Differentiable Subdivision Surface Fitting

Tianhao Xie

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science at

Concordia University

Montréal, Québec, Canada

August 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Tianhao Xie**

Entitled: **Differentiable Subdivision Surface Fitting**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Sudhir Mudur

_____ External Examiner
Dr. Eugene Belilovsky

_____ Examiner
Dr. Sudhir Mudur

_____ Supervisor
Dr. Tiberiu Popa

Approved by _____
Lata Narayanan, Chair
Department of Computer Science and Software Engineering

_____ 2022

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Differentiable Subdivision Surface Fitting

Tianhao Xie

In this paper we present a powerful differentiable surface fitting technique to derive a compact surface representation for a given dense point cloud or mesh, with application in the domains of graphics and CAD/CAM. We have chosen the Loop subdivision surface, which in the limit yields the smooth surface underlying the point cloud, and can handle complex surface topology better than other popular compact representations, such as NURBS(Non-uniform rational basis spline). The principal idea is to fit the Loop subdivision surface not directly to the point cloud, but to the IMLS (Implicit moving least squares) surface defined over the point cloud. As both Loop subdivision and IMLS have analytical expressions, we are able to formulate the problem as an unconstrained minimization problem of a completely differentiable function that can be solved with standard numerical solvers. Differentiability enables us to integrate the subdivision surface into any deep learning method for point clouds or meshes. We demonstrate the versatility and potential of this approach by using it in conjunction with a differentiable renderer to robustly reconstruct compact surface representations of spatial-temporal sequences of dense meshes.

Acknowledgments

I would like to express my deepest grateful to my supervisor Dr. Tiberiu Popa and co-advisor Dr. Brian A. Barsky for their helpful instruction during my Master program. Also, I would like to appreciate Dr. Sudhir Mudur for the discussion of technical problems. What's more, I would like to thanks my colleague Nasir Khalid for offering the pictures in Chapter 6 about CLIP-Mesh. Finally, I would like to thanks for all the people who give me help and support both in academic and life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Geometric representation

In computer graphics, the representation of the data is important. Three-dimensional geometric data has many representations depending on the context in which it is used. One such classification is by the number of parameters or control variables needed to define a 3D surface. One extreme case would be to use raw geometric data obtained from 3D acquisition in the wild using 3D sensors or photogrammetric techniques; this is usually represented as a large and unstructured point cloud, where the sampled points (parameters) are noisy and the geometry is often incomplete. At the other end of the spectrum, for simulation, CAD, shape optimization, animation, and other modeling and analysis applications, a compact and precise representation is desired; examples are implicit algebraic surfaces Bajaj (1992), bi-parametric surfaces using splines and NURBS(Non-uniform rational basis spline) Bartels, Beatty, and Barsky (1987); Piegl and Tiller (1996) or subdivision surfaces DeRose, Kass, and Truong (1998); Montes, Thomaszewski, Mudur, and Popa (2020). These compact surface representations use a relatively small number of control variables which are adjusted to yield a desired surface. In between these two extremes are a wide range of representations, from point-based surface definitions such as Moving Least Squares (MLS) surfaces, signed distance functions to standard polygonal meshes endowed with additional properties such as texture, normal and displacement maps. Conversion between these various representations is fundamental to any geometric processing pipeline, and many have been developed over the years Berger et al. (2017).

1.2 Subdivision Surface

Among all these representations, subdivision surfaces are particularly appealing to many high level applications such as surface optimization and analysis, simulation, modeling, and animation DeRose et al. (1998): not only they are very compact, they do not require explicit NURBS patch decomposition and alignment as NURBS do Sharma et al. (2020), which makes them ideal to use for fitting more complex surface topology. A subdivision surface is represented by a compact polygonal mesh which gets subdivided by introducing new vertices, using, for example, Loop subdivision formulation Stam (1998); in the limit, this subdivision process leads to a smooth shape. With the ubiquity of colour and depth sensing, dense point clouds with very large number of points have become very easy to acquire, but are difficult to use for a number of reasons, including size, ambiguity of the underlying surface, shape editing difficulties, etc. Fitting a compact representation, such as a subdivision surface, would address a number of these problems.

Fitting subdivision surfaces to point clouds has major challenges. For a start, they require an initial control mesh that is capable of representing the desired surface topology in the limit. This is difficult to derive and consequently it is usually guessed, unless a mesh is extracted by other means. Existing fitting methods Cheng et al. (2004); Estellers, Schmidt, and Cremers (2018); Marinov and Kobbelt (2005) rely on an optimization function that uses iterative point-to-point correspondences. This optimization function is non-differentiable, is not robust to noise and outliers, and also tends to fail if the initial guess of the control mesh is too far from the solution, especially in the tangential direction.

The reason is that this point-to-point fitting strategy is not only non-differentiable but also rigid and does not easily allow for optimization along the tangent space. Optimization along the tangent space is particularly important especially when fitting to spatial-temporal data where we must fit a fixed topology template to spatially and temporally changing point clouds, which are extensively used in computer animation Mendhurwar et al. (2020); Popa, South-Dickinson, Bradley, Sheffer, and Heidrich (2010).

Some of these challenges conceptually stem simply from the fact that the gap between these representations is too large: whereas the subdivision surface represents a smooth continuous surface

in a very compact way, the point cloud is only a collection of points and associated surface normals, with no other ordering or structure.

1.3 Contribution

Our principal idea in this work is to help the fitting by bridging this gap using an intermediate representation such as the implicit moving least square surface (IMLS) Kolluri (2008), which is an implicit function whose zero surface approximates the underlying surface of the input point cloud. Thus, instead of fitting the subdivision surface directly to the point cloud, we fit it to the IMLS surface defined over the point cloud. This bridging approach has several significant advantages. The IMLS surface plays the important role of an initial fairing operator over the point cloud. It defines an elegant and robust analytical distance function that replaces the traditional point to point distance used in previous methods, and one that naturally allows for sliding in the tangent space, making it ideal for both static and spatial-temporal surface fitting.

Our major contribution in this work is a complete pipeline for fitting a Loop subdivision surface to a dense point cloud or mesh using the IMLS surface as an intermediate representation. As both the Loop subdivision surface as well as the IMLS have analytical expressions, we are able to formulate the problem as an unconstrained minimization problem of a completely differentiable function that can be solved with standard numerical solvers. Furthermore, this differentiable surface fitting provides us with unique capability to integrate the compact subdivision surface representation of a point cloud into any deep learning method. We demonstrate the versatility and potential of this approach by using it in conjunction with a differentiable renderer to robustly reconstruct compact representations of spatial-temporal surface sequences.

Chapter 2

Background

2.1 Loop Subdivision

Triangular meshes are used in many applications, such as geometry modeling and finite element simulation. Thus, the ability to define a smooth surface based on a given triangular mesh is important. In 1987, Charles Loop generalized a recurrence relation between quartic box splines and triangular mesh Loop (1987). The process of subdivision is done by iteratively apply subdivision mask, which is a list of the weights, to the mesh to generate new vertices and adjust the position of old vertices as a weighted average of the neighbor vertices. The mask for the Loop subdivision scheme are shown in Figure 2.1. In the figure, β can be chosen to be $\frac{1}{n}(\frac{5}{8} - (\frac{3}{8} + \frac{1}{4}cos(\frac{2\pi}{n})^2)$. A example of the subdivision process is shown in Figure 2.2, where the original control mesh is subdivided twice. By introducing new vertices and adjusting the position of original vertices, the subdivided mesh become smoother. It is guaranteed that, in the limit of an infinite numbers of subdivision, the Loop subdivision surface has C^2 continuity in regular vertices (valence 6) and C^1 continuity in extraordinary vertices.

2.2 Analytical evaluation of Loop subdivision

In Loop (1987), the subdivision surface can be generated by iteratively applying the mask as shown in Figure 2.1 and Figure 2.2. However, the analytical evaluation of the Loop subdivision is

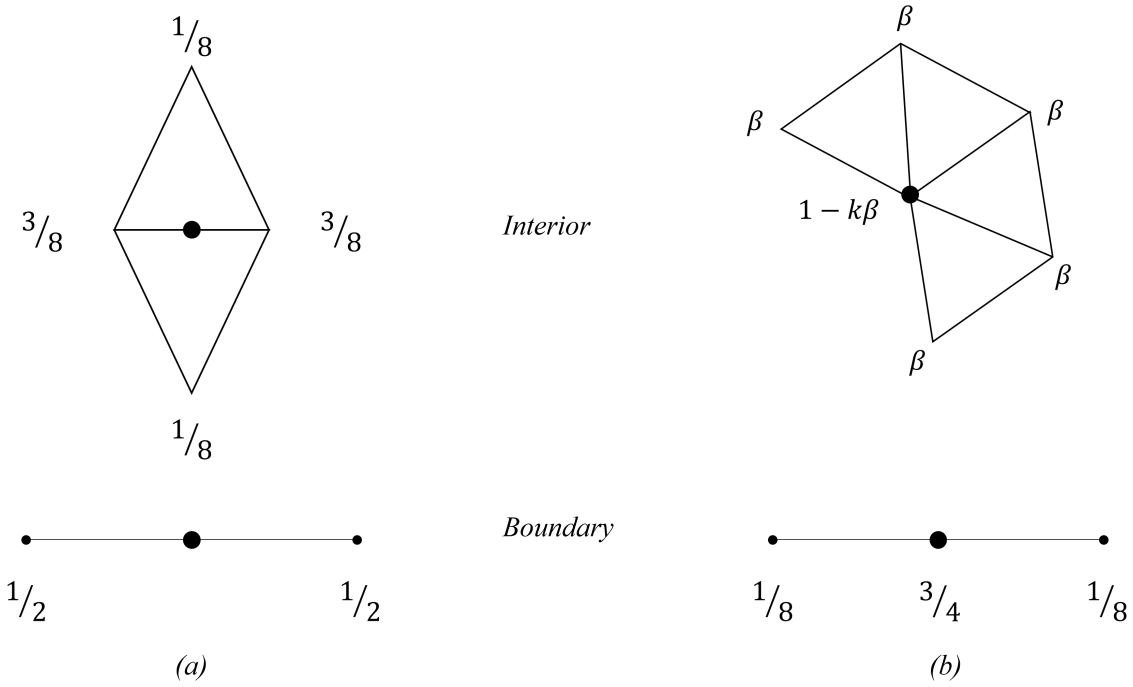

Figure 2.1: Loop subdivision mask: (a) Masks for odd vertices. (b) Masks for even vertices

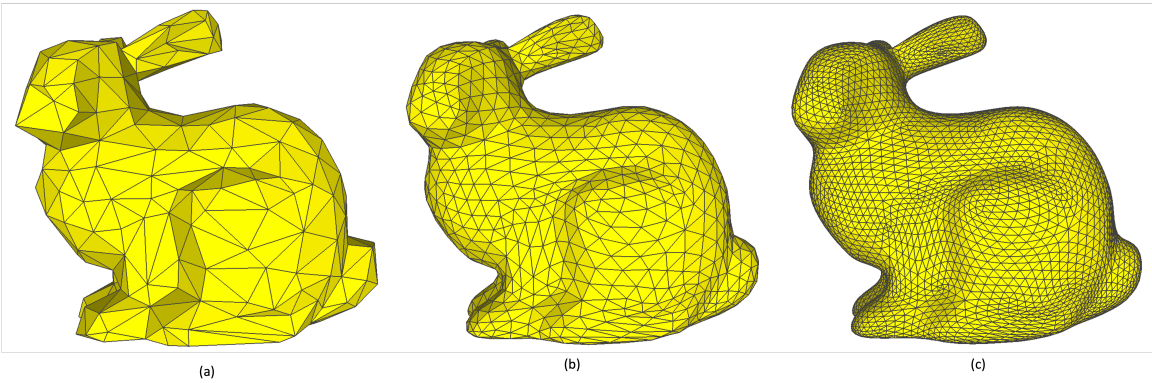

Figure 2.2: Loop subdivision process: (a) Original control mesh. (b) First level of subdivision. (c) Second level of subdivision.

required in some applications, such as surface fitting. In 1998, Jos Stam came up with an analytical evaluation method of Loop subdivision Stam (1998).

