Differentiable Subdivision Surface Fitting

Tianhao Xie
Brian A. Barsky
Sudhir Mudur
Tiberiu Popa

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-14
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-14.html

January 25, 2023

Differentiable Subdivision Surface Fitting

Tianhao Xie*
Concordia University

Brian Barsky†
University of California, Berkeley

Sudhir Mudur‡
Concordia University

Tiberiu Popa§
Concordia University

Figure 1: Fitting subdivision surface to spatial-temporal sequences

ABSTRACT

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. 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.

1 INTRODUCTION

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 [3], bi-parametric surfaces using splines and NURBS [4,40] or subdivision surfaces [11,37]. 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 [6].

Among all these representations, subdivision surfaces are particularly appealing to many high level applications such as surface optimization and analysis, simulation, modeling, and animation [11]: not only they are very compact, they do not require explicit NURBS patch decomposition and alignment as NURBS do [44], 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 [47]; 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 [9, 13, 33] rely on an optimization function that uses iterative point-to-point

*e-mail: tianhao.xie@mail.concordia.ca
†e-mail: barsky@berkeley.edu
‡e-mail: mudur@cse.concordia.ca
§e-mail: tiberiu.popa@concordia.ca

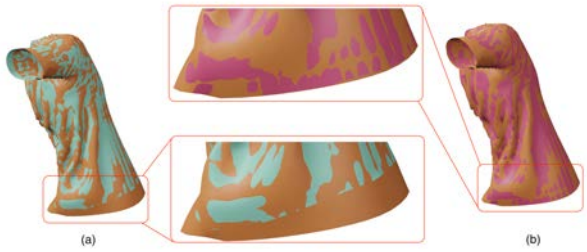

Figure 2: Comparison between the result of T-shirt data fitting. Brown color is the target. Green (a) using the geometric IMLS fit. Red (b) combining the geometric IMLS fit together with the image loss from the differential renderer. Note the drift in (a) at the bottom of the T-Shirt.

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 (see Figure 2). 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 [34, 41].

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.

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 (IMLS) [24] surface. 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.

2 Related work

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 [8, 11, 12, 30, 31]. In particular, Loop subdivision is a subdivision scheme based on quartic box spline on triangular meshes [31]. 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 [47], which was based on conversion from Box splines to B-Nets [25]. This analytical and differentiable evaluation makes this scheme ideal for differentiable shape optimization and we will use it in our novel subdivision fitting pipeline.

2.1 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 [55, 57]. 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.* [16] and Lavoue *et al.* [27] fit subdivision surfaces to CAD models by minimizing the squared distance energy. Litke *et al.* [29] used quasi-interpolation to fit Catmull-Clark subdivision surface to a given shape within a prescribed tolerance. Ma *et al.* [32] 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 [9, 13, 33, 34]. Cheng *et al.* [9] 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 [47]. Then, they solve a linear system of the control mesh variables to minimize the squared distance between the sample points and target shape. Marinov *et al.* [33] introduced an algorithm based on exact closest point search on Loop surfaces which combines Newton iteration and non-linear minimization. In more recent research, Esteller *et al.* [13] 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 [9] and [33], 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 [17] 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 used a neural network to generate NURBS from input point sets [56]. DeepFit incorporated a neural network to learn point-wise weights for weighted least squares polynomial surface fitting [5]. Sharma *et al.* described a method using neural networks to fit B-spline patches to input point cloud data [44]. 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.

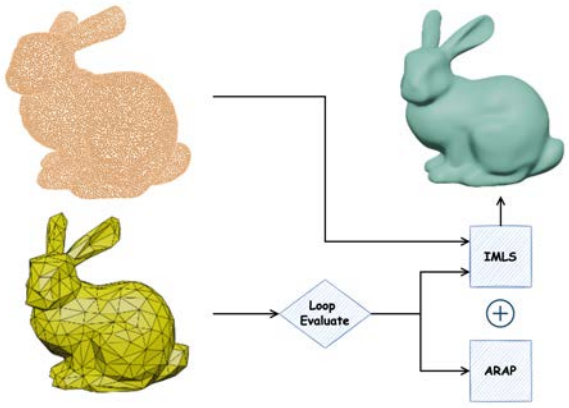

Figure 3: Overview of fitting subdivision surface to a static point cloud.

