric fusion reconstruction approaches [29, 28]. However, we found that blending the whole template volume data will degrade the quality of the source mesh, because the inferred model cannot always preserve high curvature details. Therefore, we design a novel selective fusion pipeline that modifies specifically on the gap regions. The insight is that we can complete the mesh by simply providing additional depth information of the hole regions. To accomplish this,

we render, using Blender, new depth images of the aligned template model from the optimized angles covering the hole regions on the source model. Merging these depth images as "virtual" scans, we generate the complete mesh by a second pass through the reconstruction pipeline. Our method transforms some crucial 3D computations in the traditional template-based mesh completion pipeline, such as filling regions selection, into more efficient 2D processing.

To address the potential occlusion issue in our method and obtain a better completion result, we also segment the mesh into different parts before the template fusion step as in the traditional template-based mesh completion methods. Note that the segmented parts are more rigid, leading to simpler and faster deformation and alignment.

3.2 Method

3.2.1 Overview

As shown in Fig.3.2, the inputs of our algorithm are two meshes: an incomplete source mesh generated from multi-view depth images, and a water-tight template mesh inferred by the learning-based method. Moreover, as a post-processing step of the image-based reconstruction framework, we assume that the input images and camera parameters are given. Our proposed system fills the gap regions on the source mesh by performing the following steps sequentially:

- We segment both source and template mesh into multiple regular parts;
- We align all the template parts towards the corresponding source parts with a few iterations of non-rigid ICP [83];
- We render depth images from new angles covering the gap regions with the aligned template surface;
- We complete the mesh by running the depth merging reconstruction pipeline again to fuse

new depth into the model.

The details will be elaborated in the following sections. Note that, although our experiment focuses on human models, we expect the idea to generalize well with other contents, but further experimentation is needed.

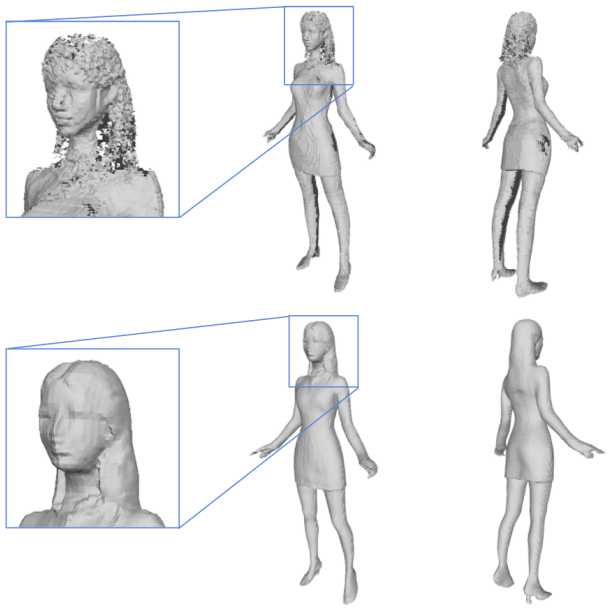


Figure 3.2: Source mesh (top) and template mesh (bottom).

3.2.2 Segmentation

To robustly produce regular 3D parts, we build our mesh segmentation based on the state-of-the-art 2D semantic segmentation method [74]. We back-project the segmentation results on multiple views to assign labels for all vertices on the meshes.

The projection-based assignment is occasionally problematic with inter-class boundaries, where minor pixel bias leads to the ghosting effect. Therefore, we shrink the area around boundaries of the 2D segmentation results, as shown in Fig.3.3. Note that this operation also makes our segmentation more consistent between views.

Thanks to the state-of-the-art pixel aligned learning-based surface estimation network [29, 28], the 2D segmentation images have a strong correlation with both source and template meshes. To further mitigate artifacts, after back-projection, we apply a smoothing operator to diffuse projected regions along the mesh topology. We iteratively apply the smoothing operator until the projected regions fully cover the desired corresponding mesh regions. The intermediate projection-segmentation results are demonstrated in Fig.3.3.

Figure 3.3: Projection-based mesh segmentation. Left to right: 2D segmentation, back-projection assignment, smoothing with diffusion, final segmentation.

3.2.3 Virtual Scans

Robust automatic gap region detection is the critical step in selective fusion. With the help of multi-view foreground masks, we can obtain the visual hull surface through the space carving algorithm [62]. Specifically, for the intermediate volumetric data, we only retain the outermost layer of voxels lying within the masks after projection. The gap regions are represented as the remaining voxels occluded in all views. The occluded voxels can be easily determined as the ones without a valid value from the truncated signed distance function (TSDF). The overall detection step can be computed alongside the original surface reconstruction pipeline without spending additional overheads. Note that the voxels detected with the aforementioned method are sparse and cannot be used for 3D selective fusion with volumetric data. An extra time-consuming region

growing method has to be done to obtain all gap voxels accurately. However, we bypass this procedure and apply these sparse points as guidance to set up virtual cameras.

The camera allocation problem can be formulated as follows: for each segmented part, find the minimal number of cameras with their FOVs collectively covering all gap voxels on all parts. In practice, we can loose the condition as our method is not subject to the overlapping of the FOV coverage. Thus, we design a greedy algorithm based on a plane segmentation method in [140]. We add a voxel into a point cluster as it satisfies:

$$|P_v - P_c| < T_a \tag{3.1}$$

$$|(P_v - P_c) \cdot V_c| < T_b \tag{3.2}$$

where P_v and P_c are the positions of a voxel and a cluster, respectively; T_a and T_b are empirically determined thresholds; and V_c is the normal vector of the best-fit plane of a cluster. We continuously group gap voxels with the above conditions until no more voxels can be grouped. After the partition, if the number of clusters exceeds the allowable number of virtual cameras, we further reduce the number of clusters by combining clusters together based on the above two conditions and the condition:

$$cos^{-1} \frac{V_{c1} \cdot V_{c2}}{|V_{c1}||V_{c2}|} < T_c \tag{3.3}$$

limits the angle between two clusters' best-fit planes' normal vectors.

Using V_c and bounding box corner positions of the cluster, we can calculate the virtual camera pose as shown in Fig.3.4. For each virtual camera and its corresponding body part segment, we render the z-depth buffer and a particular back-face mask to identify the non-interest gaps resulting from mesh segmentation. The back-face mask is generated by thresholding the dot product of the view vector with the normal vector of the surface at the fragment shader.

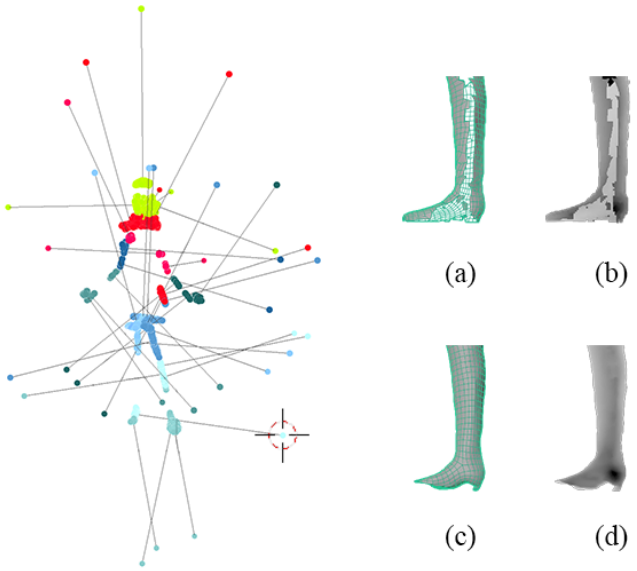


Figure 3.4: Virtual camera poses are shown on the left with colored dots indicating different parts; virtual scans of the right leg are shown on the right: (a) source mesh (b) source depth (c) deformed template mesh (d) deformed template depth.

3.2.4 Virtual Fusion

During the final process, we apply the depth data from the virtual views to accurately fill holes on the source mesh. We pass the virtual depth images to the TSDF-based reconstruction pipeline [29], in which virtual depth images are treated in the same way as real depth images but with one exception: real-view data are prioritized. Namely, we only allow virtual data to add information to voxels that have not been updated by any real camera. In this way, virtual data fill the gap regions without degrading the original detailed surfaces.

3.3 Experiment

3.3.1 Implementation Details

To evaluate our algorithm, we test on four handcrafted synthetic human models. We render the depth images, RGB images, and foreground masks of the models from multiple fixed

angles, simulating real-world multi-view reconstruction scenarios. We modify InfiniTAM [56] to support the fast multi-view depth maps merging and reconstruct the source meshes for completion. Moreover, we feed multi-view RGB images to PIFu [106] to generate the template meshes.

We utilize the pre-trained model from CDCL [74] that segment the human model into 14 body parts and background on 2D images. The segmentation is inferred at scale 1, 0.75, and 0.5 to refine the boundary between each part and generate an indexed image for each real-camera view.

After that, we apply the idea in section 3.2.2 to map multi-view 2D segmentation onto 3D mesh. Then, we deform and align the model by part using Levenberg-Marquardt based non-rigid deformation method in [29]. As the body part is relatively rigid compared with the full body, our algorithm converges faster (on average three iterations).

Lastly, we calculate the virtual camera poses and create virtual scans as discussed in section 3.2.3. We re-run the InfiniTAM software with modification in section 3.2.4 and produce the final complete output.

3.3.2 Results

We measure the mesh quality in bidirectional RMSE (w.r.t the ground truth mesh).

As shown in Table 3.1, our algorithm outperforms the traditional mesh completion method [72] and the full volumetric fusion method for all three models. Fig.3.5, Fig.3.6, and Fig.3.7 display the per-vertex error maps, demonstrating the distance from each vertex on the ground truth mesh to the closest vertex on the result mesh. In (a), the gap regions are highlighted in red on the mesh and displayed as the large-error regions on the error-map. Although the template mesh (b) is poorly inferred, our method robustly generates a cleaner mesh with less error in the gap regions comparing with the other methods.

Surface details are demonstrated in Fig.3.8. In (a) and (b), the model's necklace is well preserved with our method comparing to the over-smoothing result with full fusion method. In

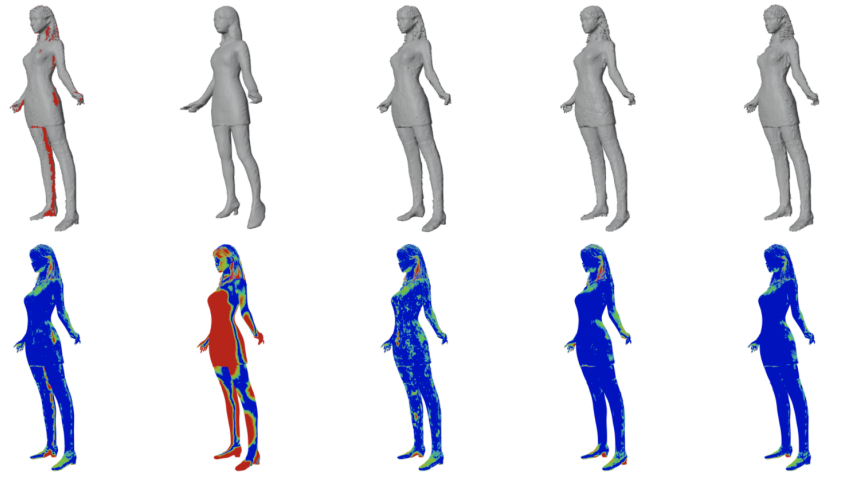


Figure 3.5: Resulting meshes and their per-vertex error maps colorized in three levels: red for large error, green for medium error, blue for minor error. Note that Liepa denotes the results from the traditional hole filling method; Full Fusion denotes the results from volumetric fusion method applying whole template surface to the source; Selective Fusion is our proposed method.

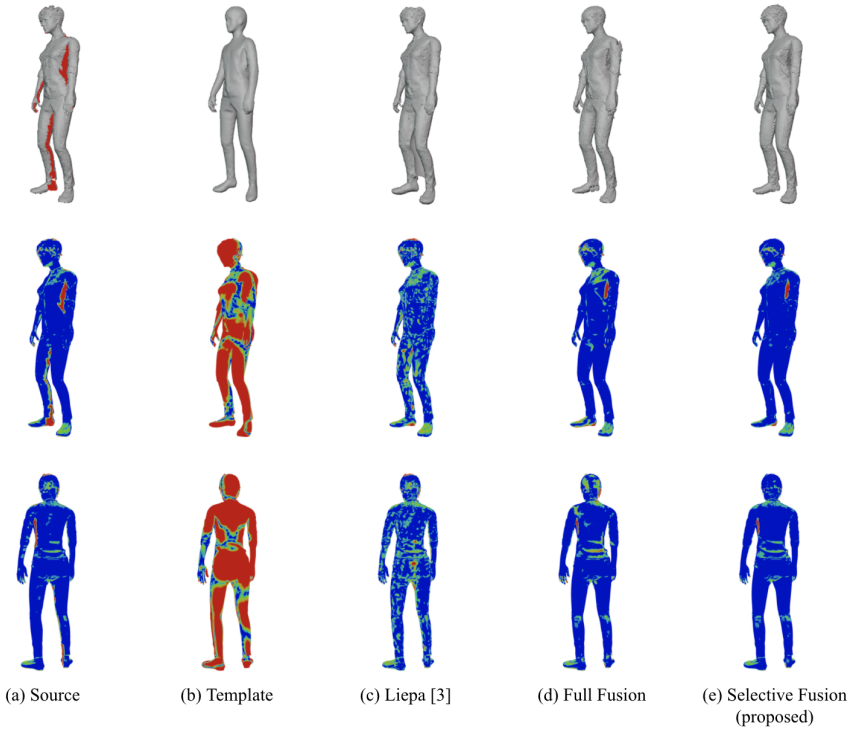


Figure 3.6: Resulting meshes of Kate and per-vertex error maps.

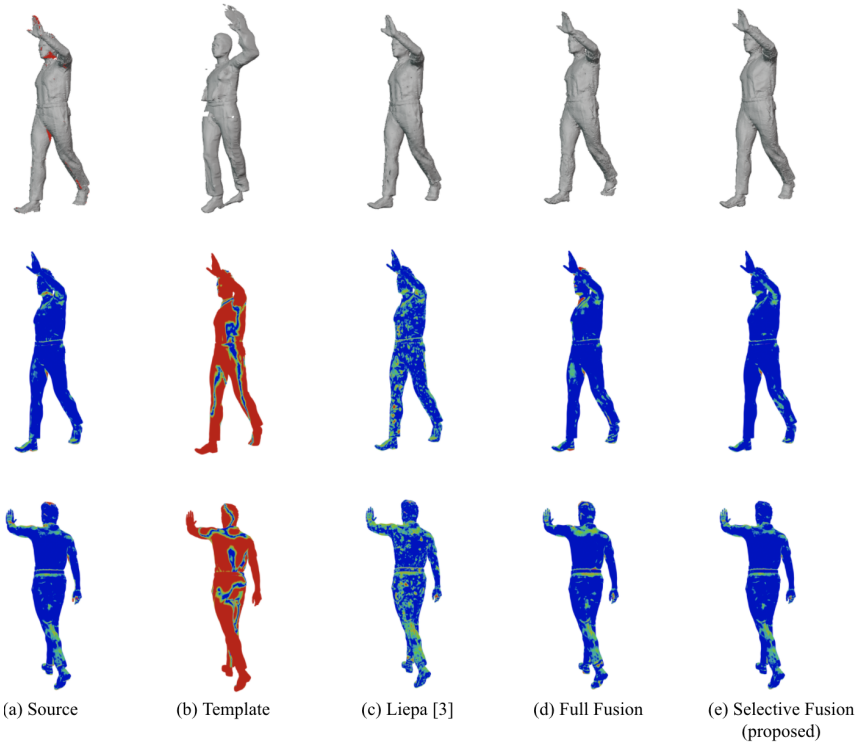


Figure 3.7: Resulting meshes of Yonni and per-vertex error maps.

(c) and (d), our method produces more visually appealing surface for the model's hair and nose region comparing with the traditional mesh completion method [72].

