Surface Creation on Unstructured Point Sets Using Neural Networks

Mehmet Ersin Yumer, Levent Burak Kara

Visual Design and Engineering Laboratory
Carnegie Mellon University
Email: {meyumer, lkara}@cmu.edu

Abstract

We present a new point set surfacing method based on a data-driven mapping between the parametric and geometric spaces. Our approach takes as input an unstructured and possibly noisy point set representing a two-manifold in $\mathbb{R}^3$. To facilitate parametrization, the set is first embedded in $\mathbb{R}^2$ using neighborhood preserving locally linear embedding. A learning algorithm is then trained to learn a mapping between the embedded 2D coordinates and the corresponding 3D space coordinates. The trained learner is then used to generate a tessellation spanning the parametric space, thereby producing a surface in the geometric space. This approach enables the surfacing of noisy and non-uniformly distributed point sets. We discuss the advantages of the proposed method in relation to existing methods, and show its utility on a number of test models, as well as its applications to modeling in virtual reality environments.

1. Introduction

In this paper, we present a new surface design method that can take as input 3D point sets, and can generate freeform open surfaces through a neural network based regression algorithm. In this work, point sets of interest can be sparse, unstructured, and unevenly distributed, and devoid of normal vector information. Such point sets frequently arise with the use of new generation input devices such as 3D optical or magnetic trackers in VR environments (Fig. 1) where the points are sampled from trackers attached to the users' hands or any part of their bodies. Such point sets are considerably different in nature than the widely studied class of range data, where dense point sets are sampled directly from the surface they represent. In surface design from point tracking, however, one rarely obtains a full and dense coverage of the intended surface. Moreover, point sampling may exhibit significant non-uniformity based on the users' motion speed and their focus on particular regions of the design. The long term goal of the proposed work is thus to provide industrial surface design algorithms that can operate on tracking data to produce surfaces with controllable aesthetic qualities and associated mechanisms enabling further detailed refinement on the initial data.

As one step toward this goal, we present a neural network based surface regression method that takes as input open or closed point sets in $\mathbb{R}^3$, and generates free form surfaces through a parametric embedding and tessellation in $\mathbb{R}^2$. The parametric embedding is achieved through a local neighborhood preserving method. Once a parametrization of the input point set is computed, a mapping between the parametric coordinates of input points in $\mathbb{R}^2$ and their corresponding 3D design space coordinates is trained on a multi-layer, feed-forward, back-propagation neural network. A tessellation created in the parametric domain is then fed to the trained network which results in the synthesis of a two-manifold surface in the design space. A key advance in the proposed work is that the surface complexity is dictated by the network topology that iteratively minimizes the under-fit and over-fit to the available data. We focus on the creation of surface patches that capture the underlying geometry intended by the designer in such cases, yet without compromising from surface quality. This approach is in contrast to methods that require the designer to study the underlying point set to decide the degree or functional form of the fitted surfaces. We demonstrate that the proposed approach can be used for creating free-form surfaces from arbitrary point sets, as well as from point sets arising from tracking data. We also present a surface stitching method to enable the creation of water-tight and possibly non-manifold shapes in the VR environment, where our patch based surface creation technique is utilized. We also demonstrate our method's applicability to hole filling on polygonal surfaces. Specifically, our contribution lies in a flexible neural network based surface creation method from unorganized point sets. This approach utilizes a non-linear parametric embedding [1], that enables our algorithm to learn a generative mapping from the parametric to the geometric space. Moreover, the proposed stitching algorithm enables the different surfaces created using our approach to be unified into water tight models.

2. Related Work

In this section we review the previous work in surface creation, fitting, and approximation of point sets based on the surface representations used; parametric, mesh based and implicit, followed by a review of the use of neural networks in this field.

Parametric Surfaces: Parametric surfaces are one of the most widely used representations as they enable compact description,

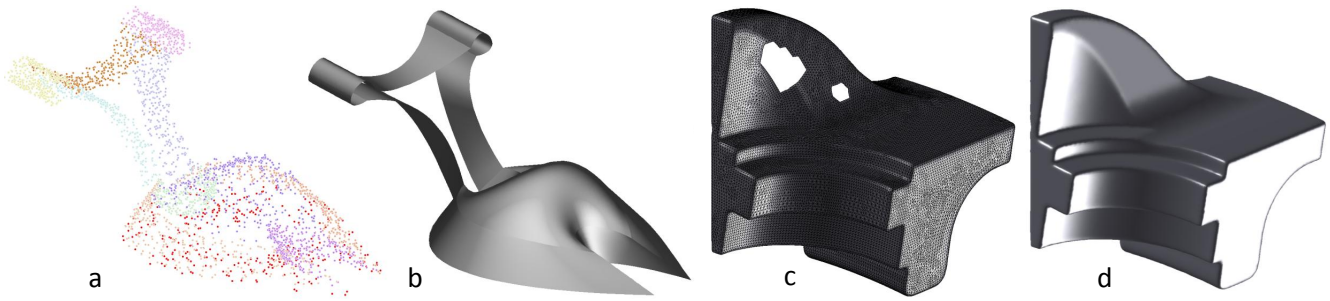

Figure 1: Applications of our surfacing method. (a-b) Patch based point set surface regression. (c-d) Hole filling.

and straightforward tessellation with arbitrary resolutions. Gregorski *et al.* [2] introduced a B-spline surface reconstruction method for point sets. Their approach utilizes a quad-tree like data structure to decompose the point set into multiple smaller point sets. Least squares quadratic fitting of each sub-point set is then followed by the degree elevation to B-spline surfaces and blending. Bae *et al.* [3], focusing primarily on laser range scanned data, introduced orthogonal coordinate transformations for NURBS surface fitting. The point set is first transformed into an orthogonal coordinate system, followed by B-spline fitting which is finally converted to NURBS surfaces. Adaptive fitting techniques introduced by Pottmann *et al.* [4, 5] utilize an active contour model which gradually approximates the targeted model shape. This iterative approximation minimizes a quadratic functional composed of an internal surface energy for smoothness and an approximation error for fitting. Lin *et al.* [6] introduce an iterative NURBS curve and surface fitting methodology to a given point set which is able to interpolate the point set. The major restriction of their approach is that the point set has to be pre-ordered. Following a similar approach, boundary condition satisfying NURBS surface fitting is also achieved [7]. The neural network in our method is similar to parametric surface definitions in the sense that it enables arbitrary resolution tessellation straightforwardly and has a compact definition. However, the proposed method differs from parametric fitting in that the functional form of the surface is dictated by the optimized network topology rather than requiring the user to decide the parameters of the fit. As shown in the following examples, the proposed method can be readily modified to fit a prescribed functional form such as a parametric surface of a given order, when desired.

Mesh-based Surfaces: Mesh-based or polygonal surfaces enable a straightforward encoding and rendition of surfaces. In particular, they have been used extensively for surfacing point sets arising from range scanners. In an early work, Hoppe *et al.* [8] used local linear approximations of the point set to create a mesh-based surface that approximates the point set. The first provably correct mesh-based surface fitting algorithm is presented by Amenta *et al.* [9, 10]. Given a sufficiently dense point sampling from the original surface, the approach guarantees the resulting surface to be topologically correct while interpolating the input samples. Gopi *et al.* [11] introduced a sampling criteria such that the fitted surface is guaranteed to be topologically correct and also provided algorithms that create mesh-based representations of such point sets [12]. Based on Delaunay tetrahedralization of a given point set, Attene *et al.* [13] introduced a method for closed genus-*n* triangulation fitting provided that the points are sampled from a real object. In 2005, Kuo *et al.* [14] approached the surface fitting problem with a region growing algorithm that gradually adds new triangles to an initial triangulation starting from a seed region of the point set. Dey *et al.* [15] presented a mesh-based surface fitting method applicable to noisy point sets as long as the noise level is within a specified threshold. Many mesh-based surface fitting algorithms typically require a smoothness or fairness criterion to be minimized, which may require considerable post processing after the initial surface fit [16].

Implicit Surfaces: Implicit representations enable compact mathematical descriptions and rapid set operations. However, the tessellation and rendering of such representations is a significant obstacle requiring specialized algorithms for visualization. Juttler *et al.* [17] introduced an approach which results in implicit least squares reconstruction of spline surfaces tailored toward reverse engineering. A widely used family of implicit surfaces are the radial basis functions (RBF). Kojekine *et al.* [18] used an octree structure to reduce the computational time associated with RBF spline based volume reconstruction. Ohtake *et al.* [19] used implicit surfaces as a way to facilitate intersection checks on mesh-based geometries. They also employed a similar approach together with compactly supported RBFs for range scanner point cloud surface fitting. Wu *et al.* [20] introduced a combined approach where they use multiple RBFs, where individual RBFs construct seed regions that are coalesced into larger regions through a partition of unity functional. A key drawback of the implicit approaches is the need for specialized visualization mechanisms. Nonetheless, the proposed approach is conceptually similar to RBFs in the way it takes a purely data-driven approach to surface generation. The main advantage of the proposed work is in its ability to generate an arbitrary tessellation directly within the parametric space. This allows an explicit control of the mesh topology and density, and lends itself to a straightforward geometry creation and visualization in the form of a polygonal model.