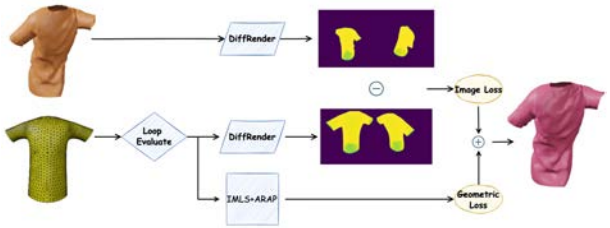

Figure 4: Overview of fitting subdivision surfaces to a spatial-temporal sequence by combining Implicit Moving Least Squares (IMLS) with differential rendering (DR) optimization

2.2 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 [2, 18, 49, 50]. 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 [45, 51]. In [51], Sussmuth *et al.* 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 [36, 41, 43]. In [36], Mitra *et al.* directly compute the motion of the scanned object in all frames and estimate the time-deforming object by kinematic properties. In [43], Sharf *et al.* used a space-time solid incompressible flow prior to reconstruct moving and deforming objects from point data. In [41] a template is constructed gradually by mapping consecutive frames in a pyramidal fashion. In [54], Wand *et al.* reconstructed 3D scanner data by pairwise scanning alignment. Tevs *et al.* [52] 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 [28, 38, 58]. Our method described next is distinct from all the above, specifically in our formulation using the IMLS surface as an intermediate for fitting.

3 Method 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 3 and 4. 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 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 [34], 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 [13] or other visual queues such as optical flow [7, 41]. Recently, with the development of differentiable renderers, rendered image difference metrics can be used to optimize shape [19]. 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.

3.1 Loop subdivision

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 [22] method in MeshLab [10]. We then simplify this triangulation using quadratic edge collapse [14] until we obtain the desired number of vertices requested by the user.

Although Loop subdivision can be evaluated iteratively, for optimization purposes it is desirable to have an analytical expression of the surface. Jos Stam [47] derived an analytical evaluation of the Loop subdivision surface of any point on the control mesh, but the

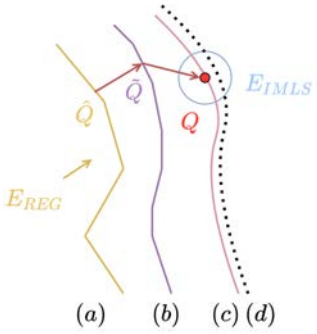

Figure 5: A schematic view of our optimization process. (a) Control mesh (M^0). (b) Control mesh after one level of subdivision (M^1). The vertices of this mesh are the Loop subdivision control points. (c) Loop subdivision surface. (d) Target point cloud. We optimize for M^0 by using an IMLS fitted to the point cloud and an ARAP regularizer on the control mesh M^0.

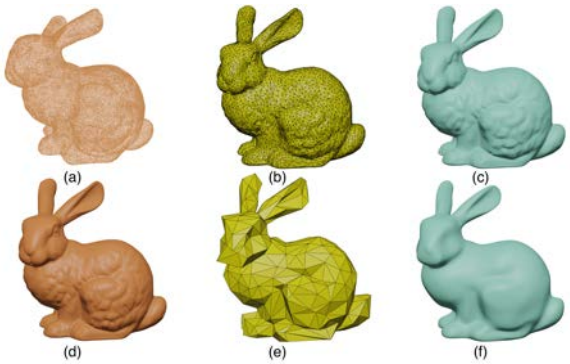

Figure 6: Stanford Bunny [48]:(a) Point cloud with 72,027 vertices. (b) Optimized control mesh with 4667 vertices. (c) Subdivision surface of (b). (d) Screened Poisson reconstructed mesh with 155,008 vertices. (e) Optimized control mesh with 314 vertices. (f) Subdivision surface of (e).

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 $M^1(V^1,E^1)$. It can be easily proven that 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. Therefore, our control mesh for analytical evaluation of the subdivision surface is M^1, but we only optimize for the vertices of M^0. This process is illustrated in Figure 5. 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 [31]. 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 [47]. 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.

4 Subdivision Surface Fitting

4.1 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:

$$\min_{V^0} E_{dist}(L(M^0,\hat{Q}))+\alpha\cdot E_{reg}(M^0,\bar{M^0}) \tag{1}$$

where $E_{dist}(\cdot)$ is the IMLS fit energy [39] (eq. 4), $L(\cdot)$ 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 [46] (eq. 5) and α is the weight of the regularizer term. The overview of the fitting model is shown in Figure 3.

