Theses and Dissertations

December 2015

Quality Enhancement of 3D Models Reconstructed By RGB-D Camera Systems

Chuanbo Wang
University of Wisconsin-Milwaukee

QUALITY ENHANCEMENT OF 3D MODELS RECONSTRUCTED BY

RGB-D CAMERA SYSTEMS

by

Chuanbo Wang

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Computer Science

at

The University of Wisconsin-Milwaukee

December 2015

ABSTRACT

QUALITY ENHANCEMENT OF 3D MODELS RECONSTRUCTED BY RGB-D CAMERA SYSTEMS

by

Chuanbo Wang

The University of Wisconsin-Milwaukee, 2015
Under the Supervision of Professor Zeyun Yu

Low-cost RGB-D cameras like Microsoft's Kinect capture RGB data for each vertex while reconstructing 3D models from objects with obvious drawbacks of poor mesh and texture qualities due to their hardware limitations. In this thesis we propose a combined method that enhances geometrically and chromatically 3D models reconstructed by RGB-D camera systems. Our approach utilizes Butterfly Subdivision and Surface Fitting techniques to generate smoother triangle surface meshes, where sharp features can be well preserved or minimized by different Surface Fitting algorithms. Additionally the global contrast of mesh textures is enhanced by using a modified Histogram Equalization algorithm, in which the new intensity of each vertex is obtained by applying cumulative distribution function and calculating the accumulated normalized histogram of the texture. A number of experimental results and comparisons demonstrate that our method efficiently and effectively improves the geometric and chromatic quality of 3D models reconstructed from RGB-D cameras.

TABLE OF CONTENTS

List of Figures

Acknowledgments

Chapter 1: Introduction

Low-cost range sensors such as Microsoft's Kinect sensor have attracted researchers in many areas including computer vision[9][10][11], augmented reality (AR) [12], robotics[13][14], human computer interaction(HCI)[15][16][17] and even medical imaging[18]. 3D Reconstruction of real-world objects, human body and indoor environment is an essential component in the systems mentioned above. However the quality of 3D reconstruction using Kinect sensor is limited by its noisy output due to its low-cost hardware.

Researchers have proposed different methods for improving the reconstruction quality. A possible taxonomy of these methods could be based on the targeting data set. The first group of researches enhance the reconstruction quality by recovering the depth map captured by the Kinect sensor. [19] proposes a new 3D reconstruction workflow based on hole-filling algorithms of 2D depth image combined with 3D point cloud filtering algorithms. [20] proposes a depth prediction algorithm based on fitting a polynomial on the unknown depth locations to generate a noise-free depth map. [21] presents a depth filtering algorithm taking advantage of the color and motion information from the RGB camera of the Kinect sensor to increase the quality of the depth map. However these works only enhance the geometrical quality of 3D models with the raw RGB frames.

Another group of researches aim at correcting the calibration of depth and color data acquired from the Kinect sensor. [22] presents a maximum likelihood solution for the joint depth and color calibration. [23] presents a calibration algorithm for a depth and

color camera pair that is optimal in the sense of the postulated principles using a planar checkerboard pattern. However their works do not enhance the data itself, therefore they still suffer from the artifacts of the depth map and the raw color image.

A third group of developers combine other cameras with the Kinect sensor as a substitute of the color camera component in the Kinect sensor. [24] and [25] mount a digital single-lens reflex (DSLR) camera to the Kinect sensor and the system uses color images captured by the DSLR instead of the color camera in the Kinect sensor. But their works at least triple the hardware cost of the sensor system.

In this thesis we propose a combined method that enhances geometrically and chromatically colored 3D models reconstructed from the data acquired by the Kinect sensor. Our method makes several contributions to the field of 3D reconstruction from RGB-D camera systems. In Chapter 2, we introduce the hardware structure and specifications of the Kinect sensor and demonstrate how our system generates colored triangle meshes from the output of the KinectFusion system. We explore possible subdivision algorithms and smoothing algorithms for smoothing the surface mesh reconstructed from RGB-D camera systems in Chapter 3, and analyze the algorithms in terms of effectiveness. In Chapter 4, we demonstrate how our system enhances chromatically the texture of 3D mesh models reconstructed from the Kinect sensor. We also describe the extension of histogram equalization algorithm to 3D and colored cases. In Chapter 5, we present and analyze the results of our system. And we discuss the limitations of our system and ideas for future research in Chapter 6.

Chapter 2: 3D Reconstruction from RGB-D Camera Systems

Our mesh enhancement system uses a Microsoft Kinect sensor with KinectFu-sion[1] for 3D surface reconstruction. Mesh triangulation is performed and the RGB information is mapped to each vertex of the 3D surface model. In Figure 2.1, we demonstrate the steps our system performs to construct a colored triangle surface mesh from a Kinect sensor. In the rest of this chapter, we describe the hardware components and specifications of the Kinect sensor, and explain how our system uses KinectFusion to generate 3D surface models.

2.1 Kinect Hardware

Our system uses KinectFusion, which take data from each component in the Kinect sensor(Figure 2.2). The Kinect sensor has a infra-red projector and two cameras, a depth camera and a normal RGB camera. The infra-red projector, locates on the left side, emits a pre-defined pattern of hundreds of infra-red rays into the space and projects dots all over the scene. The depth camera on the right side is a monochrome CMOS sensor that captures reflected Infra-Red rays(Figure 2.3) from the scene. By analyzing the deformation of these dots, the Kinect sensor calculates the depth and surface information of the objects in the scene and generates a depth map of it. The infra-red projector combined with the depth camera is the core component of the Kinect sensor, the depth sensor, which has a default sensing range of minimum 800 millimeters and maximum 4000 millimeters[7]. The Kinect sensor can however be switched to a "near mode" that provides a sensing range of 500 millimeters to 3000 millimeters in-

stead of the default range. The camera in the middle is a 8-bit color VGA video camera

that stores three channel data in the resolution of 640x480 at 30 fps. It also supports the

resolution of 1280x960 at a lower frame rate of 12 fps and other color spaces such as

YUV.

The device has a viewing angle of 43 degrees vertically and 57 degrees horizon-

tally. It also has a tilt motor that allows vertically an additional ±27 degree range, which

effectively doubles the volume that the field of view could surround. Apart from the cam-

eras, the Kinect sensor has a multi-array microphone containing four microphones,

which allows the device to find the location of the sound source and the direction of the

audio wave. An example of a color frame and a depth frame is showed in Figure 2.4.
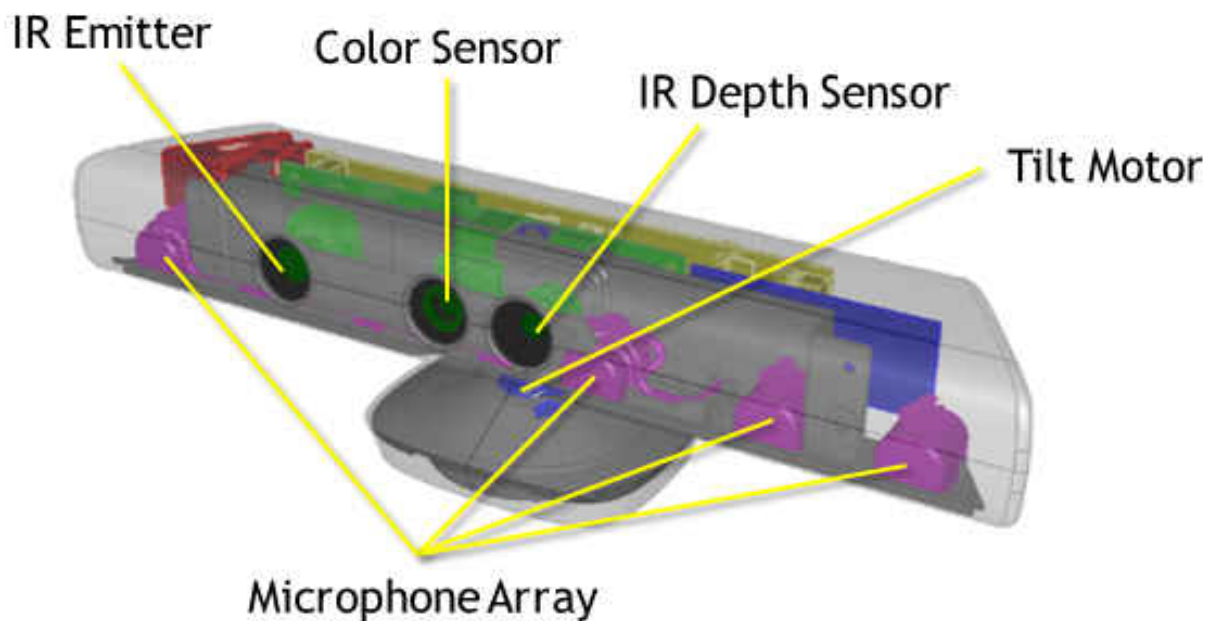
Figure 2.2 Infra-Red Projector, RGB Camera, Depth
Camera[7]
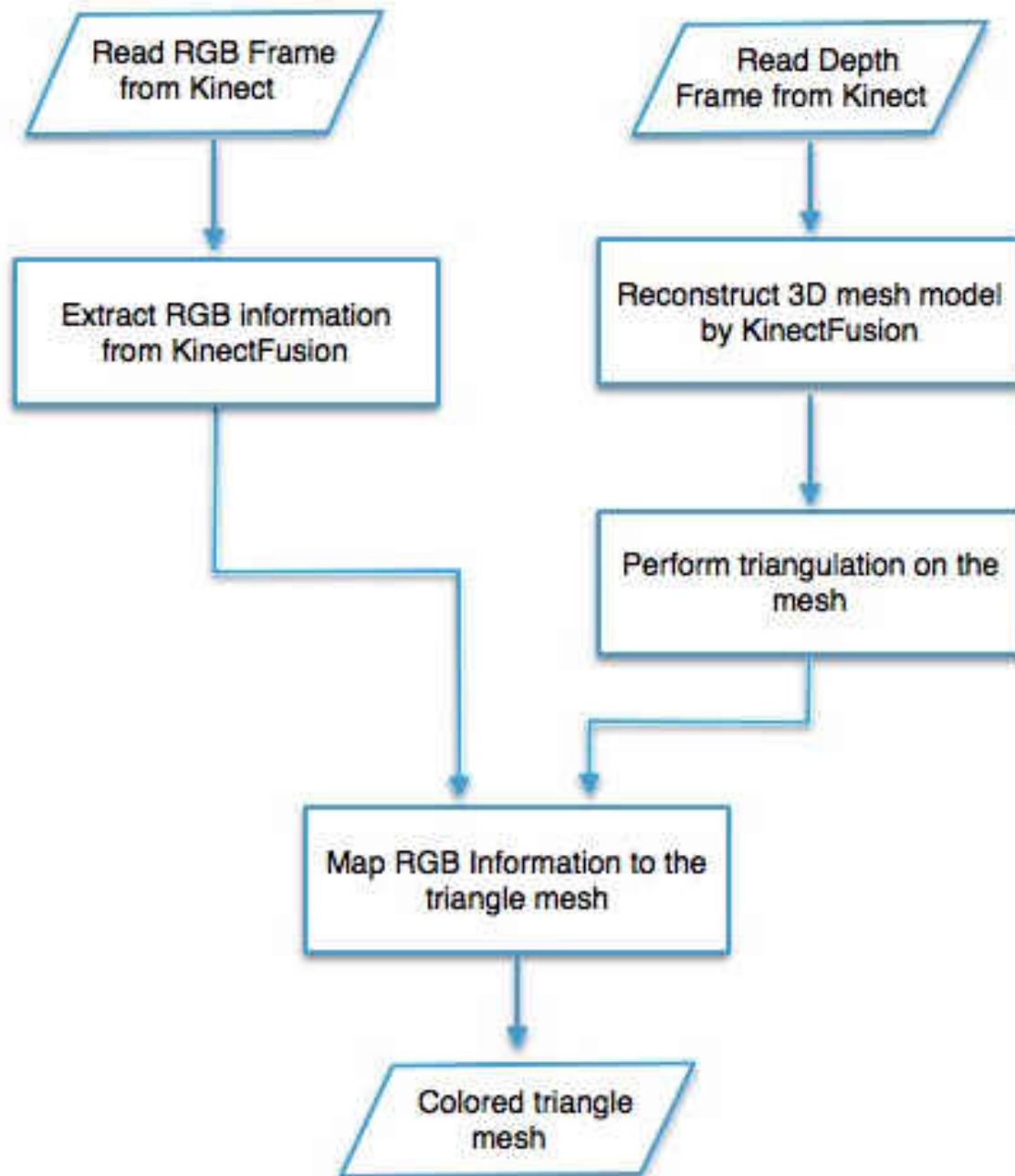
Figure 2.1 Our system uses KinectFusion to create a point cloud within a selected distance range, and then the point cloud is triangulated into a triangle mesh. Color information captured from the Kinect is used as the input of the mapping algorithm to create a triangle mesh with RGB values on each vertex.

