Optimizing B-Spline Surface Reconstruction for Sharp Feature Preservation

Daniel Lee
Carnegie Vanguard High School
Houston, USA
daniel8lee58@gmail.com

Isabella Quan
Westlake High School
Austin, USA
isabella.quan6@gmail.com

Chris Wu
Westlake High School
Austin, USA
cw86459@eanesisd.net

Jason Wu
Clements High School
Sugar Land, USA
jason.stephen.wu@gmail.com

Dan Tamir*
Department of Computer Science
Texas State University
San Marcos, USA
dt19@txstate.edu

Naphtali Rishe
School of Computing and Information Sciences
Florida International University
Miami, USA
rishe@cs.fiu.edu

***Abstract*—Methods of surface reconstruction from 3D point clouds have received much attention in recent years due to their vast array of applications and the increasing supply of accurate 3D data. Providing smoothness, local modification, and robustness to noise, the B-spline surface fitting is one of the most popular of such methods. However, a problem encountered when using B-spline surface reconstruction is the representation of sharp features: corners and edges tend to be smoothed out. We propose an approach to sharp feature preservation which relies on curvature analysis of the B-spline surface. B-spline patches that have high curvature and are surrounded by patches with low curvature are identified as those representing sharp features. The location of sharp features is then determined through interpolation from low-curvature patches surrounding the identified patches. Finally, these features are preserved through repeated addition of points to the point cloud. We evaluate our sharp feature preservation algorithm at varying levels of noise, demonstrating its high accuracy at low noise and moderate robustness as the noise increases.**

***Index Terms*—computer graphics, analytical modeling**

I. Introduction

Surface reconstruction is a growing area of research that investigates methods of obtaining 3D computer models of physical objects from data gathered through various scanning techniques. Recent advancement in both scanning technology and computer graphics has fueled increased demand for robust and accurate surface reconstruction methods, as these methods have widespread applications in areas including engineering, medical imagery, and computer-generated imagery (CGI) animation [1]. Conventional methods of surface reconstruction tend to rely on modeling objects as polygonal meshes, most ubiquitously those generated through Delaunay triangulation [2].

One alternative approach to surface reconstruction which has received significant attention is parametric surface fitting. Parametric surfaces exhibit useful properties that make them favorable in certain scenarios compared to polygonal meshes. Such surfaces have C^2 continuity, permitting more accurate representations of smooth objects, in addition to greater robustness to noise [3]. Further, parameterization allows calculation of the coordinates of any point on the surface, enabling one to conduct a more accurate analysis of surface features [10].

The predominant technique of parametric surface reconstruction is based on the generation of B-spline surface patches, through an implementation known as the non-uniform rational B-spline (NURBS) method [3]. One issue with B-spline surfaces is that due to smoothness, sharp features such as corners and edges tend to be represented poorly. This is a problem, which, despite recent progress, still remains and limits the effectiveness of B-spline and NURBS surface fitting methods [4], [11].

This paper seeks to address the problem of sharp feature preservation in B-spline surface reconstruction. We propose a solution that preserves sharp features by analyzing the curvature of the parametric surface, identifying B-spline surface patches belonging to sharp features, using those patches to interpolate the actual locations of sharp features, and then adding points to the point cloud to accurately represent those features. We hypothesize that local curvature analysis of B-spline patches will allow for accurate detection and preservation of sharp features up to moderate levels of noise.

The novelty of this approach stems from the fact that while sharp feature preservation during surface reconstruction has been tackled in the past, existing approaches consist primarily of first detecting points of the point cloud which belong to sharp features, and then constructing the B-spline with added expression of these points. A downside of such approaches is that they are more sensitive to falsely identifying noisy points as sharp features, in comparison to surface analysis, which is not as easily influenced by noise [12]. In addition, our use of point interpolation allows for a more accurate representation of edges and corners which may not be defined

*Corresponding Author.

978-1-7281-3783-4/20/$31.00 ©2020 IEEE

by the initial control points. Lastly, a method of sharp feature optimization which detects edges and corners from the surface domain rather than the point cloud has the potential to improve efficiency and reduce computational cost.

The rest of this paper is organized as follows. In Section 2, we provide background information on surface reconstruction and especially on parametric B-spline fitting. In Section 3, we review the relevant work that has been done on sharp feature preservation for surface reconstruction. We describe our approach to the problem in Section 4. We outline the experiment we use to test our algorithm and present the results of this experiment in Section 5. We evaluate these results in Section 6. We conclude and discuss future research in Section 7.