IMLS fit energy Oztireli *et al.* introduced an Implicit Moving Least Squares(IMLS) surface in [39], which gave us a definition for the point cloud surface

$$f(x)=\frac{\sum n_i^T(x-x_i)\phi_i(x)}{\sum\phi_i(x)} \tag{2}$$

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.

$$\phi(r)=(1-\frac{r^2}{h^2})^4, \tag{3}$$

We can use the implicit surface definition in equation 2 to derive a fit energy [37]

$$E_{dist}=\sum_i(\frac{\sum_k N_k^T(Q_i-P_k)\phi(\|Q_i-P_k\|)}{\sum_k\phi(\|Q_i-P_k\|)})^2, \tag{4}$$

where P_k and N_k are the 3D positions and normals of points in the input point cloud (Figure 5d) and Q_i are points on the subdivision surface sampled from the control mesh (Figure 5a-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 9.

Regularizer We experimented with several reguralizers and the As-Rigid-As-Possible (ARAP) regularizer [46] yields the best results. The ARAP regularizer does not penalize any isometric deformations allowing local rotations, but it penalizes local stretch. More specifically:

$$E_{reg}=\sum_i\sum_{j\in N(i)} w_{ij}\|(V_i^0-V_j^0)-R_i(\bar{V_i^0}-\bar{V_j^0})\|^2, \tag{5}$$

where $N(i)$ is the set of vertices adjacent to V_i^0, $\bar{V_i^0}$ 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 [35]. At every iteration R_i can be computed analytically using SVD decomposition on the local co-variance matrix [53].

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 [1]. 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.

4.2 Fitting a sptial-temporal model

Differentiable rendering The emergence of differentiable rendering (DR) [26,42] 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 [19–21,42]. In image space, DR based

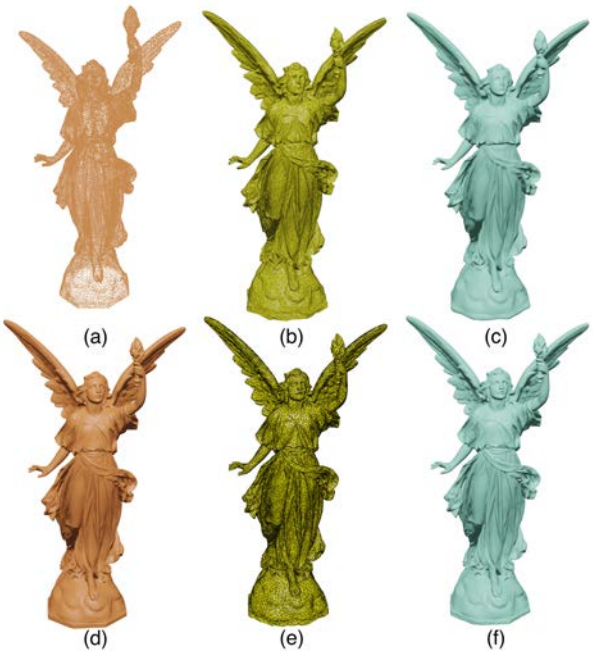

Figure 7: Stanford Lucy [48]:(a) Point cloud with 49,987 vertices. (b) Optimized control mesh with 8002 vertices. (c) Subdivision surface of (b). (d) Screened Poisson reconstructed mesh with 262,909 vertices. (e) Optimized control mesh with 20,002 vertices. (f) Subdivision surface of (e).

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.

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 2 (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 2 (b).

In every iteration's forward pass, we use the DR to render the target mesh in different camera positions k which give us target images I^k_{TARGET} 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. Suppose the number of pixels for rendered images is N, we compute image loss l_{image} by

$$l_{image} = \sum_k \frac{(I^k_{PRED} - I^k_{TARGET})^2}{N}. \quad (6)$$

As for the geometric loss, we compute it by the same method for computing energy provided in section 3. Thus, the total loss l_{total} is

$$l_{total} = E_{dist}(L(M^0), \hat{Q}) + \alpha \cdot E_{reg}(M^0, \bar{M^0}) + \beta \cdot l_{image}. \quad (7)$$

In our implementation we use the DR available in PyTorch3D [42] and for the backward pass, we use the gradient descent method, Adam optimizer [23], to optimize the control mesh.