Figure 2.3 Infra-Red dots projected on the scene[27]

Figure 2.4 An example of a color frame and a depth frame

2.2 KinectFusion

KinectFusion is a 3D reconstruction system developed by Microsoft Research in 2011. It takes live depth data from a moving Kinect camera and creates high-quality and geometrically accurate 3D models in real-time by allowing a user to pick up a standard Kinect sensor and move around the target object. The system keep tracking the 6 degrees of freedom pose of the Kinect sensor and fuse the data from depth frames to build a 3D surface model[1].

KinectFusion reconstructs 3D models in four main steps. First, the system preprocesses the raw depth map from the Kinect sensor and generates a dense vertex map and a normal map. After that, the location and orientation of the Kinect sensor are also estimated from the depth data by an iterative closest point algorithm. In the third step, the system integrates each depth frame into the reconstructed 3D model using a volumetric, truncated signed distance function. Finally, a ray-tracing algorithm computes the surface positions. Figure 2.5 provides the workflow of KinectFusion system.

Compute Surface Vertex and Normal maps	Estimate the 6DOF Pose of the Sensor	Integrates Depth data into a Global 3D volume	Ray-tracing the 3D Volume to compute Surface Positions
Surface Measurement	Pose Estimation	Volumetric Integration	Ray-tracing
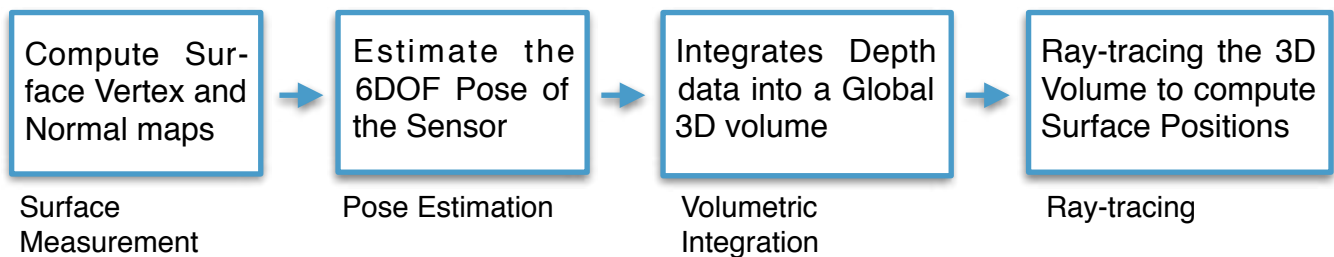
Figure 2.5: The overall system workflow of KinectFusion

2.2.1 Surface Measurement

First, the KinectFusion take the raw depth data from the Kinect sensor as input and converts it into the floating point depth values in meters, which is the distances from pixels in the scene to the depth sensor, using a function called NuiDepthPixelToDepth. Then vertex positions are computed in the camera space from the depth values. And the normal of each point is obtained by calculating the cross product of the neighbor vertex above it and the neighbor vertex on its right side. After that, the system generate a point cloud with location and normal for each point, which is integrated into a the global 3D reconstruction at the third step.

2.2.2 Pose Estimation

KinectFusion requires the pose of the sensor to integrate each depth frame into the final reconstruction since the sensor is moved by a user to capture color and depth frames from slightly different viewpoints. The system estimates the pose of the sensor using a interactive closest point(ICP) algorithm, which finds a rigid 6 degree-of-freedom alignment transform from the point cloud generated from the current depth frame to the one generated from the last depth frame. With the transforms between frames, the system keeps track of the current sensor pose and its relation to the initial starting frame.

2.2.3 Volumetric Integration

During the third phase, KinectFusion integrates the depth data into a single volumetric representation of the space around the camera. The point cloud generated from the current depth frame is converted into world coordinate system first, given the pose

of the camera from the previous step. Instead of fuse point clouds, the system take advantage of a volumetric signed distance functions(SDF). The SDF assigns positive numbers to voxels outside the surface, negative numbers to voxels inside the surface and zero to voxels on the surface. And the absolute values of the numbers are the distances from the voxels to the surface in the direction of the view ray.

2.2.4 Ray Tracing

Finally, the system reconstructs a surface by ray tracing the SDF. Ray tracing is performed from the location and the orientation of the camera, which is generated in the second step, and stops when a zero crossing(when the number assigned to a voxel by SDF is zero from the third step) is detected indicating the surface.

2.2.5 The Triangle Mesh File Format

The version of KinectFusion we use is v1.8 and it supports three formats for the output 3D model: PLY, STL, and OBJ. The only format that support RGB information extraction is PLY. However, KinectFusion generates PLY files as point clouds without information of edges or faces. To address this issue, we perform mesh triangulation to the PLY file and generate an OFF file. Then we map the RGB information of the PLY file to the OFF file by simply compare the location of each vertex. The result of this process is a colored COFF file, which is a triangle surface mesh with RGBA channels on each vertex.

Chapter 3: Geometry Processing

3D reconstruction from the Kinect sensor with KinectFusion has advantages over similar systems: low cost, easy setup, fast scanning, portable. However there is a trade-off between cost and accuracy. The Kinect system is able to capture object details with a minimum size of approximately ten millimeters, which is also the minimum radius of curvature on a curved surface to be reconstructed[3]. With a minimum camera/object distance and a maximum voxels per meter resolution, KinectFusion generates 3D models that can be dramatically enhanced with respect to the geometric quality. To break the limit of resolution of the Kinect sensor, our system first applies butterfly subdivision to the 3D model as a pre-processing step. Then we improve the smoothness of pre-processed meshes using surface fitting methods. In this chapter, we first explain our choice of surface subdivision methods followed by how it works in our system. Then we discuss respectively two mesh smoothing methods based on surface fitting approach.

3.1 Surface Subdivision

Our system utilizes surface subdivision method to increase the density and improve the smoothness of surface meshes before smoothing. Surface subdivision, in the field of 3D computer graphics, is a method of representing a smooth surface via the specification of a coarser piecewise linear polygon mesh. The smooth surface can be calculated from the coarse mesh as the limit of a recursive process of subdividing each polygonal face into smaller faces that better approximate the smooth surface[4].

3.1.1 Choices of Subdivision Method

Surface subdivision schemes roughly fall into two subcategories: approximating and interpolating. An example of approximating subdivision schemes for triangle meshes is the Loop scheme, which applies average mask to all vertices including the original ones and the new vertices. The butterfly subdivision scheme is an interpolating scheme that only applies average mask to newly created vertices based on the original ones. This differs from the Loop scheme in respect of generating new vertices on the limit surface. Since the goal of the geometry processing of our system is enhancing the quality of a 3D model reconstructed by Kinect from a real object, we choose the interpolating subdivision scheme, butterfly scheme to keep the geometry features of the mesh while being smoothed.

3.1.2 Butterfly Subdivision

Butterfly subdivision is an interpolating subdivision scheme defined for triangle mesh only. At each step of the scheme, each triangle is divided into four smaller triangles by three new points each added to an edge of the triangle. On each edge, a edge point, P', is created following an eight-point rule demonstrated in Figure 3.1[4], which is also the reason that this method is called butterfly. P' is defined as

$$P' = \frac{1}{2}(P_1 + P_2) + 2\omega(P_3 + P_4) - \omega(P_5 + P_6 + P_7 + P_8) \quad (1)$$

where ω is a tension parameter. P1 and P2 are two end points of the edge, P3 and P4 are two other points that form the two triangles that share the edge. And P_k, k = 5, 6, 7,

8, are four points that form triangles(other than the two triangles mentioned above) with edge (P_i, P_j), i = 1, 2, j = 3, 4.
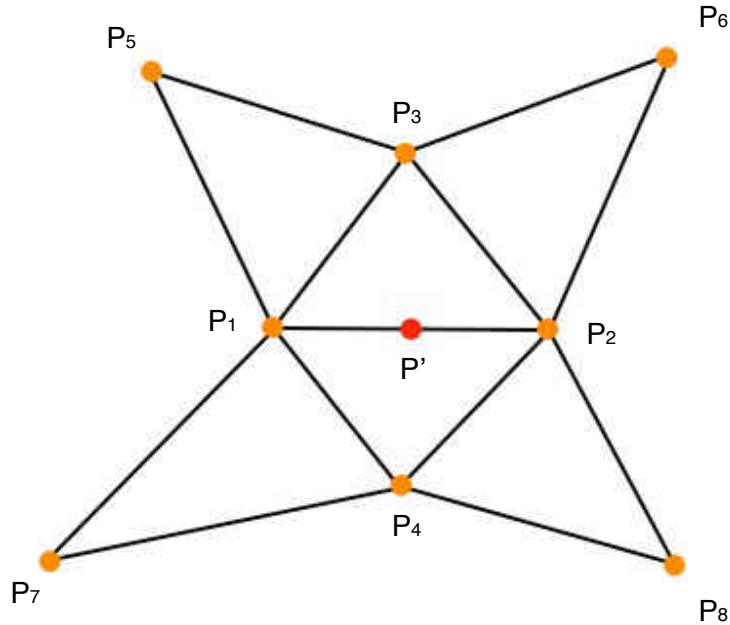
Figure 3.1 The 8-point scheme of butterfly subdivision.
P' is the point newly created on the edge(P_1, P_2).