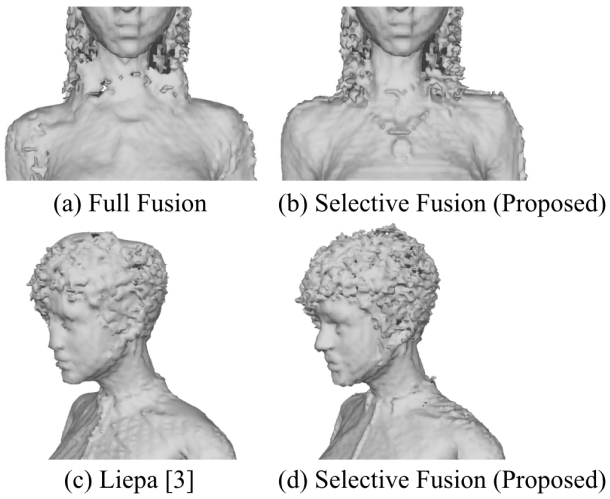
3.4 Conclusion

We present our work on mesh completion with virtual scans. Unlike the traditional template-based mesh completion approaches and the volumetric fusion methods, we apply the learning-based model as the template and design a novel fusion pipeline to selectively blend the inferred surface into the incomplete source mesh. The results show the excellent performance of our algorithms, both visually and numerically.

Chapter 3, in part, is a reprint of the material as it appears in the paper: K. Chen, F. Yin, B. Wu, B. Du, and T. Nguyen, "Mesh Completion With Virtual Scans", *IEEE International Conference on Image Processing (ICIP)*, 2021: 3303-3307. The dissertation author was the

Table 3.1: Bidirectional RMSE (in millimeters) w.r.t the ground truth model.

RMSE	Kate	Kayla	Yonni
Source Mesh	11.9247	12.1793	16.7406
Template Mesh	44.4534	15.7061	145.3785
Liepa [72]	17.7706	13.2849	16.4881
Full Fusion	12.4552	13.1374	16.6880
Selective Fusion	**10.6584**	**12.0903**	**16.2864**



(a) Full Fusion (b) Selective Fusion (Proposed)

(c) Liepa [3] (d) Selective Fusion (Proposed)

Figure 3.8: Comparison of details on resulting meshes.

primary investigator and author of this paper.

Chapter 4

Efficient Registration for Human Surfaces via Isometric Regularization on Embedded Deformation

4.1 Introduction

Immersive 3D digital humans will play the most vital roles in the future metaverse stages. To obtain ready-to-use dynamic human models, surface registration technique is one of the key procedures. The goal of surface registration is to find a mapping that optimally aligns vertices of the source with target models. This type of vertex-wise transformation field can be further applied to produce geometrically uniform mesh sequences for animation and generate temporal-consistent content for 4D videos [22]. Moreover, in recent exciting researches [29][98], real-time holoportation is achieved by progressively refining incomplete volumetric surfaces using per-vertex correspondences obtained with an accelerated implementation of surface registration.

Traditional mesh alignment methods are frequently used to address the registration problem. They are formulated to estimate the deformation field between two surfaces iteratively,

following the Iterative Closest Point (ICP) [9][104]. Due to the limitation that only local correspondences are established, the naive ICP method tends to be trapped into local minima.

A number of approaches [50][14][67] attempt to tackle this issue by providing additional global correspondence on some sparse 3D feature points. Recently, many approaches are proposed to improve the robustness of 3D correspondence with deep learning methods [134][131]. Furthermore, some researches also seek to obtain the registration directly with an end-to-end framework [13]. However, none of these results perform well in human-related scenarios because they are not specifically trained for human models. They can only generate sparse correspondences, and, most importantly, fail to handle large variants on surface details as well as noise and holes introduced by consumer-level 3D scanners. At the application level, learning-based methods adopt very complex framework structures; hence, they cannot execute efficiently for real-time processing.

Another branch of researches tackles the local minimum issue by performing regularization during deformation. The most popular practice is to regularize on local rigidity, such as ARAP [113]. These methods are generally capable of achieving smooth deformation but fail in more challenging scenarios. For human models, it is common that some parts of the surface, such as arms and legs, shown as the red region in Fig. 4.1, must be deformed more rigidly to avoid being collapsed into trivial shapes while other parts of the surface, such as torso and cloth wrinkles, shown as the green region in Fig. 4.1, should be deformed less rigidly in order to align with high-frequency surface details on the target.

The embedded deformation [115] is proposed as an efficient registration solution. The GPU version [29] with an improved implementation is capable of real-time processing. However, the conventional embedded deformation can hardly support the aforementioned region-adaptive regularization, because it does not directly regularize on surface vertices but on a deformation graph designed for acceleration. [67] proposes a method which alters regularization across regions by dynamically modifying the deformation graph in terms of the rigidity loss. This

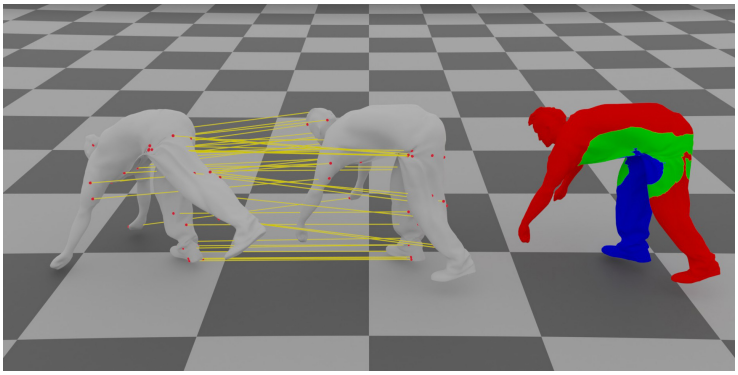


Figure 4.1: Example demonstrating the various rigidity requirements for different surface regions. Left: source mesh. Middle: target mesh. Right: distance map in three level (red: large; green: medium; blue: small).

design is not elegant because the underlying deformation graph may not reflect the true surface condition, especially at the end of the iterations. It is hard to determine the locations, where regularization weights should be adjusted, and the magnitude of those adjustments. [67] relies on a pre-established binary tree structure to double the number of deformation nodes, which is not flexible in practice. Even worse, with respect to implementation, the design is unfriendly to acceleration. Changing the deformation graph, especially the number of EDNodes, often requires re-allocating and re-assigning whole GPU memory for the best performance.

In this chapter, we aim to design an efficient non-rigid registration solution with region-adaptive regularization. We adopt the embedded deformation framework and propose a novel regularization method. We follow the isometric deformation idea in [50] and define the regularization loss on vertex clusters instead of individual vertices. Due to the complexity of optimization, we propose a more efficient two-step solution, alternating between two deformations: an isometric deformation and an embedded deformation with cluster-based isometric regularization. We also progressively refine the member of each cluster guided by the regularization loss in each step. To showcase the effectiveness of our method, we perform extensive experiments on mesh alignment tasks. As current 3D human datasets are mostly captured in high-quality studio environments, we intentionally design various cases to simulate complicated real-world scenarios. The results

reflect the advantage of our method compared with other state-of-the-art.

4.2 Related Works

Non-rigid registration has been a well-explored topic for decades in the fields of computer graphics and image processing. One of the well-known approaches is the non-rigid Iterative Closest Point (ICP) method [9][5], in which correspondences are based solely on local proximity, and transformation is formulated by a general deformation. The traditional ICP algorithm is intuitive and simple, but highly dependent on the initial condition and can easily get trapped into local minima. To resolve this issue, Robust Point Matching (RPM) [43] is proposed, which combines deterministic annealing and soft-assign optimization to convexify the objective function. [21] improves RPM using thin-plate splines (TPS) in non-rigid registration. On the other hand, [89] exploits properties of kinematics to avoid computing pair-wise correspondences, but is slower and less robust to general non-rigid motion.

Correspondence. Many researches improve the reliability of correspondences. [114, 5, 90] use sparse human-labeled landmarks to relax the demand of high quality correspondences. [84] proposes an L_2E estimator to build better dense correspondences. Recently, deep learning techniques[145, 134, 131] have shown good performance in finding reliable matches. However, [120, 145] provides only sparse correspondences while [117, 134, 131] generate dense correspondences but either require additional complex operations or the assistance of powerful GPUs to infer in a reasonable amount of time.

Deformation Model. The choice of deformation models embodies the assumptions made on the deformation. Assuming isometry, [14] applies diffusion distance and Gromov-Hausdorff distance for non-rigid shape matching. [50] applies an additional pruning procedure to the closest-point correspondences using the geodesic distance. It also classifies components with similar transformations into a rigid cluster, which reduces the optimization complexity. However, the

cluster growing mechanism has higher complexity compared with our approach. Representing points with Gaussian mixture models (GMM), [53] treats the registration as minimizing the distance measure between two distributions. [54][75][139] use the deformation of vertices on a coarse cage mesh template to manipulate the source. It keeps the original surface structures and details but the cage often fails to model subtle changes due to small degrees of freedom. [115] encodes the mesh into an abstract deformation graph, for which a deformation is parameterized by a collection of affine transformations on the nodes. It decouples the complexity of the deformation from that of the geometry, preserving local details in an efficient optimization process. [68] extends the method with a unified optimization that simultaneously solves for point correspondences, confidence weights, and a warp field that aligns the source with the target. [146] casts 3D surface registration into a graph matching problem. [29] adopts [115] and designs an optimization framework achieving real-time alignment performance.

Regularization. Due to high degrees of freedom, the deformation model can easily represent unreasonable deformations. Various regularization terms are proposed to improve the robustness of the resultant meshes following the human intuition of deformation. [5] includes a term in the cost function that requires the template to be stiff. As-rigid-as-possible energy[113] is introduced to preserve the local geometry through minimizing the transformation deviation from neighboring vertices. [76][21] regulate the deformation smoothness through differential coordinates (first-order) and the TPS transformation (second-order). [114] designs terms for the affine transformation of each triangle to ensure the smoothness while avoiding drastic changes to the shape. [90] guides close displacement vectors to point to similar directions through analyzing the low-frequency component of the field. [68] designs a regularization term to penalize the hole regions caused by occlusion. [49] constructs a statistical shape model and a noise model that detect outliers based on their sparsity and incorporates the models into the classic regularization terms. [84] introduces the manifold regularization to the point set registration problem.

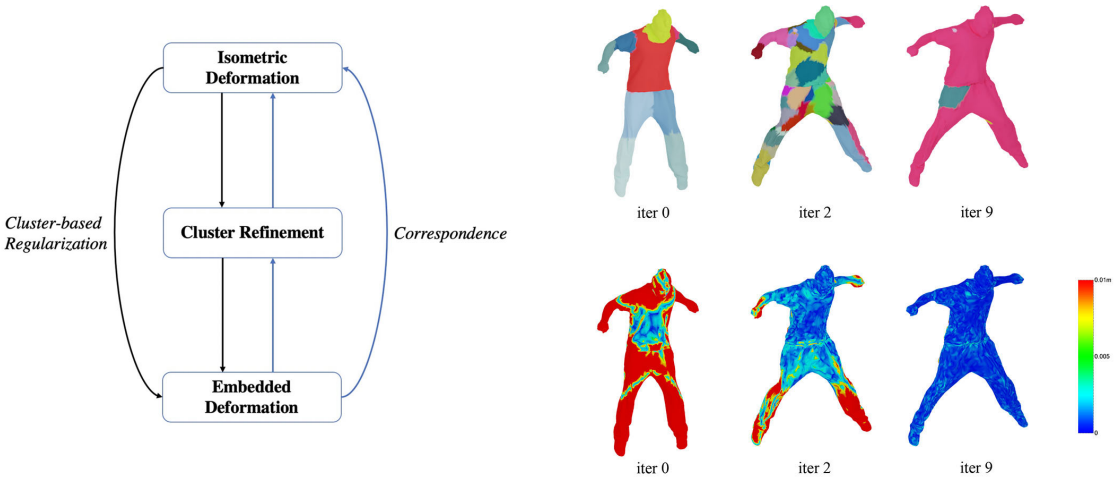


Figure 4.2: Overview of the alignment procedure. Region-adaptive regularization is achieved with cluster refinement shown in the upper right corner. Cluster patches are colorized differently. Error reduction through iterations is demonstrated in the lower right corner.

4.3 Method

4.3.1 Overview

Fig. 4.2 shows the overview of our proposed approach. The source mesh (shown as iter 0) progressively deforms towards the target surface (shown as iter 9) with region-adaptive rigidity and local correspondences. For the two-step optimization strategy, one isometric deformation is performed to specify the regularization target for each cluster. With the help of this regularization, the non-rigid embedded deformation warps the model. Then new local correspondences are generated for the transformed vertices, which are applied to guide both isometric deformation and embedded deformation in the next iteration. Moreover, clusters are refined in each iteration according to the rigidity loss from each of the two steps with a simple but efficient implementation. With tolerable amount of overhead involved beyond the embedded deformation, the whole solution can be executed in real-time.

In Sections 4.3.2 and 4.3.3, we describe the formulation of the mesh alignment task and the

baseline approach i.e. embedded deformation. Next, we highlight the proposed region-adaptive regularization, two-step optimization, and cluster refinement in Sections 4.3.4-4.3.6. Furthermore, the experiment and implementation details are discussed in Section 4.4. At last, both qualitative and quantitative results are demonstrated in Section 4.5, followed by the conclusion in Section 4.6.

4.3.2 Problem Formulation

Let $\mathcal{S} = \{\mathcal{V}, \mathcal{N}\}$ be the source surface consisting of vertices $\mathcal{V} = \{x_i \in \mathbb{R}^3\}$ and vertex normals $\mathcal{N} = \{\mathbf{n}_i \in \mathbb{R}^3\}$. Similarly, let $\mathcal{T} = \{\widetilde{\mathcal{V}}, \widetilde{\mathcal{N}}\}$, $\widetilde{\mathcal{V}} = \{\tilde{x}_i\}$, and $\widetilde{\mathcal{N}} = \{\tilde{\mathbf{n}}_i\}$ be the target surface, vertices, and normals, respectively. Conventional mesh alignment aims to compute a warping field aligning two surfaces, which is typically solved via optimization. The basic loss function can be written as:

$$\mathcal{L} = \mathcal{L}_{data} + \alpha_{reg}\mathcal{L}_{reg} \tag{4.1}$$

where the data term $\mathcal{L}_{data}$ penalizes the geometry deviation between source and target. This is conventionally formulated with point-to-point distance and point-to-plane distance:

$$\mathcal{L}_{data} = \sum_i \alpha_{pt}\|\tilde{x}_i^K - x_i^K\|^2 + \alpha_{pl}|\tilde{\mathbf{n}}_i \cdot (\tilde{x}_i^K - x_i^K)|^2 \tag{4.2}$$

For ICP-based methods, the data term sums up the distances of all correspondence pairs, which are gathered by searching for the closest point in each iteration with KDTree [8]. $(\tilde{x}_i^K, x_i^K)$ denotes the i-th correspondence pair in the K-th iteration.

4.3.3 Embedded Deformation

The embedded deformation method represents the 3D warping field with an underlying *deformation graph* $\mathcal{G} = \{\mathcal{V}_g, \mathcal{E}\}$, where EDnodes $\mathcal{V}_g = \{n_j \in \mathbb{R}^3\}$ are the selected anchor

points to control the deformation of a local region, and edges $\mathcal{E}$ between EDnodes are used for regularization, specifying the influence each node has on the shared surface vertices. Typically, EDnodes are gathered by sampling on the source surface. [115] proves that simple uniform-sampling can achieve near-optimal results.

The local deformation of each EDnode is defined as an affine transformation specified by a matrix $A_j \in \mathbb{R}^{3\times3}$ and a translation vector $t_j \in \mathbb{R}^3$. Each vertex is transformed to a new position x_i^K in K-th iteration according to its surrounding EDnodes:

$$x_i^K = \sum_j \overline{w}_{ij}\Big(A_j(x_i - n_j) + t_j + n_j\Big) \tag{4.3}$$

where $\overline{w}_{ij}$ normalizes the influence weight $w(x_i, n_j)$ among the EDNodes controlling the same vertex:

$$w(x_i, n_j) = max(0, (1 - \frac{d^2(x_i, n_j)}{r_i^2})^3) \tag{4.4}$$

r_i is a parameter for the influential radius of each EDNode. In [67], geodesic distance is applied as $d(x_i, n_j)$ to avoid geometric artifacts. However, there is no efficient way to compute geodesic distance. We follow the real-time implementation [29] adopting the Euclidean distance. In addition, in the approach of [67], the deformation graph is modified such that EDNodes' initial positions n_j vary across iterations to better represent the deformed surface. However, changing the number of EDNodes would involve large memory switching cost that is not ideal for GPU computation.