Use of Neural Networks: Barhak *et al.* [21] utilized neural network self organizing maps for 2D grid parametrization and surface reconstruction from 3D points sets. The result of the neural network is used to create a 3D surface iteratively with the help of a gradient descent algorithm. Similarly,

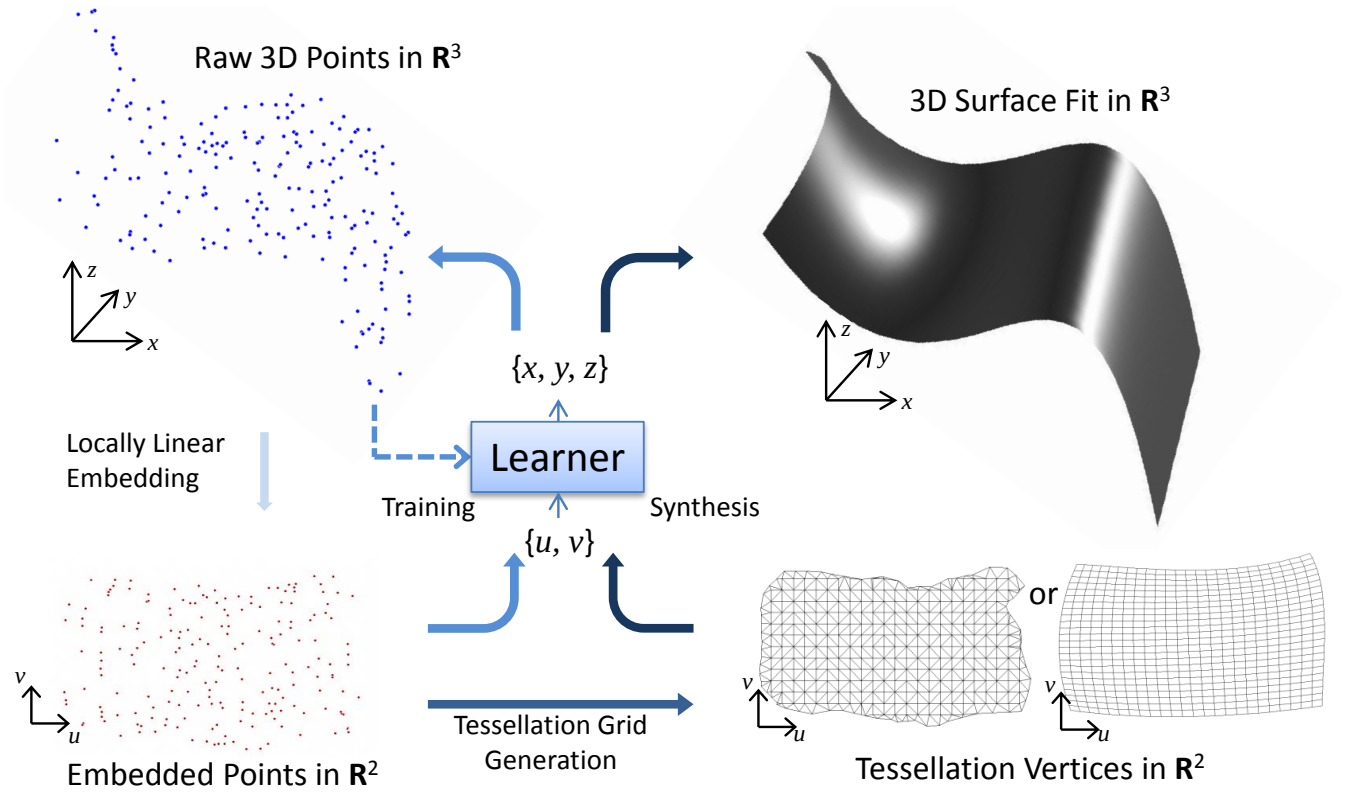

Figure 2: Neural network surface regression of unstructured point sets. First, a neighborhood preserving embedding is used for parametrization followed by neural network training. Then the final surface is synthesized by the trained network.

Galvez *et al.* [22] and He *et al.* [23] utilized neural networks for parametrization and point ordering, rather then surface creation. Khan *et al.* [24] introduced an approach for constructing surfaces from boundary curves that are required to be planar. Their approach addresses the boundary-to-surface learning problem rather than the point-to-surface learning problem. Krause *et al.* [25] implemented a neural gas neural network [26] for approximating a point set with disconnected triangles. These triangles do not necessarily span the whole surface and additional post-processing is required to ensure connectivity and water-tightness of the final surface. The method presented in this paper differs from their approach in that the network output in our work is the structured and a topologically valid manifold. Additionally, rather than deforming the initial triangles to a given point set similar to mesh-based approaches, the network presented in this paper serves as a direct tessellation engine from which a surface can be generated with an arbitrary resolution. Other related work [21, 22, 23, 24] introduced above do not utilize neural networks for direct surface creation from arbitrary point sets as we do, but rather use the network for point ordering or for surfacing planar boundary curves.

3. Overview

In this paper, we present a data-driven, learning based surface creation method for unstructured point sets. Unstructured point sets of interest may be sparse, unevenly distributed, and noisy. Our approach consists of four main steps; embedding of the point set in $\mathbb{R}^2$, training of the learner, creation of the tessellation, and generation of the surface in 3D (Figure 2). Embedding involves the creation of a unique 2D parametrization of the given point set. Training is a learning of the one-to-one mapping between the parametric coordinates and the 3D coordinates. A tessellation is then created in the parametric space that spans the embedded points. After the mapping is learned, it is used for synthesis in which a fully connected two-manifold surface is created in 3D from the tessellation in the parametric space.

4. Parametrization

Our parametrization approach is invariant to translations and rotations, and can be used for surfaces that fold onto themselves without creating self-intersections. This capability is achieved using a local neighborhood preserving nonlinear transformation from the 3D space to the 2D parametric space. The resulting global nonlinear transformation is the outcome of locally linear transformations. Therefore, the embedding of the 3D coordinates into the 2D parametric space boils down to the solution to a sparse linear system as shown by Roweis *et al.* [1]. In the following paragraphs, we explain the parametrization tailored to our purposes; for unstructured point set parametrization, and for incomplete mesh parametrization.

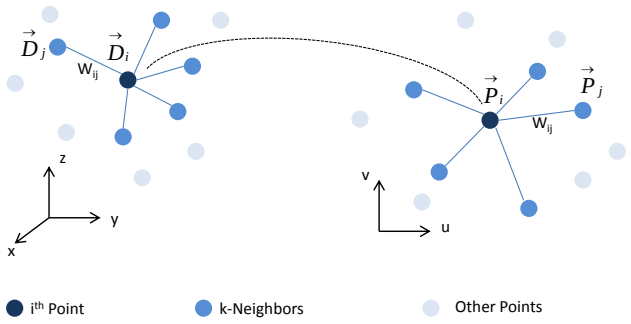

Figure 3: Locally linear embedding for unstructured point set parametrization.

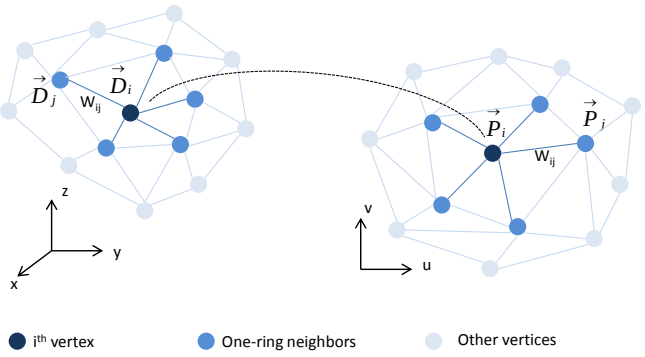

Figure 4: Locally linear embedding for mesh parametrization.

4.1. Unstructured Point Set Parametrization

For parameterizing unstructured point sets, the algorithm relies on the local neighborhood information. Specifically, the following procedure makes an implicit assumption; every point in the point set can be reconstructed by a linear combination of its nearest neighbors.

1. A neighborhood N_i for a fixed number of neighbors is calculated for every $\vec{D}_i$ in $\mathbb{R}^3$ based on the Euclidian distances. $\vec{D}_i$ represents the position vector of point i (Fig. 3).
2. A sparse neighborhood weight matrix W is computed by minimizing Eqn. 1, subject to two constraints; rows of (W) sum to 1, and W_{ij} corresponding to $\vec{D}_j$ that is not in N_i is equal to zero.

$$\theta(W) = \sum_i |\vec{D}_i - \sum_j W_{ij}\vec{D}_j|^2. \tag{1}$$

3. Using W calculated in step 2, minimizing Eqn. 2 results in the 2D parametric embedding $\vec{P}_i$ of the point $\vec{D}_i$ for all points in the original 3D space.

$$\phi(W) = \sum_i |\vec{P}_i - \sum_j W_{ij}\vec{P}_j|^2. \tag{2}$$

The above described, sequential minimization of two different cost functions is reduced to a single eigenanalysis computation as described in the following paragraphs.

4.2. Incomplete Mesh Parametrization

We utilize incomplete mesh parametrization for constructing hole-filling learners that can fill in the missing parts of a given mesh by using the mesh vertices that surrounds the corresponding gap. Although the surrounding points can be simply cast as an unstructured point set, our approach enables the existing edge information to construct the desired manifold that the mesh contains. To this end, instead of using a fixed number of nearest neighbors, we use the one ring neighbors for every vertex while forming the weight matrix, W, in Eqn. 1.

4.3. Deterministic Solution of the Embedding Problem

The minimization of Eqn. 2 with fixed weights results in the 2D embedding, $\vec{P}_i$. This optimization has to be constrained in order to attain a unique solution. First constraint involves preventing rigid body translations by anchoring the centroid of the point set at the origin:

$$\sum_i \vec{P}_i = \vec{0}. \tag{3}$$

Also, to avoid degenerate solutions [27], the embedding vectors are constrained to have unit covariance:

$$\frac{1}{n}\sum_i \vec{P}_i \otimes \vec{P}_i = I_2, \tag{4}$$

where $\otimes$ is the Kronecker product, I_2 is the 2×2 identity matrix, and n is the number of points.

Under these constraints the error function to be minimized in Eqn. 2 can be rewritten as:

$$\phi(W) = \sum_{ij} M_{ij}(\vec{P}_i \cdot \vec{P}_j), \tag{5}$$

where M is a symmetric matrix defined as:

$$M = (I_n - W)^T(I_n - W), \tag{6}$$

where I_n is the $n \times n$ identity matrix. The optimum solution to Eqn. 5 is found by the two eigenvectors of Eqn. 6 that correspond to the second and third smallest eigenvalues of M [27], since the first eigenvector is the free-free mode of the matrix M which has equal values in all degrees of freedom. That is, for every point, the parameter space coordinates are:

$$\begin{aligned} {}^uP_i &= {}^2\xi_i \\ {}^vP_i &= {}^3\xi_i \end{aligned} \tag{7}$$

where ${}^k\xi$ is the k^{th} eigenvector of M.