For a triangular mesh, we define the regular face as having three regular vertices and the irregular face as having more than one extraordinary vertex. The regular face can be parametrized using triangular Bezier patches derived from box splines Lai (1992). When evaluating a regular face, a triangular patch which defined by 12 control vertices is needed, as shown in Figure 2.3. The basis functions corresponding to each of the control vertices are shown in Appendix A. The regular face (dotted face in Figure 2.3) can be analytically expressed as:

$$s(v, w) = C^T b(v, w), \ (v, w) \in \Omega, \tag{1}$$

where C is a 12×3 matrix with 12 control vertices of the patch as shown in Figure 2.3. $b(v, w)$ is the vector of basis functions and it is defined over the barycentric coordinate:

$$\Omega = \{(v, w) \mid v \in [0, 1] \ and \ w \in [0, 1 - v]\}. \tag{2}$$

As for the irregular face, we can only evaluate the irregular face with exactly one extraordinary vertex. An irregular patch with an extraordinary vertex whose valence $K = 7$ is shown in Figure 2.4. This irregular patch is defined by $N = K + 6 = 13$ control vertices. Jos Stam used the projection of the control points into the eigen space of the Loop subdivision matrix S to evaluate the irregular face. When the extraordinary vertex's valence $K > 3$, the subdivision matrix S is non-defective and can be diagonalized,

$$S = V \Lambda V^{-1}, \tag{3}$$

where V contains the eigen vectors and Λ is diagonal eigen values. Let $\hat{C}_0 = V^{-1} C_0$ be the projection of initial control vertices C_0 into the eigen space of S and $\Phi(v, w)$ be the eigenbasis function, the irregular face (dotted face in Figure 2.4) can be analytically expressed as:

$$s(v, w) = \hat{C}_0{}^T \Phi(v, w). \tag{4}$$

The detailed definition of $\Phi(v, w)$ is referred to Stam (1998).

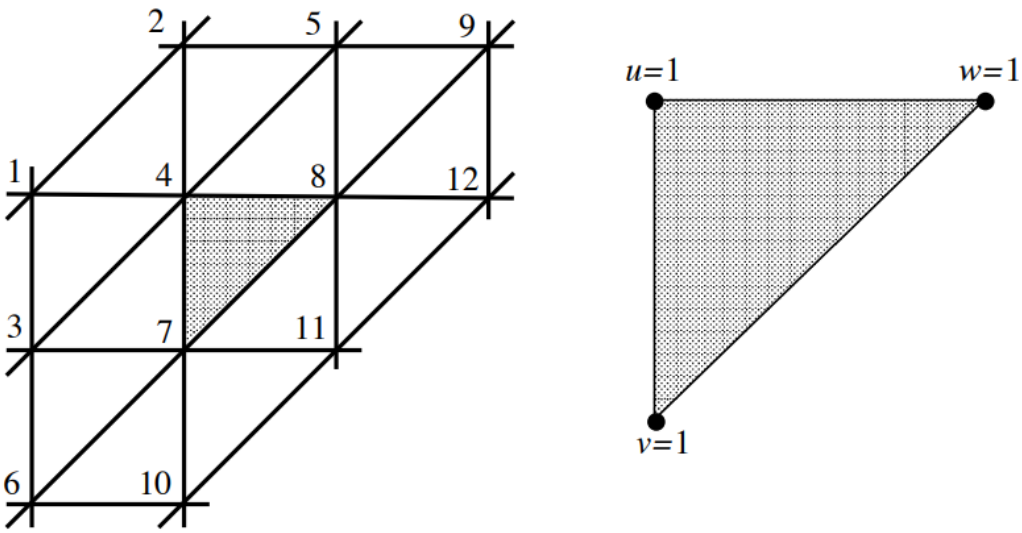

Figure 2.3: A triangular patch to evaluate regular face Stam (1998).

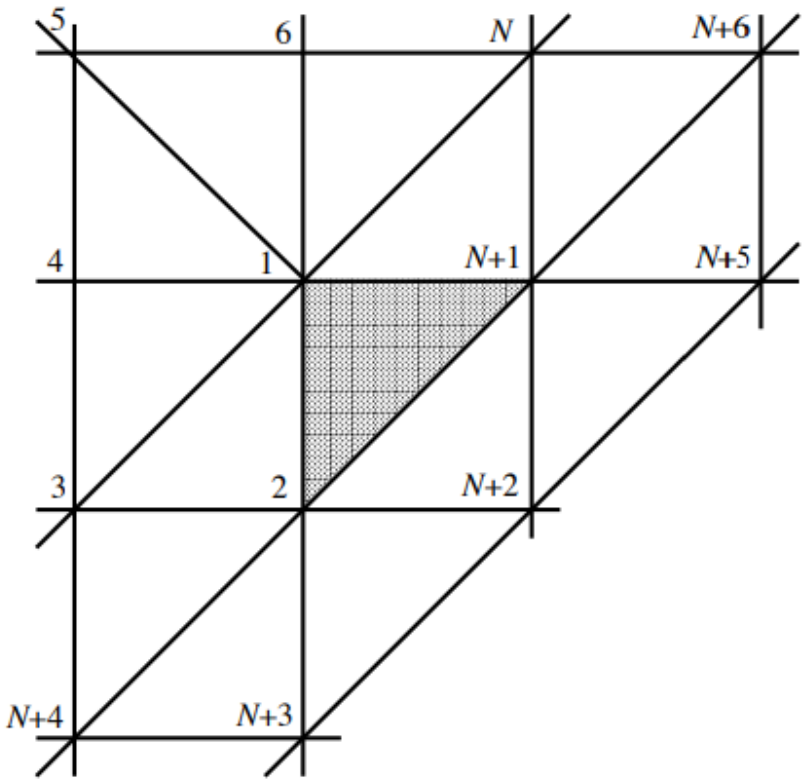

Figure 2.4: A triangular patch to evaluate irregular face with an irregular vertex whose valence $K = 7$ Stam (1998).

Chapter 3

Literature Review

The subdivision process defines a smooth curve or surface as the limit of a sequence of mesh refinement steps starting from a control mesh. This makes the final surface to be controlled by the small number of control vertices in the starting mesh, thus resulting in a very compact surface representation. Several subdivision schemes have been developed over the years and widely used in different applications Catmull and Clark (1978); DeRose et al. (1998); Doo and Sabin (1978); Liu et al. (2021); Loop (1987). In particular, Loop subdivision is a subdivision scheme based on quartic box spline on triangular meshes Loop (1987). It is guaranteed that, in the limit, the subdivision surface has C^2 continuity in regular vertices (degree 6) and C^1 continuity in irregular vertices. In 1998, Jos Stam developed an analytical evaluation method of Loop subdivision Stam (1998), which was based on conversion from Box splines to B-Nets Lai (1992). This analytical and differentiable evaluation makes this scheme ideal for differentiable shape optimization and we will use it in our novel subdivision fitting pipeline.

3.1 Fitting subdivision surface to target shape

It is a common task to fit a smooth surface representation to a target shape in computer graphics. One typical solution for this task is to fit a piecewise smooth surface to the target, such as a B-spline surface or a subdivision surface. Considerable work has been done on fitting B-splines to point clouds by squared distance minimization Wang, Pottmann, and Liu (2006); Zheng, Bo, Liu, and

Wang (2012). Since our focus in this work is on fitting subdivision surfaces, we will limit our discussion of related work primarily to subdivision surface fitting.

Hoppe et al. (1994) and Lavoué, Dupont, and Baskurt (2005) fit subdivision surfaces to CAD models by minimizing the squared distance energy. Litke, Levin, and Schroder (2001) used quasi-interpolation to fit Catmull-Clark subdivision surface to a given shape within a prescribed tolerance. Ma, Ma, Tso, and Pan (2002) described a method to fit a Loop subdivision surface to a dense triangular mesh by linear least square fitting.

The geometric data captured in the wild is almost always in the form of an unstructured point cloud, with noise, outliers, and missing geometry. A large body of work has focused on fitting subdivision surfaces to point clouds data Cheng et al. (2004); Estellers et al. (2018); Marinov and Kobbelt (2005); Mendhurwar et al. (2020). Cheng et al. (2004) fit the subdivision surface by iteratively minimizing a quadratic approximant of the squared distance function of a target shape. Their approach first samples points on the Loop subdivision surface based on a method by Stam Stam (1998). Then, they solve a linear system of the control mesh variables to minimize the squared distance between the sample points and target shape. Marinov and Kobbelt (2005) introduced an algorithm based on exact closest point search on Loop surfaces which combines Newton iteration and non-linear minimization. In more recent research, Estellers et al. (2018) used second-order approximation of the squared distance function and the tangent space alignment to achieve robust fitting of subdivision surface for shape analysis. Similar to methods in Cheng et al. (2004) and Marinov and Kobbelt (2005), Esteller *et al.* also sampled the points on the subdivision surface to establish the error function – error between the subdivision surface and the target shape. These methods need to solve a sequence of constrained least-squares problems to minimize the error function. The method in Ilic (2006) could be optimized by gradient-descent method. However, instead of fitting the limit surface, they could only fit a specific level of subdivision surface to the target shape. In contrast to many of these methods, our proposed solution frames the fitting problem as an optimization of a completely differentiable function that can be solved using standard differentiable optimization methods.

Some learning-based methods to fit a surface to a target shape have also been previously proposed. Most of these approaches fit parametric polynomial surfaces of some form to point clouds.

Yumer and Kara (2012) used a neural network to generate NURBS from input point sets. Ben-Shabat and Gould (2020) introduced DeepFit, which incorporated a neural network to learn point-wise weights for weighted least squares polynomial surface fitting. Sharma et al. (2020) described a method using neural networks to fit B-spline patches to input point cloud data . Our fitting method is not deep learning based; however, being differentiable, it can be used to bridge the gap between deep learning-based methods in a 3D domain and traditional subdivision surface techniques.

3.2 Spatio-temporal surface reconstruction

Reconstructing representations for time-varying 3D data is a common problem in graphics animation and simulation. A common approach is to fit a template mesh to the consecutive time-series point cloud or mesh. This is used to reconstruct coherent dynamic geometry from time-varying point clouds captured by real-time 3D scanning techniques. One widely used method is to reconstruct meshes for all frames first and then to fit a template mesh to all reconstructed meshes Allen, Curless, and Popović (2002); Kähler, Haber, Yamauchi, and Seidel (2002); Stoll, Karni, Rössl, Yamauchi, and Seidel (2006); Sumner and Popović (2004). These methods always need additional markers or landmarks which must be specified by the users. Another method is to generate a template from first frame and then fit the template directly to the remaining frames Shinya (2004); Süßmuth, Winter, and Greiner (2008). Süßmuth et al. (2008) followed the Multi-level Partition of Unity (MPU) Implicits approach to reconstruct the implicit function that approximates the time-varying surface defined by the time-varying point cloud and used the As-Rigid-As-Possible constraint to the moving of the points. When comparing this approach to our method, 1) it does not fit a subdivision surface to the 4D data and thus the final resulting surface was not smooth; 2) unlike our distance field energy, they used an implicit function to represent the point cloud surface, which must be optimized by solving a sequence of least squares problems.

A few other methods perform template-free reconstruction Mitra et al. (2007); Popa et al. (2010); Sharf et al. (2008). Mitra et al. (2007) directly compute the motion of the scanned object in all frames and estimate the time-deforming object by kinematic properties. Sharf et al. (2008) used a space-time solid incompressible flow prior to reconstruct moving and deforming objects from point

data. In Popa et al. (2010) a template is constructed gradually by mapping consecutive frames in a pyramidal fashion. Wand et al. (2009) reconstructed 3D scanner data by pairwise scanning alignment. Tevs et al. (2012) introduced Animation Cartography, an intrinsic reconstruction of shape and motion, based on robust estimation of dense correspondences under topological noise and landmark tracking in temporally coherent and incoherent data. In addition, there are also some real-time reconstruction methods for general objects, such as Li, Adams, Guibas, and Pauly (2009); Newcombe, Fox, and Seitz (2015); Zollhöfer et al. (2014). Our method described next is distinct from all the above, specifically in our formulation using the IMLS surface as an intermediate for fitting.