The overall optimization function of the embedded deformation can be written as:

$$\min_{\{A_j, t_j\}} \mathcal{L} = \min_{\{A_j, t_j\}} \mathcal{L}_{data} + \alpha_{reg}\mathcal{L}_{reg} + \alpha_{rot}\mathcal{L}_{rot} \tag{4.5}$$

The data term $\mathcal{L}_{data}$ can be calculated by applying x_i^K in equation (4.3) to equation (4.2) .

The rotation term $\mathcal{L}_{rot}$ measures the deviation of transformation A_j from the orthogonal

matrix:

$$\mathcal{L}_{rot} = \sum_j \Big((a_1^T a_2)^2 + (a_1^T a_3)^2 + (a_2^T a_3)^2 + (1 - a_1^T a_1)^2 + (1 - a_2^T a_2)^2 + (1 - a_3^T a_3)^2 \Big) \quad (4.6)$$

where a_1, a_2, and a_3 are the column vectors of A_j.

The regularization term $\mathcal{L}_{reg}$ ensures the smoothness of the deformation, traditionally defined on the edges of deformation graph $\mathcal{E}$:

$$\mathcal{L}_{reg} = \sum_k \sum_j w_{kj} \| A_j(n_k - n_j) + t_j + n_j - n_k - t_k \|^2 \quad (4.7)$$

w_{kj} can be computed in the same way as equation (4.4).

4.3.4 Cluster-based Regularization

We introduce a cluster-based regularization term inspired by the isometric deformation [50] to replace the original one in equation (4.7).

$$\mathcal{L}_{reg} = \sum_i \| A_{c(i)} x_i + t_{c(i)} - x_i^K \|^2 \quad (4.8)$$

where $c(i)$ denotes the cluster that the vertex i belongs to. Vertices in one cluster share the same cluster-wise rotation $A_{c(i)} \in SO(3)$ and translation $t_{c(i)} \in \mathbb{R}^3$. With new parameters involved, equation (4.3) should be modified accordingly as:

$$x_i^K = \sum_j \overline{w}_{ij} \Big(A_{c(i)} \big(A_j(x_i - n_j) + t_j + n_j \big) + t_{c(i)} \Big) \quad (4.9)$$

The optimization function would be written as:

$$\min_{\substack{\{A_j, t_j\} \cup \\ \{A_{c(i)}, t_{c(i)}, c(i)\}}} \mathcal{L} = \min_{\substack{\{A_j, t_j\} \cup \\ \{A_{c(i)}, t_{c(i)}, c(i)\}}} \mathcal{L}_{data} + \alpha_{reg} \mathcal{L}_{reg} + \alpha_{rot} \mathcal{L}_{rot} \quad (4.10)$$

Regularization demonstrated in equation (4.8) has certain advantages over the original design in equation (4.7): The regularization is explicitly defined on surface vertices; therefore, it captures the surface condition more accurately. This is crucial for a smooth deformation, especially at the end of iterations when EDNodes are not uniformly located. On the other hand, we decouple the proposed regularization and the underlying deformation graph in a way that it eliminates the necessity to modify the deformation graph in order to adjust regularization. Moreover, regularizing on vertex groups enables the control over regional rigidity instead of purely vertex-level local rigidity. It can be further applied to the region-adaptive regularization described in the following section.

4.3.5 Efficient Two-step Solution

Directly optimizing with the regularization in the last section is not easy. Note that equation (4.8) is in a complicated format, involving the new position x_i^K obtained from equation (4.9). Moreover, there is no sufficient solution to optimize the clusters $c(i)$. [50] obtains the clusters with pre-calculated features and complex grouping methods in each iteration, which cannot be accelerated with parallel computing.

We propose a two-step optimization procedure combined with a cluster refinement strategy. Firstly, we invoke an isometric deformation, i.e. each vertex warps rigidly along with its cluster. Each vertex is mapped to a target position $\dot{x}_i^K$, which will be applied to the regularization in the next step:

$$\dot{x}_i^K = A_{c(i)}^K x_i^{K-1} + t_{c(i)}^K \tag{4.11}$$

where $A_{c(i)}^K$ and $t_{c(i)}^K$ represent the transformation of each cluster in the current iteration. Note that, as cluster components $c(i)$ are modified during the process, we compute cluster transformations in an incremental way where parameters are re-initialized in each iteration, measuring the

transformation from iteration K-1 to iteration K. For the general setup, all vertices are assigned to a single cluster in the initial ICP iteration, i.e. invoking a global rigid deformation. Later on, the clusters are separated and combined for an accurate representation of the regional rigidity. For specific deformation models, such as human bodies used in our experiments, we can utilize results from a 3D mesh segmentation network as the initial value for better performance. The optimization function of the K-th iteration can be written as:

$$\min_{\{A^K_{c(i)},t^K_{c(i)}\}} \mathcal{L}_{iso} = \min_{\{A^K_{c(i)},t^K_{c(i)}\}} \mathcal{L}_{data(c)} + \alpha_{reg(c)}\mathcal{L}_{reg(c)} \tag{4.12}$$

The regularization term can be written as:

$$\mathcal{L}_{reg(c)} = \sum_i \sum_j \|A^K_{c(i)}x^{K-1}_i + t^K_{c(i)} - A^K_{c(j)}x^{K-1}_i - t^K_{c(j)}\|^2 \tag{4.13}$$

Equation (4.13) penalizes on the transformation deviation between adjacent vertex pairs (i, j). It is similar to ARAP [113], except that the regularization loss within each cluster is equal to zero. That is, $\mathcal{L}_{reg(c)}$ only calculates the loss on edges across the clusters.

In the second step, we invoke embedded deformation, leveraging the cluster-based transformations obtained from the first step as regularization targets. The optimization function can be written as:

$$\min_{\{A_j,t_j\}} \mathcal{L}_{nrig} = \min_{\{A_j,t_j\}} \mathcal{L}_{data(n)} + \alpha_{reg(n)}\mathcal{L}_{reg(n)} + \alpha_{rot}\mathcal{L}_{rot} \tag{4.14}$$

where vertex x_i is deformed with:

$$x^K_i = \sum_j \overline{w}_{ij}\Big(A_j(x_i - n_j) + t_j + n_j\Big) \tag{4.15}$$

and the regularization term is defined specifically as:

$$\mathcal{L}_{reg(n)} = \sum_i \| A^K_{c(i)} x^{K-1}_i + t^K_{c(i)} - x^K_i \|^2 = \sum_i \| \dot{x}^K_i - x^K_i \|^2 \tag{4.16}$$

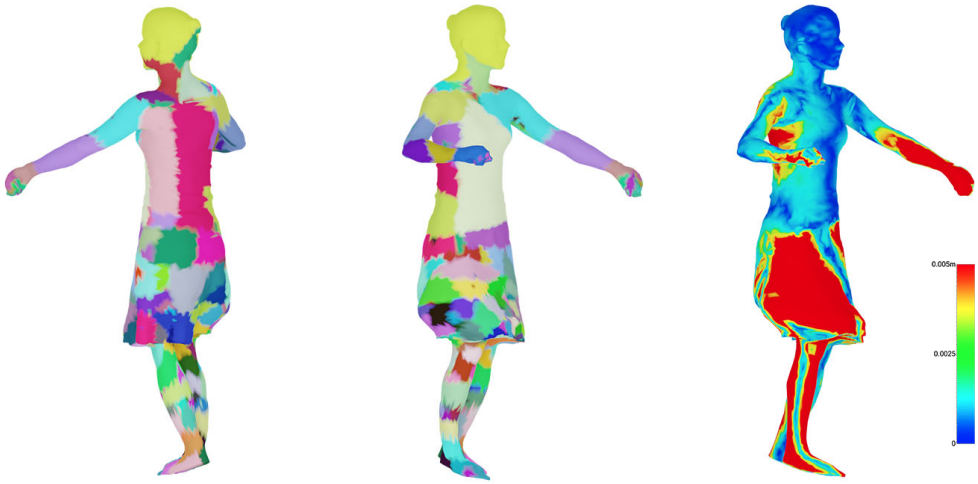
Note that in the second step, cluster-based transformation $A^K_{c(i)}$ and $t^K_{c(i)}$ are treated as fixed values, implying that equation (4.16) is in the point-to-point distance format similar to the data term in equation (4.2). Consequently, the regularization weight $\alpha_{reg(n)}$ is no longer latent, but an exact ratio between non-rigid and rigid deformation, making the regularization process more comprehensible.

4.3.6 Adaptive Cluster Refinement

In order to realize the region adaptive feature, we dynamically change the cluster components during each iteration. Unlike the complicated re-grouping method in [50], we separate and combine clusters from the ones generated in the previous iteration. This separation is managed with non-rigid regularization term $NR_i = \| \dot{x}^K_i - x^K_i \|^2$. Before each non-rigid deformation, we randomly generate sub-clusters by sampling N_c anchors in each cluster and assigning vertices in the cluster to these anchors to form sub-cluster candidates. After optimization, we calculate the separation error E_s of each sub-cluster candidate by averaging the NR_i of all its vertices. The sub-cluster would be labeled as a new cluster if E_s is larger than the threshold T_s.

Similarly, the combination process is controlled by the cluster regularization term $CR_i = \| A^K_{c(i)} x^{K-1}_i + t^K_{c(i)} - A^K_{c(j)} x^{K-1}_i - t^K_{c(j)} \|^2$. The combination error E_c between two clusters is computed by averaging all the CR_i of edges across the cluster boundaries. After each isometric deformation, we merge two clusters if E_c is smaller than the threshold T_c.

Fig. 4.3 demonstrates the cluster distribution of intermediate results during deformation. Rigid regions (shown in blue) are represented by larger clusters while non-rigid regions (shown in red) are represented by smaller clusters.



(a) (b) (c)

Figure 4.3: Examples of intermediate results. (a)(b) Cluster distribution from two viewpoints. (c) Error map of regularization loss.

4.4 Experiments

To validate the performance and effectiveness of the proposed methods, we perform extensive experiments on three elaborately designed cases to simulate the complicated real-world scenarios and compare our method with the embedded deformation, the ARAP-based deformation, and their variants.

4.4.1 Dataset

The experiments are based on the MIT human dataset [132], which contains 10 subsets of meshes. All meshes are in real human scale with heights around 1.8 meters. Each subset records a different action with diverse surface appearances and consists of a sequence with more than 150 frames. The original video is captured at 25FPS, a relatively low frame rate. Some subsets, such as *bouncing, jumping* and *handstand*, are challenging as they involve fast motion.

In our ***general*** test case, we adopt the original frame rate and select 50 consecutive meshes in each subset (500 samples in total). Moreover, to simulate real-world 3D scanned surfaces, we

intentionally generate holes on the 500 meshes to create the ***incomplete*** mesh test case. Lastly, in the ***inaccurate correspondence*** test, we experiment on extreme large-scale deformation cases by selecting meshes 5 frames apart in the *bouncing, handstand* and *marching* subsets (300 samples in total). Under this setting, ICP-based methods usually have to be guided by global correspondences to obtain reasonable results. In our experiment, we leverage the body-part segments as weak correspondences. Note that the segments are also employed as cluster initialization. We compute transformations of corresponding body-part segments and apply them in the first iteration before the closest points searching. Note that comparing with the learning-based registration methods [134, 131], our segmentation-based method provides very sparse correspondences, but in a more efficient way and fits our proposed method well. The weak correspondences are applied to all the methods in the large-scale deformation cases equally for a fair comparison. We also add random noise to these correspondences by slightly altering the transformations to simulate inaccurate correspondences and test the robustness of the methods.

4.4.2 Incomplete Mesh Generation

Occlusions due to limited coverage of physical scans inevitably introduce gap regions on resultant meshes. Such occlusions mainly appear as narrow strips, for example, the image of an arm projected onto the torso. To better re-enact this scenario, we manually introduce discontinuities with controllable shapes on the surfaces in our mesh dataset.

Given a mesh with a set of vertices $\mathcal{V}$. We uniformly sample a few vertices on the mesh and add them to the selection set, then iteratively choose the top-most and bottom-most neighbors of the selection set along a direction (such as the z-axis, as shown in Fig. 4.4) and add them to the selection set. This gives us a vertical line segment $\mathcal{V}_s^{\sigma}$ on the mesh surface after σ iterations. To further expand resultant crease, we iteratively append all the 1-ring neighbors of the currently selected set. This gives us a region masked by vertices $\mathcal{V}_s^{\sigma+\phi}$ after ϕ iterations. We then remove $\mathcal{V}_s^{\sigma+\phi}$ (and their adjacent edges and faces) from our mesh. Fig. 4.4 provides an overview of the

incomplete mesh generation process.

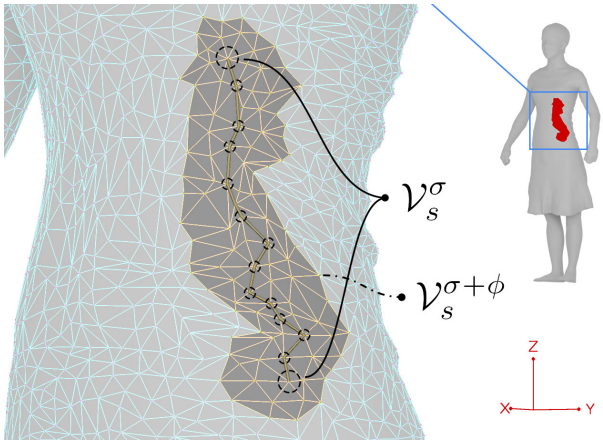


Figure 4.4: Incomplete mesh generation process. V_s^{σ}: Vertices marked with black circles. $V_s^{\sigma+\phi}$: Masked region.

4.4.3 3D Body-part Segmentation

To infer body-part labels in 3D, the majority of the methods require annotated meshes as supervision. However, most public 3D human-related datasets do not provide body-part annotations. Others may contain models with less various surfaces, such as unclothed and hairless models in [63], or have labels without distinguishing left and right [144], which is critical to our experiment as we also require correspondence. Thus, we prepare the annotation for the MIT dataset by ourselves. We generate the ground-truth labeling through 2D projection-based as well as 3D deformation-based methods to avoid the time-consuming manual labeling process and also make the annotation more consistent across meshes.

For the projection-based method, we segment the 2D image [74] from each camera view in the MIT dataset and then back-project the segmentation results using calibrated camera parameters. Such assignment is occasionally problematic with inter-class boundaries, where minor pixel bias leads to the ghosting effect. Therefore, we shrink the area around boundaries of the 2D segmentation results. Afterwards, to tolerate the precision problem of the calibration

information provided, we assign labels to vertices by voting the segmentation results from all views. The resultant mesh may contain unassigned vertices due to the shrinking procedure. We diffuse the labels along the mesh topology until all blank regions are fully covered [116]. This process is illustrated in Fig. 4.5 (a).

Considering challenging cases that the initial 2D body-part segmentation network malfunctions, as shown in Fig. 4.5, we leverage deformation to propagate labels from neighboring frames in the sequence since deformation retains the order of vertices [114]. However, long-term continuous deformation may fail for sequences with fast motion due to various reasons. For instance, as the color labels propagate through a sequence, shifting can accumulate. Moreover, two originally separate body segments attached to each other may change the surface topology of the mesh. When those two segments subsequently separate again, artifacts are generated in the form of erroneous surfaces that attempt to retain the connection between those two segments. Manual relabeling has to be involved in these failure cases.

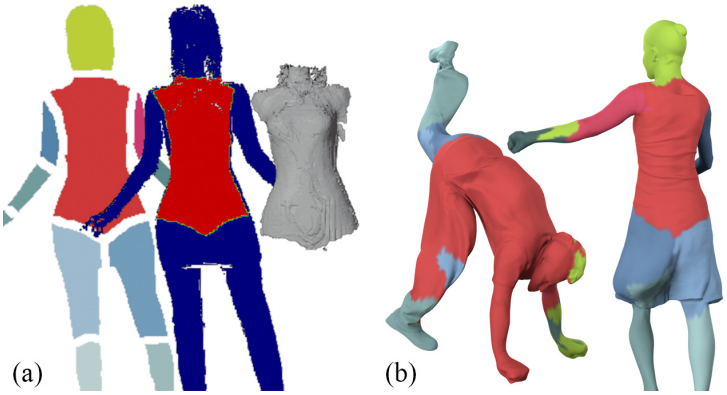


Figure 4.5: (a). Left to right: 2D segmentation, back-projection assignment, final torso segmentation. (b). Failure cases of projection-based segmentation

Both back-projection and label propagation may encounter corresponding compromised cases, which occur for human body motions relatively more often, making the use of neural network a more straight-forward and efficient method. Nevertheless, the network can have a general and scalable architecture for multiple segmentation tasks. It confers to our system the

potential of application beyond human objects and the structure can be further simplified to accelerate the inference to real-time.