II. Background

Methods of surface reconstruction often take as their input point clouds in $\mathbb{R}^3$ obtained from 3D scanning devices. Inputs sometimes also include oriented or unoriented normal vectors for each point, as well as other types of scanner information [7]. In the present work, we focus on methods that generate models based on point clouds alone. The most popular approach to surface reconstruction is to represent the model as a polygonal mesh, most commonly as one composed of triangles generated through Delaunay triangulation [2].

Parametric surface fitting is a popular surface reconstruction alternative to mesh generation. The most common form of this technique is NURBS, which is based on B-spline curves and surfaces. B-spline curves are generalizations of Bézier curves. A Bézier curve for the points $\{p_0, p_1, p_2, p_3\}$ in $\mathbb{R}^3$ is the parameterized curve

$$\boldsymbol{p}(u) = \boldsymbol{u}^T \boldsymbol{M}_B \boldsymbol{p}$$

where $0 \leq u \leq 1$,

$$\boldsymbol{u} = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}, \ \boldsymbol{p} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix},$$

and $\boldsymbol{M}_B$ is a 4 by 4 matrix of constants, known as the Bézier geometry matrix [10]. The values of $\boldsymbol{M}_B$ are such that $\boldsymbol{p}(u)$ satisfies the following conditions: $\boldsymbol{p}(0) = p_0$, $\boldsymbol{p}(1) = p_3$, $\boldsymbol{p}'(0) = \frac{p_1 - p_0}{3}$, and $\boldsymbol{p}'(1) = \frac{p_3 - p_2}{3}$. These conditions ensure that the Bézier curve is contained in the convex hull of $\{p_0, p_1, p_2, p_3\}$. For an arbitrary number of points, the Bézier curve approximation for those points is often constructed by repeatedly joining the Bézier curves of every consecutive set of 4 non-overlapping points.

A B-spline curve for the points $\{p_0, p_1, p_2, p_3\}$ is similar to the Bézier curve for those points, but only approximating the region from p_1 to p_2 [10]. Because of this, the B-spline curve for any set of points must be constructed for the points p_i to p_{i+3}, p_{i+1} to p_{i+4}, and so on. Construction of B-splines thus requires three times as much work as for Bézier curves, but allows for both C^1 and C^2 continuity at joint points, unlike Bézier curves, which exhibit only C^0 continuity at joints. B-spline surfaces share the same smoothness properties and are constructed by parameterizing a B-spline curve over an additional parameter v, so that

$$\boldsymbol{p}(u, v) = \boldsymbol{u}^T \boldsymbol{M} \boldsymbol{P} \boldsymbol{M}^T \boldsymbol{v} = \sum_{i=0}^{3} \sum_{j=0}^{3} b_i(u) b_j(v) p_{ij}.$$

Here, $\boldsymbol{M}$ is again a 4 by 4 matrix of constants, $\boldsymbol{P}$ is the 4 by 4 matrix of p_{ij} for $0 \leq i, j \leq 3$,

$$\boldsymbol{v} = \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix},$$

and $b_i(u)$ and $b_j(v)$ are cubic functions known as blending polynomials.

An important feature of the B-spline surfaces is that they preserve C^0, C^1, and C^2 continuity. This smoothness aids in surface approximation but precludes accurate representation of edges and corners. However, derivative continuity of B-spline surfaces can be broken at certain points by repeating those points multiple times in the point cloud. Figure 1 demonstrates this effect.

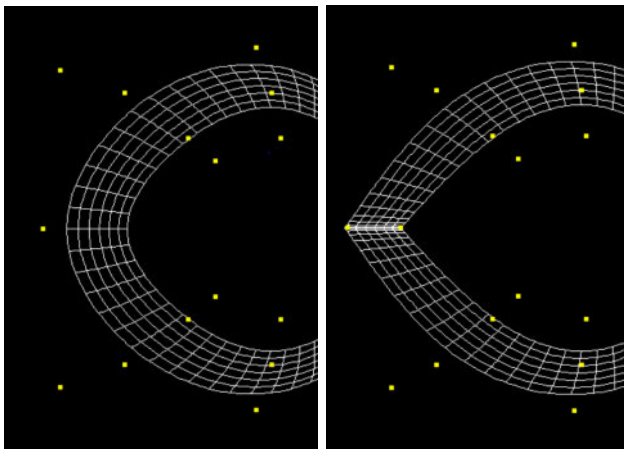

Fig. 1. Sharp Feature Preservation. Using repeated points to preserve sharpness in a B-spline surface.