Chapter 4

Method

4.1 Overview

The input to our pipeline is either a static target shape in the form of a point cloud P or a temporal sequence of target shapes S^i in the form of a set of triangular meshes. We note that we do not require the triangular meshes to have the same connectivity. For the spatial-temporal case. we employ meshes as target shapes instead of point clouds only because there are currently no available reliable differentiable renderers for point clouds. And we need differentiable rendering since we want to combine it with our differentiable fitting to reconstruct spatial-temporal surfaces. But we would like to emphasize here that our method poses no conceptual limitations for using point clouds even for the spatial-temporal case.

The output of our method is a subdivision surface defined by a control mesh M^0. For the spatial-temporal fitting, the vertices of M^0 will have different 3D positions in each frame. The overview of our method is presented in Figures 4.1 and 4.2. From the point cloud we first create an initial control mesh. We optimize the vertex positions of this control mesh by minimizing the distance between the subdivision surface defined by this control mesh and the underlying IMLS surface defined by the target point cloud.

Similar to previous work, we compute this distance by sampling points on the subdivision surface, but in our formulation the sampled points are expressed as a differentiable analytical function of the control mesh and the distance function used is also a differentiable analytical function. This

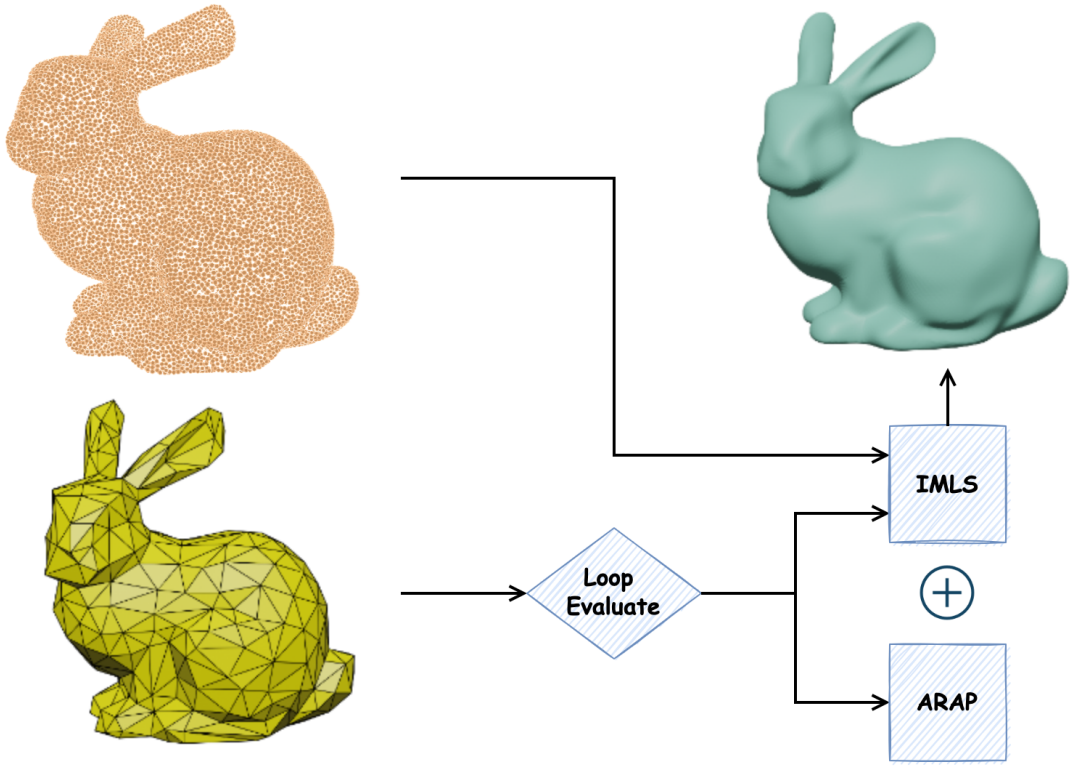

Figure 4.1: Overview of fitting subdivision surface to a static point cloud.

results in an unconstrained optimization problem of a differentiable analytical function that can be solved efficiently using standard off the shelf numerical methods. For the spatial-temporal case, we fit the subdivision surface defined by the control mesh iteratively to the temporally changing sequence of shapes, using the solution from one frame as a initial guess for the subsequent frame. Although this approach is popular and widely used Mendhurwar et al. (2020), it often fails due to accumulated drift arising from the inherently local nature of the geometric distance. Consequently, additional information is used to correct it, usually either in the form of boundary constraints Estellers et al. (2018) or other visual queues such as optical flow Bozic et al. (2020); Popa et al. (2010). Recently, with the development of differentiable renderers, rendered image difference metrics can be used to optimize shape Kanazawa, Tulsiani, Efros, and Malik (2018). Adding the image difference loss from the differential renderer complements our pipeline, adding a global structure to our local geometric fit thus eliminating the drift and yielding a more accurate fit.

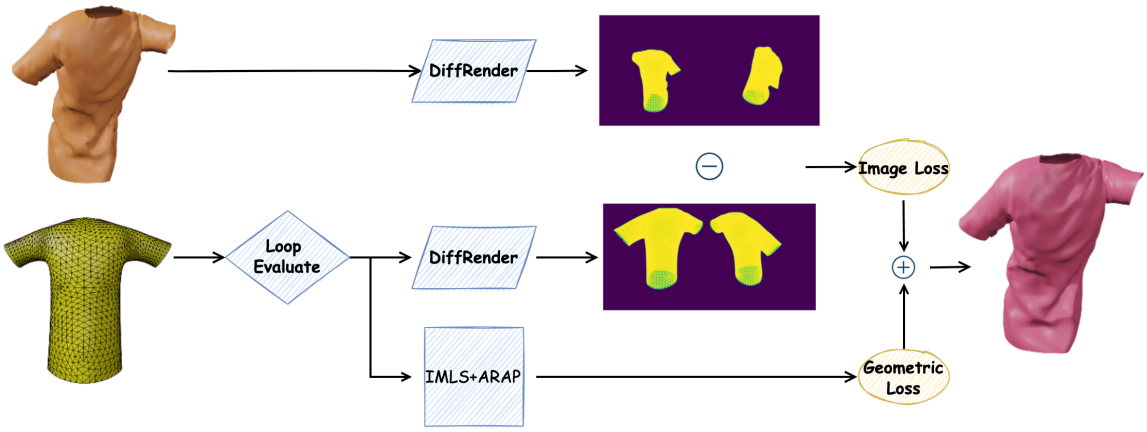

Figure 4.2: Overview of fitting subdivision surfaces to a spatial-temporal sequence by combining Implicit Moving Least Squares (IMLS) with differential rendering (DR) optimization

4.2 Acquisition of the template mesh

The first step in our process is to create the control mesh for the Loop subdivision surface $M^0(V^0, E^0)$. Although the position of the template mesh vertices will be determined by our optimization, the number of vertices as well as the topology of this mesh must be determined a priori. For this, we compute an initial triangular mesh that fits the point cloud using existing meshing methods; we used Screened Poisson Kazhdan and Hoppe (2013) method in MeshLab Cignoni et al. (2008). We then simplify this triangulation using quadratic edge collapse Garland and Heckbert (1997) until we obtain the desired number of vertices requested by the user.

There are some limitations for the generated template mesh. First, the template mesh can't have non-manifold, which will lead to failure during the subdivision process. Second, in the template mesh, there shouldn't have vertices with valence equals to two, which can't be evaluated by the algorithm. However, it's not recommended to remesh the control mesh to get better topology. According to the experiment, the remeshing process always make some vertices far away from the point cloud that can lead to bad fitting resolution in these areas. To solve this problem, there are some tools in MeshLab can remove the bad vertices by edge collapse.

4.3 Analytical derivative of Loop subdivision evaluation

Although Loop subdivision can be evaluated iteratively, for optimization purposes it is desirable to have an analytical expression of the surface. In chapter 2, we briefly introduce the Jos Stam's algorithm of analytical evaluation of the Loop subdivision surface of any point on the control mesh. For the purpose of optimization, we also need to compute the partial derivative of the Loop subdivision surface for every control point. Based on equation 3 and 4, for every control vertex, we can compute the partial derivative of the limit surface $s(v, w)$. For regular vertex,

$$\frac{\partial s(v,w)}{\partial p} = \frac{\partial C^T}{\partial p} \cdot b(v,w). \tag{5}$$

For extraordinary vertex,

$$\frac{\partial s(v,w)}{\partial p} = \frac{\partial C^T}{\partial p} \cdot (V^{-1})^T \cdot \Phi(v,w). \tag{6}$$

Suppose p_j is the j^{th} control vertex in C, $\frac{\partial C^T}{\partial p_j}$ is a matrix that j^{th} column is ones and all other entries are zeroes. Since both the evaluations of regular vertex and extraordinary vertex are linear, the Jacobian matrix of loop evaluation are constant for a certain mesh.

4.4 Limitation of Jos Stam's algorithm

Jos Stam's scheme only works on the condition that no two adjacent vertices on the control mesh are extraordinary vertices (i.e. degree different from six). As it is very difficult to guarantee this condition especially when the control mesh has thousands of vertices, a solution is to apply just the first subdivision step obtaining a mesh $M^1(V^1, E^1)$. Since the subdivision process will isolate all the extraordinary vertices, M^1 satisfies the above condition, however the number of vertices of M^1 are nearly four times as many as in M^0 making the representation far more verbose than desirable. A key observation here is that even though the number of vertices of the mesh obtained after one level of subdivision M^1 is much larger, the added vertices can be computed analytically from the original mesh M^0 thus maintaining the same number of degrees of freedom in controlling the subdivision surface. Suppose S^0 is the subdivision matrix applied to M^0, the Jacobian matrix of M^0 is J^0 and the Jacobian matrix of M^1 is J^1. Because of the observation stated before, if we

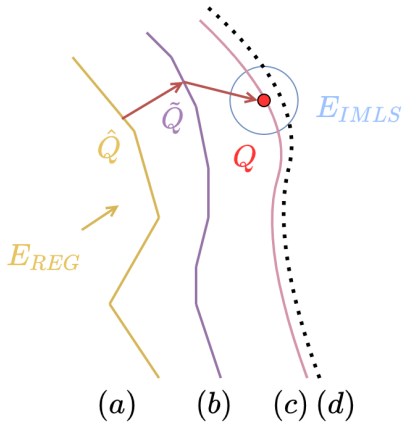

Figure 4.3: A schematic view of our optimization process. (a) Control mesh (M^0). (b) Control mesh after one level of subdivision (M^1). The vertices of this mesh are the Loop subdivision control points. (c) Loop subdivision surface. (d) Target point cloud. We optimize for M^0 by using an IMLS fitted to the point cloud and an ARAP regularizer on the control mesh M^0.

keep all derivatives of the original control vertices in the top rows of the J^1, we can compute the Jacobian matrix of M^0 as

$$J^0 = topRows(S^0 \cdot J^1). \tag{7}$$

Therefore, our control mesh for analytical evaluation of the subdivision surface is M^1, but we can only optimize for the vertices of M^0. This process is illustrated in Figure 4.3. Given a point $\hat{Q}$ on the control mesh M^0, in order to compute its position on the final smooth 3D surface we first compute its position $\tilde{Q}$ on M^1 by using the Loop subdivision mask Loop (1987). Then after adjusting the triangle index and getting new barycentric coordinates, we follow it by computing the position on the limit surface as per Stam Stam (1998). This operator $L(\cdot)$ that maps the point $\hat{Q}$ on the control mesh to the point Q onto the final subdivision surface is both analytical and differentiable.