In scheme (1), ω serves as a tension parameter that as ω tends to zero the limit surface is tightened towards the original control mesh. More specifically, when ω gets close to 0, the new point tends to be added on the midpoint of the edge. ω is found to satisfy the necessary conditions for C^1 continuity when $0 < \omega < \omega_0$, $\omega_0 > 1/16$[2]. In Figure 3.2 we show a triangle of the original surface mesh and the resulting piecewise linear surface mesh generated after one iteration of subdivision using $\omega = 1/16$. And Figure 3.3 shows the overall comparison of the 3D model of a grabbing hand before and after the butterfly subdivision scheme with a tension parameter $\omega = 1/16$.

12

3.2 Isotropic Surface Fitting

The first mesh smoothing method we perform after subdivision is an isotropic surface fitting method based on local surface fitting with optimum vertex projection technique. This algorithm fits an analytical quadric surface for each vertex using the least-square fitting method, then the center of the maximum inscribe circle(MIC) at each vertex is calculated. Finally, the vertex is updated with the projection point of the center of the maximum inscribe circle onto the local fitted surface. Figure 3.4 shows the work-flow of the surface smoothing procedure.
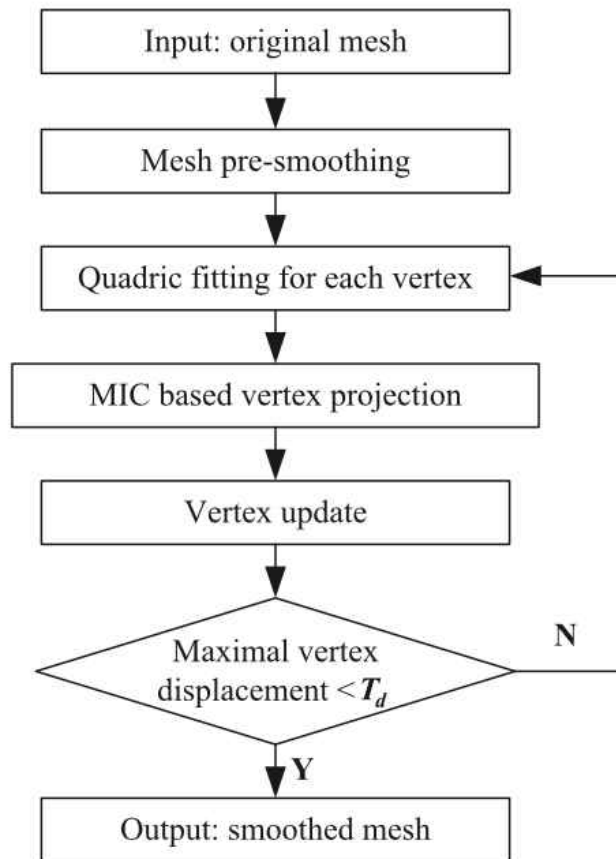
Figure 3.2 The workflow of isotropic surface fitting[5]
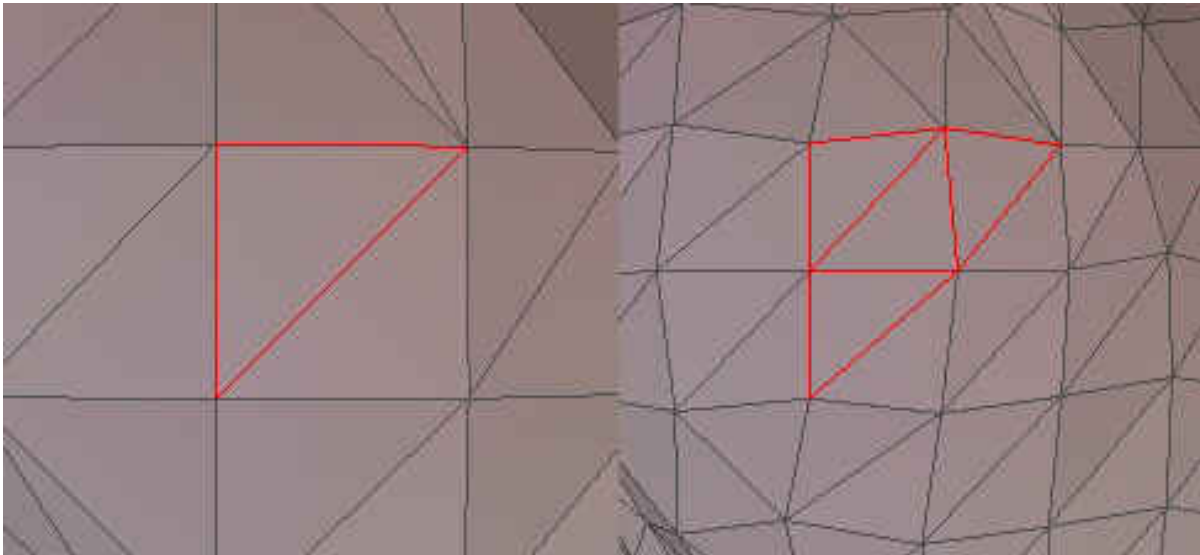
(a) A marked triangle in the original surface mesh

(b) The marked triangle is subdivided into four smaller triangles

Figure 3.3 A comparison of a triangle before and after subdivision.
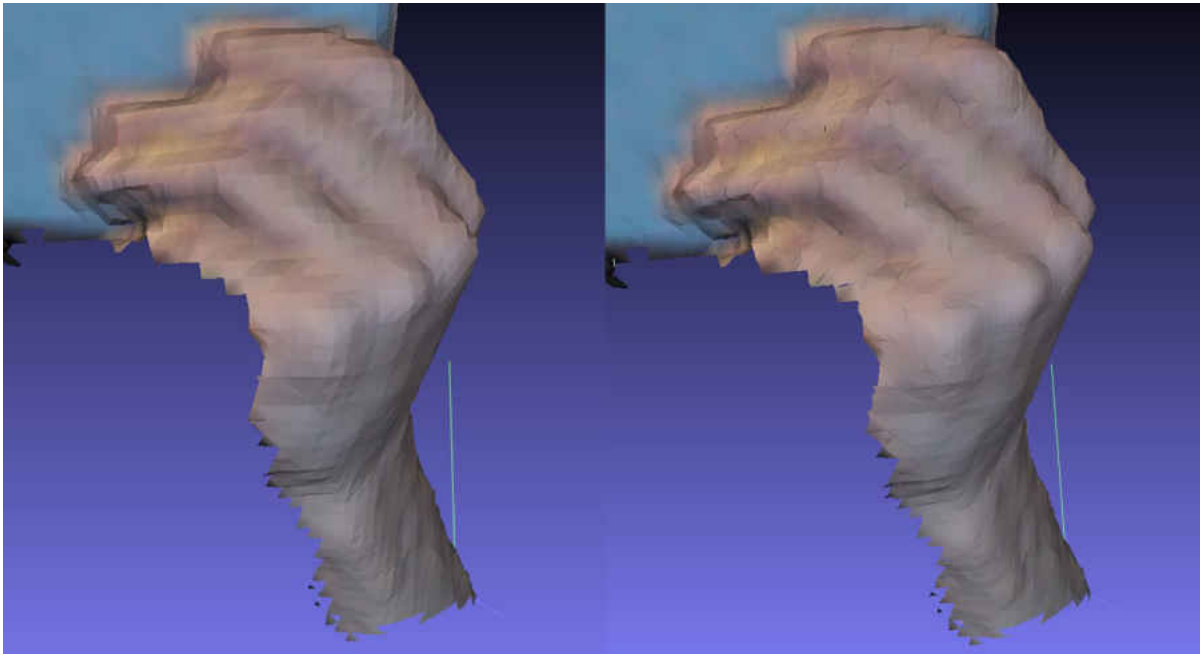
Figure 3.4 The left image shows the original surface mesh of a grabbing hand reconstructed from KinectFusion. The right image is the result of butterfly subdivision with tension parameter $\omega = 1/16$

The isotropic surface fitting method effectively enhances the smoothness of a mesh, but also improves the angle quality in the mesh benefits from the optimized vertex projection algorithm. As shown in Figure 3.5, given a vertex in the original mesh and a fitted plane generated from a adaptive number of rings of its neighbors, all vertices in the neighborhood is projected onto the plane and a 2D polygon is formed from these projection points. The center of the MIC of this 2D polygon is then found and projected back onto the quadric surface along the normal vector of the plane, and the projection point is the final new position of the vertex in the original mesh.

Figure 3.5 The optimized vertex projection algorithm[5]

3.3 Anisotropic Surface Fitting

As an alternative triangle meshing approach, we perform anisotropic surface fitting for smoothing the triangle mesh reconstructed by KinectFusion. Anisotropic surface fitting is a feature-preserving version of the isotropic surface 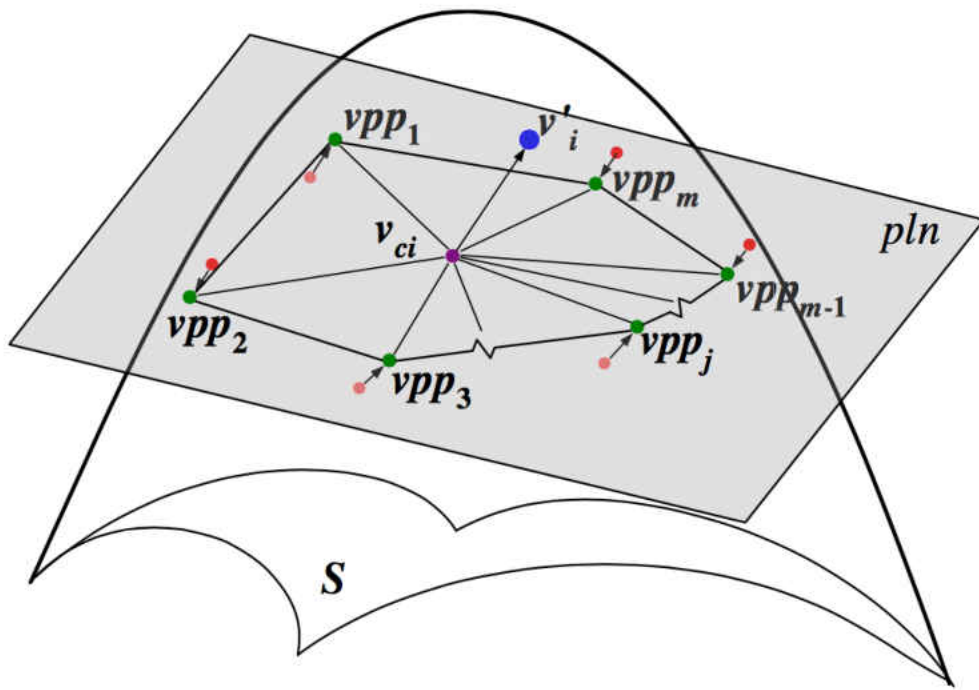fitting method we discussed in the previous section. An analytical quadric surface is fitted for each vertex using the least-square fitting method, and the center of the maximum inscribe circle at each vertex is calculated and then projected onto the fitted quadric surface to serve as the new position of the original vertex. However, instead of finding rings of neighbor vertices of the original vertex for fitting, the anisotropic method selects part of its neighbors by filtering the other part based on a normal-weighted distance function, which takes not only the distance between the vertex and its neighbor, but also the normal difference into consideration[6]. Figure 3.6 shows the workflow of anisotropic surface fitting algorithm.