With this calculation, the two minimization problems associated with Eqn. 1 and Eqn. 2 is avoided. The calculation of M, from the weight matrix, W, is straightforward. Note that M is a sparse matrix as the number of points in the point set is typically much greater than the number of nearest neighbors practically. Moreover, we need to calculate only the smallest three

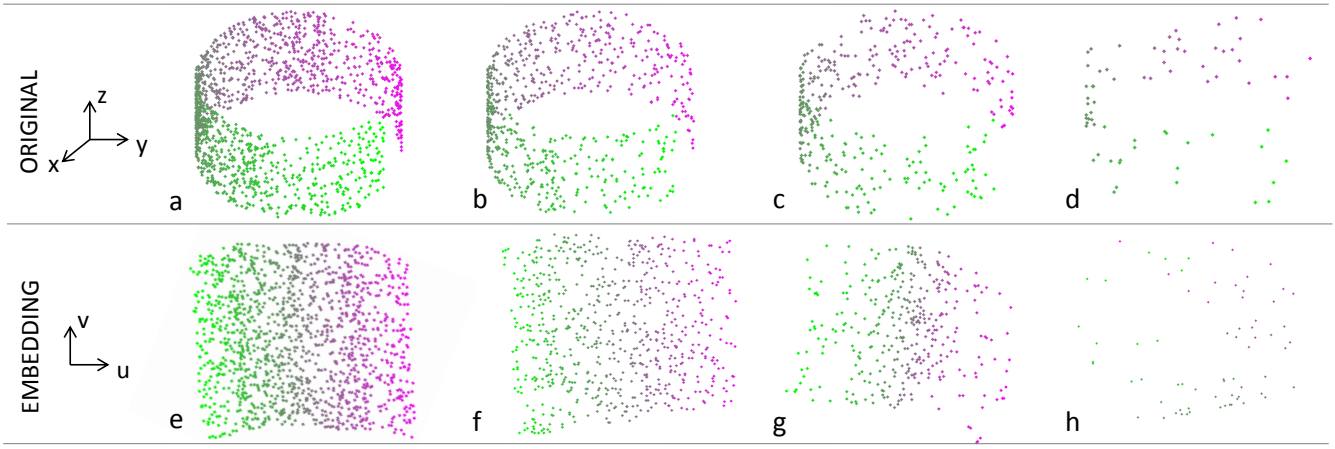

Figure 5: Parametrization of an open surface with uniformly sampled random points. Open cylinder data points (a-d) and corresponding parametrization (e-h) for 1400, 700, 350, 70 points respectively.

eigenvectors of M need to be calculated, which requires minimal computational effort. Azariadis et al. [28] also introduces a parametrization suitable for patch by patch surface fitting. Their method requires an iterative fitting of the dynamic parametrization, whereas we accomplish the parametrization with a one-step eigenanalysis as described. Moreover, our parametrization does not require the pre-determination of the boundary curves in contrast to [28].

To demonstrate our parametrization scheme, an open cylinder is utilized. Note that these types of point sets are not correctly embedded using global linear techniques such as the Principal Component Analysis or its variants. Fig. 5 shows the embedding results on this cylinder. The sampling in the original space becomes increasingly sparse to the point that the underlying geometry is no longer discernible (Fig. 5d). As shown, the parametric embedding faithfully captures the intended geometry in the embedded space when the sampling rate is sufficiently high. This capability diminishes with increasing sparsity. However, with increasing sparsity, the underlying surface concurrently becomes less discernible to the human eye as well. One downside of this approach, however, is that if the point set represents a surface that intersects itself, this approach will fail to compute a correct parametrization of the surface. To overcome this challenge, segmentation approaches might be utilized [29].

5. Learning Based Surface Regression

In this section, we present our learning based surface regression method for surface creation from point sets, and hole-filling for incomplete meshes. A *learner* in our framework (Fig. 2), represents a function that can continuously map an $\mathbb{R}^2$ space to an $\mathbb{R}^3$ space. This learner can then be trained using the one-to-one relationship between the parametric coordinates and the 3D coordinates of the original point set. We utilize feed-forward multi layer neural networks as learners in our approach for a number of desired properties they entail:

1. The topology of a neural network, thus the degrees of freedom it imposes on the created surface, is not fixed as opposed to a pre-defined functional form such as a polynomial function.
2. Trained neural networks are good interpolaters [30], which is a desired property since the tessellation in the embedded space corresponds to vertices to be generated from the parametrizations of the point set. In this context, interpolation refers to the regression function within the inner range of the data points.
3. The functional form of each neuron in a given neural network can be defined to tailor desired properties (such as continuity and differentiability) in the resulting surface.

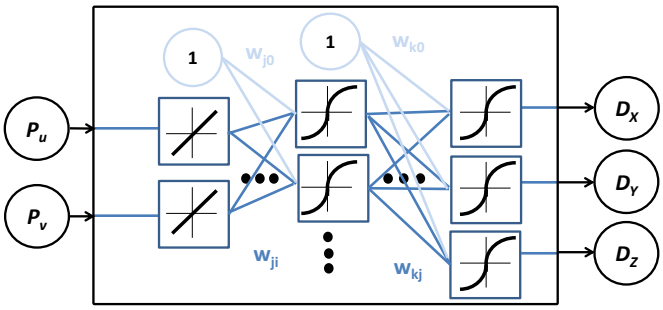

Figure 6: The multi-layer feed-forward network.

Neural networks can be viewed as functions that map an input space to an output space (Fig. 6). This general idea can be exploited in a number of ways. For instance, if we use a single layer-single neuron in the neural network topology given in Fig. 6, with a linear activation function, a mapping from the embedded space to the 3D space will result in a planar surface (Fig. 7). Similarly, the network can be designed to regress any input space to any output space using a prescribed polynomial or parametric form. However, we focus on the general form of feed forward neural networks with adaptive topology as shown in Fig. 6 with nonlinear activation functions. The topology of the network has a significant impact on the final surface created,

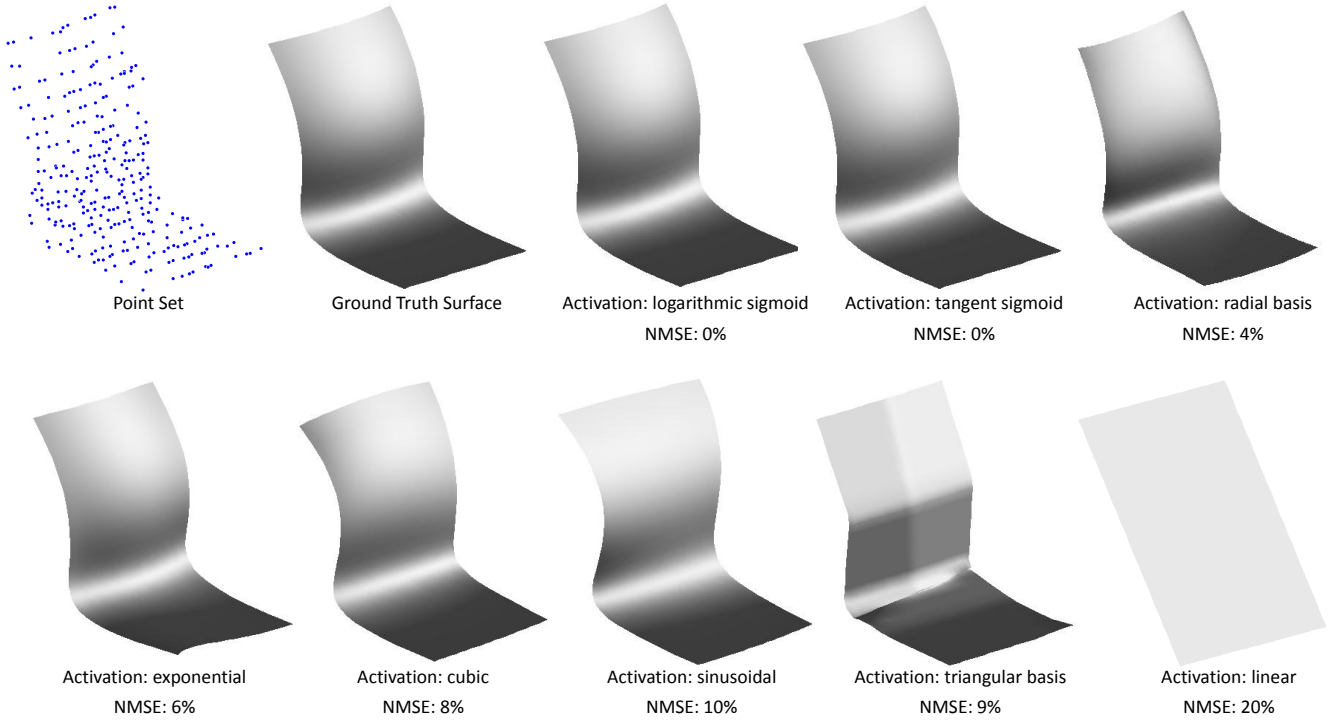

Figure 7: Activation functions, and their effect on the resulting surface. *NMSE is the normalized mean square error with respect to the bounding box diagonal of the point set.*

since it implicitly dictates the degrees of freedom of the surface. Our training scheme adaptively selects the desired topology.

5.1. The Learning Problem

The general form of the feed forward neural network learner we utilize is shown in Fig. 6. Here, $\vec{P}$ is the 2D parametric coordinate of the 3D input point $\vec{D}$ as computed using the techniques described in the previous section. Hence, the surface constructing problem boils down to a learning problem from $\mathbb{R}^2$ to $\mathbb{R}^3$.