Chapter 5

Subdivision Surface Fitting

5.1 Fitting a static model

Given a point cloud $P = \{P_i\}$ with associated normals $N = \{N_i\}$, we fit our template control mesh $M^0(V^0, E^0)$ using the following optimization:

$$\min_{V^0} \; E_{dist}(L(M^0, \hat{Q})) + \alpha \cdot E_{reg}(M^0, \bar{M^0}) \tag{8}$$

where $E_{dist}(\cdot)$ is the IMLS fit energy Öztireli, Guennebaud, and Gross (2009) (eq. 11), $L(\cdot)$ is the 3D position on the subdivision surface of a set of points $\hat{Q}$ sampled from the control mesh, $\bar{M_0}$ is the undeformed control mesh, $E_{Reg}(\cdot)$ is the ARAP regularizer Sorkine and Alexa (2007) (eq. 12) and α is the weight of the regularizer term. The overview of the fitting model is shown in Figure 4.1.

5.1.1 IMLS fitting energy

Öztireli et al. (2009) introduced an Implicit Moving Least Squares(IMLS) surface, which gave us a definition for the point cloud surface

$$f(x) = \frac{\sum n_i^T (x - x_i)\phi_i(x)}{\sum \phi_i(x)} \tag{9}$$

where ϕ is a locally supported kernel function that vanishes beyond the cut-off distance h. h is the radius we search for neighbor points and needs to be manually selected.

$$\phi(r) = (1 - \frac{r^2}{h^2})^4, \tag{10}$$

However, especially in real captured data, the point cloud density is always not similar everywhere. To avoid searching failure when h is fixed, if we can't get enough neighbor points on target point cloud, then h will increase from the h_0, until we get enough neighbours or arrive at the threshold. Larger values of h_0 will lead to surface smoothing while lower values of h_0 will give more details. The selection of h_0 will depend on the scale, density level and noise level of the point cloud.

We can use the implicit surface definition in equation 9 to derive a fit energy Montes et al. (2020)

$$E_{dist} = \sum_i (\frac{\sum_k N_k^T (Q_i - P_k)\phi(\|Q_i - P_k\|)}{\sum_k \phi(\|Q_i - P_k\|)})^2, \tag{11}$$

where P_k and N_k are the 3D positions and normals of points in the input point cloud (Figure 4.3d) and Q_i are points on the subdivision surface sampled from the control mesh (Figure 4.3a-c). For simplicity, in all our examples we only use the vertices of the control mesh, but we analyse the pros and cons of using more sampled points in the following sections and illustrated in figure 6.4.

5.1.2 Differentiation of IMLS fitting energy

As shown in equation 11, the analytical derivative of the IMLS energy is obviously difficult to derive. Thus, to make sure the derivative correct, we actually apply an auto-differentiation algorithm to compute the derivative in our code. By using the chain rule, the complex formula is decomposed to basic arithmetic operation and without having the analytical derivative, we can get the numerical value of the derivative at every point.

For easy to replicate our work, we also derived the analytical derivative of IMLS energy in the Appendix B.

5.1.3 Regularizer

We experimented with several reguralizers and the As-Rigid-As-Possible (ARAP) regularizer Sorkine and Alexa (2007) yields the best results. The ARAP regularizer allows local rotations, but it penalizes local stretch. More specifically:

$$E_{reg} = \sum_{i} \sum_{j \in N(i)} w_{ij} \|(\bar{V_i^0} - \bar{V_j^0}) - R_i(V_i^0 - V_j^0)\|^2, \tag{12}$$

where $N(i)$ is the set of vertices adjacent to V_i^0, V_i^0 is the initial vertex position and R_i is the local estimation rotation matrix for the one ring of vertices around vertex i. w_{ij} is the standard cotangent Laplacian weight Meyer, Desbrun, Schröder, and Barr (2003). At every iteration R_i can be computed analytically using SVD decomposition on the local co-variance matrix Umeyama (1991).

For the purpose of optimization, we also need to compute the derivative of ARAP energy,

$$\frac{\partial E_{reg}}{\partial V_i^0} = \sum_{j \in N(i)} 4 w_{ij} \left[(\bar{V_i^0} - \bar{V_j^0}) - \frac{1}{2}(R_i + R_j)(V_i^0 - V_j^0) \right]. \tag{13}$$

The details are shown in Appendix C.

5.1.4 Optimization

The optimization of the control mesh vertex positions V^0 is a non-linear optimization problem, which can be solved by using a non-linear solver, such as Google Ceres solver Agarwal, Mierle, and Others (n.d.). However, it will be slow if the size of the control mesh is large (i.e. thousands of vertices). In that case, we use gradient descent method, which is widely used in learning-based problem optimization.

After having the derivatives of the Loop evaluation(equation 7), IMLS energy(equation 19) and the regularizer(equation 13), the total energy function 8 can be optimized by the gradient,

$$\frac{\partial E_{total}}{\partial M^0} = \frac{\partial E_{dist}(L(M^0), \hat{Q})}{\partial L(M^0)} \cdot J^0 + \alpha \cdot \frac{\partial E_{reg}(M^0, \bar{M^0})}{\partial M^0}, \tag{14}$$

where J^0 is the Jacobian matrix of the Loop evaluation, which can be computed as shown in chapter 4.

5.2 Fitting a spatial-temporal model

5.2.1 Differentiable rendering

The emergence of differentiable rendering (DR) Laine et al. (2020); Ravi et al. (2020) paved the way for a new set of tools in 2D to 3D surface reconstruction. It allows 3D shape optimization and modeling from rendered 2D images Kanazawa et al. (2018); Kato et al. (2020); Kato and Harada (2019); Ravi et al. (2020). In image space, DR based optimization can give us a global loss energy when fitting to a mesh, which is complementary to our local geometric IMLS loss. Inspired by this, we introduce a new pipeline for fitting subdivision surface to spatial-temporal (4D) mesh data by combining our method with DR.

5.2.2 Optimization

Similar to the static case, given a control mesh M^0 and a sequence of spatial-temporal target meshes S^i, we are sequentially fitting the control mesh to each target mesh, using the solution of the current frame as an initial guess for the next one. Optimizing using only the geometric energy functionals described above leads to temporal drift as it can be seen in Figure 6.8 (a). Instead we add a image loss term that provides a global stabilization of the optimization, eliminating the drift as can be seen in Figure 6.8 (b).

In every iteration's forward pass, we use the silhouette DR with to render the target mesh in different camera positions k which give us target images I^k_{TARGET}. As shown in Figure 5.1, we use silhouette DR with 20 different camera positions which cover the 360° of the space to render a T-shirt mesh.

At same time, we use the same DR to render the limit surface of the template mesh which gives us predicted images I^k_{PRED} in same camera positions as used for rendering the target images.

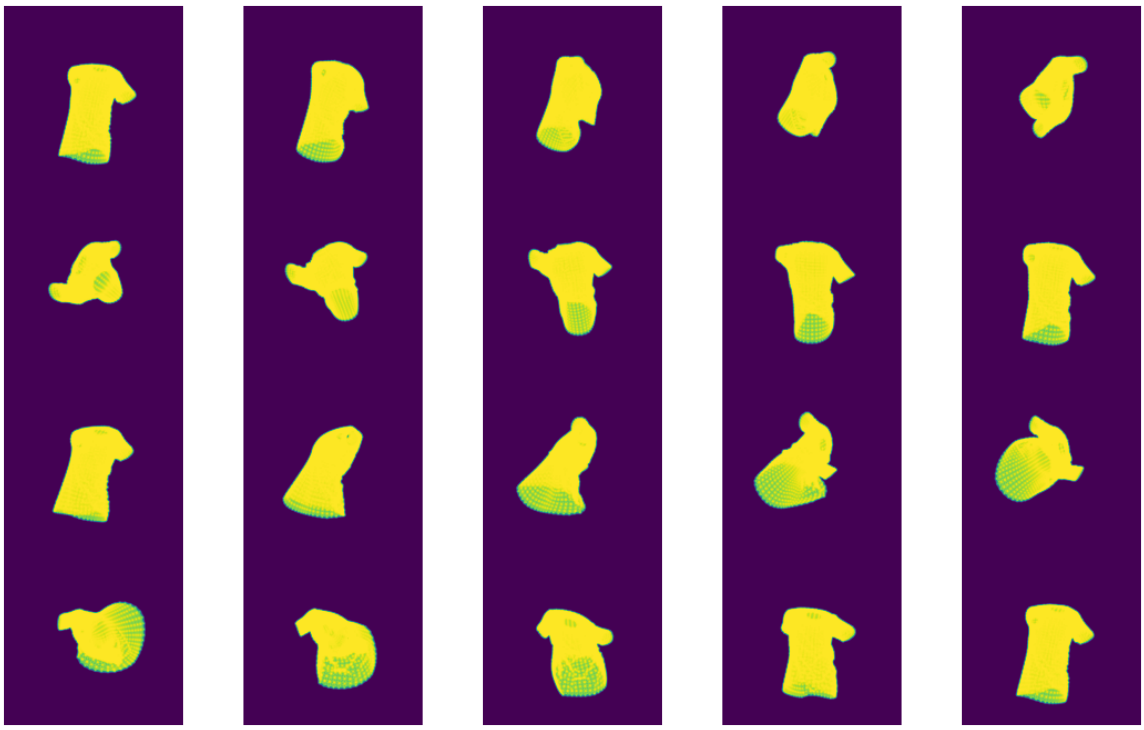

Figure 5.1: Target images rendered by silhouette DR in different camera positions

Suppose the number of pixels for rendered images is N, we compute image loss l_{image} by

$$l_{image} = \sum_{k} \frac{(I^k_{PRED} - I^k_{TARGET})^2}{N}. \tag{15}$$

As for the geometric loss, we compute it by the same method for computing energy provided in chapter 4. Thus, the total loss l_{total} is

$$l_{total} = E_{dist}(L(M^0), \hat{Q}) + \alpha \cdot E_{reg}(M^0, \bar{M^0}) + \beta \cdot l_{image}. \tag{16}$$

In our implementation we use the DR available in PyTorch3D Ravi et al. (2020) and for the backward pass, we use the gradient descent method, Adam optimizer Kingma and Ba (2014), to optimize the control mesh.

Chapter 6

Results and Discussion

6.1 Static surface fitting

We used our method to fit a number of synthetic models (the Stanford Bunny and Lucy, see Figures 6.1 and 6.2 as well as a point cloud acquired using the Microsoft Azure Kinect device (a Koala toy) Figure 6.3.

The starting searching radius h_0 and weight of ARAP regularizer α were selected manually. An automatic increasing scheme for $h0$ are applied in the case that neighbors searched are not enough, as detailed in chapter 5. We scaled the point clouds to a unit box before fitting, to increase the numerical stability of the optimization. For the Stanford Bunny and Lucy, we used $h_0 = 0.0005$. For the toy Koala, we used $h_0 = 0.05$. As for the α, it depends on the noise level of the point cloud. When the point cloud is noisy, you need a bigger weight, such as 0.1. When the point cloud is very clean, α should be set to very small, such as 0.01. For the Stanford Bunny and Lucy, we set $\alpha = 0.01$. For the toy Koala, we set $\alpha = 0.1$.

For the bunny(Figure 6.1) the original point cloud has $72,027$ vertices and the reconstructed mesh using Screened Poisson Kazhdan and Hoppe (2013) has $155,008$ vertices. We demonstrate two reconstructions. The first one with a template mesh of 4667 vertices (Figure 6.1 (c)) that shows no visual difference to the original, but uses only around 3% of the Screened Poisson reconstruction. The second one uses only 314 vertices, or only 0.2% of the Screened Poisson reconstruction (Figure 6.1 (f)). While a number of details are lost, the main shape is still reconstructed fairly well.