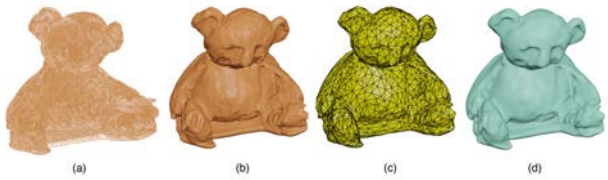

Figure 8: Kinect scanned koala:(a) Point cloud with 1,018,126 vertices. (b) Screened Poisson reconstructed mesh with 276,529 vertices. (c) Optimized control mesh with 2,465 vertices. (d) Subdivision surface of (c).

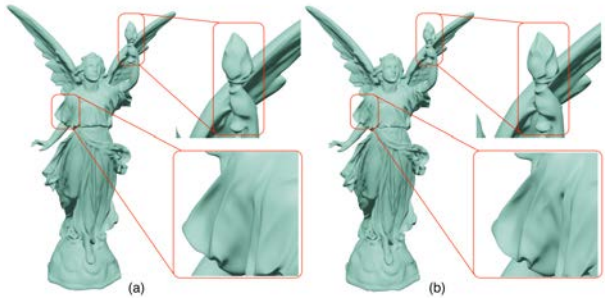

Figure 9: Comparison between (a) using only the control mesh vertices to compute the IMLS fit, and (b) using the vertices after one level of subdivision.

5 Results and Discussion

5.1 Static surface fitting

We used our method to fit a number of synthetic models (the Stanford Bunny and Lucy, see Figures 6 and 7 as well as a point cloud acquired using the Microsoft Azure Kinect device (a Koala toy) Figure 8 The starting searching radius h_0 and weight of ARAP regularizer α were selected manually. 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) the original point cloud has $72,027$ vertices and the reconstructed mesh using Screened Poisson [22] has $155,008$ vertices. We demonstrate two reconstructions. The first one with a template mesh of 4667 vertices (Figure 6 (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 (f)). While a number of details are lost, the main shape is still reconstructed fairly well.

For the more detailed and complicated Lucy model (Figure 7), with only 3% of the Screened Poisson reconstruction vertices, we could retain most of the intricate objects and folds.

In Figure 8 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

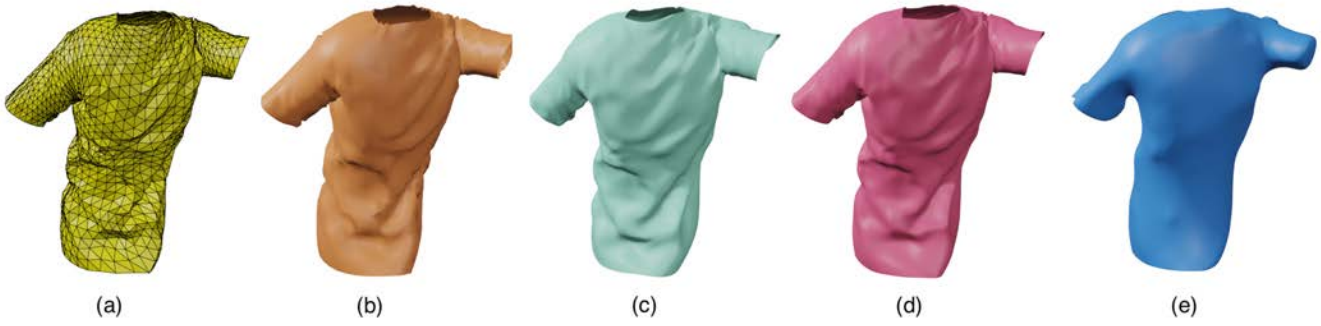

Figure 10: Fitting result for t-shirt simulation: (a) Optimized control mesh of using both IMLS and DR. (b) Simulation result from Blender [15]. (c) Fitting result by only IMLS energy(section 3). (d) Fitting result by combining IMLS and DR(section 4.2). (e) Fitting result by only DR.