We adopt the state-of-the-art KP-FCNN architecture from [124] to train the network with data prepared through the aforementioned workflow. The final instance mIOU on the testset of MIT dataset is 79.8. The visualization of the results from various subsets is shown in Fig. 4.6. Note that the isometric regularization proposed is not bound to the 3D segmentation network we used in this chapter. Any black-box annotation methods can be applied here.

Figure 4.6: Results of our 3D body-part segmentation network on various poses.

4.4.4 Segmentation as Initial Correspondence

Nearest-neighbour-based correspondence from ICP is limited to the locality of a source vertex. In cases where a source vertex and its correct target counterpart are far away and possibly impeded by other vertices in between, ICP alone cannot produce the correct deformation results. To achieve large-scale deformation, we introduce an initial correspondence to the algorithm. Note that we only compute and apply the correspondences in the first iteration, unlike [50] which

requires accurate correspondences in the whole process. The initial correspondence is generated through the network proposed above that segments each human mesh into ten parts.

Given the labels, we first calculate all body segment centers on both the source and target meshes. A body segment center x_c^θ for a segment θ is defined as the center of the segment's bounding box.

Next, we find the principle components for all body segments on both meshes, where the principle component v_{pc}^θ of a body segment θ is the eigenvector associated with the 2nd largest eigenvalue of the covariance matrix of all vertices in that segment. Since a principle component can have two orientations, to eliminate this ambiguity, we also introduce a reference vector for each segment. A segment's reference vector v_r^θ is defined as the vector pointing from the segment center x_c^θ to the average position of the vertices x_r^θ that bound another specific segment:

$$v_r^\theta = (\frac{1}{N_r}\sum x_r^\theta) - x_c^\theta \tag{4.17}$$

The right image in Fig. 4.7 exemplifies the set of reference vectors for the mesh on the left. Since each reference vector points to a unique direction for a given mesh, we disambiguate the orientation of the principle components by comparing them to their corresponding reference vectors. That is, we define a principle component v_{pc}^θ to be correctly oriented if the angle between it and its corresponding reference vector v_r^θ is less than 90 degrees by checking whether the following holds:

$$(v_{pc}^\theta)^T v_r^\theta > 0 \tag{4.18}$$

If the above equation is not true for a pair of principle component and reference vector, we reverse the principle component. This process is visualized in Fig. 4.7, where, for the person's right thigh, the angle between its principle component (labeled with blue arrow) and reference vector (labeled with yellow arrow) is less than 90 degrees. On the other hand, for the person's

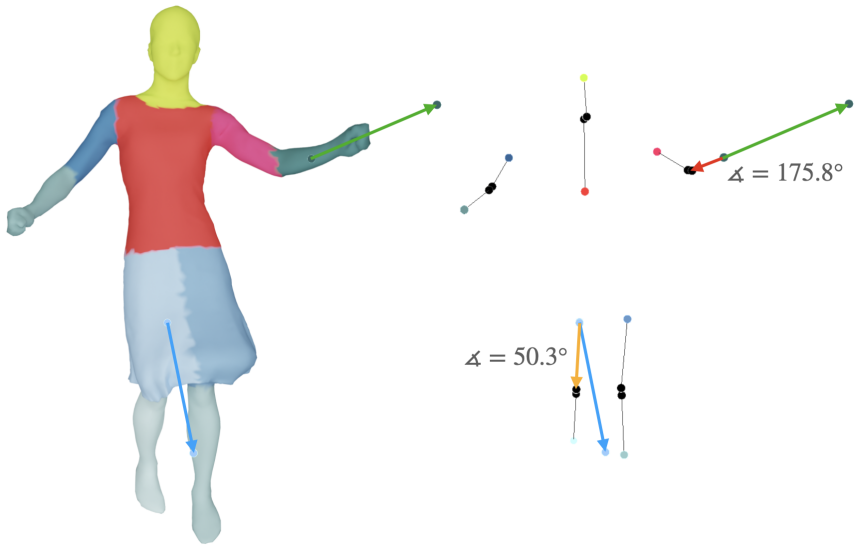


Figure 4.7: Left: example of labeled mesh with two selected principle components. Right: example of reference vectors with two selected principal components.

left forearm, the angle between its principle component (labeled with green arrow) and reference vector (labeled with red arrow) is greater than 90 degrees, so we reverse the principle component in the latter case. We perform this operation on both the source and target mesh to ensure that all principle components are correctly matched. Note that although reference vectors alone can capture the transformations between segments, principle components are more accurate in this regard.

With segment centers and principle components correctly matched between the source and target meshes, we calculate the segment-wise rotation and translation. Then, for each source vertex, we find the vertex on the target mesh that is closest to the transformed version of the source vertex. In this way, we are able to relocate each source vertex to a similar position on the target mesh with respect to the segment that the vertex belongs to in the first iteration, thereby speeding up the converging time in easy deformation cases as well as realizing almost impossibly hard cases.

To simulate uncertainty in the real-world cases and test the robustness of current global registration methods, we add Gaussian distributed noise to both the rotation and translation

of each segment with variances equal to 10 degrees and 10% of the segment bounding box's diagonal, respectively.

4.4.5 Implementation Details

We implement all the methods with high-performance implementations. We optimize equations (4.5, 4.12, 4.14) with Levenberg–Marquardt [65] algorithm. We solve the optimization problem with the GPU version of Optlang [27]. We use an efficient KDTree implementation [10] to find closest points. All experiments run on a PC with Intel i7-10700K CPU, NVIDIA 2080TI GPU, and 32GB of RAM.

We explore extensively on regularization weights and all the other parameters including N_{max}: the maximum number of clusters allowed, R_s: sampling ratio of the sub-clusters, T_s: cluster separation threshold, and T_c: cluster combination threshold. The regularization weight tuning will be discussed in detail in the ablation study section. For the remaining parameters, the experiments show that they do not have significant influence on the final results in a reasonable value range. Therefore, in the following section, we only show results with the setup: $N_{max} = 1000$, $R_s = 6$, $T_s = 1 \times 10^{-3}$, and $T_c = 5 \times 10^{-4}$.

4.5 Results

4.5.1 Quantitative Evaluation

In this section, we demonstrate the numerical results of our experiments. We compare seven algorithms in terms of the deviation between the output mesh and the target ground truth mesh, calculated as the sum of the bi-directional RMSE (Root Mean Square Error): distance of all vertices on the output mesh to their closest points on the ground truth surface, and similarly the inverse point-to-plane distance of ground truth vertices to the output surface:

$$f(A,B) = \sqrt{\frac{1}{N_v} \sum_{x \in \mathcal{V}_A} (\min_{p \in \mathcal{S}_B} \|x - p\|^2)} \tag{4.19}$$

$$RMSE = f(A,B) + f(B,A) \tag{4.20}$$

where $\mathcal{V}_A$ is the set of all vertices on mesh A and $\mathcal{S}_B$ is the surface of mesh B.

Let EDO as the embedded deformation with relaxation on the regularization weight. EDO fix stands for the embedded deformation without relaxation. ARAP is the traditional vertex-based deformation method with ARAP regularization. We also compare with other non-rigid deformation works [141, 5, 69] with open-sourced implementations using the default parameters in [141] for the same dataset. The results are shown as RPTS, NICP and RNRR. For our proposed methods, EDC represents embedded deformation with isometric regularization. We test with two different EDNode sizes. Baseline EDO and EDC use $\frac{1}{20}$ of all their vertices as EDNodes, whereas EDO 2x and EDC 2x use $\frac{1}{10}$. As our proposed EDC method decouples the deformation model and regularization, in the EAC 2x Re method, we resample the EDNodes in each iteration so that the underlying model reflects more accurate surface geometry and is not affected by accumulated error of deformation. Note that all our methods are separately tuned on a validation set with 50 samples in each test case. The final parameters are shown in Table 4.1.

Comparing the results of EDO and EDO fix in Table 4.2, we observe that relaxation improves the original embedded deformation, which demonstrates the importance of balancing the rigidity of different parts of the surface during deformation. Comparing the results of EDO and EDO 2x, we see that increasing the number of EDNodes does not always improve the final results as the local minimum issue may occur without proper regularization. Although we exhaustively search for optimal parameters, in the general case, we cannot make EDO with more EDNodes perform better. However, due to its superiority, EDC can easily achieve better results by either using more EDNodes or resampling them. Note that we fail to achieve a comparable results with

Table 4.1: Optimal value of regularization parameters. For EDO and its variations: maximum regularization weight. For EDC and its variations: inter-cluster regularization weight (left) for the isometric deformation step and cluster-based regularization weight (right) for the nonrigid embedded deformation step. Note that we decrease the regularization weights in all the methods exponentially with a fixed base $b = 0.9$.

	General / Incomplete Mesh	Inaccurate Corr
EDO	100	20
EDO 2x	200	20
EDO fix	50	2
ARAP	4	0.25
EDC	1 / 2	0.05 / 0.5
EDC 2x	0.5 / 2	0.05 / 0.5
EDC 2x Re	0.5 / 2	0.05 / 0.5

the source code of RPTS, NICP and RNRR, it could partially result from the fact they don't contain meticulous relaxation design.

4.5.2 Optimization Curve

We track the registration process by calculating the RMSE between the intermediate results and the target ground truth after each deformation iteration. We take the average RMSE among 50 samples in each of the three test cases. The optimization curves are shown in Fig. 4.8.

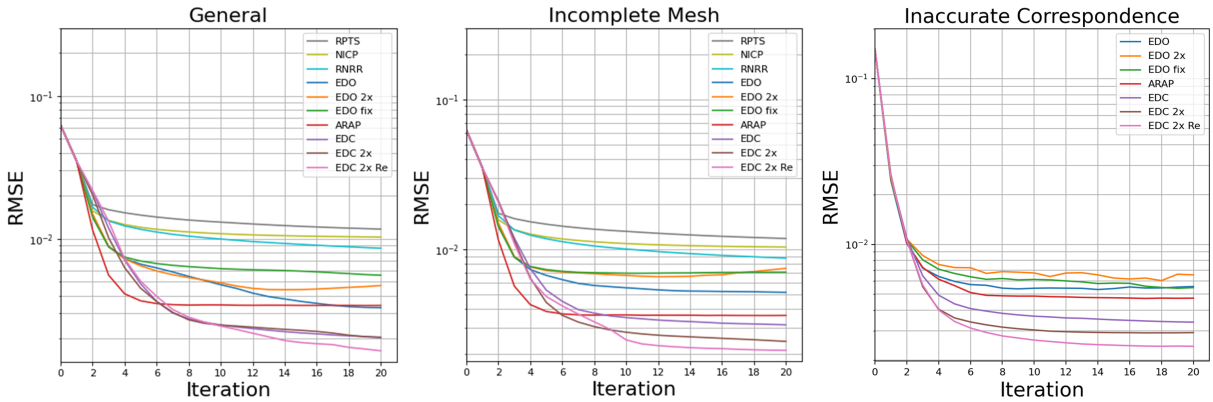


Figure 4.8: Optimization curves of the seven algorithms in three test cases: average bi-directional RMSE among all samples v.s. iterations.

Table 4.2: Bi-directional RMSE between the deformed meshes and the ground truth in meters. Our proposed method (EDC), its variants, and the best results are highlighted. Note that we don't test RPTS, NICP, RNRR in inaccurate correspondence case.

	General	Incomplete Mesh	Inaccurate Corr
RPTS	0.008805	0.008989	/
NICP	0.008311	0.008449	/
RNRR	0.007295	0.007602	/
EDO	0.003839	0.004500	0.006014
EDO 2x	0.004367	0.005155	0.006783
EDO fix	0.004168	0.004909	0.006327
ARAP	0.003835	0.004980	0.005809
EDC	0.003386	0.003541	0.004408
EDC 2x	0.003042	0.003707	0.003624
EDC 2x Re	**0.002775**	**0.003325**	**0.003086**

Our proposed methods (EDC, EDC 2x, and EDC 2x Re) achieve the best performance in all three test cases by converging with smaller errors in contrast to the other methods. For a fair comparison, in the general and incomplete mesh test, identical rotation and translation are applied to mesh vertices in the first iteration regardless of deformation methods. In these two cases, our proposed methods, which begin with one cluster, converge at the same rate as EDO methods but slower than ARAP in early iterations. However, ARAP is optimized with ten times more parameters and runs much slower (shown in Table 4.3). In the third experiment, we apply the inaccurate correspondences to all three methods in the first iteration. Our methods, which start with ten clusters, converge faster than the other methods. Note that, as our proposed methods and ARAP all directly regularize on surface vertices, the optimization curves are smoother and more stable at the end of iterations.

4.5.3 Qualitative Evaluation

In Fig. 4.10, we demonstrate deformed output meshes from different algorithms (shown on the right) along with the optimization curve for the specific case (shown on the left). We use the

point-to-plane inverse distance (from target to source) as the error, assign it to each ground truth vertex, and render the target mesh into a colormap. Note that, on the left side, we show different objects for different experiment cases. In the general test (upper), we render the source mesh and target-to-source error map to demonstrate the difference between the two. In the incomplete mesh test case (middle), we render the incomplete target mesh and complete-to-incomplete error map to display the discontinuity we test with. In the inaccurate correspondence case (lower), to visualize the large-scale divergence, we render the source mesh by itself and error map from the transformed source mesh to the target mesh where we directly apply the noisy cluster-wise transformations.

Our proposed method achieves the best visual appearance in all three experiments by creating intuitive results without affecting any part of the model (indicated by fewer red regions) as well as preserving the surface details (indicated by fewer green regions).

4.5.4 Running time

We record the running time of five ICP iterations without early stop for six methods in the general test case. The mean values are shown in Table 4.3. As EDO and EDC use simplified deformation model, they only take around $\frac{1}{7}$ running time in five iterations compared to ARAP methods. Due to only a small amount of overhead involved in the isometric deformation step, the overall EDC costs relatively the same time as EDO. It can be further accelerated with smaller max cluster sizes and better implementation.

4.6 Ablation Study

We tune all algorithms with meticulous parameter search. While Table 4.1 summarizes the parameters with the best performance on validation sets, Fig. 4.9 shows the tuning process for EDO and EDC with a heat map for each method. Since both heatmaps share the same color

Table 4.3: Running time and number of optimization parameters for each algorithm. Note that RNRR runs on CPU while the other methods run on GPU

	Time (ms)	# Parameters
EDO	23.408	6000
ARAP	153.966	140028
RNRR	357.440	/
EDC Cluster	4.874	<600
EDC Nonrigid	20.051	6000
EDC Combined	24.925	6000

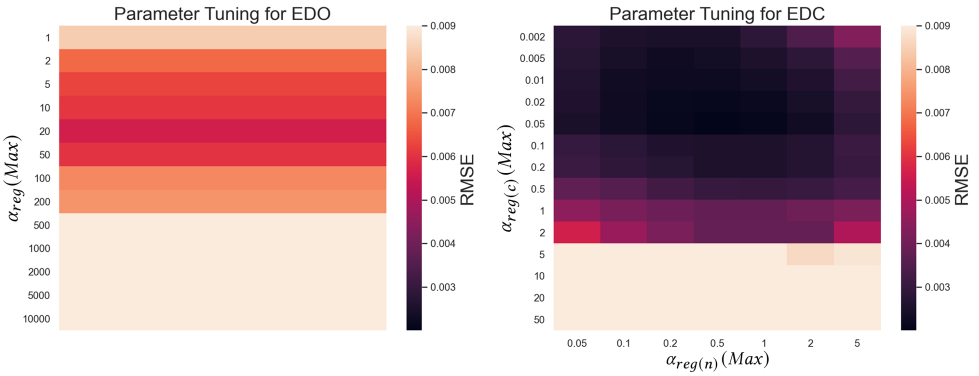


Figure 4.9: Examples of the parameter tuning process shown as heat maps.

scales, we clearly see that EDC attains lower RMSE on the majority of combinations of its two parameters compared with the best-performing case for EDO. Moreover, since the parameters are tuned with approximately the same multiples apart (i.e. parameter $p^{next} \approx 2p$), we find that EDC is less sensitive to parameter changes as its optimal performance does not vary significantly in a large span of parameter values.