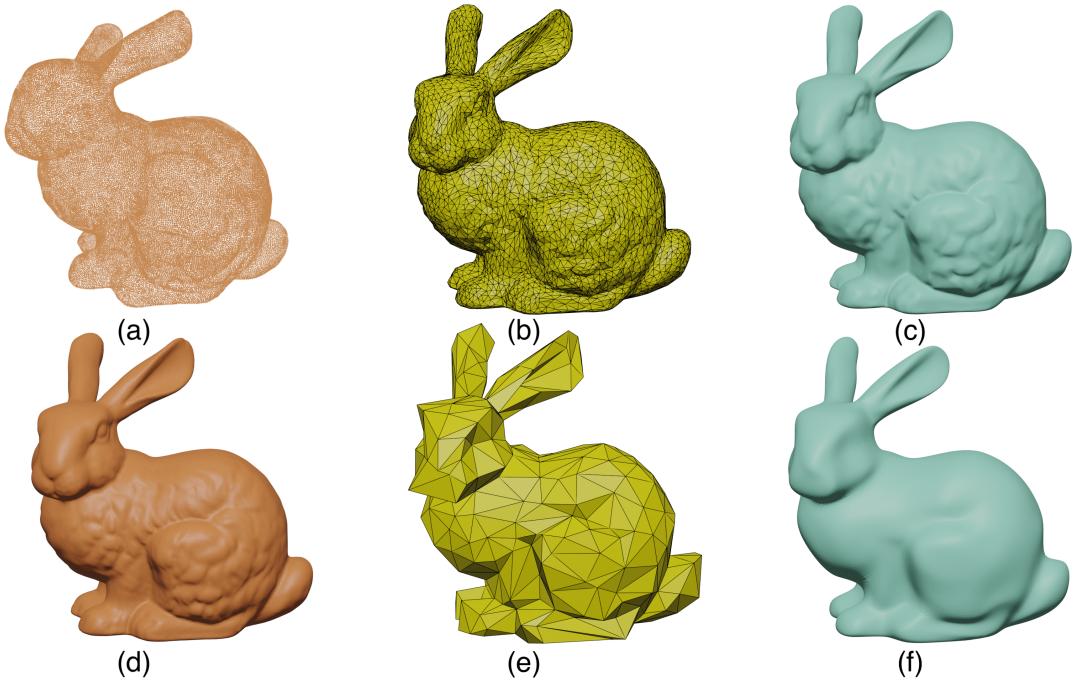

Figure 6.1: Stanford (n.d.) Bunny:(a) Point cloud with 72,027 vertices. (b) Optimized control mesh with 4667 vertices. (c) Subdivision surface of (b). (d) Screened Poisson reconstructed mesh with 155,008 vertices. (e) Optimized control mesh with 314 vertices. (f) Subdivision surface of (e).

For the more detailed and complicated Lucy model (Figure 6.2), with only 3% of the Screened Poisson reconstruction vertices, we could retain most of the intricate objects and folds.

In Figure 6.3 we show the reconstruction of a koala toy. The physical scanned model is furry so while the original reconstruction is very detailed it also contained a lot of noise. With only 0.2% of the original number of vertices and 0.8% of Screened Poisson reconstruction vertices, we provide a reconstruction that retains the shape and many of the important details.

The performance of the IMLS distance depends on the number of sampled points on the subdivision surface that we use in the computation. By default in all our examples we only use the points in the control mesh. However, it is possible to select more samples. Figure 6.4 shows this trade off. Figure 6.4 (a) is the reconstruction of the Lucy model using only the vertices in the original control mesh. Figure 6.4 (b) is the reconstruction using the vertices obtained after one level of subdivision (i.e. four times more). The result is slightly improved, some areas contain more detail, but the optimization takes about three times as long.

6.2 Spatial-temporal fitting

We tested our spatial-temporal method on two sequences: a synthetic sequence generated using a cloth simulation of a T-Shirt in Blender Hess (2010), and a spatial-temporal capture of a cow toy using a multi-view stereo setup Bradley, Popa, Sheffer, Heidrich, and Boubekeur (2008). Both sequences have 30 frames and, in both cases, we made a template from the first frame. The results for both sequences are shown in Figure 6.5, where the yellow mesh with wireframe is the template and the pink mesh is the fitted subdivision surface for spatial-temporal sequences.

For the cloth sequence we used for simulation a mesh of 2000 vertices that we randomly re-sampled in every frame to simulate a real capture to $100,000$ vertices . The template mesh has 2046 vertices(or 2% of the total vertices). For the puppet sequence the target mesh has around $123,000$ vertices and the template mesh of 1252 vertices (1% of the total vertices).

The settings for the DR are adapted from the PyTorch3D Ravi et al. (2020) tutorial. We used Soft Silhouette shader whose image size is 256×256, blur radius is $log(1/(1e^{-4} - 1) * 1e^{-4})$ and faces per pixel is 100. When rendering the target shape, we had 20 different camera views in total. However, in every iteration, we only randomly select 2 views to render the images of template to reduce unnecessary rendering time. The T-Shirt sequence has a lot of geometric details that is well preserved in the reconstruction. In contrast, the puppet sequence has less detail and in some cases some reconstruction artifacts (see Figure 6.7 (f)) stay fixed in the reconstruction due to the continuity properties of the subdivision surfaces.

In Figures 6.6 and 6.7 we compare the IMLS fitting scheme with the DR fitting scheme. Using the DR fitting scheme by itself results in the loss of a lot of details: Figures 6.6 (e), 6.7 (e), (i) This is not unexpected as we only use the silhouette loss. However, the geometric detail between IMLS and IMLS+DR is very similar (Figures 6.6 (c), (d), Figures 6.7 (c), (d), Figures 6.7 (g), (h)). The main gain from adding the DR term is the reduced drift (Figure 6.8). We also perform a quantitative evaluation using the Hausdorff distance between the target mesh and the subdivision surface. For the subdivision surface, we computed the Hausdorff distance using 3 iterations of subdivision. Results are presented in Figure 6.9. The combination of IMLS + DR largely outperforms either of them used separately. Figure 6.10 shows the execution time of our method.

6.3 Implementation detail

This project was implemented in 4 main function modules: Loop subdivision evaluation, Energy(IMLS and ARAP) function, Auto-differentiation and differentiable rendering. Given an input template mesh, the Loop subdivision evaluation module compute the limited position of the input mesh and compute the constant Jacobian matrix of the input mesh. The energy function module was used to compute the energy in every iteration during the optimization process and the derivative of the energy was computed by the auto-differentiation module. These three modules were implemented by C++. We used Libigl Jacobson, Panozzo, et al. (2018) to do some basic geometry processing, such as generating Loop subdivision matrix and the cotangent matrix. When computing IMLS energy, we need to search for the neighbor points, which was done by libANN, a library for approximate nearest neighbor searching Arya and Mount (1998). What's more, we used Eigen as our basic matrix operation library Guennebaud, Jacob, et al. (2010).

Since the DR we used is from PyTorch3D which implemented by python, we used the torch C++ interface to achieve the interaction between our main C++ code and the Differentiable rendering module. We used the CustomClassHolder in torch library to write an interface file, which can be compiled as a PyTorch library that the function in it can be called in PyTorch. Now, we can call the Loop subdivision evaluation, Energy(IMLS and ARAP) function and Auto-differentiation modules in PyTorch to work with DR and optimized by the PyTorch Adam optimizer.

As for the results, the code has been run on a computer with a CPU i9 12900 and GPU is RTX3060 Ti.

6.4 Other application of the Loop subdivision module

As shown in section 4.3, we modified Jos Stam's scheme to apply differentiable loop subdivision for meshes having adjacent extraordinary vertices, which is normal in real captured mesh. This subdivision surface evaluation module was also be applied in a text-to-mesh mesh generation pipeline. In Khalid et al. (2022), we presented a technique for zero-shot generation of a 3D model using only a target text prompt which called CLIP-Mesh. Without a generative model or any 3D supervision our method deforms a control shape of a limit subdivided surface along with

a texture map and normal map to obtain a 3D model asset that matches the input text prompt and can be deployed into games or modeling applications. The overview of this method is shown in Figure 6.11. Since the limited surface of Loop subdivision was guaranteed to be smooth, the generated results was improved visually by applying loop subdivision module. The comparison between generated mesh with and without limit subdivision module is shown in Figure 6.12, the generated mesh with subdivision module has less self-intersection and better continuity. The experiments also showed that adding the limit subdivision module can improve the quantative metric efficiently for all datasets Khalid et al. (2022).

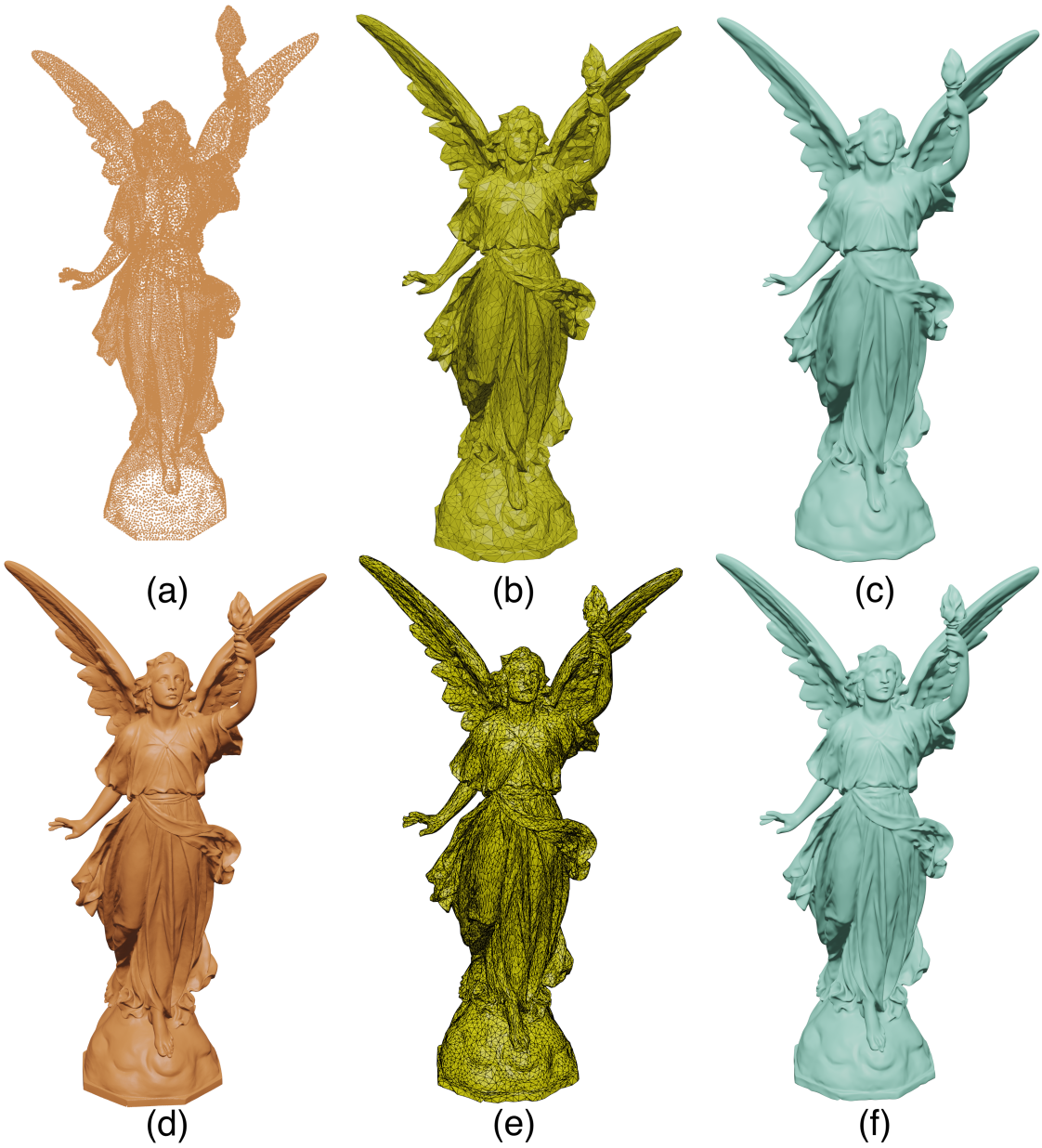

Figure 6.2: Stanford (n.d.) Lucy:(a) Point cloud with 49,987 vertices. (b) Optimized control mesh with 8002 vertices. (c) Subdivision surface of (b). (d) Screened Poisson reconstructed mesh with 262,909 vertices. (e) Optimized control mesh with 20,002 vertices. (f) Subdivision surface of (e).