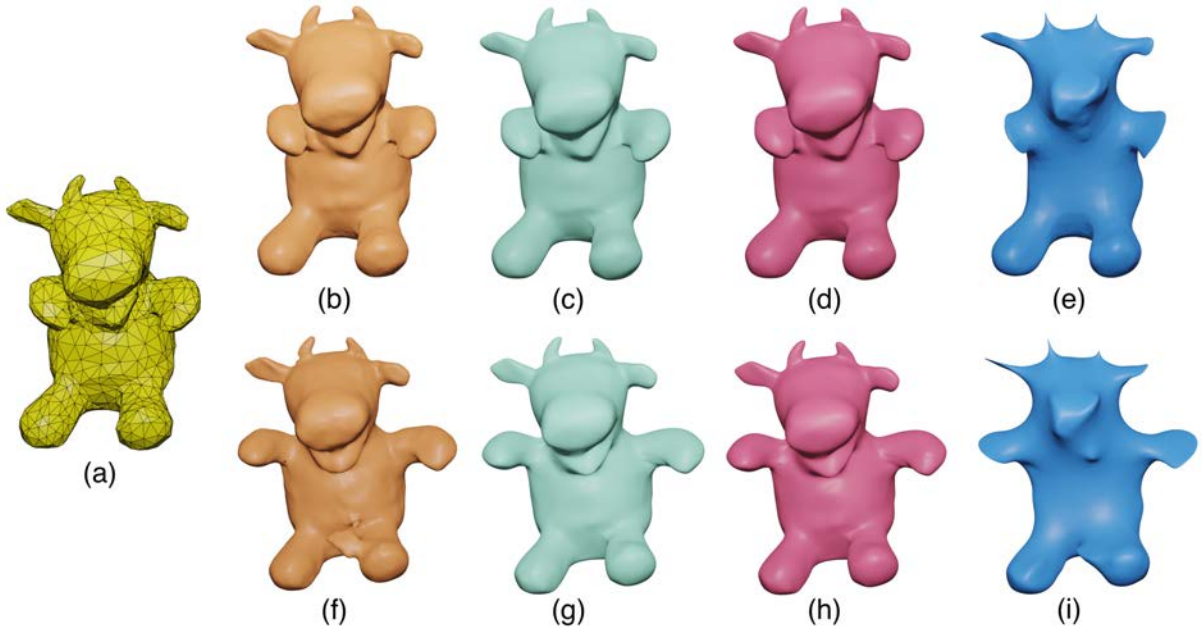

Figure 11: Fitting result for real scanned puppet. (a) Template control mesh with 1,252 vertices. (b) Reconstructed mesh with 123,234 vertices for start frame. (c)Fitted subdivision surface using IMLS energy(section 3 for start frame. (d) Fitted subdivision surface using combination of IMLS energy and DR(section 4.2) 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.

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 9 shows this trade off. Figure 9 (a) is the reconstruction of the Lucy model using only the vertices in the original control mesh. Figure 9 (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.

5.2 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 [15], and a spatial-temporal capture of a cow toy using a multi-view stereo setup. Both sequences have 30 frames and in both cases we made a template from the first frame. 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 (or 2% of the total vertices). The template mesh has 2046 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 [42] 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 11 (f)) stay fixed in the reconstruction due to the continuity properties of the subdivision surfaces.

In Figures 10 and 11 we compare the IMLS fitting scheme with the DR fitting scheme. Using the DR fitting scheme by itself results

	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 12: 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 13: Running time (in seconds) for fitting subdivision surface to static model and fitting subdivision surface to spatial-temporal sequence(30 frames)

in the loss of a lot of details: Figures 10 (e), 11 (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 10 (c), (d), Figures 11 (c), (d), Figures 11 (g), (h)). The main gain from adding the DR term is the reduced drift (Figure 2). 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 12. The combination of IMLS + DR largely outperforms either of them used separately. Figure 13 shows the execution time of our method. The code has been run on a computer with a CPU i9 12900 and GPU is rtx3060 ti.

6 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 differential 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. Thus, especially in the case of spatial-temporal examples, the frame-to-frame motion of the data must be relatively small.

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 differential renderers. In the future, 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. [30].

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

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

[3] C. L. Bajaj. Chapter 2: Surface fitting using implicit algebraic surface patches. In H. Hagen, ed., *Topics in Surface Modeling*, pp. 23–52. SIAM, 1992. doi: 10.1137/1.9781611971644.ch2

[4] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[5] Y. Ben-Shabat and S. Gould. Deepfit: 3d surface fitting via neural network weighted least squares. In *European Conference on Computer Vision*, pp. 20–34. Springer, 2020.

[6] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva. A survey of surface reconstruction from point clouds. In *Computer Graphics Forum*, vol. 36, pp. 301–329. Wiley Online Library, 2017.

[7] A. Bozic, P. Palafox, M. Zollhöfer, A. Dai, J. Thies, and M. Nießner. Neural non-rigid tracking. *Advances in Neural Information Processing Systems*, 33:18727–18737, 2020.

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

[9] K.-S. Cheng, W. Wang, H. Qin, K.-Y. Wong, H. Yang, and Y. Liu. 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. IEEE, 2004.