Our sharp feature detection algorithm relies on finding the curvature of the parametric surface used to approximate the model. The two measures of a surface's curvature at a specified point are its mean curvature and Gaussian curvature, which are respectively equal to the arithmetic mean and the product of the surface's two principal curvatures at that point. We use the absolute value of the mean curvature to determine how "curved" surface patches are. Finding the mean curvature of a parametric surface $P(u, v)$ involves finding the first and second partial derivatives of P, as well as the normal vector of P [20]. These are used to calculate what are called the fundamental coefficients of P, given by:

$$E = P_u \cdot P_u$$

$$F = P_u \cdot P_v$$

$$G = P_v \cdot P_v$$

and

$$L = P_{uu} \cdot n$$

$$M = P_{uv} \cdot n$$

$$N = P_{vv} \cdot n,$$

where

$$n = \frac{P_u \times P_v}{|P_u \times P_v|}.$$

The mean curvature is then equal to

$$\frac{EN - 2FM + GL}{2(EG - F^2)}.$$

We select the mean curvature for use in the present algorithm because it yields nonzero values for features resembling edges and corners, while the Gaussian curvature is zero at valleys and ridges due to one principal curvature being zero.

III. Related Work

Because sharp features are generally of interest when conducting any form of surface reconstruction, numerous methods have been developed for identifying such features. These methods can generally be divided into three categories. First are those which require a polygonal mesh as their input and identify sharp features using the normal and connectivity information of the mesh. Second are those which operate directly on the point cloud and identify sharp features through analysis of point neighborhoods [12]. Third are those which detect sharp features from a parametric surface constructed from the point cloud. In our research, the input of our algorithm is either a raw point cloud or a B-spline surface, not a polygonal a mesh. Although it is possible to construct a polygonal mesh from the point cloud and then apply a mesh-based sharp feature detection method, mesh generation without any additional connectivity or normal information tends to chamfer edges and corners, making it sub-optimal for sharp feature detection [13]. In addition, mesh methods of sharp feature detection serve primarily as post-processing steps for mesh-based surface reconstruction, which does not fit our goal of B-spline surface optimization. For these reasons, we do not need to explore the state of the art on the detection of sharp features from mesh input. Rather, we will direct our attention to methods that identify sharp features from either the point cloud directly or from a parametric surface input.

Methods of sharp feature detection from the point cloud typically begin with identifying point neighborhoods, often through a k-nearest neighbor algorithm, and then analyze these neighborhoods to identify points that belong to sharp features. We will first address point cloud detection methods, which, rather than specifically finding sharp features, extract line-type features more generally. Such features include edges, creases, and border loops, and while they tend to have high local curvature, they are not necessarily sharp. Gumhold et al. [18] propose a method for identifying surface features: they first use a Delaunay tetrahedralization and subsequent Riemann tree to obtain connectivity information of the point cloud, allowing them to find point neighborhoods. They then analyze these neighborhoods using principal component analysis (PCA), yielding a dense set of points representing line-type features. A minimum-spanning tree is then used to reduce these points to a sparse set representing feature lines. Pauly et al. [19] use a similar method; they identify point neighborhoods and utilize PCA coupled with a minimum spanning graph to approximate sharp features. However, Pauly improves upon Gumhold's method by adding a multiscaling analysis of neighborhoods of varying sizes. This makes their algorithm more robust, especially to noisy data points, but increases computational cost. An important limitation of both Gumhold's and Pauly's methods is that they detect any visually prominent surface feature, while neither method differentiates between sharp and non-sharp features.