4.7 Conclusion

In this chapter, we improve embedded deformation with a novel isometric regularization method. We propose a two-step procedure that successfully decouples the deformation model and regularization, and efficiently achieves region adaptive regularization. We verify the strength of our algorithm with extensive experiments.

The proposed deformation framework can be further improved. Firstly, we introduce several more parameters in our algorithms. Although we have shown that most of the parameters such as N_{max}, R_s, T_s and T_c do not significantly influence the final results, they still have to be adjusted in a new dataset where meshes have different scales and numbers of vertices. Better solutions to set the parameters, such as dynamically updating T_s and T_c given the statistics from the input meshes, can be implemented in future work. Secondly, our algorithm potentially converges slower in the first several iterations due to the naive cluster separation strategy currently employed. We will consider designing a more robust cluster refinement method as well as exploring on generating better initial clusters for faster convergence. Lastly, we can further accelerate our algorithm by replacing the general non-linear least squares solver (Optlang) with a made-to-measure optimizer.

Chapter 4, in part, is a reprint of the material as it appears in the paper: K. Chen, F. Yin, B. Wu, B. Du, and T. Nguyen, "Efficient Registration for Human Surfaces via Isometric Regularization on Embedded Deformation", submitted to *IEEE Transactions on Visualization and Computer Graphics(TVCG)*, 2021. The dissertation author was the primary investigator and author of this paper.

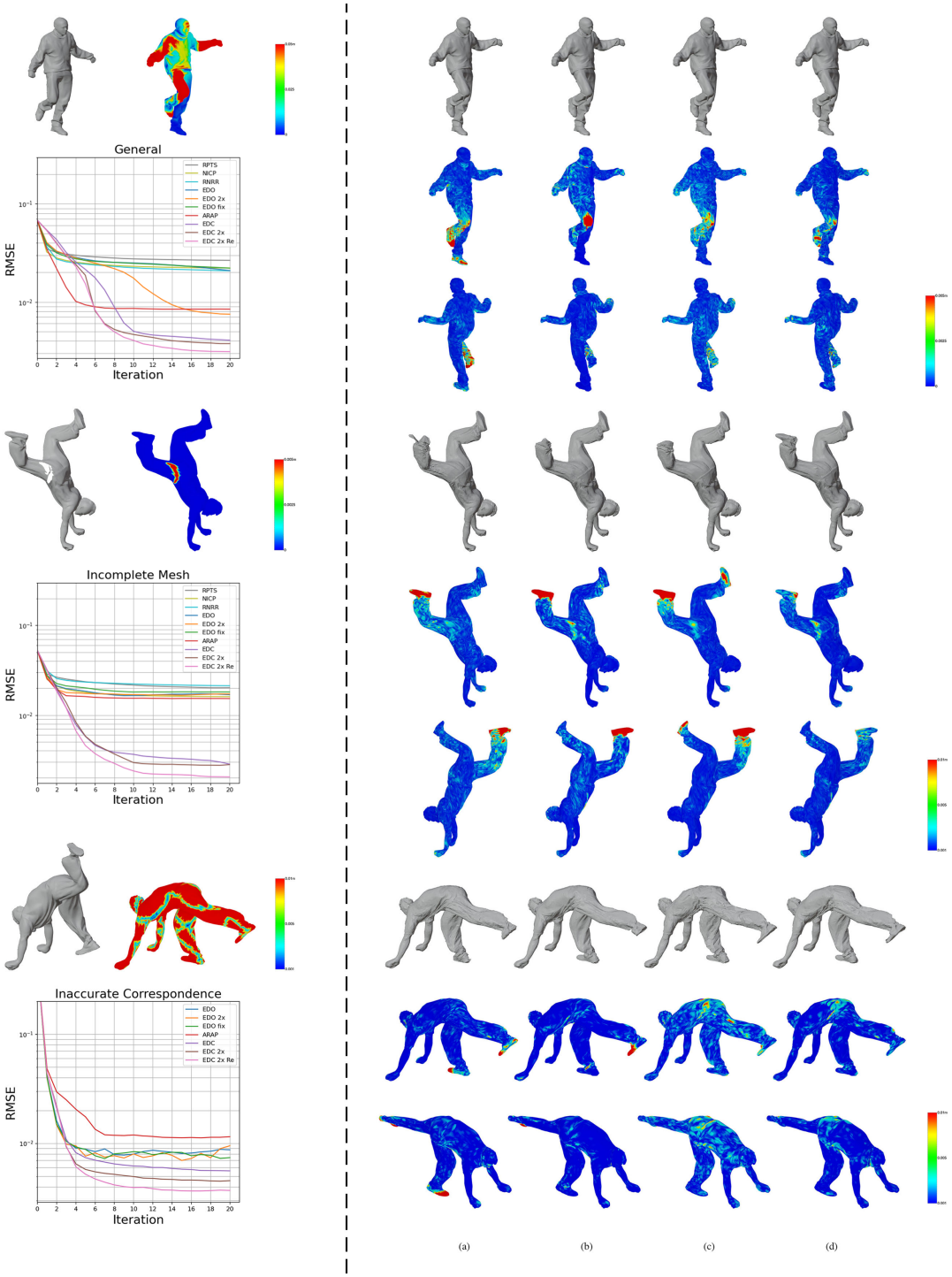


Figure 4.10: The optimization curves (left) and final outputs (right) for three selected cases. The deformed meshes and error maps from two viewpoints are displayed for four methods: (a) ARAP, (b) EDO, (c) EDC, and (d) EDC 2x Re.

Chapter 5

FaceFusion: Towards Lightweight High-fidelity 4D Faces

5.1 Introduction

Reconstructing photo-realistic dynamic 3D faces is one of the most important techniques in human digitization. It opens the gate to many metaverse applications such as virtual communication, VTuber live show, immersive interactive video games.

In the past few years, considerable success has been made in model-based single image face reconstruction. With the help of Deep Convolutional Neural Network (DCNNs), one can obtain a ready-to-use face mesh by estimating the coefficients of any 3D Morphable Model (3DMM) [11, 16, 51, 70, 40, 12], as well as generating a texture map given a frontal view or near-frontal view image [35, 121, 126, 24]. Recently, a lot of work [108, 97, 80, 23, 38, 73] has been done to improve texture generation. Some of them [38, 73] is capable of producing impressive models consisting of most of the high-frequency details in the input images by exploiting Generative Adversarial Networks (GANs).

However, generating a high-fidelity dynamic face given a video input is still challenging.

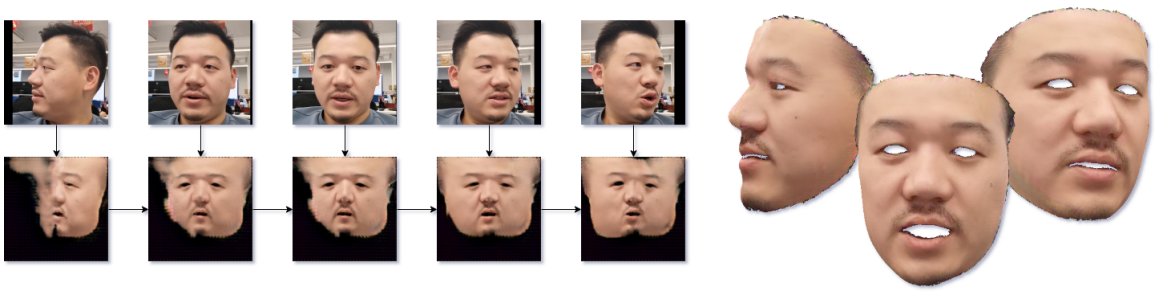


Figure 5.1: Our method reconstructs high-fidelity 3D face models by leveraging temporal information. Instead of costly inferring the unseen region and modeling precise surface, we stitch and refine the full-detail raw textures which are directly collected from the source pixels. Our method enables lightweight reconstruction and can be employed to many real-time applications. As shown above, after continuously blending new inputs, we obtain a complete texture. Three generated models are displayed on the right without any post-processing.

Firstly, the trivial solution, generating model and texture map for each individual frame, is not satisfying, because most of the single image reconstruction methods cannot handle side-view face images as reliable as front-view ones. It is a common observation that a network infers different appearances for the same surface region captured from other viewpoints, especially in occluded areas, hence there is no guarantee of view consistency across the whole sequence. In addition, we cannot follow tracking or blendshapes solution in the animation industry to apply baked texture maps to models in other frames, because the generated high-resolution texture maps are often non-uniformed. They contain expression variance facial information such as wrinkles, skin details, and hairs. Although some works [37, 7] attempt to decouple those content from the texture map, they either have to adopt a more complicated procedure or involve customized model. Most of them can only be performed offline.

In this chapter, we propose an efficient solution to generate high-fidelity face models from video input. Instead of expecting a perfect output from network in one step, we adopt a refinement strategy and focus on leveraging temporal information. In the first step, we follow the model-based single-view reconstruction framework and generate a rough face geometry with the state-of-the-art real-time reconstruction network [47]. After that, we texture the surface with

a projection-based method in which we only map the RGB value from image space to texture space, and remove some areas with apparent artifacts without introducing additional complicated structure to infer the occluded region.

The initial face model is far from satisfying with respect to three types of inconsistency:

- Geometry *v.s.* Input inconsistency. Due to the simplification of 3DMM parameters and network structure, the result of [47] doesn't always fit the input image accurately enough. Particularly, [47] tends to generate an open mouth surface even the mouth is closed in the input image.

- Texture *v.s.* Geometry inconsistency. The misalignment error can be propagated to the texture map. For example, if the estimated model is oversized, the background pixels will be introduced to the texture.

- Inter-view inconsistency. Although all models are mapped into the identical UV coordinates, the texture from different views can still mismatch, especially for the high-frequency details such as beard, mustache and moles. Directly averaging two texture maps would easily result in a ghosting effect.

Therefore, we design a FaceFusion network to stitch two incomplete texture maps from different viewpoints as well as address the aforementioned inconsistency. We are inspired by the structure in end-to-end image stitching networks [93, 94], and feed two texture maps into an encoder-decoder structure, and generate a high-quality texture map free from the ghosting effect. We also integrate a mesh refinement module to fix the inaccurate geometry. Specifically, we deform surfaces based on a projected 2D deformation map which is estimated with convolution framework, unlike in the other work, memory and time consuming Multilayer perceptron (MLP) or Graph Convolutional Network (GCN) has to be involved. With differentiable rasterization, two networks can be collaboratively trained without any 3D or texture space supervision. After the

above training procedure, we finetune the network to support progressive fusing which one of inputs is an accumulated texture generated with all of the previous frames.

5.2 Related Works

Single-view 3DMM-based method. With the developing popularity of metaverse applications, the demand for 3D face reconstruction is also rapidly growing. The seminal work of 3D Morphable Models (3DMM) [11] is introduced by Blanz and Vetter. The 3DMM performs a promising result of face reconstruction with the optimal combination of shape and albedo, given a single view image. After that, many researchers [31, 12, 52, 16] are exploring on different aspects of 3DMM. Focusing on the 3DMM limitation of facial shape, skin reflectance and illumination, Tewari *et al.* [122] introduce a self-supervised learned parametric face model based on just monocular images. To improve the linear 3DMM, taken two networks as the non-linear 3DMM decoders, Tran *et al.* [126, 127, 125] propose an innovative framework to learn non-linear 3DMM from numerous unconstrained face images. The new framework shows better performance than the linear one. After that, Deng *et al.* [24] further uses identity information and proposes a hybrid-level weakly-supervised training for CNN-based 3D face reconstruction. Based on Basel Face Model [52] and expression basis, this method performs more accurate shapes. For Multi-view Geometry Consistency Net (MGCNet) [111], Shang *et al.* present a self-supervised pipeline for monocular 3D Face reconstruction and introduce a new multi-view geometry consistency loss. However, these 3DMM-based methods rely on 3DMM texture data, so the restoration of facial details is not accurate.

Single-view non-3DMM-based method. 3DMM provides great prior knowledge of the face and also give some limitation of shape and texture. Many efforts are devoted to generating 3D faces directly. Feng *et al.* [33] choose to directly generate the 3D face dense alignment for face restructure with the Position map Regression Network (PRNet). Furthermore, Chen *et al.* [18]

and Gecer *et al.* [37] propose the networks to generate texture, shape and normal map directly. However, these methods acquire a lot of computation, which is difficult to run in real-time.

Multi-view reconstruction and Unconstrained reconstruction. Compared with the limited data of a single-view, multi-view reconstruction benefits from more data and is also an attractive problem. Multi-view 3D face network (MVFnet) [137] proposed by Wu *et al.* is a 3DMM parameters regressor that is trained with the novel self-supervised view alignment loss. The input of this network is multi-view face images taken simultaneously under the same lighting condition, which is a strong limitation. Bai *et al.* [6] propose the non-rigid multi-view stereo (NRMVS) to regress the face shape directly without requiring the input images taken simultaneously. Moreover, Gao *et al.* [34] propose a semi-supervised manner with adversarial loss that face reconstruction is facilitated using unconstrained photos. These methods use new data, but the effect is not ideal.

5.3 Overview

Figure 5.2 demonstrates our reconstruction pipeline. In Section 5.4, we describe the pre-processing step in which we generate rough geometries and textures, and also transform them into the format that is more suitable for DCNN. Next, we highlight the proposed FaceFusion network in Section 5.5, specifying our design and loss functions. Furthermore, in Section 5.6, we discuss the fine-tuned network to achieve our final purpose, improving the rough models by leveraging temporal information. After that, the experiment and implementation details are discussed in Section 5.7. At last, both qualitative and quantitative results are demonstrated in Section 5.8, followed by the conclusion in Section 5.9.

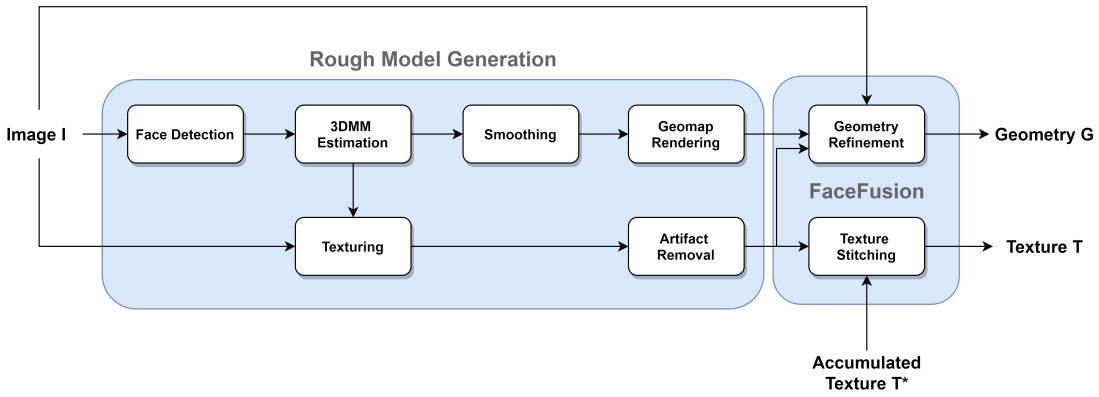


Figure 5.2: Overall reconstruction pipeline.

5.4 Rough Model Generation

The input video is processed frame-by-frame. Firstly we detect the face position with a robust face detector [15]. Then we send cropped face image to [47] to estimate 3DMM coefficients as well as camera position. Note that, our solution sets no constraints on the reconstruction approach in this step. It can work with more complicated algorithms or other parameter-rich 3DMM models for better initial guesses. We choose [47] only for efficiency consideration.

Next, we create texture map by mapping the RGB values from the input image to the UV space $\mathcal{T} \in \mathbb{R}^{3\times512\times512}$. It can be achieved through an inverse-rendering procedure. Specifically, we exchange texture coordinates and image-space coordinates of the generated face model and render with the input image as texture. This implementation could be problematic as both back-oriented faces and occluded faces can be painted as well. In addition, as the estimated model doesn't always align with the input, the texels that lie on the surface boundaries could be disseminated with colors of the wrong surface, as shown in Figure 5.3. Traditionally, the back-oriented issue could be tackled with back-face culling while the occluded faces issue is addressed with a depth test method. However, as 3DMM is an one-layer model, we resolve all three issues with large-angle removal method. For each pixel (or fragment in GPU), we calculate the angle between its normal and the viewing angle. We set all pixels with angles large than 70

degree to be black.