3.3.1 Anisotropic Neighborhood Selection

In this approach, the key of feature preserving is the selection of anisotropic neighboring vertices using the normal-weighted distance function(NDF)(Figure 3.7). Each vertex is presented by a triangle that has the most similar normal direction to the normal vector of the vertex since the NDF is supposed to work on triangles. Then the NDF between the triangle indicating each neighboring vertex and the triangle indicating the center vertex is calculated, and neighboring vertices whose distance are less than a predefined threshold is selected to join the anisotropic neighborhood. After this proce-
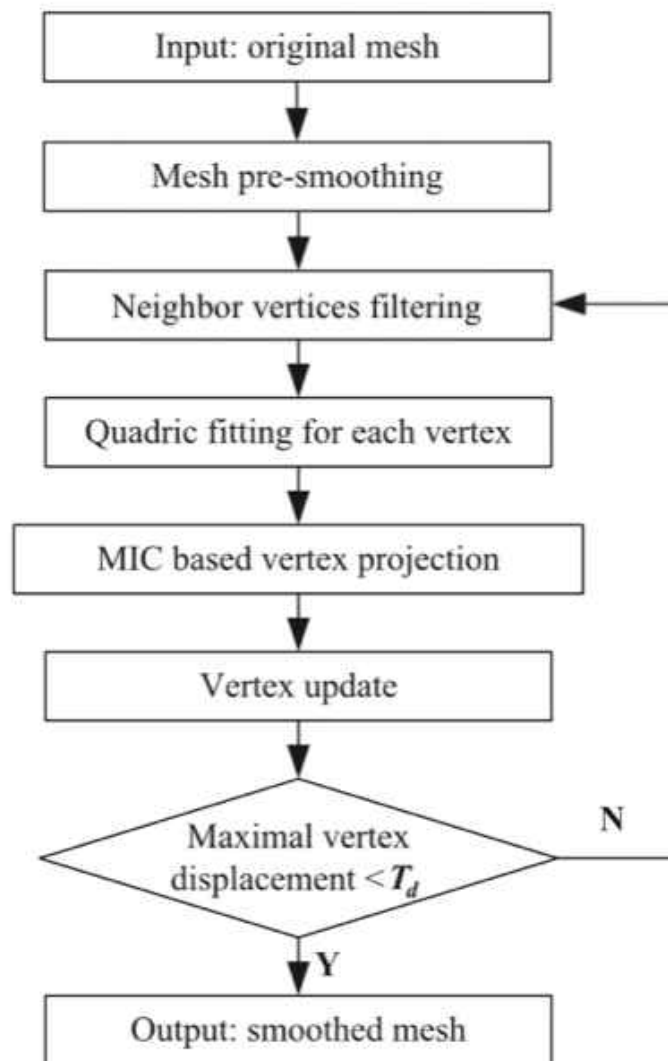
Figure 3.6 the workflow of anisotropic surface fitting algorithm[6]

dure, the quadric surface fitted from the anisotropic neighbors of each vertex only geometrically mimics the surface patch with low curvatures around the vertex. Therefore, instead of being smoothed, sharp features of the mesh are preserved as the boundaries between two surface patches.

The results of isotropic surface fitting and anisotropic surface fitting are compared in Figure 3.7. Generally the mesh is effectively smoothed using both methods. However they still have some flaws respectively. The isotropic method blurs the bound-

ary between the hand and the object. This is predictable since the isotropic method blurs sharp features(high curvature) such as boundaries, edges and corners. The anisotropic method effectively recovers the sharp boundary between objects, yet it fails to remove the gaps caused by vertices fall inside the surface. This is because the method does not consider these vertices for fitting the quadric surface since they are filtered out by the anisotropic neighbor selection algorithm.

To sum it up, the isotropic method is more suitable for smoothing meshes reconstructed by KinectFusion in most cases. More specifically, the isotropic method is more suitable for meshes with high level of noise and without sharp features. Since KinectFusion has a minimum detail detection size and a maximum number of voxels reconstructed per meter due to the hardware limitation, the sharp features of mesh models reconstructed are not very sharp. Therefore the biggest advantage of the anisotropic method over the isotropic method is not useful under these scenarios.
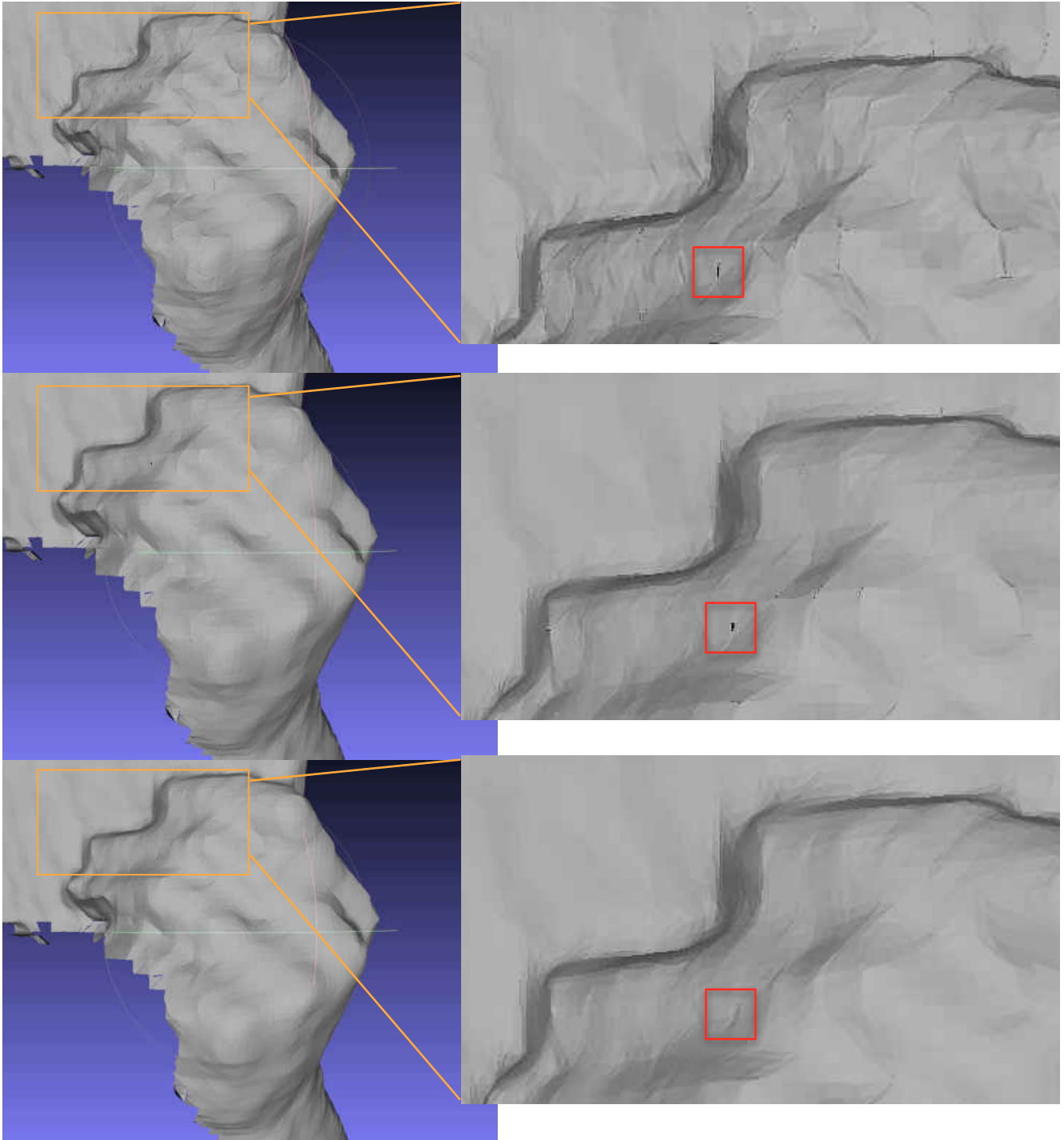
Figure 3.7 On the top is the original mesh after pre-processing, the middle image demonstrates the mesh after the anisotropic surface fitting, and the one on the bottom is the mesh after the isotropic surface fitting. A gap caused by vertices fall inside the surface is marked in red rectangles. It is fixed by the isotropic method as shown in the image on the bottom.

Chapter 4: Texture Enhancement

3D reconstruction using a Kinect sensor benefits researchers from different areas such as Human-Computer Interaction, Machine Vision and Indoor Environment Reconstruction. Most of these applications apply image processing techniques to the depth map for accuracy and error correction. However, they utilize raw color information for the reconstruction without taking advantage of image processing techniques. The texture qualities of 3D models reconstructed from the Kinect sensor are highly influenced by the lighting conditions. Under less-than-ideal lighting conditions, the contrast of a 3D model texture could be too high or too low, which makes the details of the model obscure to users. We address these issues with a modified histogram equalization algorithm that is extended to colored 3D cases from the histogram equalization for 2D grayscale image. In this chapter, we briefly review the traditional histogram equalization algorithm. Then we introduce the 3D histogram equalization.

4.1 Review of Histogram Equalization on 2D grayscale image

Histogram equalization is one of the most simple, yet powerful techniques in imaging enhancement. For the purpose of adjusting image intensities to enhance contrast, histogram equalization simply counts the frequency of grayscale levels of each pixel in the image, and assigns a new grayscale level to pixels with the same grayscale level based on the statistics[8].

Given a image with n pixels. The pixels in the image has a intensity range from 0 to L - 1. The image's histogram for pixel value i is defined as

$$p_x(i) = p(x = i) = \frac{n_i}{n}, \quad 0 \le i < L$$

where L is the number of possible intensity values, n_i is the number of occurrence of grayscale level i, and n is the total number of pixels in the image. $p_x(i)$ is in fact the normalized histogram for intensity value i. Then we accumulate these normalized histograms and define the cumulative distribution function as

$$cdf_x(i) = \sum_{j=0}^{i} p_x(j),$$

In the histogram equalized image, the new intensity of all pixels with intensity i in the original image will be

$$f(i) = \text{floor}((L - 1) \cdot cdf_x(i))$$

By assigning new intensity values, histogram equalization distributes intensities better on the histogram. To be more specific, it increases the contrast of areas with lower local contrast by spreading out the most frequently occurred intensity value(Figure 4.1). As shown in Figure 4.2, the histogram is stretched horizontally, and the area marked with red square in the histogram is filled after the equalization. This means that pixels with low intensity values is more, which corresponds to the whitened area such as the bridge of the nose in Figure 4.1.