There have been several approaches proposed specifically for detecting sharp features from point clouds. Demarsin et al. [16] identify sharp features using normal estimation; they first construct Delaunay triangulations of each point's k-nearest neighbors, which allows for normal vectors to be estimated for each point along the object's surface. Using normal information, they then locate points that are likely to belong to sharp features and cluster the point cloud according to the locations of these features. Similar to Gumhold's Riemann tree method, Demarsin uses a minimum spanning tree to extract sharp feature lines from the point cloud. An important note on Demarsin's method is that it focuses solely on detecting closed edges, while in the present work, we aim to detect all sharp features. Similarly to Demarsin, Weber et al. [14] use the triangulation of local point neighborhoods to estimate surface normals. However, they introduce the use of a Gaussian map to detect when normal vector behavior indicates that a point belongs to a sharp feature. One downside of Weber's approach is that it is quite sensitive to noisy points around sharp features. Bazazian et al. [12] propose an improvement on Weber's method. They still apply a Gaussian map to point neighborhoods found through a k-nearest neighbor search. However, they estimate normals using PCA and analysis of the eigenvalues of each neighborhood's covariance matrix, as opposed to simple triangulation. The result is an algorithm that is faster and more robust to noise than Weber's. Despite this improvement, Bazazian's approach achieves under 90% precision with no noise present, an accuracy that we hypothesize could be surpassed through curvature analysis.

There have been relatively few methods proposed that identify sharp features from parametric surface input. Weber et al. [17] extend their Gaussian map approach to improve sharp feature preservation in moving least squares (MLS) surface reconstruction, and their optimization proves successful up to roughly 0.5% noise. However, MLS surface reconstruction, despite being a parametric surface fitting method, is different from NURBS and B-spline techniques, which are the focus of the present work. Some strategies have been proposed for preserving sharp features in B-spline surfaces, including that of Leal et al. [4], who use an evolutionary algorithm to find

optimal weights of points in NURBS patches to minimize their error in approximating sharp features. Their method is able to achieve a 14% reduction in the distance between the model and the NURBS surface, although their algorithm is computationally expensive. A few others, particularly [5] and [6], have tackled NURBS surface optimization with the same approach of employing algorithms to find optimal point weights for error minimization. Some limitations of such strategies are that they are not tailored specifically to sharp features and tend to have high computational costs [4], whereas our method focuses specifically on sharp feature representation without having to optimize the entire surface.

IV. Sharp Feature Preservation

We now describe our approach to sharp feature preservation for B-spline surface reconstruction. The first step of our method is to construct an initial B-spline surface modeling the point cloud. This parametric surface consists of many B-spline patches, each representing four points. We then calculate k for each patch, defined as the absolute value of the mean curvature at the *center point* of a patch, where the center point for a B-spline patch parameterized by $\boldsymbol{p}(u, v)$ for $0 \leq u, v \leq 1$ is defined as $\boldsymbol{p}(\frac{1}{2}, \frac{1}{2})$.

Once the curvature of each patch has been calculated, we establish a threshold $\epsilon > 0$ such that if $k < \epsilon$ for a patch, that patch is classified as flat, and if $k > \epsilon$, that patch is classified as curved. After classifying them as flat or curved, we compare each patch to its 8 neighboring patches. If a curved patch has 4 or more flat neighbors, we classify it as sharp. These patches are predicted to belong to sharp features of the original model.

Our sharp feature detection algorithm is based on the fact that B-spline patches representing sharp features will have high curvature and be neighbored by patches with low curvature. Because any non-sharp region of a surface will resemble a flat plane when magnified to a high enough degree, this is a valid assumption to make when the point cloud has reasonable resolution near sharp features.

After identifying sharp patches, we locate sharp features themselves by interpolating based on the surrounding flat patches. Because a sharp feature is either a corner or an edge, we expect it to be either an intersection of two planes (an edge) or an intersection of more than two planes (a corner). Because every sharp patch must be surrounded by at least four flat patches, we use these flat patches to approximate planes and then find their intersections, predicting the locations of sharp features. Doing this allows us, for each sharp patch, to generate a point which we predict to lie on the sharp feature that the patch represents.

Once we have found points corresponding to the sharp feature represented by every sharp patch, we add each of these points to the point cloud three times. We then regenerate the B-spline surface at and around sharp patches, and the repeated points break continuity and allow for sharp feature preservation.

V. Experiment and Results

We test our sharp feature preservation algorithm using a point cloud consisting of a box with a cylinder on top. The point cloud contains roughly 34 000 uniformly distributed points and includes a total of 14 edges and 8 corners, as shown in Figure 2. The geometric simplicity of the model allows us to know the exact location of each edge and corner, which will facilitate precise evaluation of our accuracy in preserving sharp features.