Figure 5.3: (a) 3DMM model projects to source image. (b) Generated texture after back-face culling. (c) Generated texture after removing large-angle faces.

Moreover, in the preprocessing step, we also render vertex position XYZ value to the texture space. We feed that as the representation of geometry to the following geometry refinement network.

5.5 Face Fusion

5.5.1 Texture Stitching

The overall training framework is shown in Figure 5.4. In the texture stitching module, we follow [94] as it reveals the capability to stitch two coarsely aligned images and generate high-resolution output without ghosting effects. In our task, stitching two texture maps seems easier in terms of the fact that less distortion is involved when mapping from image space to the regularized UV space. However, unlike in the common stitching problem, we have no access to the high-resolution texture ground truth. Also, the content of the rough textures could be slightly different.

Therefore, we utilize the self-supervised framework in the single-view reconstruction task, render the generated face model back to the image plane, and train with image space losses. Note that, although DCNN is powerful in the high-resolution image generation and transformation

tasks [133], it has to be fed with pixelwise aligned input and output pairs. Mismatch could cause artifacts in the final generation. For that reason, geometry refinement has to be trained with the texture stitching module simultaneously in order to fulfill better outcome.

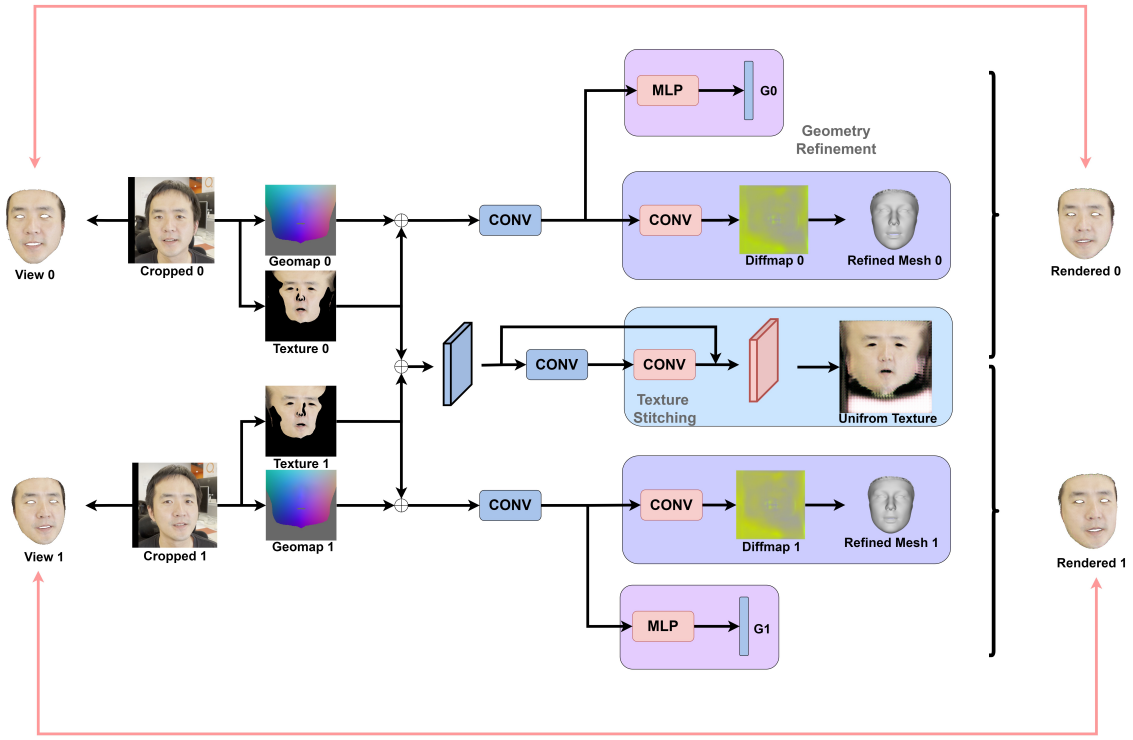


Figure 5.4: We train FaceFusion network to combine two rough models in a self-supervised manner. Note that two models are fed into the identical geometry refinement network and they share weight during training. Network and implementation details are shown in the supplementary material.

5.5.2 Geometry Refinement

In the geometry refinement module, instead of optimizing on the latent coefficients of 3DMM, we directly deform the vertices of the surface, allowing a more accurate and customized adjustment beyond the pre-defined model space.

As 3DMM surface is quite dense, consisting of near 40K vertices and 80K faces, directly applying MLP and GCN to predict the position of each vertex is not ideal for both training and inference. Alternatively, we follow the work in [3] to estimate a 2D deformation map

$\mathcal{D} \in \mathbb{R}^{3\times512\times512}$. The deformation map carries the movements information of all of the vertices. Specifically, the displacement of a vertex can be calculated by mapping it to the corresponding pixel and sampling the neighborhood with the bilinear interpolation:

$$\Delta v_i = I_b(\mathcal{D}(i;\delta_r)) \tag{5.1}$$

where I_b denotes the bilinear interpolation function and δ_r stands for the network parameters. Note that, deformation map decouples the network from underlying models and can work with any 3DMM model as well as arbitrary number of vertices. In addition, it can be trained with pure convolution structure.

Besides, the global transformation including: scale vector $s \in \mathbb{R}^3$, rotation quaternion $q \in \mathbb{H}^4$ and translation vector $t \in \mathbb{R}^3$ are estimated as well in another branch in the decoder. (See details in Figure 5.4). As these parameters are in low dimensions, we simply utilize MLP structure to estimate.

In summary, the new position of the vertex i after deformation can be written as:

$$v_i^* = S \cdot R \cdot \left(v_i + I_b(\mathcal{D}(i;\delta_r))\right) + t \tag{5.2}$$

where S, R are scale and rotation in matrix format.

5.5.3 Differentiable Renderer

With help of a differentiable renderer, we transform the 3D result to the 2D image plane and train the whole framework end-to-end. Thanks to the simplicity of face shape and our refinement policy, we can employ a differentiable rasterizer [78] to accomplish the deformation, with no necessity to exploit a more complicated raytracing-based method.

As we maintain our 3D models in image space coordinates and apply the global transformation in the geometry refinement module, in the rasterizer, we simply utilize the orthographic

camera with fixed focal length and principal point. Additionally, since our generated texture has been baked with lighting information, and in our application, the environment lighting is relatively stable. We fix the lighting parameters in the renderer as white ambient.

5.5.4 Loss Functions

Feature Loss

Following [93] and [94] , we penalize the difference between rendered appearance of reconstructed model $I \in \mathbb{R}^{3\times512\times512}$ and input image $I^0 \in \mathbb{R}^{3\times512\times512}$ by applying weight-averaged ℓ_1 loss of last five VGG layers outputs:

$$\mathcal{L}_{vgg} = \sum_k w_k ||\mathcal{F}^k(I) - \mathcal{F}^k(I^0)||_1 \tag{5.3}$$

Compared with pixel-wise ℓ_1 loss, the feature losses are proven to be more robust to variations such as pose, illumination as well as minor misalignment.

Adversarial Loss

We integrate the improved multi-scale discriminator [133] into our framework and demonstrate that it can effectively work with differentiable renderer if basic alignment is guaranteed. Given input and output resolution 512×512, we exploit three PatchGAN discriminators to process on three scales: 512×512, 256×256, 128×128. We denote the k-th discriminator as $D_k = D_k(s,x)$. When training with two-frame rough textures, s is the input texture for the specific view, x is the rendered image. We denote the GAN loss as $\mathcal{L}_{GAN}(G, D_k)$, where G represents the whole FaceFusion network. We also add the featuring matching loss in [133]:

$$\mathcal{L}_{fm}(G, D_k) = \sum_1^T \frac{1}{N_i} ||D_k^{(i)}(s,x) - D_k^{(i)}(s, G(s))||_1 \tag{5.4}$$

where i denotes the i-th layer feature of the discriminator.

Silhouette Loss

VGG-based feature losses can handle low-frequency information in the source image very well while GAN Loss are capable of recovering high-frequency details. However, both losses are defined on the RGB space. As shown in Figure 5.5, with only RGB losses, when the output silhouette doesn't match the source silhouette, the network tends to incorrectly fit the texture with the corresponding color at the imprecise position instead of correcting the surface. Therefore, we define a silhouette loss to guide the deformation explicitly with the silhouette $\mathcal{H} \in \mathbb{R}^{\times 512 \times 512}$.

$$\mathcal{L}_{silh} = \sum ||\mathcal{H}^{'}(I) - \mathcal{H}(I^0)||_1 \tag{5.5}$$

The differentiable rasterizer [78] highlights its global gradient design in which vertex can be affected by all surrounding pixels. In the default settings, [78] generate soft silhouettes with value range from 0 to 1 instead of binary mask (shown in Figure 5.5). In this way, the inner vertices may be deformed as well, leaving a very noisy surface after optimization. To solve the issue, we intentionally set all pixels five pixels away from the boundaries to be one. We denote the fix silhouette as $\mathcal{H}^{'}$. This implementation can efficiently achieve our deformation purpose without wasting great efforts tuning the renderer-related hyper-parameters (σ, γ, blur radius).

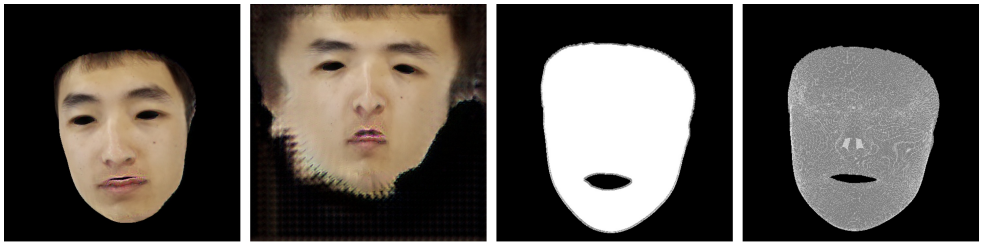


Figure 5.5: (a) Failure case deforming with RGB-only losses. (b) Texture generated with RGB-only losses. (c) Inside-hard silhouette. (d) Inside-soft silouette

Regularization Loss

In order to generate smooth surface, we also include Laplacian Loss and Normal Consistency Loss to regularize on the surface deformation. Laplacian Loss measures on the distance of a vertices v_i to the center of mass of its 1-ring neighbours $\mathcal{N}_i$:

$$\mathcal{L}_{lap} = \sum_i ||\delta_i||_2^2 \tag{5.6}$$

where $\delta_i = v_i - \frac{1}{||\mathcal{N}_i||}\sum_{j\in\mathcal{N}_i} v_j$

Normal Consistency Loss is defined on normal disparity between two adjacent faces with normals n_i and n_j:

$$\mathcal{L}_{nc} = \sum_{ij} \big(1 - \frac{n_i \cdot n_j}{||n_i||||n_j||}\big)^2 \tag{5.7}$$

Combining all of the aforementioned losses, we train FaceFusion network given two-frame rough models with:

$$\min_G \Big(\big(\max_{D_k} \sum_{n\in 0,1} \sum_k \mathcal{L}^n_{GAN}(G, D_k) \big) + \sum_{n\in 0,1} \lambda_{fm} \sum_k \mathcal{L}^n_{fm}(G, D_k) + \lambda_{vgg}\mathcal{L}^n_{vgg} + \lambda_{silh}\mathcal{L}^n_{silh} + \lambda_{lap}\mathcal{L}^n_{lap} + \lambda_{nc}\mathcal{L}^n_{nc} \Big) \tag{5.8}$$

where $n \in 0, 1$ denotes the losses from two views accordingly.

5.6 Progressive fusion

We extent the two-frame fusion framework to broader scenarios where we not only blend with rough models, but also with the fused model obtained in the previous frames. As shown in Figure 5.6, FaceFusion that is only trained with the rough textures cannot produce a satisfying result, if we directly feed the resultant texture back to FaceFusion network along with a rough texture. We upgrade the network by fine-tuning with a new type of dataset mixed with original rough textures and stitched textures. Note that the stitched textures don't fit the previous training framework very well, because supervision with only one viewing angle doesn't satisfy the fact that stitched textures should preserve all the information in the previous images.

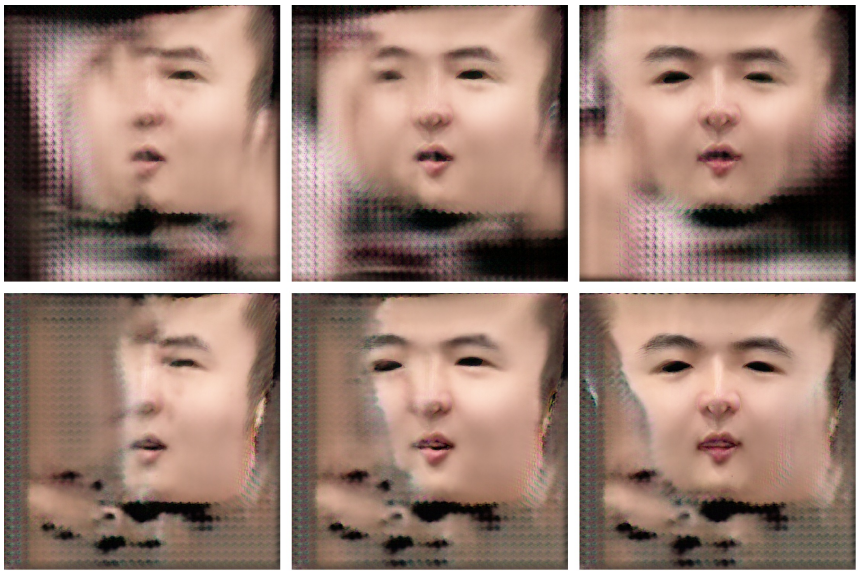


Figure 5.6: Results of progressive fusion. Top: before fine-tuning. Below: after fine-tuning.

Instead of making the accumulated result consistent with all the underlying frames, we take a simple but effective policy in which we only make the results consistent with the new input while preserve the remaining void region in the new texture similar to the accumulated texture.

Consequently, we train accumulated texture compatible FaceFusion with:

$$\min_{G}\Big(\max_{D_k}\sum_k \mathcal{L}_{GAN}(G,D_k)+\max_{D_k^T}\sum_k \mathcal{L}_{GAN}(G,D_k^T) + \lambda_{fm}\sum_k \mathcal{L}_{fm}(G,D_k)+\lambda_{fm}\sum_k \mathcal{L}_{fm}(G,D_k^T) + \lambda_{vgg}\mathcal{L}_{vgg}+\lambda_{silh}\mathcal{L}_{silh}+\lambda_{lap}\mathcal{L}_{lap}+\lambda_{nc}\mathcal{L}_{nc}\Big) \quad (5.9)$$

where we introduce a new discriminator $D_k^T = D_k^T(s,\mathcal{M}(x))$ for accumulated textures. s is the concatenation of two input textures. x is output texture. $\mathcal{M}$ is the void mask of new texture, as we only discriminate on the remaining parts in the UV space.

5.7 Experiment

5.7.1 Dataset

We experiment our method with a simple dataset. We collect turning head video of 40 objects with various expression. The videos are captured at 720×1080 resolution, 10 fps. The video length is around 15s.

We select 30 objects as the training set, 5 as validation set and test with the remaining 5 objects. To blend with two-frame rough models, we train FaceFusion network with data consisting of two randomly selected frames of a single object. To achieve progressive fusion, we iteratively train FaceFusion network with data from two randomly frames of a single object in the first run, then feed the stitched texture and a new texture of the same object in the second run.

5.7.2 Data Preprocessing

As our 3DMM model doesn't contain movable eyeballs and teeth structures, to mitigate the influence of misalignment on these parts, the captured images are preprocessed to remove

eyes, mouth, hair, and background. Firstly, we get 68 face landmarks by the face-alignment detector [15] and crop the image into 512×512 resolution with the face in the center. Simply applying face segmentation network to detect face region could result in an irregular forehead. To tackle this issue, the 3DMM face masks are obtained with a face reconstruction method [24], and the face segmentation mask of both skin and hair is obtained based on an off-the-shelf face segmentation network built on BiSeNet [142]. The final data is obtained with the intersection mask of the face parsing mask and the 3DMM mask. The results are shown in Figure 5.7.