Figure 4.1 The picture of Lena before(left) and after(right) histogram equalization, the nose bridge of Lena is whitened after equalization because the local contrast of this area is relatively low before equalization.
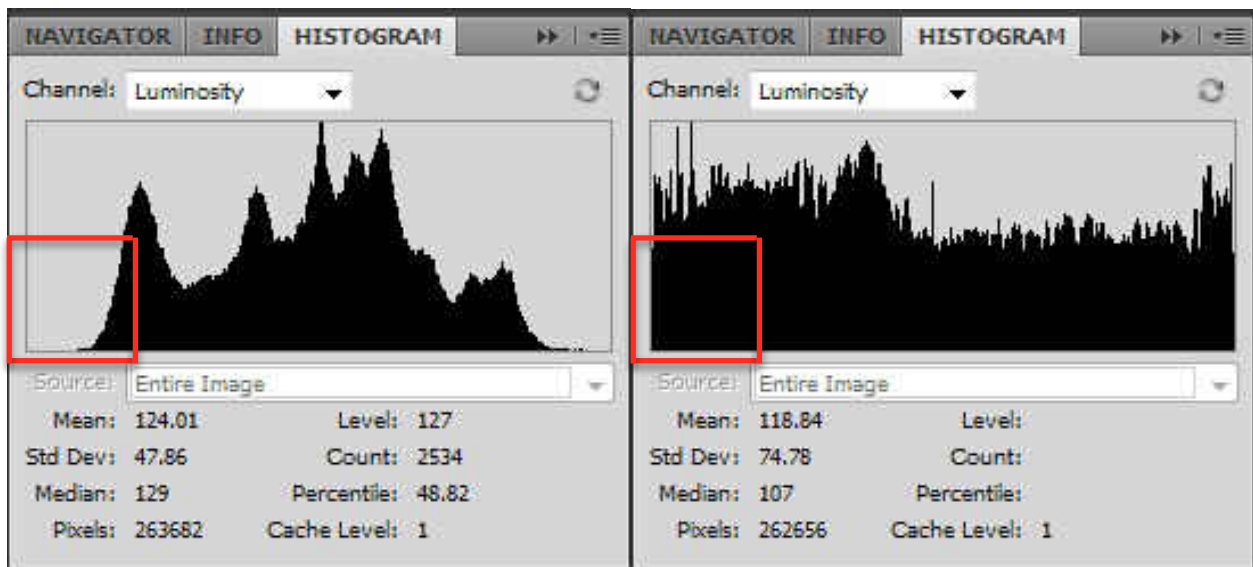

Figure 4.2 The histograms of the Lena picture before(left) and after(right) histogram equalization, the histogram is flattered after equalization.

4.2 Extending Histogram Equalization to 3D cases

We obtain contrast-enhanced textures of 3D models reconstructed from the Kinect sensor by a modified histogram equalization method, 3D histogram equalization, which inherited the main idea from the original histogram equalization method. The core of histogram equalization works perfectly in 3D cases because both the pixels in a image and the vertices in a surface mesh are discrete data sets. However, histogram equalization is only effective on luminance channels but the Kinect sensor captures color information in RGB color space. Applying histogram equalization separately on red,

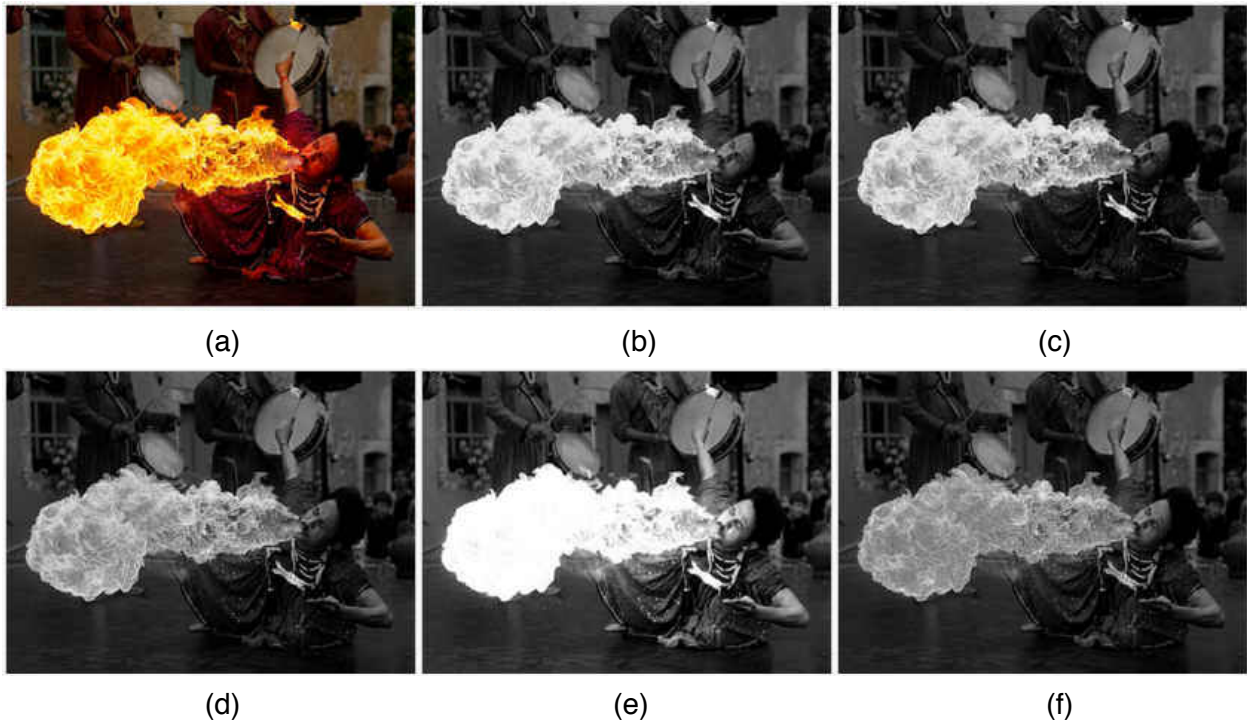(a)　　　　　　　　　　(b)　　　　　　　　　　(c)

(d)　　　　　　　　　　(e)　　　　　　　　　　(f)

Figure 4.3 (a) the original color image,
(b) the grayscale image from the L channel in CIELAB,
(c) the grayscale image from the Y channel in YCrCb,
(d) the grayscale image from the I channel in HSI,
(e) the grayscale image from the V channel in HSV,
(f) the grayscale image from the L channel in HSL.[26]

23

green and blue channel results in dramatic changes of the image's color balance since the histogram of these three channels are not uniform. To address this issue, textures of a 3D model are converted into other color spaces containing a separate luminance-like channel such as HSL, HSV, HSI, CIELAB, and YCrCb. Then 3D histogram equalization can be applied to the channel without changing the color balance. After extracting the luminance-like channel of color spaces mentioned above from a image of a fire breather(Figure 4.3), a comparison shows that the HSI color space with a intensity channel "I" is relatively easy and lossless for the conversions between itself and RGB color space.The reason that HSI is an effective intermediate for this process is that it is a common cylindrical-coordinate representations of points in an RGB color model and the distance along the cylinder axis corresponds to "intensity".

4.2.1 Conversion from RGB to HSL

In the conversion from RGB to HSI, the intensity is defined as

$$I = \tfrac{1}{3}(R + G + B)$$

The saturation is defined in line with the psychometric definition, which is chroma relative to lightness:

$$S = \begin{cases} 0 & \text{, if max(R, G, B) - min(R, G, B) = 0} \\ 1 - \dfrac{\min(R, G, B)}{I} & \text{, otherwise} \end{cases}$$

and the hue is defined in degrees:

$$H' = \begin{cases} \text{undefined,} & \text{if } C = 0 \\ \frac{G-B}{C} \bmod 6, & \text{if } M = R \\ \frac{B-R}{C} + 2, & \text{if } M = G \\ \frac{R-G}{C} + 4, & \text{if } M = B \end{cases} \qquad H = 60° \times H'$$

4.4.2 3D Histogram Equalization

After we convert the color information of all vertices from RGB to HSI, the 3D histogram equalization is applied to the texture of the 3D model. In 3D histogram equalization, the normalized histogram of the texture for intensity value i is:

$$p_x(i) = \frac{n_i}{n}, \quad 0 \le i < L$$

where n_i is the number of vertices with intensity i in the 3D model before histogram equalization, n is the total number of vertices, and L is the number of possible intensity values, in this case, 256. After 3D histogram equalization, the new intensity of vertices that has intensity i in the original texture is:

$$f(i) = \text{floor}\left((L-1) \cdot \sum_{j=0}^{i} p_x(j)\right)$$

As shown in Figure 4.4, the 3D histogram equalization effectively reveal the hidden details from the texture of a 3D model of a palm.
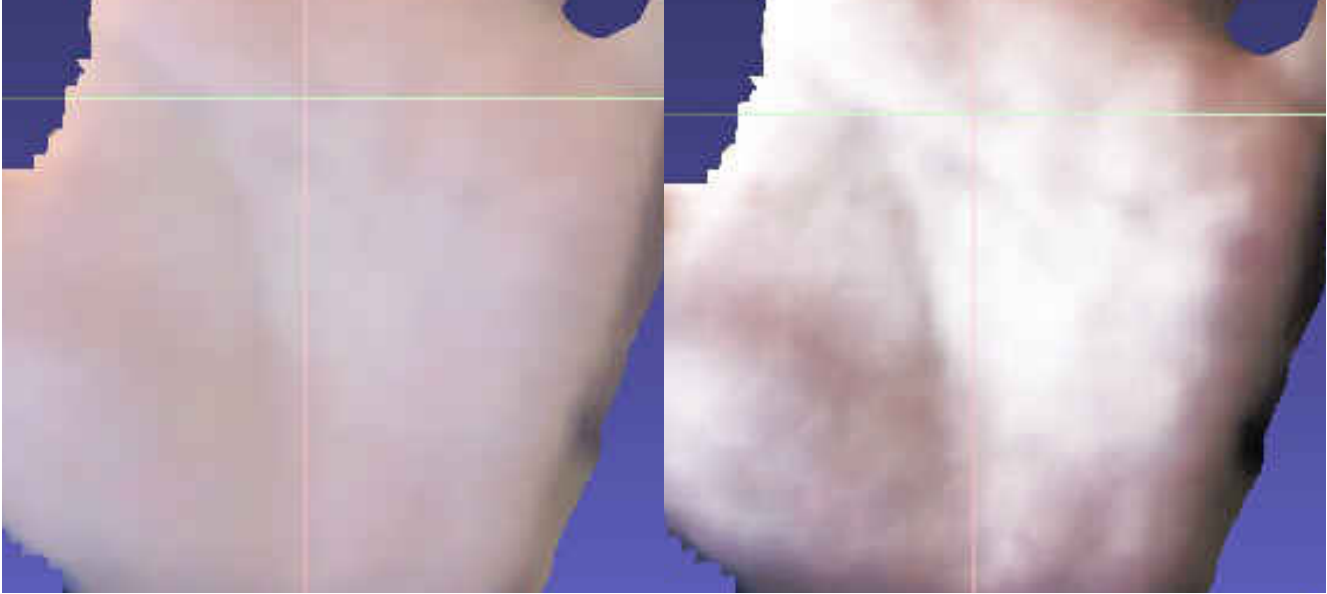
Figure 4.4 A comparison of a colored 3D model before and after the 3D histogram equalization

Chapter 5: Results and Analysis

To analyze the performance of our system, we test it on a set of 3D models reconstructed from single objects with different sizes and materials under different lighting condition. These objects are: a stuffed bear with fluffy surface around 1.5 feet long, a stuffed giraffe with cotton surface around 4 inches long, a human hand and a flat surface made of ABS plastic with the size of 6 x 9 inches approximately. The performance of texture enhancement is measured by histograms. The vertical axis of the histogram is the number of vertices with same luminance values, and the horizontal axis is the possible luminance values.