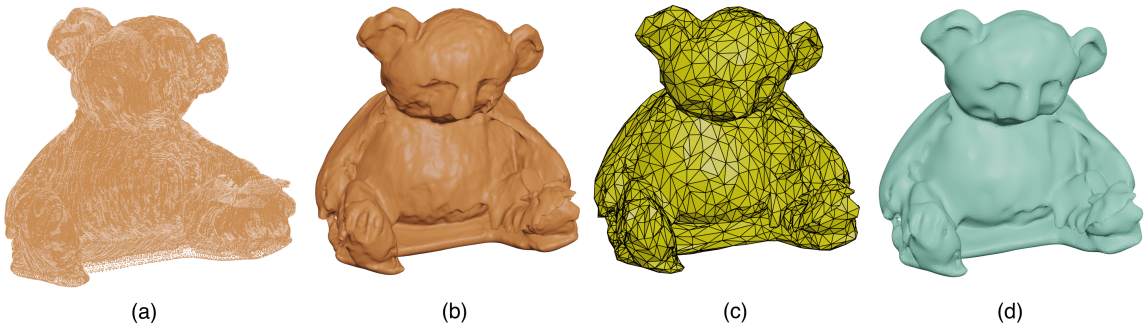

Figure 6.3: Kinect scanned koala:(a) Point cloud with 1,018,126 vertices. (b) Screened Poisson reconstructed mesh with 276,529 vertices. (c) Optimized control mesh with 2,465 vertices. (d) Subdivision surface of (c).

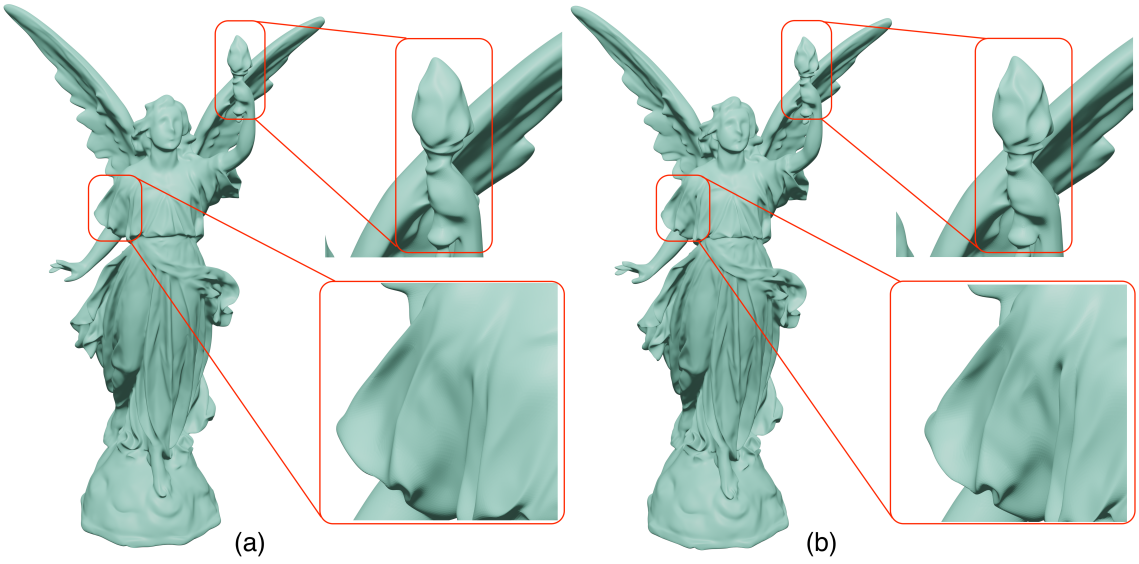

Figure 6.4: Comparison between (a) using only the control mesh vertices to compute the IMLS fit, and (b) using the vertices after one level of subdivision.

Figure 6.5: Results for spatial-temporal fitting, Top:T-shirt sequence. Bottom: Capture of a cow toy sequence.

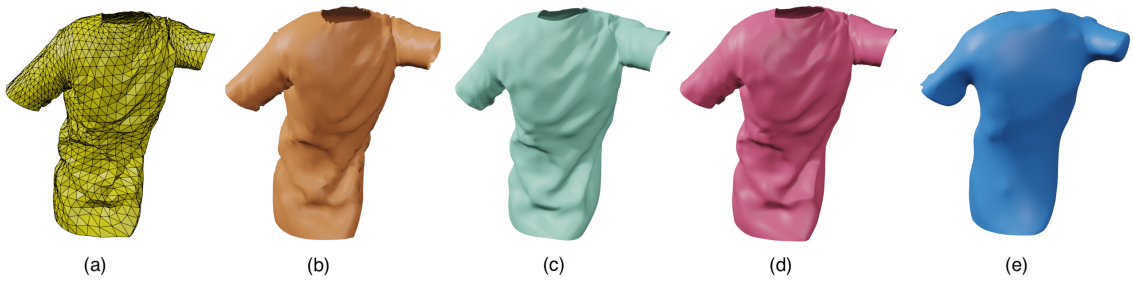

Figure 6.6: Fitting result for t-shirt simulation: (a) Optimized control mesh of using both IMLS and DR. (b) Simulation result from Blender Hess (2010). (c) Fitting result by only IMLS energy(Chapter 4). (d) Fitting result by combining IMLS and DR(section 5.2.1). (e) Fitting result by only DR.

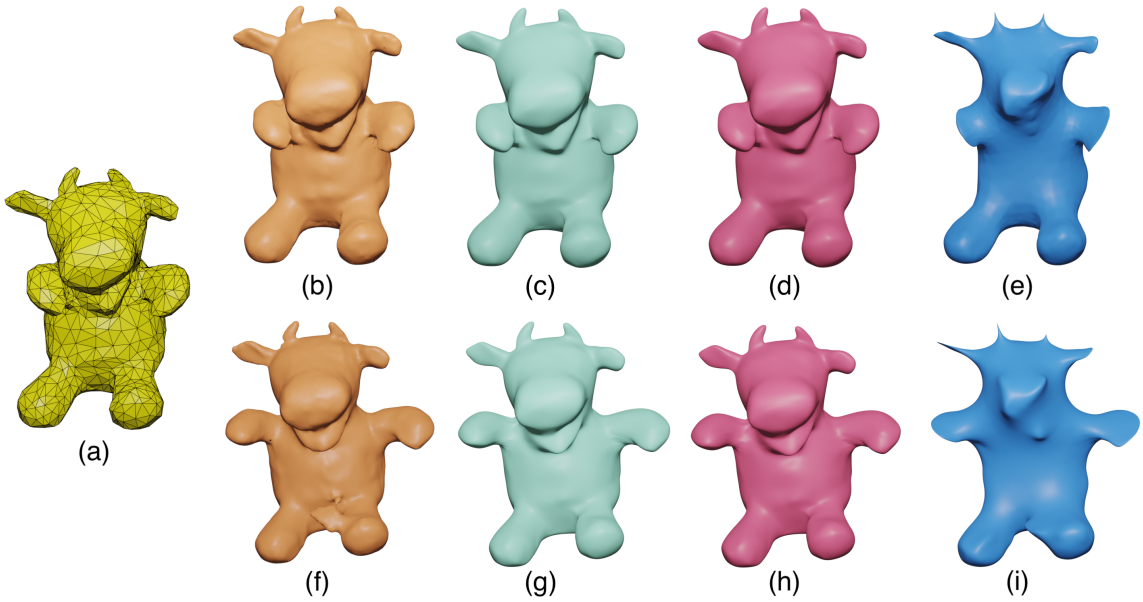

Figure 6.7: Fitting result for real scanned puppet. (a) Template control mesh with 1,252 vertices. (b) Reconstructed mesh with 123,234 vertices for start frame. (c)Fitted subdivision surface using IMLS energy(chapter 4) for start frame. (d) Fitted subdivision surface using combination of IMLS energy and DR(section 5.2.1) for start frame. (e) Fitting result only using DR for start frame. (f) Reconstructed mesh with 123,234 vertices for end frame. (g) Fitted subdivision surface using IMLS energy for end frame. (h) Fitted subdivision surface using combination of IMLS energy and DR for end frame. (i) Fitting result only using DR for end frame.

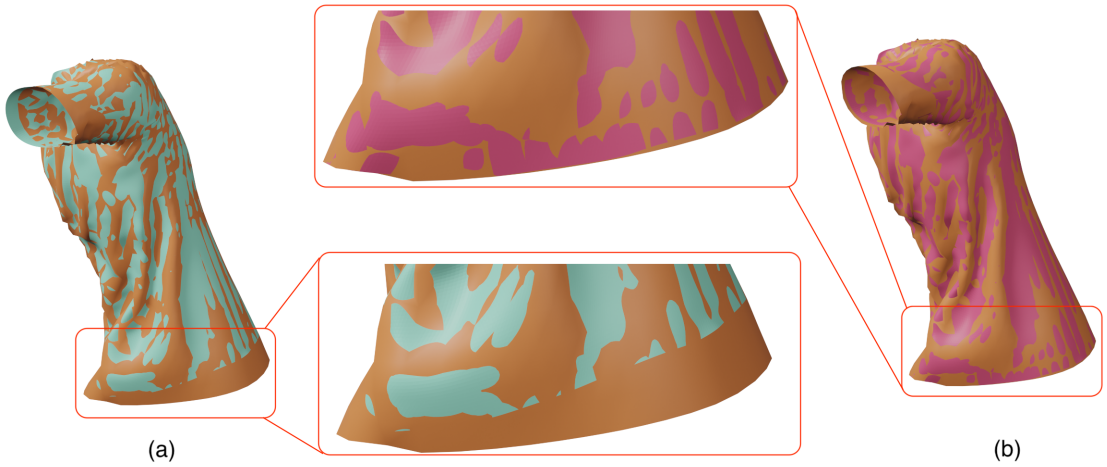

Figure 6.8: Comparison between the result of T-shirt data fitting. Brown color is the target. Green (a) using the geometric IMLS fit. Red (b) combining the geometric IMLS fit together with the image loss from the differential renderer. Note the drift in (a) at the bottom of the T-Shirt.