The functional form of a network with a single hidden layer with n neurons, that maps $\vec{P}$ to $\vec{D}$:

$$\{D\}_k = \sigma\left(\sum_{j=1}^{n} w_{kj}\sigma\left(\sum_{i=1}^{2} w_{ji}\{P\}_i + w_{j0}\right) + w_{k0}\right) \tag{8}$$

where σ is the activation function. The activation function plays an important role in the characteristics of the created surface, and the fitting error. Since it maps the input of a each neuron to its output, the propagation of the information in a feed-forward fashion over these neurons crete the surface in 3D coordinates.

Figure 7 shows the effect of varying activation functions on the resulting surfaces. In all examples, a two hidden layer neural network is used, with four neurons in each layer. The following activation functions are used:

$$Log - Sig : \sigma(a) = \frac{1}{1+\exp(-a)}, \tag{9}$$

$$Tan - Sig : \sigma(a) = \frac{2}{1+\exp(-a)} - 1, \tag{10}$$

$$Rad - Bas : \sigma(a) = \exp(-a^2), \tag{11}$$

$$Tri - Bas : \sigma(a) = \begin{array}{l} 1-|a|, if -1 \le n \le 1 \\ 0, otherwise. \end{array} \tag{12}$$

In Fig. 7, it is seen that the sigmoid activation function results in the zero mean square error, and captures the ground truth freeform surface. Moreover, combining Eqn. 9 with Eqn. 8, the surface can be shown to be infinitely differentiable. Therefore, a sigmoidal activation function enables the creation of surface patches that are C^∞ continuous.

5.2. Training

In our approach the numbers of inputs and outputs are fixed at two and three respectively as dictated by the learning problem. The number of hidden layers and the number of neurons in each layer, however, can be controlled freely. Increasing the number of neurons or the number of hidden layers will result in an increased degrees of freedom and non-linearity in the surface. This may cause an undesirable overfit to the data, if no other measures are taken [30]. On the contrary, an insufficient number of neurons and/or layers will produce a stiff map, which may result in under fitting [30]. Therefore, the number of neurons and layers must be chosen judiciously. To this end, we employ an iterative procedure for selecting these parameters as follows:

1. Decompose the available input to training (85%), validation (10%) and test sets (5%).

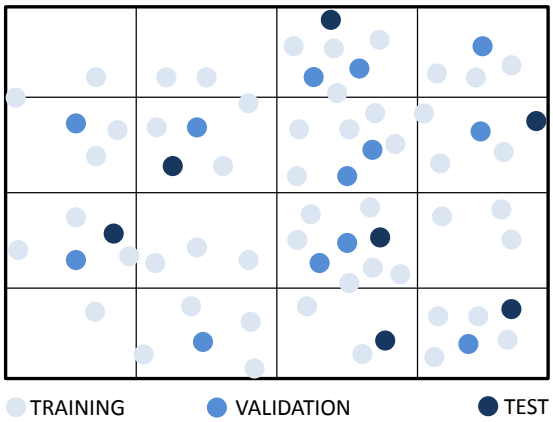

Figure 8: Example sampling prior to learning.

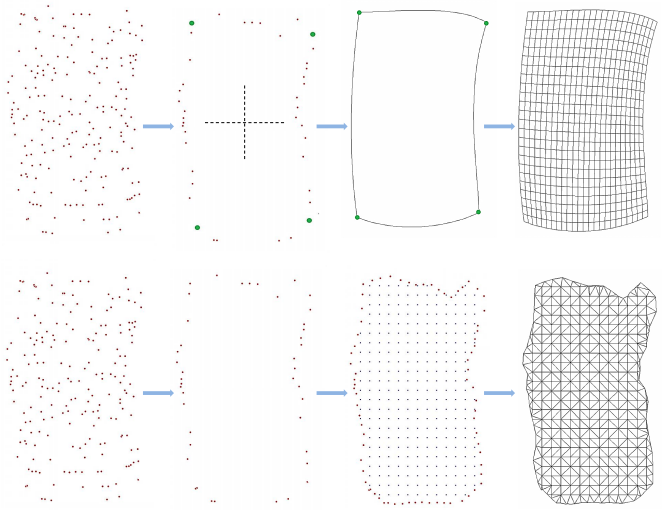

Figure 9: Tessellation Grid Generation. Regular quad grid generation (*top row*), Triangular freeform grid generation (*bottom row*).

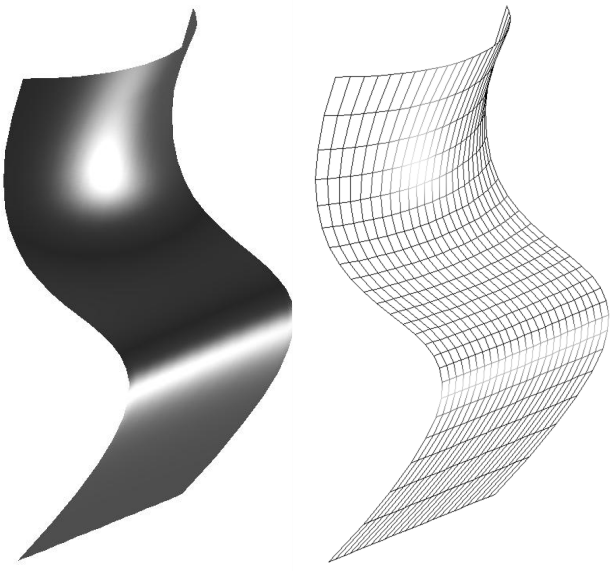

Figure 10: Tessellation of the seat back surface.

2. Initialize a network with a single hidden layer ($n_L = 1$) and single neuron ($n_N = 1$) in its hidden layers.
3. Train the network until the validation set performance (P_V) converges, with back-propagation [31].
4. Record the test set performance (P_T) for the current network configuration.
5. Increase the number of hidden neurons by 1 ($n_N \leftarrow n_N+1$). Iterate steps 3-5 until P_T converges.
6. Record n_N and P_T for current n_L.
7. If $n_L < n_{Lmax}$, increase the number of hidden layers by 1 ($n_L \leftarrow n_L + 1$). Iterate steps 3-7 until $n_L = n_{Lmax}$.
8. Report the network configuration (n_L and n_N) with the best performance P_T on the test set.

Previous works have shown that an iterative search for the network architecture prevents over-fitting and under-fitting [30, 32]. In our case, over-fitting will correspond to poor surface quality with relatively small fitting error, and under-fitting will correspond to a relatively flat surface with a high fitting error. Our primary focus is to extract the details of the underlying true manifold as much as possible, yet without compromising the surface quality.

For the above training scheme, we need to sample the validation data set as well as the training data set from the original input points. To ensure an unbiased coverage of the input space, we divide the parametric space into a number of subregions and sample points randomly from a uniform distribution from each subregion. An example is shown in Fig. 8.

5.3. Tessellation Grid Generation

Once the embedding of the point set is calculated, these points in 2D are used to generate a tessellation grid (Fig. 9). We use two different approaches, each serving a different need. For point sets that span simple regions in the parametric space (not necessarily in the 3D space), we generate a four sided quadratic grid represented as a bilinear Coons patch in the parametric coordinates. First, far most points in the four corners of the two largest PCA directions are determined. Then, points that connect these four anchors are calculated such that the resulting closed polyline encloses all of the points but the smallest area. A cubic Beziér curves is fit to each of these point sets. The curves are then used to generate a quadrilateral tessellation grid using the bilinear Coons patch governed by Eqn. 13.

$$\begin{aligned} Q(u',v') \;=\; & B(u',0)(1-v') + B(u',1)v' \\ & + B(v',0)(1-u') + B(v',1)u' \\ & - B(0,0)(1-u')(1-v') - B(0,1)(1-u')v' \\ & - B(1,0)u'(1-v') - B(1,1)u'v' \end{aligned} \tag{13}$$

In Eqn. 13, B's are the boundary curves of the patch, Q.

The second approach, targeting relatively more complex outer boundary point sets in the parametric space, generates a triangulation. First, the outer loop of the point set is determined, and uniformly resampled. This outer loop is embedded into a uniform grid, and the grid points outside the loop are discarded. The remaining points are triangulated with Delaunay triangulation, resulting in the tessellation grid.

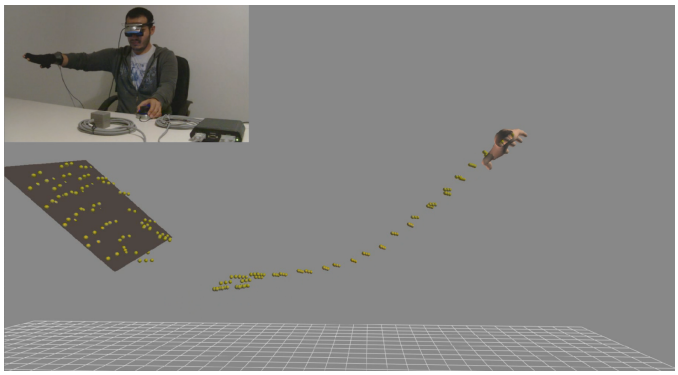

Figure 11: User Interaction.

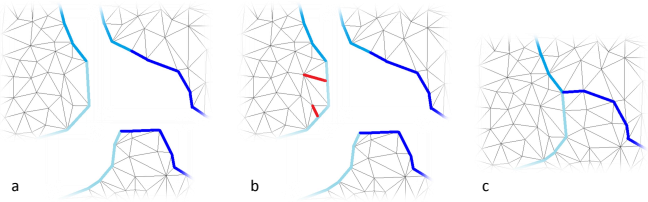

Figure 12: (a) Stitching edges demarcated by the user. (b) Necessary vertex additions for watertightness. (c) Stitched model.

5.4. Surface Synthesis

The vertices of the generated tessellation grid are then fed to the trained network, producing the surface in $\mathbb{R}^3$, while sharing the tessellation topology established in the parametric space. An example surface tessellation is shown in Fig. 10.