To exclude the possibility that our results are influenced by the settings of KinectFusion. We achieve maximum KinectFusion accuracy by choosing the implicit voxel volume to be as small as possible. We create dense voxels for detailed 3D reconstruction by selecting the resolution of KinectFusion to be 512^3 voxels per meter, which means that the lengths of voxel side is approximately 1.9 millimeter. (Memory requirements do scale with the third power of the volume resolution). The camera/object distance was approximately 1 meter in this case.

To demonstrate the results of geometry processing, the original mesh, the mesh smoothed by the isotropic approach, and the mesh smoothed by the anisotropic approach are demonstrated respectively with their normal maps. Figure 5.1 and Figure 5.2 show the results of applying our system to the fluffy teddybear and the stuffed giraffe. The results of the isotropic approach is clearly smoother than the results of the anisotropic approach because its removal of the bumpy noises is more effective. On the

27

other hand, the results of the feature-preserving approach, the anisotropic approach, show that it keeps the bumpy noises as sharp features. Thus, for objects with rich details on the surface(smaller than 1.9 millimeter), our system has better performance with the isotropic surface fitting algorithm. However, this assessment is reversed by the results display in Figure 5.3, which shows the results of our system applied to a human hand. The palm of the hand is flattened by the isotropic approach and details are filtered out as noises. In this case, the anisotropic approach successfully preserves the correct features and yields a better result. Following this line of thinking, Figure 5.4 also indicates that our system should utilize the anisotropic approach when the input meshes do not contain rich details. The orthogonal edge of the flat surface is recovered by the anisotropic approach but blurred by the isotropic one.

In Figure 5.5 and Figure 5.6, the results of 3D histogram equalization of these 3D models are presented with their histograms. As shown in the histograms, the intensity values or luminance values are effectively spread out by our system. The contrast of the textures are increased and texture details are more visible to users. We found that the 3D Histogram Equalization tends to yield overexposed results because it enhances the contrast globally.

All algorithms described have been implemented in Visual C++, running on a 3.4 GHz 4th-generation i7 desktop computer with a GeForce GTX 770.
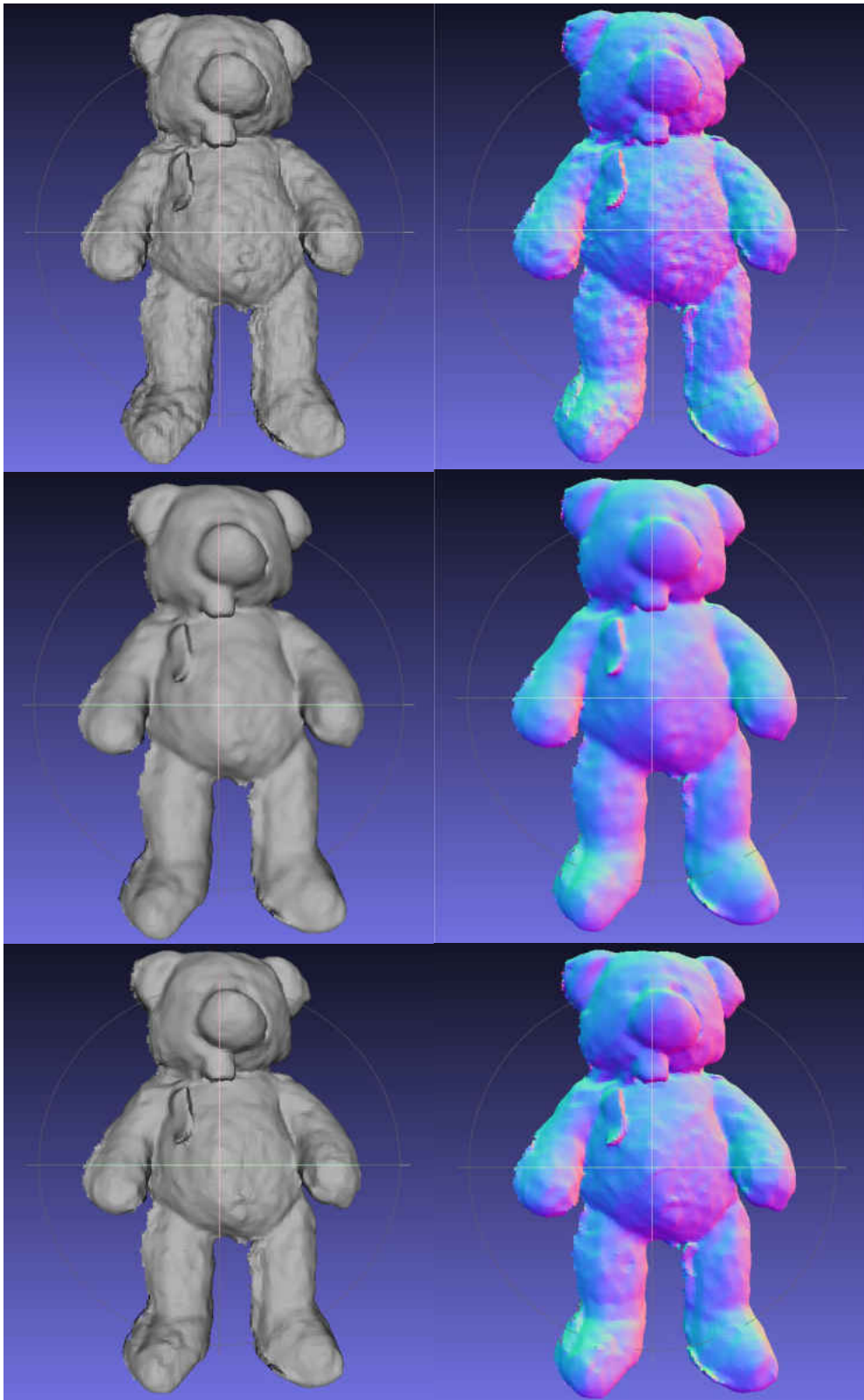
Figure 5.1 The original mesh(up), isotropic smoothed
mesh(middle) and anisotropic smoothed mesh(bottom)
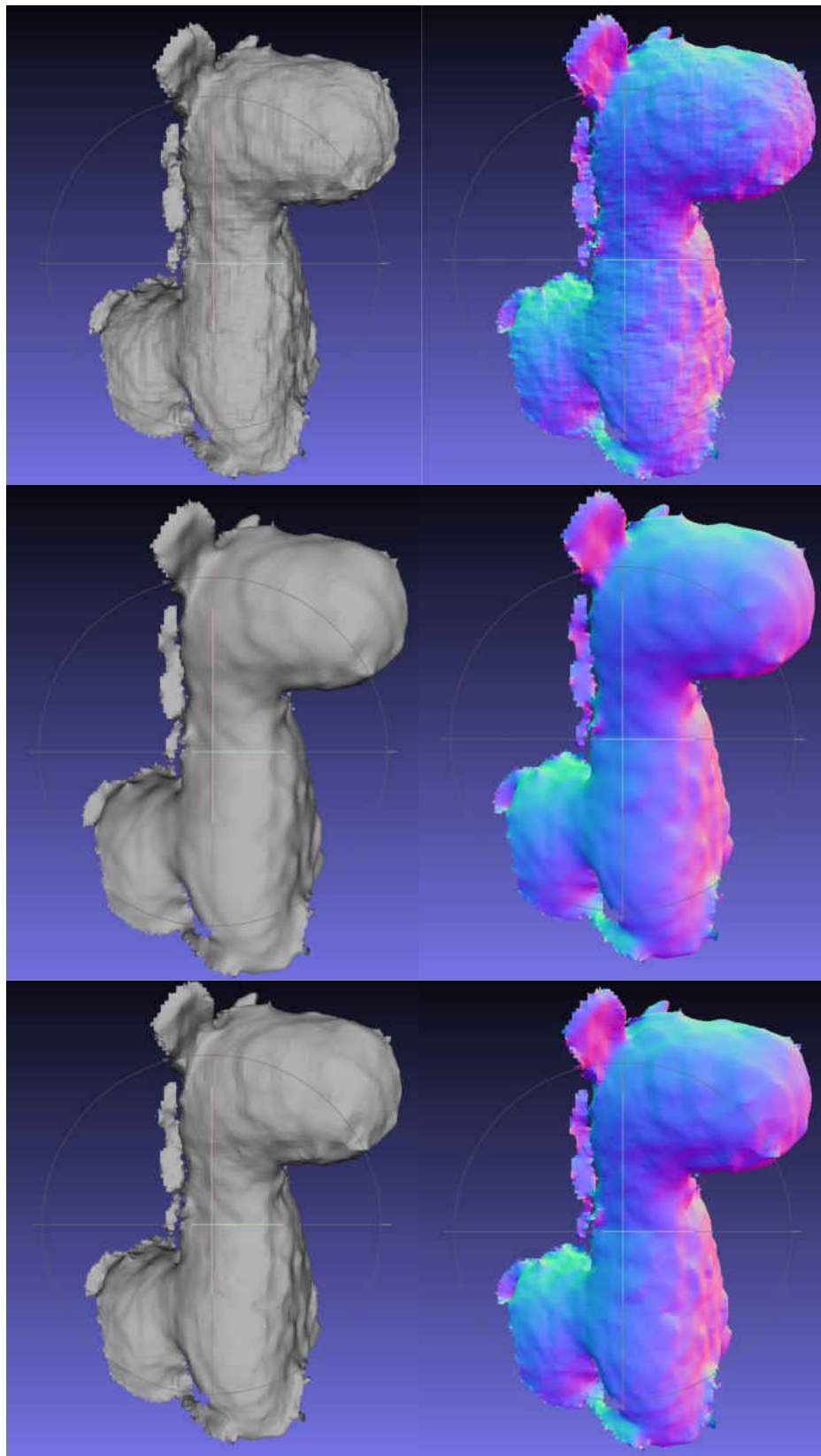and their normal maps of the fluffy bear

Figure 5.2 The original mesh(up), isotropic smoothed mesh(middle) and anisotropic smoothed mesh(bottom) and their normal maps of the cotton giraffe