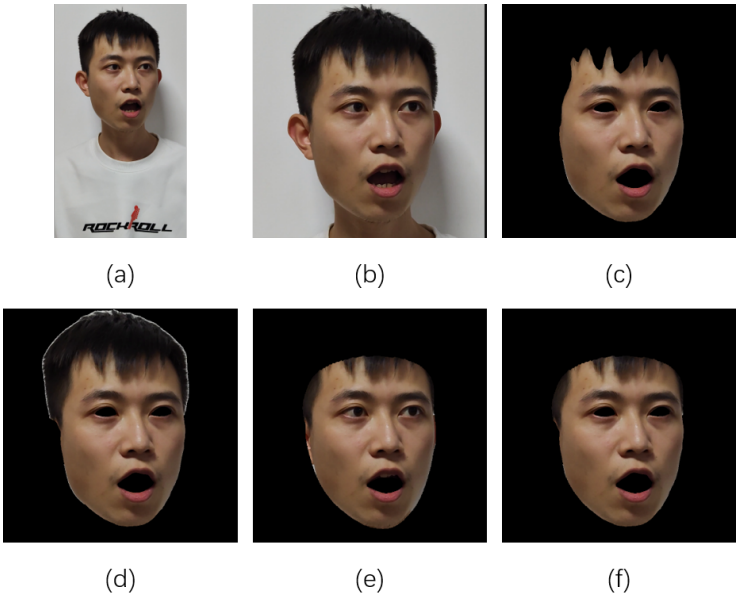


Figure 5.7: (a) Captured 720×1080 image. (b) Cropped 512×512 face image. (c) Cropped with the face segmentation mask of skin. (d) Cropped with the face segmentation mask of skin and hair. (e) Cropped with the 3DMM mask. (f) Cropped by the intersection mask of the face segmentation mask of skin and hair and the 3DMM mask.

5.8 Results

5.8.1 Implementation Details

We set $\lambda_{fm} = \lambda_{vgg} = \lambda_{silh} = 10$, $\lambda_{lap} = 1 \times 10^{-5}$, $\lambda_{nc} = 1 \times 10^{-2}$ and optimize equation (5.8) and (5.9) with batchsize equals two. More details are demonstrated in the supplementary

material. Without further speedup, both texture stitching and geometry refinement module take around 60 ms to process with NVIDIA V100 GPU.

5.8.2 Quantitative Evaluation

With five test objects, we randomly select 20 pairs of frames for each person and utilize FaceFusion network to generate 10 models. The ten models are rendered back into 20 images with the estimated camera parameters. We compare our results with three single-view reconstruction approaches [24],[111] and [126]. For each method, 20 models are generated separately and also rendered back to the specific views. FaceNet [110] is introduced to evaluate identity consistent with high-fidelity 3D face models to alleviate the misalignment issue. Given one face image, the FaceNet extracts a 512-dimensional embedding, regarded as the face's feature vector. Low Mean Squared Error (MSE) between two vectors represents the faces are more similar. Table 5.1 shows the FaceNet-MSE results of the proposed method and other comparison methods. To prove the effectiveness of the proposed method, we add two baselines. Baseline 1 is the evaluation result of the same person from different angles. Baseline 2 is the evaluation result of the same image with or without eyes and teeth.

It can be seen from the Table 5.1 that the performance of the proposed method is better than that of other comparison methods. Meanwhile, the evaluation of the proposed method is less different from baseline 1, which shows that the proposed method can generate consistent high-fidelity 3D face models.

5.8.3 Qualitative Evaluation

Figure 5.8 demonstrates the visual appearance of the reconstructed models. Compared to the other method, our proposed approach can generate faces that are more similar to the input. In addition, our method successfully recover the high-frequency details such as hairs, moles and

Table 5.1: Objective identity consistence comparison based on FaceNet-MSE [110] (10^{-4}) $\downarrow$

Method	Obj 1	Obj 2	Obj 3	Obj 4	Obj 5
Baseline 1	1.45	4.10	2.84	3.72	2.96
Baseline 2	0.91	3.58	0.83	1.07	4.51
Deng et al.[24]	11.53	14.95	8.85	8.31	10.31
MGCNet [111]	11.75	9.86	10.70	7.23	9.07
L. Tran et al. [126]	16.91	10.40	16.16	9.48	9.38
Ours	**6.86**	**5.03**	**3.76**	**6.26**	**6.40**

wrinkles in the source images. Besides, our method can robustly handle side-view images. More results are demonstrated in the supplementary material.

Figure 5.9 shows the process that texture is continuously fused through the video. Compared to the simple averaging accumulation method, our texture stitching network can generate higher resolution results with no ghosting artifacts. Without further fine-tuning, FaceFusion focuses on combining the information in the current frames, without maintaining texture contents in the previous frames.

5.9 Conclusion

In this chapter, we propose a lightweight solution to generate high-fidelity face models with certain consistency. We design a FaceFusion network, enabling both two frame blending and progressive fusion.

The proposed framework can be further improved. Firstly, we experiment our method in a large dataset with more diversity in races, ages, skin colors etc. Secondly, we'd like to extend our work from simple 3DMM model to full head. We will explore on more complicated models with eye balls, teeth and eyelid structure. Last but not least, we plan to accelerate our algorithm, further decrease the cost and apply it to real-time applications.

Chapter 5, in part, is a reprint of the material as it appears in the paper: K. Chen, B. Li, Y. Ye, and T. Nguyen, "FaceFusion: Towards Lightweight High-fidelity 4D Faces", submitted to

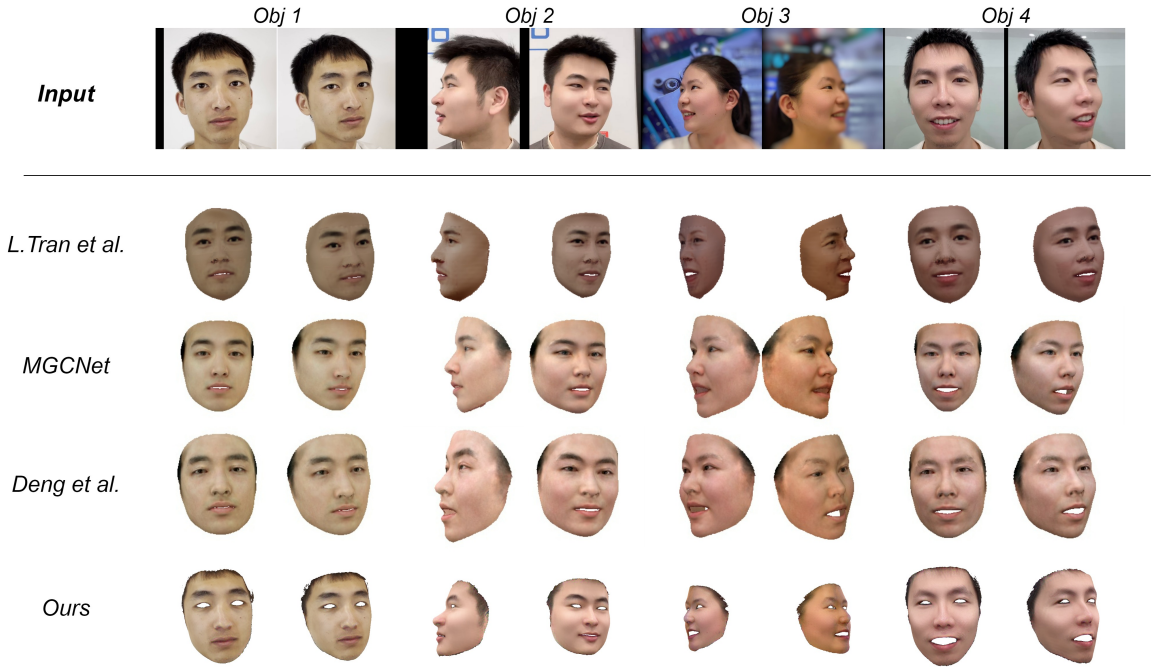


Figure 5.8: Reconstruction results of four methods. Note that our method generate single model using two frames data. The other methods generate models separately for each image.

Proceedings of the European Conference on Computer Vision (ECCV), 2022. The dissertation author was the primary investigator and author of this paper.

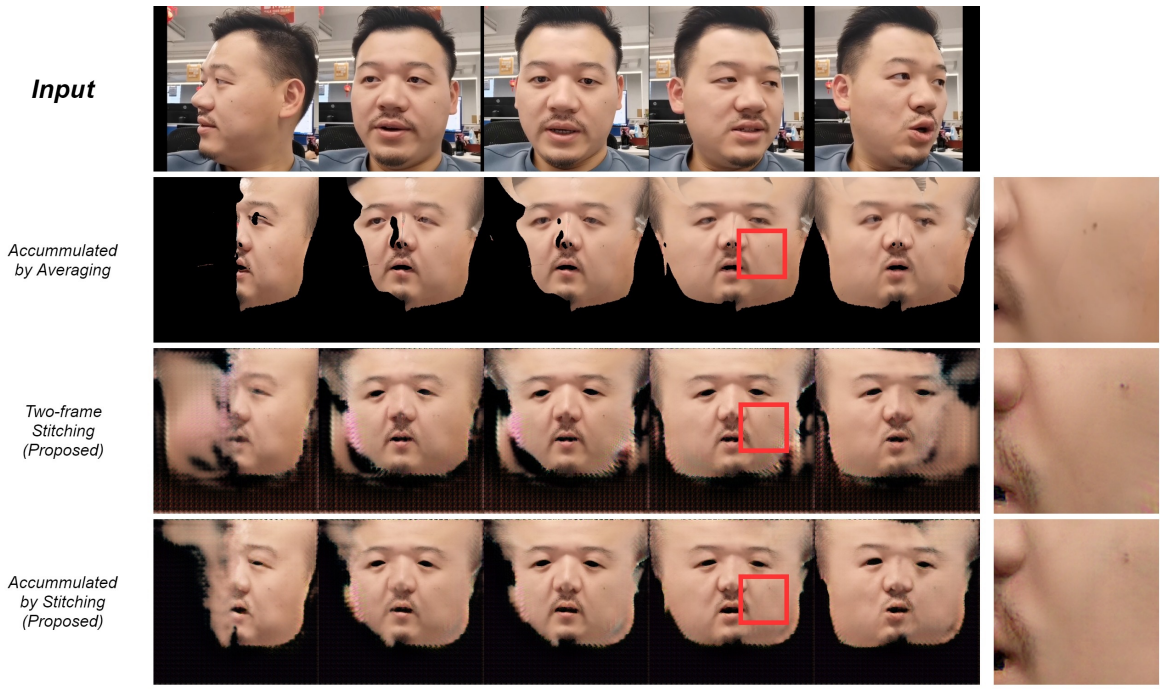


Figure 5.9: Progressive fusion results. Red block are zoomed out to show the reconstruction details

Chapter 6

Conclusion and Future Work

To achieve our central goal that enabling high quality reconstruction for multi-view system in limited-input situation, firstly, we investigate single-view 3D reasoning and improve on one of the most important modules, differentiable rasterization, in the mesh-based unsupervised learning framework. We propose a screen-space regularization method which tackles the implicit redundancy issues while keeping the overall solutions simple. Compared with the state-of-the-art differentiable rasterization design, the proposed method achieves better numerical results with much lower complexity.

The main issue of conventional approaches in fewer-camera scenarios is that the results often contain complex holes. We design an effective method to fill the gap regions on surfaces through "virtual" scans which are provided by inferred surfaces from a 3D reasoning method. Compared with the traditional mesh completion approaches, our algorithm can generate natural appearances without any user interaction. Compared with the state-of-the-art volumetric fusion methods, our approach supports selective blending, which only modifies the regions of interest and prevents adverse template inference from impacting the source.

Subsequently, we explore the scenarios with video stream input. In order to make use of temporal information to further improve the reconstruction results, it's critical to establish

correspondences between surfaces across different frames. The problem is conventionally tackled through non-rigid deformation. Among many ICP-based deformation methods, the embedded deformation is known for its efficiency. However, the embedded deformation regularizes on an underlying simplified structure, which could be problematic for intricate surfaces with various potential artifacts in human body reconstructions. Moreover, without elaborate parameter-tuning, embedded deformation may perform suboptimally, leading to slow convergence or a local minimum as all regions on the surface are assumed to share the same rigidity during the optimization. We propose a novel design that leverages isometric deformation idea and decouples regularization from simplified structures. Our method outperforms state-of-the-art methods both numerically and visually.

Furthermore, we investigate on human faces reconstruction, as faces often contain the most important and subtle information in real applications. We design a solution to integrate consecutive images of a tuning-head person with various expressions into a series of consistent high-fidelity 3D face models. Compared with the state-of-the-art single-view face reconstruction approaches, our method can resolve the ambiguities caused by occlusion and robustly handle challenging cases with extreme viewing angles.

For future works, we plan to extend our research in the following aspects. Firstly, while we propose several algorithms to address the key issues, we haven't fully integrated them together into a complete system. We will continue implementing the missing modules to realize the entire pipeline from depth sequences capture, sequential surface reconstruction to the final 4D video generation. Following this, we will accelerate the current algorithms system-wisely and explore the boundaries of performance given the hardware combinations.

There remains considerable room for further study of each step in the reconstruction pipeline. Apart from the reasonable extensions mentioned at the end of each chapter, we are interested in recently popular learning-based methods with implicit function representation [88, 99]. Related methods reveal the potential to generate more intricate surfaces. The highlighted

implicit functions in the frameworks, no matter based on occupancy or SDF, are closely related to the TSDF in our depth fusion algorithms. We would like to investigate the possible way to cooperate these learning-based methods into our pipeline.

Besides, we also want to explore another branch of human digitization research. Unlike our purpose to simplify the multi-view system, a series of work [109, 85, 3] provide successful reconstruction and animation solutions by accumulating more information, such as hair, cloth, and pose related shape variation, to the simple parametric models. Compared with free-form 3D reasoning, this type of reconstruction method can provide more robust and regularized predictions on the surface. Moreover, "animatable" models can be exploited to interpolate between large-shape-variation frames and generate more reasonable and smooth results.

More often, work is not accomplished after obtaining satisfying models. For many communication-based applications, it is necessary to consider how to compress and transmit sequences of meshes. A single HD model often contains more than 10K vertices, faces, and a high-resolution texture map. 4D video stream consisting of these models might take unacceptable brand width. Following our deformation framework, the established dense correspondences can be applied to fulfill this compression task. Instead of encoding raw positions and indices frame-by-frame, it is much more efficient to encode the inter-frame vertices' displacements and keep the connectivity unchanged in the short-time interval. And the by-product, local rigidity, estimated in our algorithm can also help to better encode these displacements.

Bibliography

[1] N. Ahuja and J. Veenstra. Generating octrees from object silhouettes in orthographic views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(2):137–149, 1989.

[2] T. Alldieck, M. Magnor, B. L. Bhatnagar, C. Theobalt, and G. Pons-Moll. Learning to reconstruct people in clothing from a single rgb camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1175–1186, 2019.

[3] T. Alldieck, G. Pons-Moll, C. Theobalt, and M. Magnor. Tex2shape: Detailed full human body geometry from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2293–2303, 2019.

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

[5] B. Amberg, S. Romdhani, and T. Vetter. Optimal step nonrigid icp algorithms for surface registration. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[6] Z. Bai, Z. Cui, J. A. Rahim, X. Liu, and P. Tan. Deep facial non-rigid multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5850–5860, 2020.

[7] L. Bao, X. Lin, Y. Chen, H. Zhang, S. Wang, X. Zhe, D. Kang, H. Huang, X. Jiang, J. Wang, et al. High-fidelity 3d digital human head creation from rgb-d selfies. *arXiv e-prints*, pages arXiv–2010, 2020.

[8] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[9] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[10] J. L. Blanco and P. K. Rai. nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees, 2014.

[11] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, 1999.

[12] J. Booth, A. Roussos, A. Ponniah, D. Dunaway, and S. Zafeiriou. Large scale 3d morphable models. *International Journal of Computer Vision*, 126(2):233–254, 2018.

[13] A. Bozic, P. Palafox, M. Zollhofer, J. Thies, A. Dai, and M. Nießner. Neural deformation graphs for globally-consistent non-rigid reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1450–1459, 2021.

[14] A. M. Bronstein, M. M. Bronstein, R. Kimmel, M. Mahmoudi, and G. Sapiro. A gromov-hausdorff framework with diffusion geometry for topologically-robust non-rigid shape matching. *International Journal of Computer Vision*, 89(2-3):266–286, 2010.