6. Implementation

We have deployed the proposed method for surface creation in a Virtual Reality (VR) design environment for rapid generation of conceptual shapes, as well as in a sketch based environment for hole filling.

In the VR environment shown in Fig. 11, the users interact with the system through a data glove and a magnetic tracker worn on the sketching hand, and a 3D mouse operated by the other hand. A 3D stereo vision enabled head set allows the user to situate themselves in the VR environment. The magnetic tracker enables six degrees of freedom absolute motion tracking of the hand whose 14 joint angles are decoded by the data glove. Tracking the user's hand enables the direct dictation of the point set that, in turn, represents the desired surface geometry. The resulting point sets are used as the input to our regression method, which leads to the final freeform surface geometry. A virtual hand in the VR environment simulates the sketching hand in real time. The other hand, operating the 3D mouse, helps the user fly in the VR environment via panning, rolling and spinning.

In the sketch-based environment, users interactively draw a region of interest around a hole on a mesh model by lasso gestures, and the hole is filled by generating new vertices and the associated topology. Once the region of interest is selected and mapped into the parametric space, the users are able to modify the new tessellation grid uniformly generated inside the hole, in order to establish any desired patterns.

6.1. Surface Stitching

To create water tight models, we use a stitching method based on Laplacian reconstruction [33]. Our formulation is applicable to arbitrary meshes and non-manifold stitching. For instance edge shared by three or more surface patches can be stitched together. An edge in this section (Fig. 12), refers to a connected open or closed polyline formed by the edges of simplicies on a meshed surface. A stitching edge does not necessarily lie on the boundary, *i.e.* stitching edges at arbitrary locations of the mesh is possible. For each surface, s, associated with the stitching operation, the differential coordinates are calculated as follows:

$$\vec{\delta}_s = [L]_s \vec{v}_s, \tag{14}$$

where $\vec{v}_s$ is the vertex position vector and $[L]_s$ is the discrete laplacian operator where row i having weights w_{ij} with $\sum_j w_{ij} = -1$, for one-ring neighbors j and $w_{ii} = 1$. Calculating the weights with the cotangent scheme [33], results in the most accurate representation of the differential coordinates.

Constraining one vertex, and reconstructing $\vec{v}_s$ from the knowledge of the differential coordinates deltas is a linear problem of the form, $[A]\vec{x} = \vec{b}$, and will yield the exact vertex positions $\vec{v}_s$. Since the Laplacian operator is rank deficient (rigid body translations not constrained), constraining a single vertex is necessary, and sufficient for a unique solution. Therefore, imposing more than one constraints will result in a least-squares problem. We exploit this nature of Laplacian reconstruction. Instead of fixing one point for each free surface in the space, we add as many constraint equations as there are vertex pairs in the stitching edges and solve the system of equations for all the surfaces involved in the stitching operation at once. The constraint equations are in the form of:

$$\vec{c}_a - \vec{c}_b = 0, \tag{15}$$

where $\vec{c}_s$ are the positions of the vertices lying on the corresponding stitching edge. Necessary vertex additions (Fig. 12) prior to solution are made to ensure one-to-one correspondence in the stitching edges.

7. Results

7.1. Comparison with Implicit Function Fitting and Direct Triangulation

Noise-added point sets sampled from an underlying surface, shown in Fig. 13, are used for a comparison study between our surface regression method, the radial basis function surface fitting [34], and direct delaunay triangulation [35] of the original point set.

Radial basis functions can be fit to arbitrary data, ranging from exact interpolation at the given points to non-interpolating but smooth fittings for high density point sets [34, 36]. It can be seen that the low density RBF surface (Fig. 13) has a similar mean square error as the surface fit created by our method,

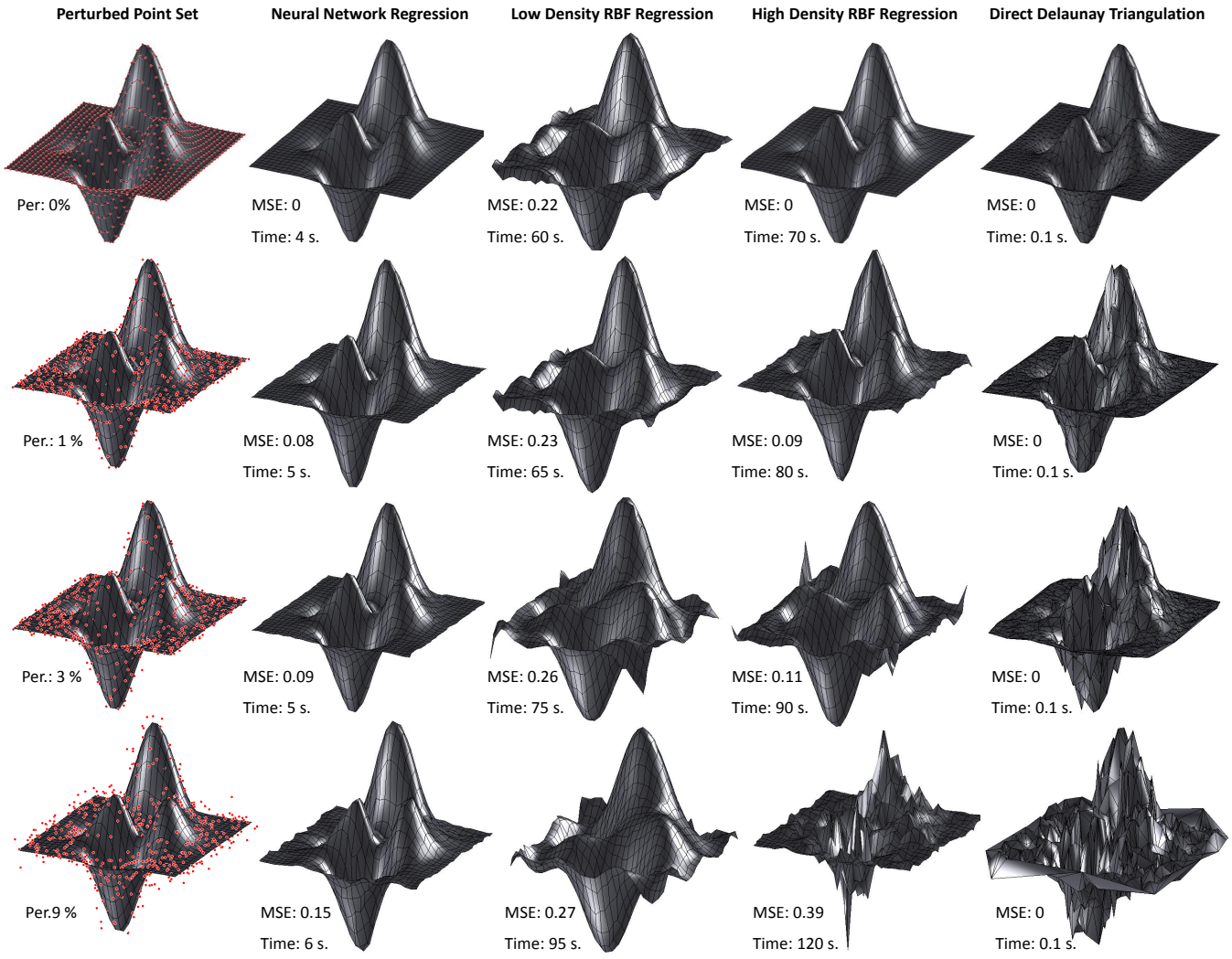

Figure 13: Comparison of surface fitting methods. *Each row is the result of reconstruction of the corresponding point set with the given method. Per.*: Random uniform perturbation error percentage in the point set. *MSE*: Mean square error of reconstruction. *Time*: Total processing time of each surface in seconds.

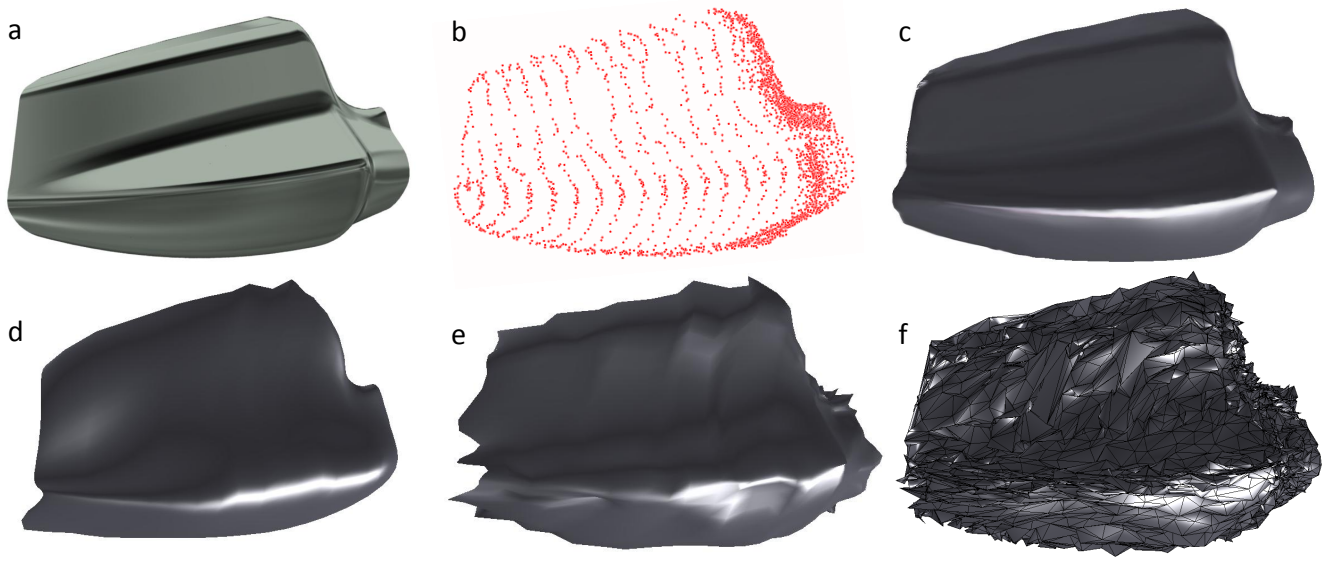

Figure 14: Automobile side view mirror back surface reconstruction. (a) Actual side view mirror model. (b) Sampled points with 1% uniform error. (c) Neural network regression. (d) Low density RBF surface fitting. (e) High density RBF surface fitting. (f) Direct delaunay triangulation.