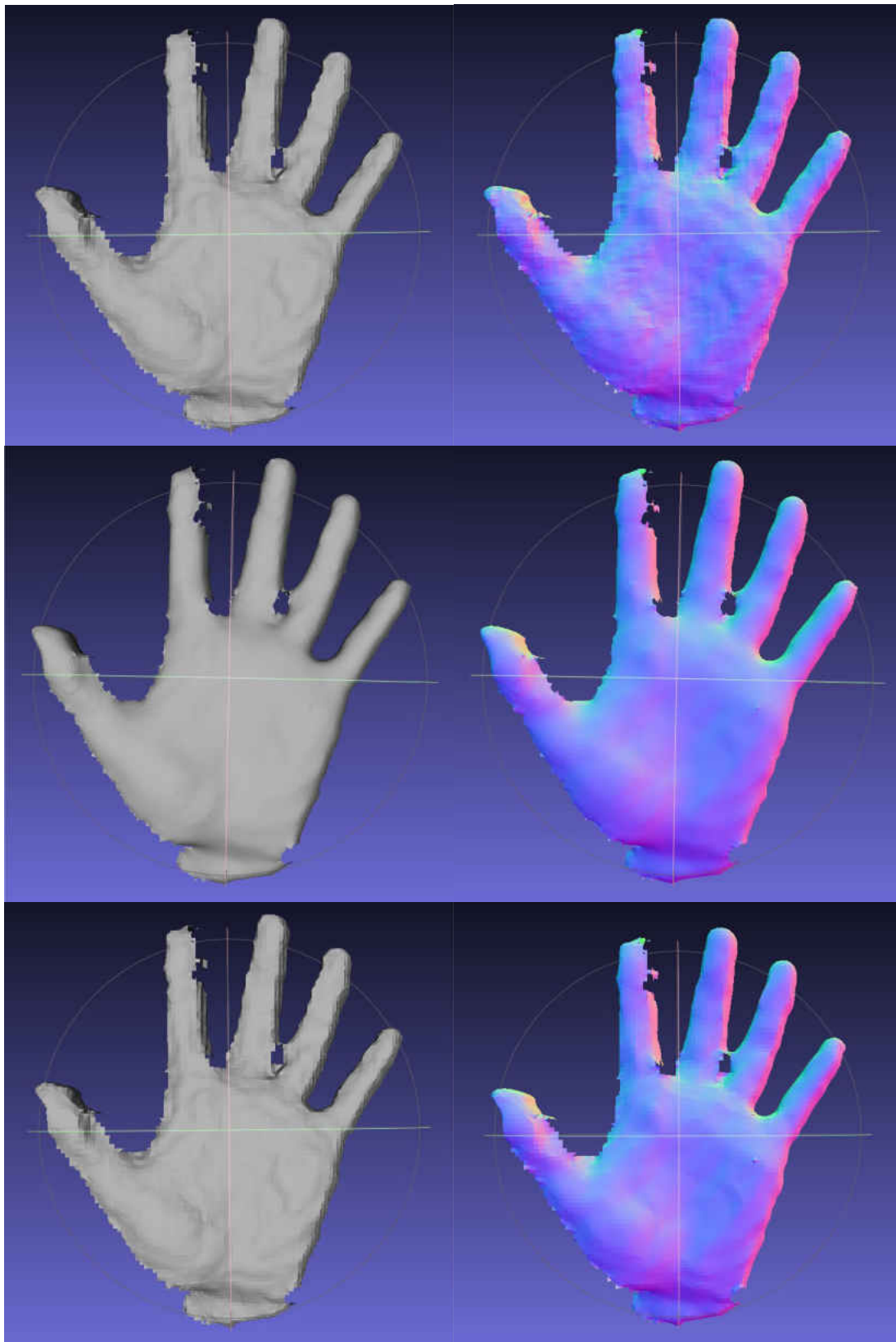**Figure 5.3** The original mesh(up), isotropic smoothed
mesh(middle) and anisotropic smoothed mesh(bottom) and their
normal maps of a human hand

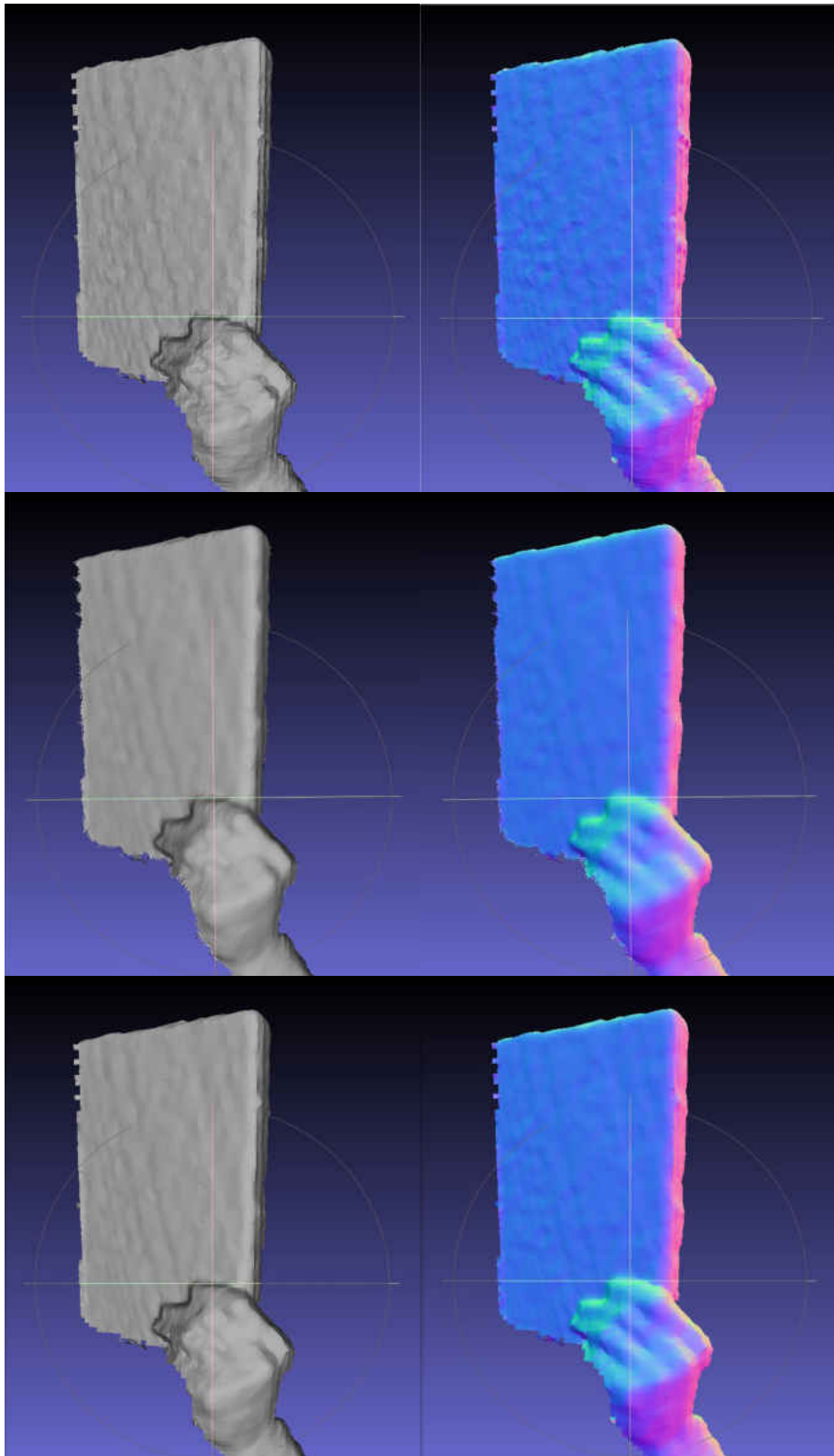**Figure 5.4** The original mesh(up), isotropic smoothed mesh(middle) and anisotropic smoothed mesh(bottom) and their normal maps of the flat surface

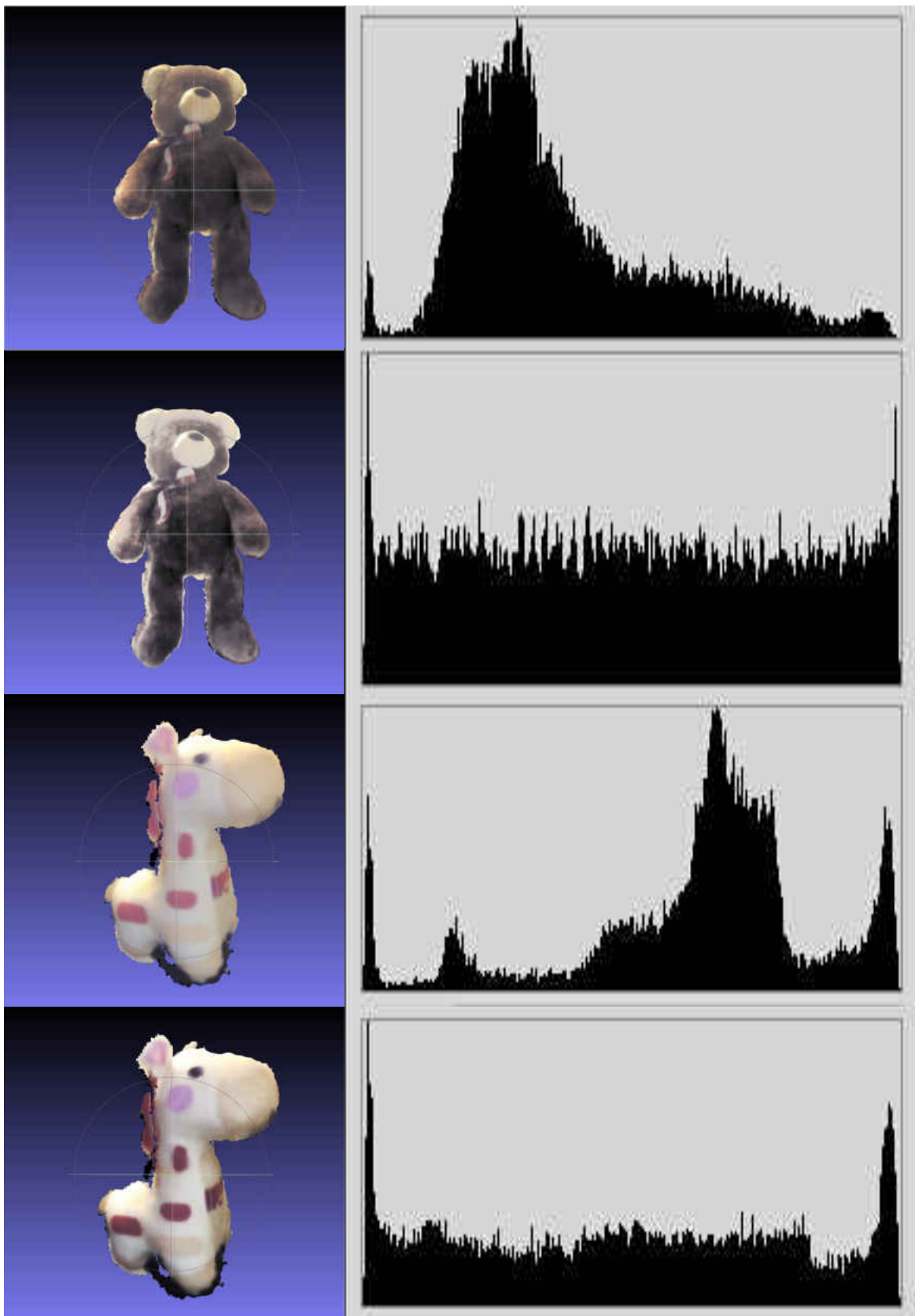**Figure 5.5** the original mesh(up), 3D histogram equalized mesh(bottom) of the fluffy bear and the cotton giraffe
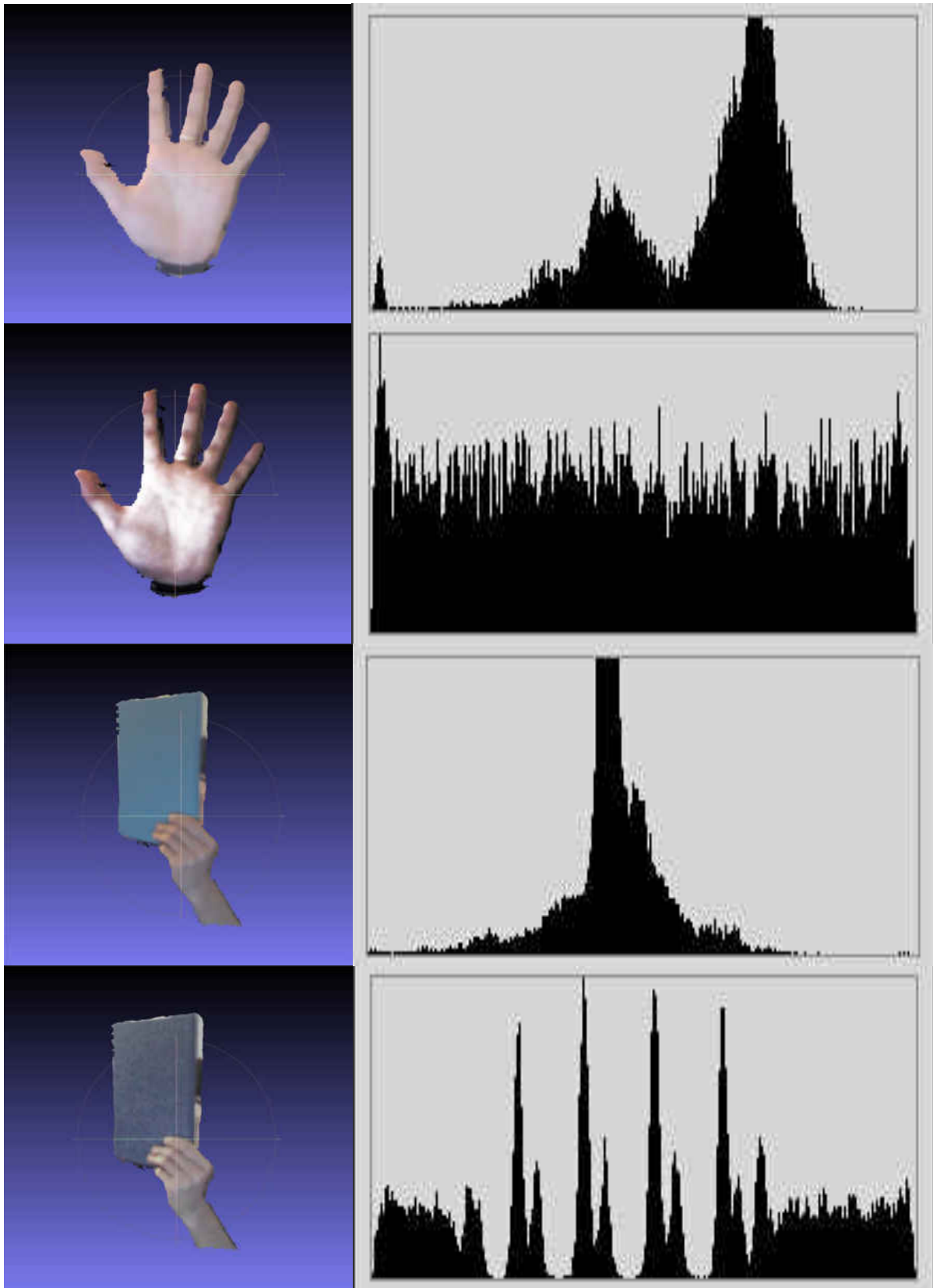
Figure 5.6 the original mesh(up), 3D histogram equalized mesh(bottom) of a human hand and the flat surface