	T-shirt		Puppet	
	Mean	RMS	Mean	RMS
IMLS	0.003673	0.007482	**0.000839**	0.002866
IMLS+DR	**0.003045**	**0.005374**	0.000868	**0.002818**
DR	0.005955	0.008770	0.013661	0.017071

Figure 6.9: Hausdorff distance between fitting result and target shape(w.r.t bounding box diagonal)

	Computing Jacobian Matrix	Optimization /frame	Total/ frame
Bunny 314	0.03	3.19	4.06
Bunny 4667	6.37	35.44	42.74
Lucy 8002	22.04	32.35	55.89
T-shirt	1.70	16.38	17.04
Puppet	0.50	10.59	12.85

Figure 6.10: Running time (in seconds) for fitting subdivision surface to static model and fitting subdivision surface to spatial-temporal sequence(30 frames)

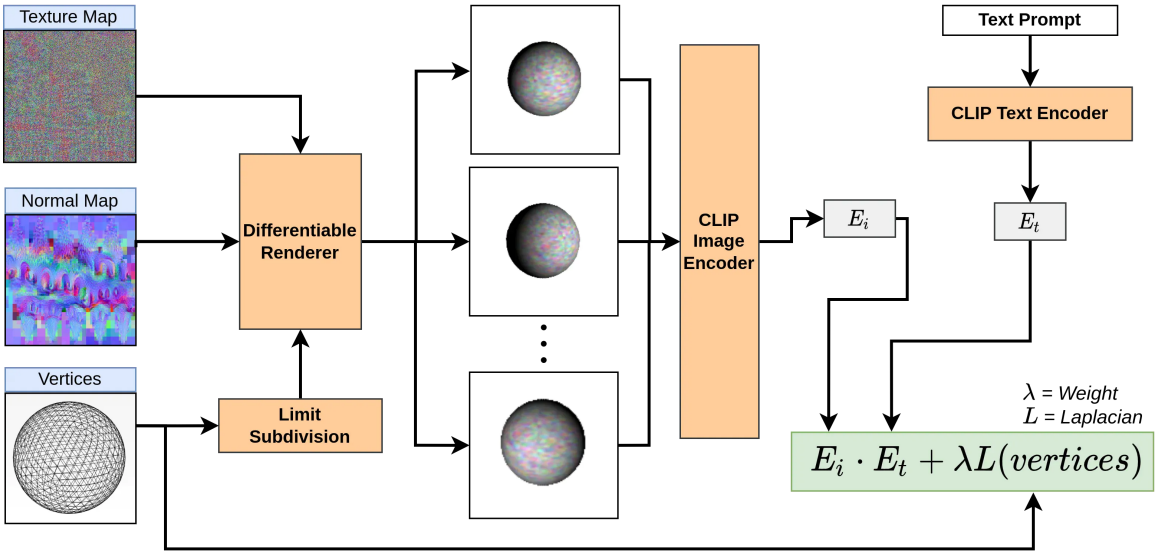

Figure 6.11: An overview of CLIP-Mesh Khalid et al. (2022)

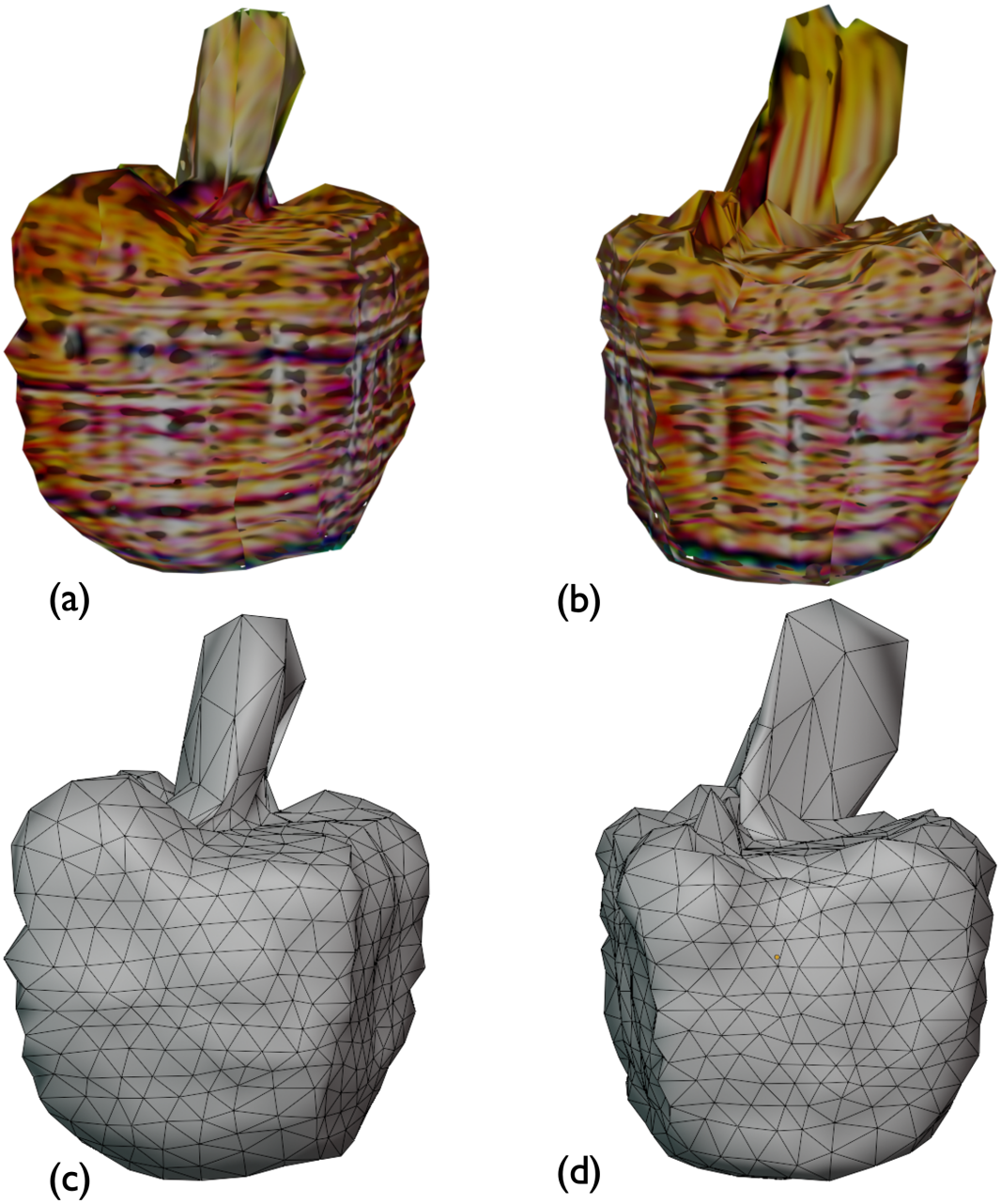

Figure 6.12: Comparison between generated mesh with and without limit subdivision module Khalid et al. (2022): (a)Generated Fruit Basket with subdivision module. (b)Generated Fruit Basket without subdivision module. (c)Topology of the generated mesh with subdivision module. (d)Topology of the generated mesh without subdivision module.

Chapter 7

Conclusion, Limitations and Future work

In this paper, we present a novel differentiable method to fit a Loop subdivision surface to a point cloud. Because our fitting method is differentiable, it can be easily integrated into deep learning-based methods for different applications. We demonstrate our method on several static point clouds as well as spatial-temporal shape sequences. The results show that our method does well in preserving surface detail while still being very compact, requiring only a small fraction (between 1% and 3%) of the data, in comparison to reconstruction methods such as Screened Poisson.

However, our method has some limitations. The spatial-temporal reconstruction relies on a differentiable renderer and the ones currently available only support mesh format. Therefore, for the spatial-temporal examples, we had to reconstruct a triangular mesh from each static point cloud.

Since the IMLS energy is based on nearest neighbor search, the optimization may fail when the distance between template control mesh and the target shape is too large or there are some holes in the point cloud. Thus, especially in the case of spatial-temporal examples, the frame-to-frame motion of the data must be relatively small. For the spatial-temporal fitting, if some frames have holes, sometimes the subdivision surface will be fitted to the wrong surface because of the wrong neighbors searched. Other than that, if the IMLS can't be built around the holes' areas due to the lack of enough neighbors, this area will keep same with the template. In the future, we would like

to explore a fitting energy can be applied even when template control mesh is far away the target shape.

Since we focused on geometric fitting, we selected an image loss based on silhouette only. In the future, it would be of interest to explore other image losses and point based differentiable renderers. What's more, we can also improve our method by using more accurate implicit surface reconstruction techniques from point-clouds such as the one proposed by Liu et al. (2021).

As for the implementation of our method, because the Loop subdivision evaluation, Energy(IMLS and ARAP) function and Auto-differentiation modules are implemented by C++, these modules can't be accelerated by GPU. Since the DR need to be ran on GPU, this leads to a wasting of transferring data between CPU and GPU. In the future, we would be interested in implementing the Loop subdivision evaluation module by CUDA, which can be accelerated by Nvidia GPU, to decrease the data transferring between CPU and GPU. What's more, the parallel structure of CUDA may give our method a great speed up.

Appendix A

Basis functions

$$
\begin{aligned}
b^T(v,w) = \frac{1}{12}(&u^4 + 2u^3v,\\
&u^4 + 2u^3w,\\
&u^4 + 2u^3w + 6u^3v + 6u^2vw + 12u^2v^2 + 6uv^2w + 6uv^3 + 2v^3w + v^4,\\
&6u^4 + 24u^3w + 24u^2w^2 + 8uw^3 + w^4 + 24u^3v + 60u^2vw + 36uvw^2 +\\
&6vw^3 + 24u^2v^2 + 36uv^2w + 12v^2w^2 + 8uv^3 + 6v^3w + v^4,\\
&u^4 + 6u^3w + 12u^2w^2 + 6uw^3 + w^4 + 2u^3v + 6u^2vw + 6uvw^2 + 2vw^3,\\
&2uv^3 + v^4,\\
&u^4 + 6u^3w + 12u^2w^2 + 6uw^3 + w^4 + 8u^3v + 36u^2vw + 36uvw^2 + 8vw^3 +\\
&24u^2v^2 + 60uv^2w + 24v^2w^2 + 24uv^3 + 24v^3w + 6v^4,\\
&u^4 + 8u^3w + 24u^2w^2 + 24uw^3 + 6w^4 + 6u^3v + 36u^2vw + 60uvw^2 +\\
&24vw^3 + 12u^2v^2 + 36uv^2w + 24v^2w^2 + 6uv^3 + 8v^3w + v^4,\\
&2uw^3 + w^4,\\
&2v^3w + v^4,\\
&2uw^3 + w^4 + 6uvw^2 + 6vw^3 + 6uv^2w + 12v^2w^2 + 2uv^3 + 6v^3w + v^4,\\
&w^4 + 2vw^3),
\end{aligned}
\tag{17}
$$

where $u = 1 - v - w$.

Appendix B

Derivative of IMLS fitting energy

If we denote $r_k = \|Q_i - P_k\|$, $\omega_k = \sum_k N_k(Q_i - P_k)\phi(r_k)$, the partial derivative of equation 11 can be expressed as:

$$\frac{\partial e_{dist}}{\partial P_i} = 2\sum_i \sum_k \frac{\omega_k}{\phi(r_k)} \frac{\frac{\partial \omega_k}{\partial P_i} \cdot \phi(r_k) - \omega_k \cdot \frac{\partial \phi(r_k)}{\partial P_i}}{\phi(r_k)^2}. \tag{18}$$

When we computing the $\frac{\partial E_{dist}}{\partial P_i}$, the $\frac{\partial E_{dist}}{\partial x_i}$, $\frac{\partial E_{dist}}{\partial y_i}$ and $\frac{\partial E_{dist}}{\partial z_i}$ are computed separately, where (x_i, y_i, z_i) is the Cartesian coordinate of P_i. In that case, the $\frac{\partial \omega_k}{\partial x_i}$ and $\frac{\partial \phi(r_k)}{\partial x_i}$ can be expressed as,

$$\frac{\partial \omega_k}{\partial x_i} = \sum_k N_k^x \cdot \phi(r_k) + \sum_k N_k \cdot (P_i - Q_k) \cdot \frac{\partial \phi(r_k)}{\partial x_i}, \tag{19}$$

where N_k^x is the x coordinate of N_k.

$$\frac{\partial \phi(r_k)}{\partial x_i} = -\frac{8}{h^2} \cdot (1 - \frac{r_k^2}{h^2})^3 \cdot (x - Q_k^x), \tag{20}$$

where Q_k^x is the x coordinate of Q_k. Similar to coordinate x, we can compute the derivative for y and z easily.