Table 1: Network degree of freedom.

SURFACE	LAYERS	NEURONS
Front of the Back Rest	2	3
Seat Section	2	2
Side of the Seat Section	3	3
Side of the Back Rest	3	3

but the patch created with our method is smoother (Fig. 13). To decrease the fitting error associated with RBF surface fit, its function density can be increased [34]. However, when the density of the RBF surface fitting function is increased, its generalization diminishes [36, 34] and the surface quality is poorer as observed in Fig. 13.

Another widely used point set surfacing method is the direct triangulation of points in the parametric domain [35]. For direct triangulation, we triangulate the point set in the parametric space created with our open-manifold parametrization method, and then projected the topology back to 3D space (Fig. 13). As expected, the mean square error is zero as the triangulation interpolates the original points. However, the surface quality is significantly compromised.

In Fig. 14, a point set sampled from the back surface of an automobile side view mirror with 1% uniform error is surfaced with our method, as well as the above described methods.

7.2. Conceptual Design in Virtual Reality Environment

In this section we demonstrate the capabilities of our surfacing technique using three examples created in a VR environment. A car seat cushioning design, an abstract shape and an asymmetric mouse is designed within this environment.

Fig. 15a shows the complete point cloud drawn by the designer representing the multiple surfaces of a car seat. Individual surfaces are demarcated by different colors, and are separated by the user with a keypress during construction. Fig. 15c shows the four point sets forming the front and left sides of the back rest and the seat sections of the seat. Fig. 15d is the corresponding 2D locally linear embedding of these point sets. Fig. 15e are the resulting constructed surfaces. Table 1 shows the number of network layers and neurons as computed from the iterative network topology optimization.

Fig. 15f shows the neural network regression of the front and back surfaces of the seat. Note that a neural network is trained and regressed for each individual point set. Table 2 shows the total network training time for different point sets. In all cases, once the network is trained, the synthesis is near instantaneous. The two main computationally expensive steps in our approach are the parametric embedding, and the network topology optimization/training. Parametric embedding is bounded by the eigenanalysis involved, which is $O(n^2)$ in our implementation where n is the number of points in the point cloud. Neural network topology optimization and training is bounded with $O(mn^2)$ where m is the number of topology iterations and n is the number of points in the point cloud. Since $n >> m$, the bound to our algorithm is effectively $O(n^2)$.

Table 2: Time for parametric embedding and topology optimization. (*On a Intel-i7 1.6 GHz, 4 GB machine.*)

Point Set Size	Parametric Embedding	Topology Opt. & Learning
100	0.2 *sec.*	0.1 *sec.*
1000	0.8 *sec.*	0.9 *sec.*
10000	10 *sec.*	10 *sec.*

Since point sets are treated independently, the synthesized surface patches do not form shared boundaries. Currently, we stitch these surfaces as explained previously.

Fig. 16 and Fig. 17 exemplifies two product shapes designed with our system; a mouse and an abstract shape. The mouse is an asymmetric model with five surfaces defining its ergonomic shape. The spaceship is an axis-symmetric model with five symmetric and three mirror surfaces. The sparsity of various point sets in Fig. 16 - 17(a) are handled by our algorithm and the regression surfaces are created as shown in Fig. 16 - 17(b). Final stitched surfaces that result in the water tight geometric shapes are shown in Fig. 16 - 17(c-d).

7.3. Applications to Hole Filling

In Fig. 18 and Fig. 19, we demonstrate the mesh hole filling capabilities of our method. Once the user draws the region of interest around the hole, these vertices are mapped into the parametric space as described previously by utilizing the connectivity of the mesh. The holes are then filled either with random points exhibiting a desired density (Fig. 18), or the user can impose tailored patterns in on the newly created vertices in the parametric domain. For instance, in Fig. 19, the user has sampled points according to the surrounding pattern to create a triangulation commensurate with the surrounding mesh.

Once the new vertices are generated inside the holes, the hole boundary and these new vertices are triangulated by Delaunay triangulation in the parametric domain. A network trained only on the input-output pairs of the surrounding vertices, is then used to tessellate the newly created vertices. The additional topology created in the parametric domain is preserved in the 3D.

8. Discussions and Conclusion

We presented a neural network regression method for freeform surface creation on point sets. Point sets of interest are primarily sparse, unstructured, unevenly distributed and can be partially detailed. Our surface regression procedure is composed of four steps; parametrization, neural network training, grid generation and synthesis. We also investigate a surface stitching method to leverage shape design using our surface regression technique.

On Point Set Parametrization. We use locally linear embedding for open two-manifold parametrization. In Fig. 5, it can be observed that the parametrization is accurate for the point sets in which the underlying geometry is discernable to the human eye, whereas the parametrization deteriorates as the sampling

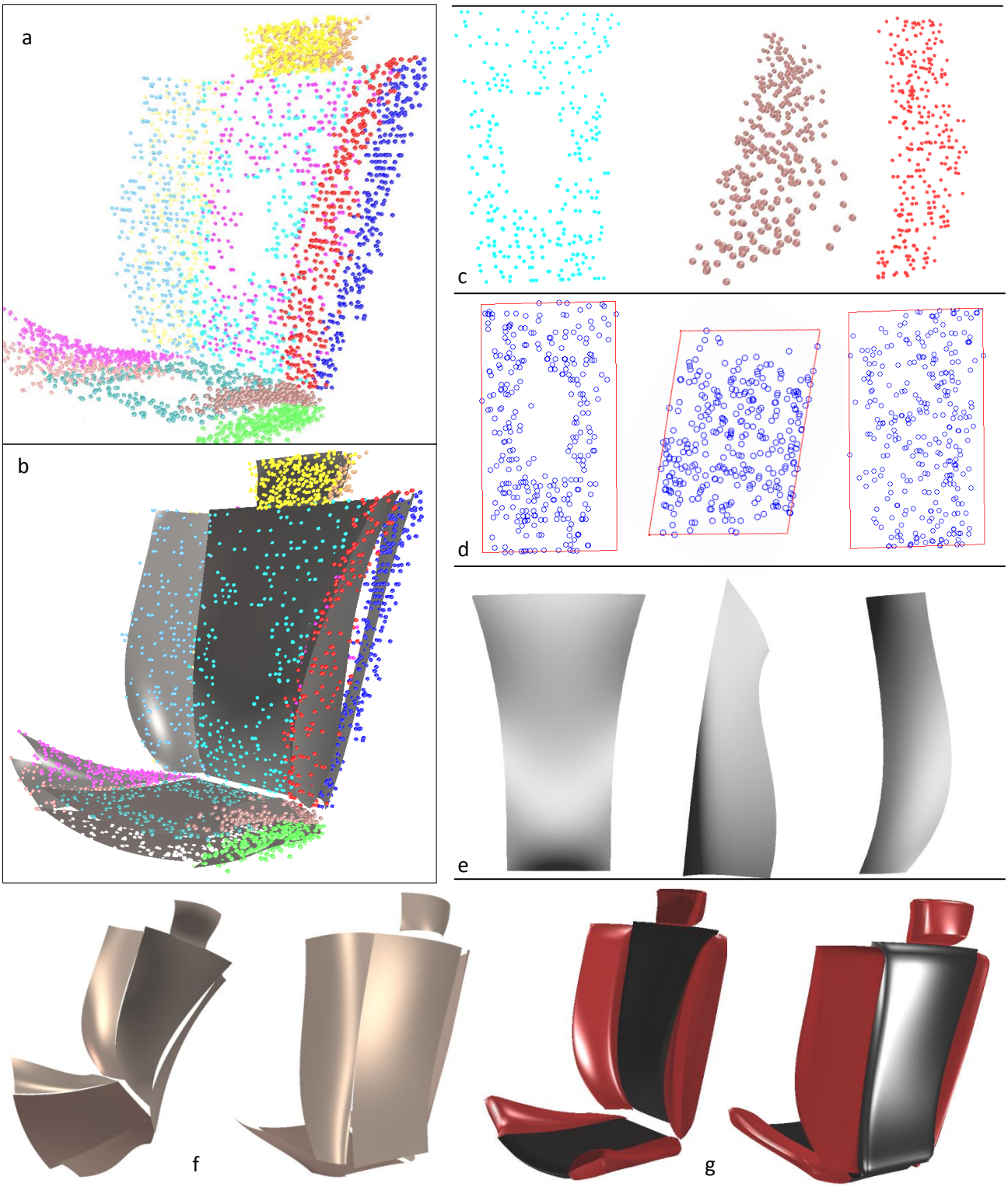

Figure 15: Conceptual design of a car seat. (a) Complete point set created by the user in a virtual reality environment where the position and orientation of the user's hand is tracked. (b) Neural network regressed surfaces. (c) Point sets for the front and left sides of the back rest and the seat section. (d) Corresponding parametrizations. (e) Corresponding regressions. (f) Regressed front and back surfaces. (g) Surfaces after stitching. *Point set size vary in 200-500.*