Chapter 6: Conclusions

This thesis has described a combined method designed to enhance the 3D models reconstructed from RGB-D camera systems. The system uses a Microsoft Kinect device and KinectFusion for color and depth data acquisition. The system performs butterfly subdivision, isotropic mesh smoothing and anisotropic mesh smoothing on the geometry of 3D models, and applies a modified 3D histogram equalization to the texture of the 3D models. Our system is tested on a variety of 3D models and it is shown that it effectively generates 3D models with improved smoothness and contrast-adjusted textures, which reveals hidden details from the original texture.

6.1 Limitations

While taking advantages of the existing techniques, our system inherits limitations from these methods. Our system extends the classic histogram equalization to 3D cases. Therefore, the texture enhancement component of our system is more useful under strong lighting conditions. Furthermore, our system suffers from the artifacts caused by the butterfly subdivision, which lowers the performance of anisotropic mesh smoothing. Thirdly, the hardware requirement of our system is relatively high comparing to other methods we mentioned in the first chapter. The KinectFusion system has a hardware requirement of Desktop PC with 3GHz (or better) multi-core processor and a graphics card with 2GB or more of dedicated on-board memory.

6.2 Future Works

There are many possibilities for future improvement with our system to address the issues we discussed in the previous section. Our 3D histogram equalization algorithm adjust the global contrast of the texture ignoring the different lighting conditions on the different parts of the scene. A future research could be adjusting the contrast for local patches of the texture using an adaptive 3D histogram equalization method. Another interesting idea for further research is denoising the texture of Kinect-based 3D models following the thought of machine learning. A set of 3D models of everyday objects could be used by the system to learn about the pattern of how they interact with each other, then the system could make predictions and de-noise the texture on the boundary areas of the scene. Additionally, a possible future work is novel subdivision algorithms that is designed to condense and smooth the 3D models constructed from RGB-D camera systems.

References

[1] Newcombe, Richard A., Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. "KinectFusion: Real-time dense surface mapping and tracking." In Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on, pp. 127-136. IEEE, 2011.

[2] Dyn, Nira, David Levine, and John A. Gregory. "A butterfly subdivision scheme for surface interpolation with tension control." ACM transactions on Graphics (TOG) 9, no. 2 (1990): 160-169.

[3] Meister, Stephan, Shahram Izadi, Pushmeet Kohli, Martin Hämmerle, Carsten Rother, and Daniel Kondermann. "When can we use KinectFusion for ground truth acquisition?." In Proc. Workshop on Color-Depth Camera Fusion in Robotics, vol. 2. 2012.

[4] "Subdivision Surface." Wikipedia. September 20, 2015. https://en.wikipedia.org/wiki/Subdivision_surface(accessed November 24, 2015)

[5] Wang, Jun, and Zeyun Yu. "Quality mesh smoothing via local surface fitting and optimum projection." Graphical models 73, no. 4 (2011): 127-139.

[6] Wang, Jun, and Zeyun Yu. "Feature-preserving mesh denoising via anisotropic surface fitting." Journal of computer science and technology 27, no. 1 (2012): 163-173.

[7] "Kinect for Windows Sensor Components and Specifications." Mircosoft. https://msdn.microsoft.com/en-us/library/jj131033.aspx(accessed November 24, 2015)

[8] Rafael C. Gonzalez. Digital Image Processing (3rd Edition). Prentice Hall, 2007.

[9] Xia, Lu, Chia-Chih Chen, and Jake K. Aggarwal. "Human detection using depth information by Kinect." In Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on, pp. 15-22. IEEE, 2011.

[10] Dutta, Tilak. "Evaluation of the Kinect™ sensor for 3-D kinematic measurement in the workplace." Applied ergonomics 43, no. 4 (2012): 645-649.

[11] Han, Jungong, Ling Shao, Dong Xu, and Jamie Shotton. "Enhanced computer vision with microsoft kinect sensor: A review." Cybernetics, IEEE Transactions on 43, no. 5 (2013): 1318-1334.

[12] Izadi, Shahram, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton et al. "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera." In Proceedings of the 24th annual ACM symposium on User interface software and technology, pp. 559-568. ACM, 2011.

[13] Zou, Yuhua, Weihai Chen, Xingming Wu, and Zhong Liu. "Indoor localization and 3D scene reconstruction for mobile robots using the Microsoft Kinect sensor." In Industrial Informatics (INDIN), 2012 10th IEEE International Conference on, pp. 1182-1187. IEEE, 2012.

[14] Stowers, John, Michael Hayes, and Andrew Bainbridge-Smith. "Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor." In Mechatronics (ICM), 2011 IEEE International Conference on, pp. 358-362. IEEE, 2011.

[15] Shum, Hubert PH, Edmond SL Ho, Yizhang Jiang, and Shinichi Takagi. "Real-time posture reconstruction for microsoft kinect." Cybernetics, IEEE Transactions on 43, no. 5 (2013): 1357-1369.

[16] Zollhöfer, Michael, Michael Martinek, Günther Greiner, Marc Stamminger, and Jochen Süßmuth. "Automatic reconstruction of personalized avatars from 3D face scans." Computer Animation and Virtual Worlds 22, no. 2-3 (2011): 195-202.

[17] Frati, Valentino, and Domenico Prattichizzo. "Using Kinect for hand tracking and rendering in wearable haptics." In World Haptics Conference (WHC), 2011 IEEE, pp. 317-321. IEEE, 2011.

[18] Ruchanurucks, Miti, Amornrat Khongma, Panjawee Rakprayoon, Teera Phatra-pornnant, and Taweetong Koanantakool. "Kinect quality enhancement for triangular mesh reconstruction with applications in burn care." Transactions of the Institute of Measurement and Control (2013): 0142331213490596.

[19] Wu, Lei, and Senchun Chai. "Depth Filtering in 3D Reconstruction of Indoor Scenes Based on Kinect." In Computational Intelligence and Design (ISCID), 2014 Seventh International Symposium on, vol. 1, pp. 356-359. IEEE, 2014.

[20] Saygili, Gorkem, C. Balim, H. Kalkan, and Emile Hendriks. "Estimating the missing Kinect depth information by polynomial fitting." In Signal Processing and Communications Applications Conference (SIU), 2013 21st, pp. 1-4. IEEE, 2013.

[21] Matyunin, Sergey, Dmitriy Vatolin, Yury Berdnikov, and Maxim Smirnov. "Temporal filtering for depth maps generated by Kinect depth camera." In 3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2011, pp. 1-4. IEEE, 2011.

[22] Zhang, Cha, and Zhengyou Zhang. "Calibration between depth and color sensors for commodity depth cameras." In Computer Vision and Machine Learning with RGB-D Sensors, pp. 47-64. Springer International Publishing, 2014.

[23] Herrera, C., Juho Kannala, and Janne Heikkilä. "Joint depth and color camera calibration with distortion correction." Pattern Analysis and Machine Intelligence, IEEE Transactions on 34, no. 10 (2012): 2058-2064.

[24] Li, Zhongrui, Chao Sun, Won-Sook Lee, and Ig-Jae Kim. "Hair modeling using kinect sensors and DSLR cameras." In Ubiquitous Robots and Ambient Intelligence (URAI), 2014 11th International Conference on, pp. 22-27. IEEE, 2014.

[25] "DEPTHKIT: RGB+D FILMMAKING TOOL." DepthKit. http://www.rgbdtoolkit.com/ (accessed November 24, 2015)

[26] "HSL and HSV." Wikipedia. The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=HSL_and_HSV&oldid=687890438 (accessed November 24, 2015).

[27] Joseph L. Flatley. "Visualized: Kinect + night vision = lots and lots and lots of dots." Engadget. http://www.engadget.com/2010/11/08/visualized-kinect-night-vision-lots-and-lots-and-lots-of-do/(accessed November 24, 2015)