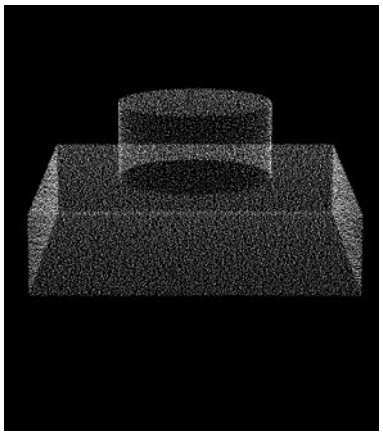

Fig. 2. A Point Cloud used to test our algorithm.

We first model this point cloud with a normal B-spline surface and then apply our sharp feature preservation algorithm. We gauge our algorithm's accuracy by finding the average error of our optimized surface in approximating the sharp features of the original model. Our error estimates are found by calculating the average distance from the model to the parametric surface at each corner point and at 100 equally-spaced points along each edge, with distances normalized to the average distance between adjacent points of the point cloud. We calculate this error for four categories of sharp features: straight edges, the lower circular edge, the upper circular edge, and corners, as shown in Figure 3. We also test our algorithm's robustness to noise, analyzing each sharp feature separately in a bounding box of side length equal to one-eighth that of the full model. We add to each original point a random vector of magnitude equal to the noise level, represented as a percentage of the length of the bounding box, and then evaluate the accuracy of our algorithm at varying levels of noise. We also record noise levels as decibels of signal-to-noise ratio (SNR); this aids in expressing the power of noise in relation to the strength of the point cloud.

At all tested noise levels, our algorithm has been highly successful in detecting sharp features of the model with our curvature threshold ϵ set at 0.10. Patches representing straight edges, circular edges, and corners were all identified accurately; edges consisted of a 1-patch-wide band of patches, while corners consisted of a single patch. No patches on flat faces of the model, nor any on the lateral face of the cylinder, were identified as sharp. Our algorithm was thus 100% accurate at detecting sharp patches.

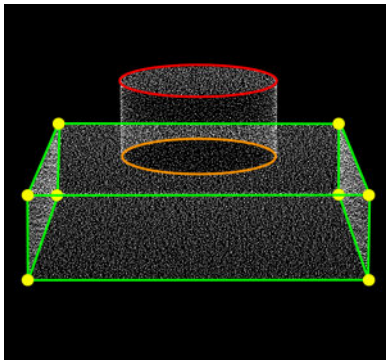

Fig. 3. Sharp Features. A Point Cloud with straight edges (green), lower circular edge (orange), upper circular edge (red), and corners (yellow) marked.

After detection of sharp patches, our algorithm used surrounding flat patches to interpolate the location of sharp features, added repeated points to represent those features, and reconstructed the B-spline surface around those features for sharp feature preservation. Figure 4 displays the effect this had on different sharp features. The error values of the parametric surface in representing sharp features prior to and after optimization for various noise levels are shown in Table 1.

Noise Level	Sharp Feature	Initial Error	Final Error	Reduction
0.0%	Straight Edges	0.3955	0.0000	100.00%
	Lower Circular Edge	0.3917	0.0002	99.95%
	Upper Circular Edge	0.3997	0.0002	99.95%
	Corners	0.4958	0.0000	100.00%
0.5% (46dB)	Straight Edges	0.3952	0.0444	88.77%
	Lower Circular Edge	0.3906	0.0428	89.04%
	Upper Circular Edge	0.3992	0.0450	88.73%
	Corners	0.4951	0.0499	89.92%
1.0% (40dB)	Straight Edges	0.3939	0.0890	77.41%
	Lower Circular Edge	0.3901	0.0829	78.75%
	Upper Circular Edge	0.3979	0.0849	78.66%
	Corners	0.4933	0.0927	81.21%
2.0% (34dB)	Straight Edges	0.3920	0.2172	44.59%
	Lower Circular Edge	0.3879	0.2241	42.23%
	Upper Circular Edge	0.3966	0.2053	48.23%
	Corners	0.4903	0.2468	49.66%

TABLE I
Sharp feature error values prior to and after surface optimization, normalized to the average distance between adjacent points in the original point cloud.