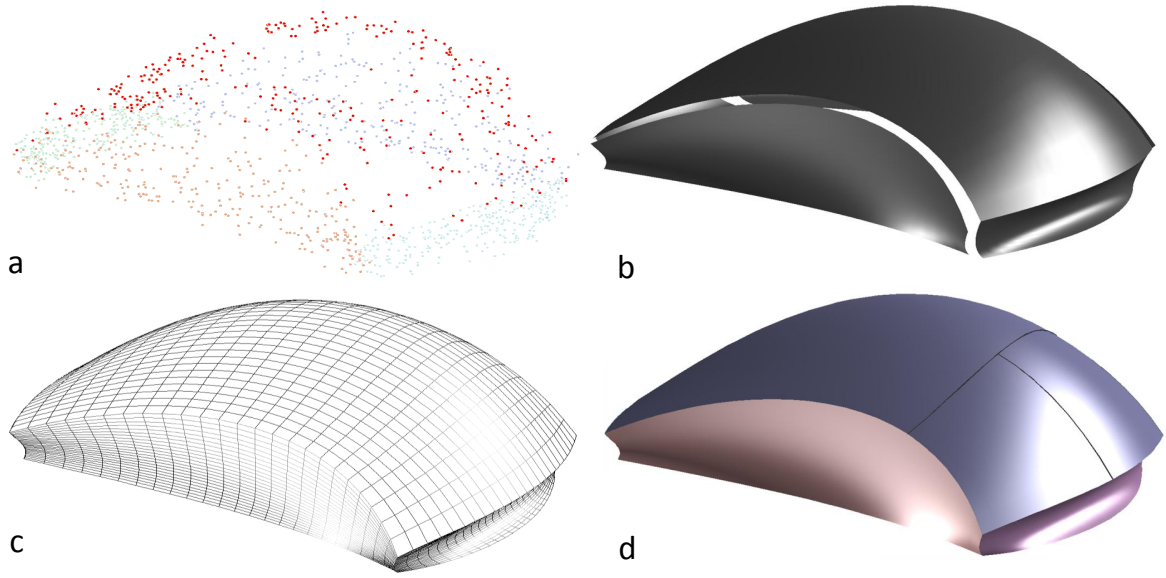

Figure 16: A mouse designed in the VR environment with our surface regression technique. (a) Point sets. (b) Neural network regression surfaces. (c-d) Finalized water-tight model. *Point set size vary in 300-600.*

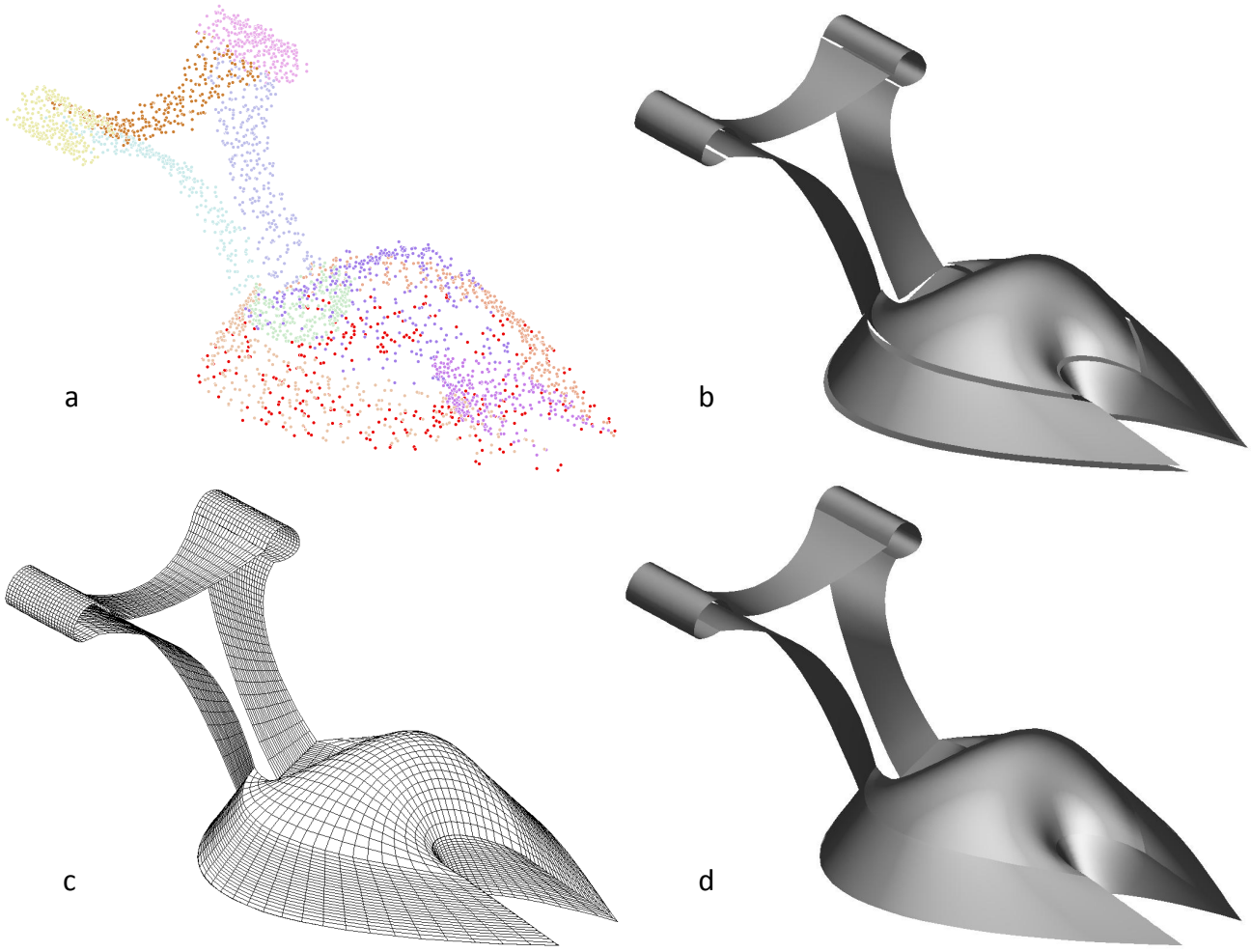

Figure 17: An abstract shape designed in the VR environment with our surface regression technique. (a) Point sets. (b) Neural network regression surfaces. (c-d) Finalized water-tight model. *Point set size vary in 400-1500.*

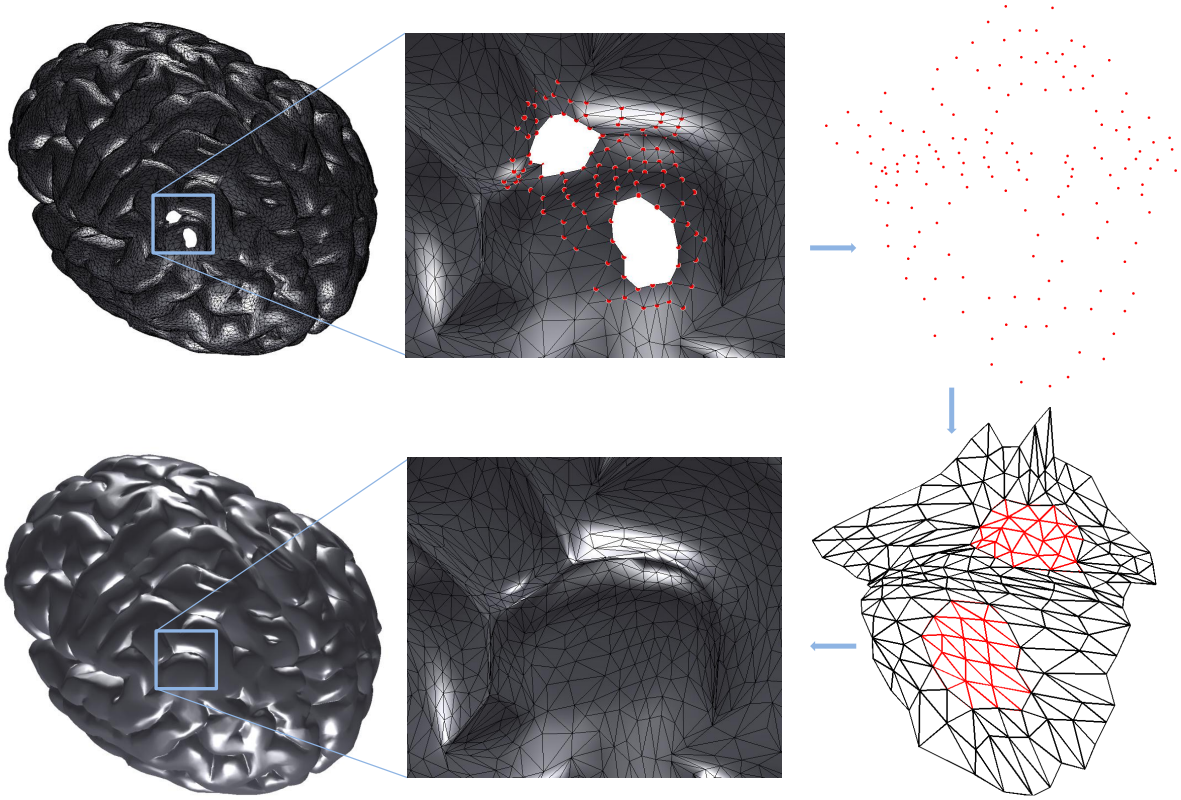

Figure 18: Cortex mesh hole filling with our algorithm.