Appendix C

Solution for rotation matrix R and the derivative of E_{reg}

Suppose S_i is the co-variance matrix of the cell i, whose formula is:

$$S_i = \sum_{N(i)} w_{ij} e_{ij} e i j^T, \tag{21}$$

where e_{ij} is mesh edge(i, j). The SVD decomposition of S_i is

$$S_i = U_i \Sigma_i \Lambda_i^T. \tag{22}$$

Then the rotation matrix R_i can be estimated as,

$$R_i = \Lambda_i U_i^T. \tag{23}$$

After we get the rotation matrix R, the partial derivative of E_{reg} needed to be derived,

$$\begin{aligned}\frac{\partial E_{reg}}{\partial V_i^0} &= \frac{\partial}{V_i^0}(\sum_{j\in N(i)} w_{ij}\|(\bar{V_i^0} - \bar{V_j^0}) - R_i(V_i^0 - V_j^0)\|^2 \\ &+ \sum_{j\in N(i)} w_{ji}\|(\bar{V_i^0} - \bar{V_j^0}) - R_j(V_i^0 - V_j^0))\|^2) \\ &= \sum_{j\in N(i)} 2w_{ij}\left[(\bar{V_i^0} - \bar{V_j^0}) - R_i(V_i^0 - V_j^0)\right] \\ &- \sum_{j\in N(i)} 2w_{ji}\left[(\bar{V_j^0} - \bar{V_i^0}) - R_j(V_j^0 - V_i^0)\right].\end{aligned} \tag{24}$$

Because the cotangent matrix is symmetric, $w_{ij} = w_{ji}$, the derivative can be expressed as,

$$\frac{\partial E_{reg}}{\partial V_i^0} = \sum_{j\in N(i)} 4w_{ij}\left[(\bar{V_i^0} - \bar{V_j^0}) - \frac{1}{2}(R_i + R_j)(V_i^0 - V_j^0)\right]. \tag{25}$$

References

Agarwal, S., Mierle, K., & Others. (n.d.). *Ceres solver.* http://ceres-solver.org.

Allen, B., Curless, B., & Popović, Z. (2002). Articulated body deformation from range scan data. *ACM Transactions on Graphics (TOG)*, *21*(3), 612–619.

Arya, S., & Mount, D. M. (1998). Ann: library for approximate nearest neighbor searching..

Bajaj, C. L. (1992). Chapter 2: Surface fitting using implicit algebraic surface patches. In H. Hagen (Ed.), *Topics in surface modeling* (pp. 23–52). SIAM. Retrieved from https://doi.org/10.1137/1.9781611971644.ch2 doi: 10.1137/1.9781611971644.ch2

Bartels, R. H., Beatty, J. C., & Barsky, B. A. (1987). *An introduction to splines for use in computer graphics and geometric modeling*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Ben-Shabat, Y., & Gould, S. (2020). Deepfit: 3d surface fitting via neural network weighted least squares. In *European conference on computer vision* (pp. 20–34).

Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Guennebaud, G., Levine, J. A., ... Silva, C. T. (2017). A survey of surface reconstruction from point clouds. In *Computer graphics forum* (Vol. 36, pp. 301–329).

Bozic, A., Palafox, P., Zollhöfer, M., Dai, A., Thies, J., & Nießner, M. (2020). Neural non-rigid tracking. *Advances in Neural Information Processing Systems*, *33*, 18727–18737.

Bradley, D., Popa, T., Sheffer, A., Heidrich, W., & Boubekeur, T. (2008). Markerless garment capture. In *Acm siggraph 2008 papers.* New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1399504.1360698 doi: 10.1145/1399504.1360698

Catmull, E., & Clark, J. (1978). Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided design*, *10*(6), 350–355.

Cheng, K.-S., Wang, W., Qin, H., Wong, K.-Y., Yang, H., & Liu, Y. (2004). Fitting subdivision surfaces to unorganized point data using sdm. In *12th pacific conference on computer graphics and applications, 2004. pg 2004. proceedings.* (pp. 16–24).

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., & Ranzuglia, G. (2008). MeshLab: an Open-Source Mesh Processing Tool. In V. Scarano, R. D. Chiara, & U. Erra (Eds.), *Eurographics italian chapter conference.* The Eurographics Association. doi: 10 .2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136

DeRose, T., Kass, M., & Truong, T. (1998). Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on computer graphics and interactive techniques* (pp. 85–94).

Doo, D., & Sabin, M. (1978). Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, *10*(6), 356–360.

Estellers, V., Schmidt, F., & Cremers, D. (2018). Robust fitting of subdivision surfaces for smooth shape analysis. In *2018 international conference on 3d vision (3dv)* (pp. 277–285).

Garland, M., & Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on computer graphics and interactive techniques* (pp. 209–216).

Guennebaud, G., Jacob, B., et al. (2010). *Eigen v3.* http://eigen.tuxfamily.org.

Hess, R. (2010). *Blender foundations: The essential guide to learning blender 2.6.* Focal Press.

Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., ... Stuetzle, W. (1994). Piecewise smooth surface reconstruction. In *Proceedings of the 21st annual conference on computer graphics and interactive techniques* (pp. 295–302).

Ilic, S. (2006). Using subdivision surfaces for 3-d reconstruction from noisy data. In *Workshop on image registration in deformable environments (deform)* (pp. 1–10).

Jacobson, A., Panozzo, D., et al. (2018). *libigl: A simple C++ geometry processing library.* (https://libigl.github.io/)

Kähler, K., Haber, J., Yamauchi, H., & Seidel, H.-P. (2002). Head shop: Generating animated head

models with anatomical structure. In *Proceedings of the 2002 acm siggraph/eurographics symposium on computer animation* (pp. 55–63).

Kanazawa, A., Tulsiani, S., Efros, A. A., & Malik, J. (2018). Learning category-specific mesh reconstruction from image collections. In *Proceedings of the european conference on computer vision (eccv)* (pp. 371–386).

Kato, H., Beker, D., Morariu, M., Ando, T., Matsuoka, T., Kehl, W., & Gaidon, A. (2020). Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*.

Kato, H., & Harada, T. (2019). Learning view priors for single-view 3d reconstruction. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 9778–9787).

Kazhdan, M., & Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, *32*(3), 1–13.

Khalid, N., Xie, T., Belilovsky, E., & Popa, T. (2022). *Text to mesh without 3d supervision using limit subdivision.* arXiv. Retrieved from https://arxiv.org/abs/2203.13333 doi: 10.48550/ARXIV.2203.13333

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

Kolluri, R. K. (2008). Provably good moving least squares. *ACM Trans. Algorithms*, *4*(2), 18:1–18:25. Retrieved from https://doi.org/10.1145/1361192.1361195 doi: 10.1145/1361192.1361195

Lai, M.-J. (1992). Fortran subroutines for b-nets of box splines on three-and four-directional meshes. *Numerical Algorithms*, *2*(1), 33–38.

Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., & Aila, T. (2020). Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (TOG)*, *39*(6), 1–14.

Lavoué, G., Dupont, F., & Baskurt, A. (2005). Subdivision surface fitting for efficient compression and coding of 3d models. In *Visual communications and image processing 2005* (Vol. 5960, pp. 1159–1170).

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

Litke, N., Levin, A., & Schroder, P. (2001). Fitting subdivision surfaces. In *Proceedings visualization, 2001. vis'01.* (pp. 319–568).

Liu, S.-L., Guo, H.-X., Pan, H., Wang, P.-S., Tong, X., & Liu, Y. (2021). Deep implicit moving least-squares functions for 3d reconstruction. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 1788–1797).

Loop, C. (1987). *Smooth subdivision surfaces based on triangles* (Unpublished master's thesis). University of Utah, USA.

Ma, W., Ma, X., Tso, S.-K., & Pan, Z. (2002). Subdivision surface fitting from a dense triangle mesh. In *Geometric modeling and processing. theory and applications. gmp 2002. proceedings* (p. 94-103). doi: 10.1109/GMAP.2002.1027500

Marinov, M., & Kobbelt, L. (2005). Optimization methods for scattered data approximation with subdivision surfaces. *Graphical Models*, *67*(5), 452–473.

Mendhurwar, K., Handa, G., Zhu, L., Mudur, S., Beauchesne, E., LeVangie, M., ... Popa, T. (2020). A system for acquisition and modelling of ice-hockey stick shape deformation from player shot videos. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition workshops* (pp. 890–891).

Meyer, M., Desbrun, M., Schröder, P., & Barr, A. H. (2003). Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics iii* (pp. 35–57). Springer.

Mitra, N. J., Flory, S., Ovsjanikov, M., Gelfand, N., Guibas, L., & Pottmann, H. (2007). Dynamic geometry registration. In *Symposium on geometry processing* (pp. 173–182).

Montes, J., Thomaszewski, B., Mudur, S., & Popa, T. (2020). Computational design of skintight clothing. *ACM Transactions on Graphics (TOG)*, *39*(4), 105–1.

Newcombe, R. A., Fox, D., & Seitz, S. M. (2015). Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 343–352).

Öztireli, A. C., Guennebaud, G., & Gross, M. (2009). Feature preserving point set surfaces based on non-linear kernel regression. In *Computer graphics forum* (Vol. 28, pp. 493–501).

Piegl, L., & Tiller, W. (1996). *The nurbs book* (second ed.). New York, NY, USA: Springer-Verlag.

Popa, T., South-Dickinson, I., Bradley, D., Sheffer, A., & Heidrich, W. (2010). Globally consistent

space-time reconstruction. In *Computer graphics forum* (Vol. 29, pp. 1633–1642).

Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.-Y., Johnson, J., & Gkioxari, G. (2020). Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*.

Sharf, A., Alcantara, D. A., Lewiner, T., Greif, C., Sheffer, A., Amenta, N., & Cohen-Or, D. (2008). Space-time surface reconstruction using incompressible flow. *ACM Transactions on Graphics (TOG)*, *27*(5), 1–10.

Sharma, G., Liu, D., Maji, S., Kalogerakis, E., Chaudhuri, S., & Měch, R. (2020). Parsenet: A parametric surface fitting network for 3d point clouds. In *European conference on computer vision* (pp. 261–276).

Shinya, M. (2004). Unifying measured point sequences of deforming objects. In *Proceedings. 2nd international symposium on 3d data processing, visualization and transmission, 2004. 3dpvt 2004.* (pp. 904–911).

Sorkine, O., & Alexa, M. (2007). As-rigid-as-possible surface modeling. In *Symposium on geometry processing* (Vol. 4, pp. 109–116).

Stam, J. (1998). Evaluation of loop subdivision surfaces. In *Siggraph'98 cdrom proceedings.*

Stanford. (n.d.). *The stanford 3d scanning repository.* http://graphics.stanford.edu/data/3Dscanrep/.

Stoll, C., Karni, Z., Rössl, C., Yamauchi, H., & Seidel, H.-P. (2006). Template deformation for point cloud fitting. In *Pbg@ siggraph* (pp. 27–35).

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

Süßmuth, J., Winter, M., & Greiner, G. (2008). Reconstructing animated meshes from time-varying point clouds. In *Proceedings of the symposium on geometry processing* (pp. 1469–1476).

Tevs, A., Berner, A., Wand, M., Ihrke, I., Bokeloh, M., Kerber, J., & Seidel, H.-P. (2012). Animation cartography—intrinsic reconstruction of shape and motion. *ACM Transactions on Graphics (TOG)*, *31*(2), 1–15.

Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *13*(04), 376–380.

Wand, M., Adams, B., Ovsjanikov, M., Berner, A., Bokeloh, M., Jenke, P., … Schilling, A. (2009).

Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Transactions on Graphics (TOG)*, *28*(2), 1–15.

Wang, W., Pottmann, H., & Liu, Y. (2006). Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics (ToG)*, *25*(2), 214–238.

Yumer, M. E., & Kara, L. B. (2012). Surface creation on unstructured point sets using neural networks. *Computer-Aided Design*, *44*(7), 644–656.

Zheng, W., Bo, P., Liu, Y., & Wang, W. (2012). Fast b-spline curve fitting by l-bfgs. *Computer Aided Geometric Design*, *29*(7), 448–462.

Zollhöfer, M., Nießner, M., Izadi, S., Rehmann, C., Zach, C., Fisher, M., ... others (2014). Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (ToG)*, *33*(4), 1–12.