When no noise was present, our optimization algorithm preserved edges and corners almost perfectly, with error reductions of over 99.90% for all sharp features. At the 0.5% noise level, our algorithm remained effective, reducing errors by nearly 90% for all sharp features. At 1.0% noise, our percent reduction dropped to around 80%, and at 2.0% noise, it fell to 40% to 50% depending on the type of sharp feature. Past 2.0% noise, our error reduction continues to decrease and our algorithm begins to incorrectly label flat patches as sharp. That indicates that our algorithm performs best at low to moderate levels of noise.

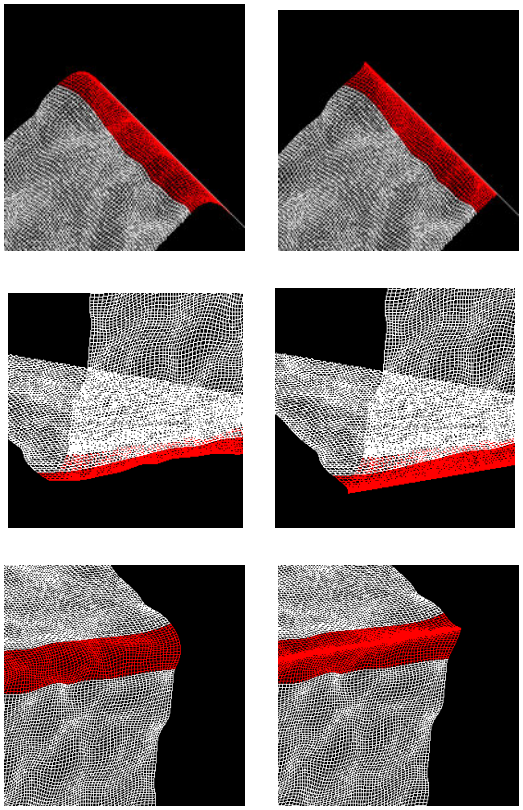

Fig. 4. Sharp Feature Optimization. *The B-spline surface before (left) and after (right) sharp feature preservation through interpolation and addition of repeated points. Here, patches identified as sharp are highlighted red. The specific sharp features represented are (from top to bottom): a straight edge, the lower circular edge, and the upper circular edge.*

VI. Evaluation

Our curvature-based sharp feature preservation algorithm for B-spline surface fitting proves successful and can tolerate up to roughly 2% noise, corresponding to 34db of SNR. We have thus demonstrated a method of surface reconstruction optimization which specifically targets and represents sharp features from a parametric B-spline surface. In contrast to most prior approaches to sharp feature preservation, our method does not require mesh input, nor does it require the point cloud itself. Rather, it is a post-processing algorithm that operates on surface patches.

Our conceptually simple algorithm shows very favorable results at low levels of noise, with only a small computational cost, unlike machine learning-based methods of NURBS optimization, which tend to require more complex analysis and longer training time. In addition, the fact that we detect sharp features based on the B-spline surface allows circumvention of many of the challenges encountered by point-cloud methods. For instance, our algorithm takes advantage of the natural resilience of B-spline surfaces to noisy control points, affording our method greater robustness to noise than point-cloud-based methods. At the same time, our method retains high accuracy; its precision is par with or surpasses that of the aforementioned methods; yet the fact that we analyze surface curvature offers

a convenient approach to sharp feature preservation free from conventional requirements of neighborhood searches, normal estimation, or PCA.

VII. Conclusion and Future Research

We have designed and implemented a novel approach to optimizing parametric surface reconstruction through curvature analysis of B-spline patches. Our algorithm provides a means of sharp feature preservation, which, unlike other methods, relies only on the parametric surface, rather than on connectivity information, normal estimates, or other point-based data. Our method proves highly successful at detecting and representing sharp features at low levels of noise and is moderately robust as noise increases.

In the future, we hope to improve upon our sharp feature preservation algorithm by testing it against increasingly complex point clouds, such as those with more intricate sharp features or with non-uniform point density, and adjusting our procedure to handle more difficult models. We would also like to increase our algorithm's effectiveness at higher levels of noise by improving the versatility of our sharp feature interpolation procedure. Finally, we would like to explore the potential of curvature analysis to optimize other forms of parametric surface fitting.

References

[1] Inge Söderkvist. *Introductory Overview of Surface Reconstruction Methods*. Department of Engineering Sciences and Mathematics, Luleå University of Technology, 2011.