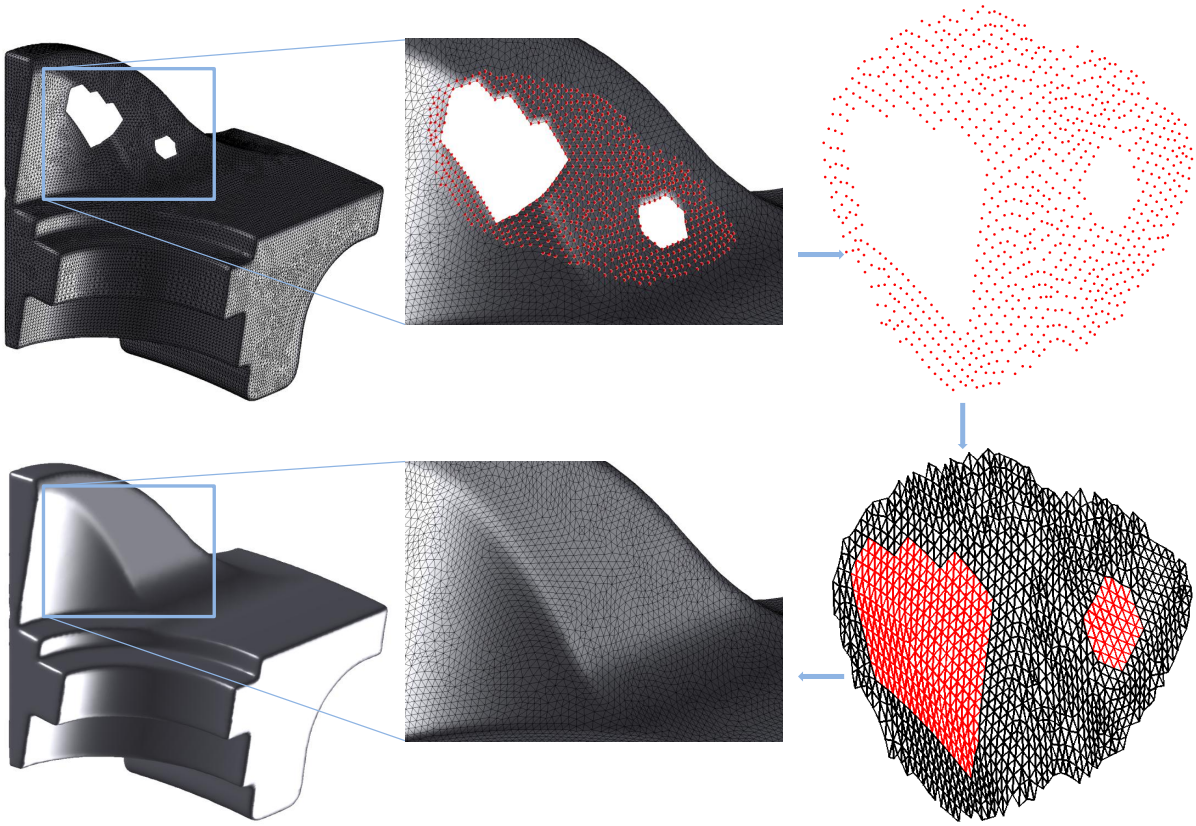

Figure 19: Fan disk mesh hole filling with our algorithm.

rate becomes prohibitively low. Point sets that are unevenly distributed and/or partially detailed in nature (Fig. 15- 17) are also successfully parameterized. These parametrizations are independent of the global position and orientation of the point set and are also able to process surfaces that fold onto themselves.

On Neural Network Training. Once the point set is parameterized, a neural network that takes the parametrization and 3D space coordinates as input-output pairs is trained for surface regression. This training process is controlled by continuously monitoring the validation and test sets over different network structures in order to concurrently minimize both under-fitting and over-fitting. Under-fitting results in a loss of information in which the regressed surface will exhibit fewer details compared to the intended one. Over-fitting, on the other hand, will result in a network structure that has excess degrees of freedom, which typically results in undulations in the synthesized surfaces. Our approach aims to curtail such phenomena through an integrated network topology optimization and learning algorithm.

On Grid Generation and Synthesis. The trained network forms a bridge between the parametrization domain and the 3D space, thereby effectively serving as a tessellation mechanism. Through an automatic or guided grid generation process in the parametric space, a surface tessellation in 3D can be obtained with a controllable resolution. One such tessellation is shown in Fig. 10. Tessellated free-form, free-boundary surface patches can be connected with our stitching mechanism to create water tight models in our system.

On Surface Stitching. We implemented our surface creation method in a VR environment where the user is enabled sketching through hand gestures in 3D. For creating watertight models, a surface stitching algorithm is developed leveraging a non-interactive scheme. Our surface stitching method is capable of connecting an arbitrary number of surfaces into manifold and non-manifold mesh representations with G^0 continuity at the stitching edges.

Users interacted with our system report highly expected results from the surface patch creation and surface stitching. We exploited the interpolation strength intrinsic to neural networks, and have shown successful hole filling operations using our surface regression method.

References

[1] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding., Science (New York, N.Y.) 290 (5500) (2000) 2323–6.

[2] B. Gregorski, B. Hamann, K. Joy, Reconstruction of B-spline surfaces from scattered data points, in: Proceedings Computer Graphics International 2000, IEEE Comput. Soc, 2000, pp. 163–170.

[3] S. Bae, NURBS surface fitting using orthogonal coordinate transform for rapid product development, Computer-Aided Design 34 (10) (2002) 683–690.

[4] H. Pottmann, S. Leopoldseder, M. Hofer, Approximation with active B-spline curves and surfaces, in: 10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings., IEEE Comput. Soc, 2002, pp. 8–25.

[5] H. Pottmann, A concept for parametric surface fitting which avoids the parametrization problem, Computer Aided Geometric Design 20 (6) (2003) 343–362.

[6] H. Lin, Constructing iterative non-uniform B-spline curve and surface to fit data points, Science in China Series F 47 (3) (2004) 315.

[7] Z. Yin, Reverse engineering of a NURBS surface from digitized points subject to boundary conditions, Computers & Graphics 28 (2) (2004) 207–212.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, ACM SIGGRAPH Computer Graphics 26 (2) (1992) 71–78.

[9] N. Amenta, M. Bern, M. Kamvysselis, A new Voronoi-based surface reconstruction algorithm, in: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, ACM, 1998, pp. 415–421.

[10] N. Amenta, M. Bern, Surface reconstruction by Voronoi filtering, Discrete and Computational Geometry 22 (4) (1999) 481–504.

[11] M. Gopi, S. Krishnan, C. Silva, Surface Reconstruction based on Lower Dimensional Localized Delaunay Triangulation, Computer Graphics Forum 19 (3) (2000) 467–478.

[12] M. Gopi, S. Krishnan, A fast and efficient projection-based approach for surface reconstruction, Proceedings. XV Brazilian Symposium on Computer Graphics and Image Processing 1 (1) (2000) 179–186.

[13] M. Attene, M. Spagnuolo, Automatic Surface Reconstruction from Point Sets in Space, Computer Graphics Forum 19 (3) (2000) 457–465.

[14] C. Kuo, H. Yau, A Delaunay-based region-growing approach to surface reconstruction from unorganized points, Computer-Aided Design 37 (8) (2005) 825–835.

[15] T. Dey, S. Goswami, Provable Surface Reconstruction from Noisy Samples, Computational Geometry 35 (1-2) (2006) 124–141.

[16] M. Siqueira, D. Xu, J. Gallier, L. G. Nonato, D. M. Morera, L. Velho, A new construction of smooth surfaces from triangle meshes using parametric pseudo-manifolds, Computers & Graphics 33 (3) (2009) 331–340.

[17] B. Juttler, A. Felis, Least-squares fitting of algebraic spline surfaces, Advances in Computational Mathematics 17 (1) (2002) 135–152.

[18] N. Kojekine, V. Savchenko, I. Hagiwara, Surface reconstruction based on compactly supported radial basis functions, Citeseer, 2004, pp. 218–231.

[19] Y. Ohtake, A. Belyaev, H.-P. Seidel, Ridge-valley lines on meshes via implicit surface fitting, ACM SIGGRAPH 2004 1 (212) (2004) 609.

[20] X. Wu, M. Yu, W. Xia, Implicit fitting and smoothing using radial basis functions with partition of unity, in: Ninth International Conference on Computer Aided Design and Computer Graphics, IEEE Computer Society, 2005, pp. 139–148.

[21] J. Barhak, a. Fischer, Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques, IEEE Transactions on Visualization and Computer Graphics 7 (1) (2001) 1–16.

[22] A. Gálvez, A. Iglesias, A. Cobo, J. Puig-Pey, J. Espinola, Bézier curve and surface fitting of 3D point clouds through genetic algorithms, functional networks and least-squares approximation, in: Proceedings of the 2007 international conference on Computational science and Its applications-Volume Part II, Springer-Verlag, 2007, pp. 680–693.

[23] X. He, C. Li, Y. Hu, R. Zhang, S. X. Yang, G. S. Mittal, Automatic sequence of 3D point data for surface fitting using neural networks, Computers & Industrial Engineering 57 (1) (2009) 408–418.

[24] U. Khan, A. Terchi, S. Lim, D. Wright, S. Qin, 3D freeform surfaces from planar sketches using neural networks, in: Neural Information Processing, Springer, 2006, pp. 651–660.

[25] F. Krause, a. Fischer, N. Gross, J. Barhak, Reconstruction of Freeform Objects with Arbitrary Topology Using Neural Networks and Subdivision Techniques, CIRP Annals - Manufacturing Technology 52 (1) (2003) 125–128.

[26] T. Martinetz, K. Schulten, Topology representing networks, Neural Networks 7 (3) (1994) 507–522.

[27] J. C. R. Horn, R. A., Matrix Analysis, Cambridge University Press, 1990.

[28] P. Azariadis, Parameterization of clouds of unorganized points using dynamic base surfaces, Computer-Aided Design 36 (7) (2004) 607–623.

[29] H. Woo, E. Kang, S. Wang, K. H. Lee, A new segmentation method for point cloud data, International Journal of Machine Tools and Manufacture 42 (2) (2002) 167 – 178.

[30] C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[31] H. G. E. Rumelhart, D. E., R. J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.

[32] J. Heaton, Introduction to Neural Netowrks with Java, Heaton Research, 2005.

[33] M. Alexa, Differential coordinates for local mesh morphing and deforma-

tion, The Visual Computer 19 (2) (2003) 105–114.

[34] M. Buhmann, Radial Basis Functions: Theory and Implementations, Cambridge Monographs on Applied and Computational Mathematics, 2003.

[35] D. T. Lee, B. J. Schachter, Two algorithms for constructing a delaunay triangulation, International Journal of Parallel Programming 9 (1980) 219–242.

[36] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, T. R. Evans, Reconstruction and representation of 3d objects with radial basis functions, in: Proceedings of the 28th Computer graphics and interactive techniques, SIGGRAPH '01, 2001, pp. 67–76.