[10] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In V. Scarano, R. D. Chiara, and U. Erra, eds., *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. doi: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136

[11] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 85–94, 1998.

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

[13] V. Estellers, F. Schmidt, and D. Cremers. Robust fitting of subdivision surfaces for smooth shape analysis. In *2018 International Conference on 3D Vision (3DV)*, pp. 277–285. IEEE, 2018.

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

[15] R. Hess. *Blender Foundations: The Essential Guide to Learning Blender 2.6*. Focal Press, 2010.

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

[17] S. Ilic. Using subdivision surfaces for 3-d reconstruction from noisy data. In *Workshop on Image Registration in Deformable Environments (DEFORM)*, pp. 1–10. Citeseer, 2006.

[18] K. Kähler, J. Haber, H. Yamauchi, and H.-P. Seidel. Head shop: Generating animated head models with anatomical structure. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 55–63, 2002.

[19] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 371–386, 2018.

[20] H. Kato, D. Beker, M. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020.

[21] H. Kato and T. Harada. Learning view priors for single-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9778–9787, 2019.

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

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

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

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

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

[27] G. Lavoué, F. Dupont, and A. Baskurt. Subdivision surface fitting for efficient compression and coding of 3d models. In *Visual Communications and Image Processing 2005*, vol. 5960, pp. 1159–1170. SPIE, 2005.

[28] 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.

[29] N. Litke, A. Levin, and P. Schroder. Fitting subdivision surfaces. In *Proceedings Visualization, 2001. VIS'01.*, pp. 319–568. IEEE, 2001.

[30] S.-L. Liu, H.-X. Guo, H. Pan, P.-S. Wang, X. Tong, and Y. Liu. 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, 2021.

[31] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, USA, 1987.

[32] W. Ma, X. Ma, S.-K. Tso, and Z. Pan. Subdivision surface fitting from a dense triangle mesh. In *Geometric Modeling and Processing. Theory and Applications. GMP 2002. Proceedings*, pp. 94–103, 2002. doi: 10.1109/GMAP.2002.1027500

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

[34] K. Mendhurwar, G. Handa, L. Zhu, S. Mudur, E. Beauchesne, M. LeVangie, A. Hallihan, A. Javadtalab, and T. Popa. 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, 2020.

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

[36] N. J. Mitra, S. Flory, M. Ovsjanikov, N. Gelfand, L. Guibas, and H. Pottmann. Dynamic geometry registration. In *Symposium on Geometry Processing*, pp. 173–182, 2007.

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

[38] 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*, pp. 343–352, 2015.

[39] A. C. Öztireli, G. Guennebaud, and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. In *Computer graphics forum*, vol. 28, pp. 493–501. Wiley Online Library, 2009.

[40] L. Piegl and W. Tiller. *The NURBS Book*. Springer-Verlag, New York, NY, USA, second ed., 1996.

[41] T. Popa, I. South-Dickinson, D. Bradley, A. Sheffer, and W. Heidrich. Globally consistent space-time reconstruction. In *Computer Graphics Forum*, vol. 29, pp. 1633–1642. Wiley Online Library, 2010.

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

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

[44] G. Sharma, D. Liu, S. Maji, E. Kalogerakis, S. Chaudhuri, and R. Měch. Parsenet: A parametric surface fitting network for 3d point clouds. In *European Conference on Computer Vision*, pp. 261–276. Springer, 2020.

[45] M. Shinya. 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. IEEE, 2004.

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

[47] J. Stam. Evaluation of loop subdivision surfaces. In *SIGGRAPH'98 CDROM Proceedings*. Citeseer, 1998.

[48] Stanford. The stanford 3d scanning repository. `http://graphics.stanford.edu/data/3Dscanrep/`.

[49] C. Stoll, Z. Karni, C. Rössl, H. Yamauchi, and H.-P. Seidel. Template deformation for point cloud fitting. In *PBG@ SIGGRAPH*, pp. 27–35, 2006.

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

[51] J. Süßmuth, M. Winter, and G. Greiner. Reconstructing animated meshes from time-varying point clouds. In *Proceedings of the Symposium on Geometry Processing*, pp. 1469–1476, 2008.

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

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

[54] M. Wand, B. Adams, M. Ovsjanikov, A. Berner, M. Bokeloh, P. Jenke, L. Guibas, H.-P. Seidel, and A. Schilling. Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Transactions on Graphics (TOG)*, 28(2):1–15, 2009.

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

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

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

[58] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, et al. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (ToG)*, 33(4):1–12, 2014.