[2] Seng Poh Lim, Habibollah Haron. *Surface reconstruction techniques: A review*. Artificial Intelligence Review, 2012.

[3] Nallig Leal, Esmeide Leal, John William Branch. *Simple Method for Constructing NURBS Surfaces from Unorganized Points*. Proceedings of the 19th International Meshing Roundtable, 2010.

[4] Nallig Eduardo Leal, Oscar Ortega Lobo, and John William Branch. *Improving NURBS Surface Sharp Feature Representation*. International Journal of Computational Intelligence Research, 2007.

[5] E. Cuartas. *Optimizacion de la representacion con superficies NURBS de imagenes de rango mediante el algoritmo de Levemberg-Marquardt*. EITI Universidad Nacional de Colombia sede Medellin, 2004.

[6] Jorn Mehnen, Weinert Weinert. *Discrete NURBS-Surface Approximation using an Evolutionary Strategy*, 2001. DOI:10.17877/DE290R-15245.

[7] Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Gael Guennebaud, Joshua Levine, Andrei Sharf, Claudio Silva. *A Survey of Surface Reconstruction from Point Clouds*. Computer Graphics Forum, Wiley, 2016, pp.27. DOI:10.1111/cgf.12802ff.

[8] Frederic Cazals, Joachim Giesen. *Delaunay Triangulation Based Surface Reconstruction: Ideas and Algorithms*. RR-5393, INRIA. 2004, pp.42. http://hal.inria.fr/inria-00070610.

[9] Jonathan Shewchuk. *Lecture Notes on Delaunay Mesh Generation*. Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 1999.

[10] Edward Angel, Dave Shreiner. *Interactive Computer Graphics: A top-down approach with shader-based OpenGL.— 6th ed.*. Addison-Wesley, Massachusetts, 2012.

[11] Taro Kawasaki, Pradeep Kumar Jayaraman, Kentaro Shida, Jianmin Zheng, Takashi, Maekawa. *An Image Processing Approach to Feature-Preserving B-Spline Surface Fairing*. Computer-Aided Design, Vol. 99, pp. 1-10, June 2018.

[12] Dena Bazazian, Josep R. Casas, Javier Ruiz-Hidalgo. *Fast and Robust Edge Extraction in Unorganized Point Clouds*. In International Conference on Digital Image Computing: Techniques and Applications (DICTA), pp. 1-8, 2015.

[13] Marco Attene, Bianca Falcidieno, Jarek Rossignac, and Michela Spagnuolo. *SharpenBend: Recovering Curved Sharp Edges in Triangle Meshes Produced by Feature-Insensitive Sampling*. IEEE Transactions on Visualisation and Computer Graphics, Vol. 11, No. 2, March/April 2005.

[14] Christopher Weber, Stefanie Hahmann, Hans Hagen. *Sharp Feature Detection in Point Clouds*. Proceedings of Shape Modelling International Conference, pp. 175-186, 2010.

[15] Khang Jie Liew, Ahmad Ramli, Nur Nadiah Abd Hamid, Ahmad Abd Majid. *Sharp edge preservation using bicubic B-spline surfaces*. ScienceAsia 43S(1):20-26, August 2017.

[16] Kris Demarsin, Denis Vanderstraeten, Tim Volodine, Dirk Roose. *Detection of closed sharp edges in point clouds using normal estimation and graph theory*. Computer-Aided Design, 39(4), pp. 276-283, 2007.

[17] Christopher Weber, Stefanie Hahmann, Hans Hagen, Georges-Pierre Bonneau. *Sharp feature preserving MLS surface reconstruction based on local feature line approximations*. Graphical Models, 74 (6), pp. 335345, 2012.

[18] Stefan Gumhold, Xinlong Wang, Rob MacLeod. *Feature Extraction from Point Clouds*. Proceedings of 10th International Meshing Roundtable, 2001.

[19] Mark Pauly, Richard Keiser, Markus Gross. *Multi-scale Feature Extraction on Point-Sampled Surfaces*. Computer Graphics Forum, 22(3), pp. 281-289, 2003.

[20] Ron Goldman. *Curvature formulas for implicit curves and surfaces.* Computer Aided Geometric Design, Volume 22, Issue 7, October 2005, pp 632-658.