[15] A. Bulat and G. Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *International Conference on Computer Vision*, 2017.

[16] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425, 2013.

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

[18] A. Chen, Z. Chen, G. Zhang, K. Mitchell, and J. Yu. Photo-realistic facial details synthesis from single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9429–9439, 2019.

[19] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[20] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[21] H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141, 2003.

[22] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (ToG)*, 34(4):1–13, 2015.

[23] J. Deng, S. Cheng, N. Xue, Y. Zhou, and S. Zafeiriou. Uv-gan: Adversarial facial uv map completion for pose-invariant face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7093–7102, 2018.

[24] Y. Deng, J. Yang, S. Xu, D. Chen, Y. Jia, and X. Tong. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[25] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, 1999.

[26] V. Deschaintre, M. Aittala, F. Durand, G. Drettakis, and A. Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (ToG)*, 37(4):1–15, 2018.

[27] Z. DeVito, M. Mara, M. Zollöfer, G. Bernstein, C. Theobalt, P. Hanrahan, M. Fisher, and M. Nießner. Opt: A domain specific language for non-linear least squares optimization in graphics and imaging. *ACM Transactions on Graphics 2017 (TOG)*, 2017.

[28] M. Dou, P. Davidson, S. R. Fanello, S. Khamis, A. Kowdle, C. Rhemann, V. Tankovich, and S. Izadi. Motion2fusion: Real-time volumetric performance capture. *ACM Transactions on Graphics (TOG)*, 36(6):1–16, 2017.

[29] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, J. Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)*, 35(4):1–13, 2016.

[30] M. Droske and A. Bertozzi. Higher-order feature-preserving geometric regularization. *SIAM Journal on Imaging Sciences*, 3(1):21–51, 2010.

[31] B. Egger, W. A. Smith, A. Tewari, S. Wuhrer, M. Zollhoefer, T. Beeler, F. Bernard, T. Bolkart, A. Kortylewski, S. Romdhani, et al. 3d morphable face models—past, present, and future. *ACM Transactions on Graphics (TOG)*, 39(5):1–38, 2020.

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

[33] Y. Feng, F. Wu, X. Shao, Y. Wang, and X. Zhou. Joint 3d face reconstruction and dense alignment with position map regression network. In *ECCV*, 2018.

[34] Z. Gao, J. Zhang, Y. Guo, C. Ma, G. Zhai, and X. Yang. Semi-supervised 3d face representation learning from unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 348–349, 2020.

[35] P. Garrido, M. Zollhöfer, D. Casas, L. Valgaerts, K. Varanasi, P. Pérez, and C. Theobalt. Reconstruction of personalized 3d face rigs from monocular video. *ACM Transactions on Graphics (TOG)*, 35(3):1–15, 2016.

[36] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[37] B. Gecer, A. Lattas, S. Ploumpis, J. Deng, A. Papaioannou, S. Moschoglou, and S. Zafeiriou. Synthesizing coupled 3d face modalities by trunk-branch generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV)*. Springer, 2020.

[38] B. Gecer, S. Ploumpis, I. Kotsia, and S. Zafeiriou. Ganfit: Generative adversarial network fitting for high fidelity 3d face reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1155–1164, 2019.

[39] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, and W. T. Freeman. Unsupervised training for 3d morphable model regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8377–8386, 2018.

[40] T. Gerig, A. Morel-Forster, C. Blumer, B. Egger, M. Luthi, S. Schönborn, and T. Vetter. Morphable face models-an open framework. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 75–82. IEEE, 2018.

[41] I. Gkioulekas, A. Levin, and T. Zickler. An evaluation of computational imaging techniques for heterogeneous inverse scattering. In *European Conference on Computer Vision*, pages 685–701. Springer, 2016.

[42] A. S. Glassner. *An introduction to ray tracing*. Morgan Kaufmann, 1989.

[43] S. Gold, A. Rangarajan, C.-P. Lu, S. Pappu, and E. Mjolsness. New algorithms for 2d and 3d point matching: pose estimation and correspondence. *Pattern recognition*, 31(8):1019–1031, 1998.

[44] I. I. Groen and C. I. Baker. Scenes in the human brain: Comparing 2d versus 3d representations. *Neuron*, 101(1):8–10, 2019.

[45] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[46] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018.

[47] J. Guo, X. Zhu, Y. Yang, F. Yang, Z. Lei, and S. Z. Li. Towards fast, accurate and stable 3d dense face alignment. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16*, pages 152–168. Springer, 2020.

[48] P. Henderson and V. Ferrari. Learning single-image 3d reconstruction by generative modelling of shape, pose and shading. *International Journal of Computer Vision*, pages 1–20, 2019.

[49] H. Hontani, T. Matsuno, and Y. Sawada. Robust nonrigid icp using outlier-sparsity regularization. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 174–181. IEEE, 2012.

[50] Q.-X. Huang, B. Adams, M. Wicke, and L. J. Guibas. Non-rigid registration under isometric deformations. In *Computer Graphics Forum*, volume 27, pages 1449–1457. Wiley Online Library, 2008.

[51] P. Huber, G. Hu, R. Tena, P. Mortazavian, W. P. Koppen, W. Christmas, M. Rätsch, and J. Kittler. A multiresolution 3d morphable face model and fitting framework. In *Proceedings of the 11th joint conference on computer vision, imaging and computer graphics theory and applications*, pages 79–86. SciTePress, 2016.

[52] IEEE. *A 3D Face Model for Pose and Illumination Invariant Face Recognition*, Genova, Italy, 2009.

[53] B. Jian and B. C. Vemuri. Robust point set registration using gaussian mixture models. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1633–1645, 2010.

[54] T. Ju, S. Schaefer, and J. Warren. Mean value coordinates for closed triangular meshes. In *ACM Siggraph 2005 Papers*, pages 561–566. 2005.

[55] Y. Jwa, G. Sohn, V. Tao, and W. Cho. An implicit geometric regularization of 3d building shape using airborne lidar data. *International Archives of Photogrammetry and Remote Sensing, XXXVI*, 5:69–76, 2008.

[56] O. Kähler, V. A. Prisacariu, and D. W. Murray. Real-time large-scale dense 3d reconstruction with loop closure. pages 500–516, 2016.

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

[58] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

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

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

[61] A. Kundu, Y. Li, and J. M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3559–3568, 2018.

[62] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International journal of computer vision*, 38(3):199–218, 2000.

[63] C. Lassner, J. Romero, M. Kiefel, F. Bogo, M. J. Black, and P. V. Gehler. Unite the people: Closing the loop between 3d and 2d human representations. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[64] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[65] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

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

[67] H. Li, B. Adams, L. J. Guibas, and M. Pauly. Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics (ToG)*, 28(5):1–10, 2009.

[68] H. Li, R. W. Sumner, and M. Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Computer graphics forum*, volume 27, pages 1421–1430. Wiley Online Library, 2008.

[69] K. Li, J. Yang, Y.-K. Lai, and D. Guo. Robust non-rigid registration with reweighted position and transformation sparsity. *IEEE transactions on visualization and computer graphics*, 25(6):2255–2269, 2018.

[70] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. Learning a model of facial shape and expression from 4d scans. *ACM Trans. Graph.*, 36(6):194–1, 2017.

[71] Z. Li, T. Yu, C. Pan, Z. Zheng, and Y. Liu. Robust 3d self-portraits in seconds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1344–1353, 2020.

[72] P. Liepa. Filling holes in meshes. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 200–205, 2003.

[73] J. Lin, Y. Yuan, T. Shao, and K. Zhou. Towards high-fidelity 3d face reconstruction from in-the-wild images using graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5891–5900, 2020.

[74] K. Lin, L. Wang, K. Luo, Y. Chen, Z. Liu, and M.-T. Sun. Cross-domain complementary learning using pose for multi-person part segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[75] Y. Lipman, D. Levin, and D. Cohen-Or. Green coordinates. *ACM Transactions on Graphics (TOG)*, 27(3):1–10, 2008.

[76] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rossi, and H.-P. Seidel. Differential coordinates for interactive mesh editing. In *Proceedings Shape Modeling Applications, 2004.*, pages 181–190. IEEE, 2004.

[77] S. Liu, W. Chen, T. Li, and H. Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. *arXiv preprint arXiv:1901.05567*, 2019.

[78] S. Liu, T. Li, W. Chen, and H. Li. A general differentiable mesh renderer for image-based 3d reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[79] S. Liu, S. Saito, W. Chen, and H. Li. Learning to infer implicit surfaces without 3d supervision. In *Advances in Neural Information Processing Systems*, pages 8295–8306, 2019.

[80] S. Lombardi, J. Saragih, T. Simon, and Y. Sheikh. Deep appearance models for face rendering. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

[81] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*, pages 154–169. Springer, 2014.

[82] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[83] K.-L. Low. Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 4(10):1–3, 2004.

[84] J. Ma, W. Qiu, J. Zhao, Y. Ma, A. L. Yuille, and Z. Tu. Robust $l\_\{2\}e$ estimation of transformation for non-rigid registration. *IEEE Transactions on Signal Processing*, 63(5):1115–1129, 2015.

[85] Q. Ma, S. Saito, J. Yang, S. Tang, and M. 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.

[86] V. K. Mansinghka, T. D. Kulkarni, Y. N. Perov, and J. Tenenbaum. Approximate bayesian image interpretation using generative probabilistic graphics programs. In *Advances in Neural Information Processing Systems*, pages 1520–1528, 2013.

[87] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[88] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[89] N. J. Mitra, S. Flöry, M. Ovsjanikov, N. Gelfand, L. J. Guibas, and H. Pottmann. Dynamic geometry registration. In *Symposium on geometry processing*, pages 173–182. Citeseer, 2007.

[90] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275, 2010.

[91] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015.

[92] T. H. Nguyen-Phuoc, C. Li, S. Balaban, and Y. Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. In *Advances in Neural Information Processing Systems*, pages 7891–7901, 2018.

[93] L. Nie, C. Lin, K. Liao, M. Liu, and Y. Zhao. A view-free image stitching network based on global homography. *Journal of Visual Communication and Image Representation*, 73:102950, 2020.

[94] L. Nie, C. Lin, K. Liao, S. Liu, and Y. Zhao. Unsupervised deep image stitching: Reconstructing stitched features to images. *IEEE Transactions on Image Processing*, 30:6184–6197, 2021.

[95] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.

[96] M. Olano and T. Greer. Triangle scan conversion using 2d homogeneous coordinates. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 89–95, 1997.

[97] K. Olszewski, Z. Li, C. Yang, Y. Zhou, R. Yu, Z. Huang, S. Xiang, S. Saito, P. Kohli, and H. Li. Realistic dynamic facial textures from a single image using gans. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5429–5438, 2017.

[98] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, et al. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 741–754, 2016.

[99] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. 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.

[100] D. Paschalidou, O. Ulusoy, C. Schmitt, L. Van Gool, and A. Geiger. Raynet: Learning volumetric 3d reconstruction with ray potentials. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3897–3906, 2018.

[101] G. Patow and X. Pueyo. A survey of inverse rendering problems. In *Computer graphics forum*, volume 22, pages 663–687. Wiley Online Library, 2003.

[102] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[103] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

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

[105] S. Saito, , Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. *arXiv preprint arXiv:1905.05172*, 2019.

[106] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2304–2314, 2019.

[107] S. Saito, T. Simon, J. Saragih, and H. Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 84–93, 2020.

[108] S. Saito, L. Wei, L. Hu, K. Nagano, and H. Li. Photorealistic facial texture inference using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5144–5153, 2017.

[109] S. Saito, J. Yang, Q. Ma, and M. J. Black. Scanimate: Weakly supervised learning of skinned clothed avatar networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2886–2897, 2021.

[110] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[111] J. Shang, T. Shen, S. Li, L. Zhou, M. Zhen, T. Fang, and L. Quan. Self-supervised monocular 3d face reconstruction by occlusion-aware multi-view geometry consistency. *arXiv preprint arXiv:2007.12494*, 2020.

[112] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The princeton shape benchmark. In *Proceedings Shape Modeling Applications, 2004.*, pages 167–178. IEEE, 2004.

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

[114] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004.

[115] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. In *ACM siggraph 2007 papers*, pages 80–es. 2007.

[116] K. Takayama, O. Sorkine, A. Nealen, and T. Igarashi. Volumetric modeling with diffusion surfaces. In *ACM SIGGRAPH Asia 2010 papers*, pages 1–8. 2010.

[117] G. K. Tam, R. R. Martin, P. L. Rosin, and Y.-K. Lai. An efficient approach to correspondences between multiple non-rigid parts. In *Computer Graphics Forum*, volume 33, pages 137–146. Wiley Online Library, 2014.

[118] M. Tarini, M. Callieri, C. Montani, C. Rocchini, K. Olsson, and T. Persson. Marching intersections: An efficient approach to shape-from-silhouette. In *VMV*, pages 283–290, 2002.

[119] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358, 1995.

[120] A. Tevs, M. Bokeloh, M. Wand, A. Schilling, and H.-P. Seidel. Isometric registration of ambiguous and partial data. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1185–1192. Ieee, 2009.

[121] A. Tewari, M. Zollhöfer, P. Garrido, F. Bernard, H. Kim, P. Pérez, and C. Theobalt. Self-supervised multi-level face model learning for monocular reconstruction at over 250 hz. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2549–2559, 2018.

[122] A. Tewari, M. Zollhöfer, P. Garrido, F. Bernard, H. Kim, P. Pérez, and C. Theobalt. Self-supervised multi-level face model learning for monocular reconstruction at over 250 hz. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[123] A. Tewari, M. Zollhofer, H. Kim, P. Garrido, F. Bernard, P. Perez, and C. Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1274–1283, 2017.

[124] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[125] L. Tran, F. Liu, and X. Liu. Towards high-fidelity nonlinear 3d face morphable model. In *In Proceeding of IEEE Computer Vision and Pattern Recognition*, Long Beach, CA, June 2019.

[126] L. Tran and X. Liu. Nonlinear 3d face morphable model. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7346–7355, 2018.

[127] L. Tran and X. Liu. On learning 3d face morphable model from in-the-wild images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, July 2019.

[128] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[129] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017.

[130] B. Vallet and B. Lévy. Spectral geometry processing with manifold harmonics. In *Computer Graphics Forum*, volume 27, pages 251–260. Wiley Online Library, 2008.

[131] N. Verma, E. Boyer, and J. Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2598–2606, 2018.

[132] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. In *ACM SIGGRAPH 2008 papers*, pages 1–9. 2008.

[133] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[134] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li. Dense human body correspondences using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1544–1553, 2016.

[135] A. E. Welchman. The human brain in depth: How we see in 3d. *Annual review of vision science*, 2:345–376, 2016.

[136] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. J. Leonard, and J. McDonald. Kintinuous: Spatially extended kinectfusion. 2012.

[137] F. Wu, L. Bao, Y. Chen, Y. Ling, Y. Song, S. Li, K. N. Ngan, and W. Liu. Mvf-net: Multi-view 3d face morphable model regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 959–968, 2019.

[138] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[139] C. Xian, H. Lin, and S. Gao. Automatic cage generation by improved obbs for mesh deformation. *The Visual Computer*, 28(1):21–33, 2012.

[140] Y. Xie, T. Jiaojiao, and X. Zhu. Linking points with labels in 3d: A review of point cloud semantic segmentation. *IEEE Geoscience and Remote Sensing Magazine*, 2020.

[141] Y. Yao, B. Deng, W. Xu, and J. Zhang. Quasi-newton solver for robust non-rigid registration. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[142] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018.

[143] Y. Yu, P. Debevec, J. Malik, and T. Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 215–224, 1999.

[144] Z. Yu, J. S. Yoon, I. K. Lee, P. Venkatesh, J. Park, J. Yu, and H. S. Park. Humbi: A large multiview dataset of human body expressions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[145] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1802–1811, 2017.

[146] Y. Zeng, C. Wang, Y. Wang, X. Gu, D. Samaras, and N. Paragios. Dense non-rigid surface registration using high-order graph matching. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 382–389. IEEE, 2010.

[147] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

[148] Y. Zhou, L. Hu, J. Xing, W. Chen, H.-W. Kung, X. Tong, and H. Li. Hairnet: Single-view hair reconstruction using convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 235–251, 2018.

[149] W. Zhu, H. Wu, Z. Chen, N. Vesdapunt, and B. Wang. Reda: Reinforced differentiable attribute for 3d face reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4958–4967, 2020.