Building Customized Cushion Seats Using Stereo Visions for Disabled

Weifeng Xu, Ben Luebbert, Stephen Frezza, Sreela Sasi
Department of Computer and Information Science
Gannon University
{xu001, luebbert001, frezza001, sasi}@gannon.edu

Todd Dinner
Precision Rehab Manufacturing Inc.
Northeast, PA
tdinner@prmrehab.com

Abstract — **This paper presents a systematic approach to construct a grid-based 3D surface model for a customized cushion seat for people with disabilities. Initially, a 2D grid map is defined as the bottom surface of the cushion seat based on the required resolution. Then each point in the 2D grid map is projected onto one surface point in the 3D pixel-based point cloud. The height of each surface point is approximated using the nearest neighbor algorithm. A tool box is developed for the visualization the constructed 3D surface model. The height of the 3D surface computed using this tool box is compared with the manually designed cushion seat at Precision Rehab Manufacturing Inc. and found to be within tolerance.**

Keywords - stereo camera, grid-based 3D surface model, pixel-based point cloud, customized seat

I. INTRODUCTION

Surface model reconstruction from stereo vision is appealing to many industries, including the rehabilitation treatment equipment industry. The purpose of the paper is to provide a systematic approach for reconstructing editable 3D surface models from stereo cameras, and employ the approach to develop a system for building custom-contour seats.

Individual customization is a growing movement in the manufacture of rehabilitation equipment. Customization and comfort are both vital aspects to the function of mobility devices [4][5]. For example, an uncomfortable seat causes sores and discomfort for some users who require seating for a period of time [7]. In addition, an attuned seat may be required in the case of certain injuries or afflictions that dramatically change a body's shape. To meet such needs, we need to offer the option of customized seats, as they maintain low contact pressures for people they support, and thus reduce pressure in body soft tissue [9].

Creating custom seats is a skill- and time-intensive process. To automate the manufacture process, we need to get a mould of a person's body, and then generate a 3D surface model based upon the mould. The 3D surface model should be editable for further adjustment. Finally, the modified 3D surface model can be used for milling by a computer numerical controlled (CNC) mill. In our approach, we use a stereo vision camera to capture the object surface information as its binocular vision allows the use of depth perception. Using stereo vision cameras removes much of the human effort from the process.

Model reconstruction from the stereo vision camera is the subject of progressing study and research. Stereo photo pair systems are used to model entire scenes. The challenges for modeling entire scenes include how to address the correctness of linear transformations, how to accurately determine depth of modeled objects, and how to construct entire scenes based on these data [8]. In a more specialized domain, such as a minimally invasive medical procedure [3], in order to selectively register human laryngeal cartilage, stereo cameras were used to compute the iterative closest point. Stephen Se applied a stereo camera system as an alternative to laser rangefinders for visualization in unmanned robots. In his approach, 3D data was constructed into triangular meshes to reduce computation and improve time [6].

Our effort focuses on reconstructing a grid-based 3D surface model from a pixel-based point cloud obtained from a stereo vision camera. The pixel-based point cloud is a collection of pixel points on a display screen. These points are projected from the object surface. The surface model can be visualized, edited, and adjusted to any resolutions for the CNC milling.

The rest of the paper is organized as follows: section II provides the architecture of the system, section III describes the pixel-based point cloud, the grid-based 3D surface model, and methodology of reconstructing a grid-based 3D surface model from the point cloud, section IV demonstrates an editing tool for editing the reconstructed surface model, and section V illustrates some of the experiment results.

II. ARCHITECTURE OVERVIEW

The system (see Figure 1) consists of four components: image capturing, model reconstruction, model editing, and model milling.

In our example, we make full use of a stereo vision camera to generate a pixel-based point cloud. The point cloud is composed of a set of points with a 3D, i.e., *(x, y, z)*, real world coordinate. More specifically, the image capturing component utilizes the Bumblebee stereo vision camera [1] to capture the mould of seat surface information and generate the pixel-based point cloud. The stereo vision camera comes with two Software Develop Kit (SDK) libraries, Digiclops and Triclops. The Digiclops library uses IEEE 1394 bus as a communication interface to exchange information between computers and cameras. The Digiclops is able to configure the photographic factors, such as white balance and exposure time [2]. The Triclops library provides a mechanism for the measurement of distance from the camera lens to any pixel

in its view. The Bumblebee camera captures images with a resolution of 640 x 490.

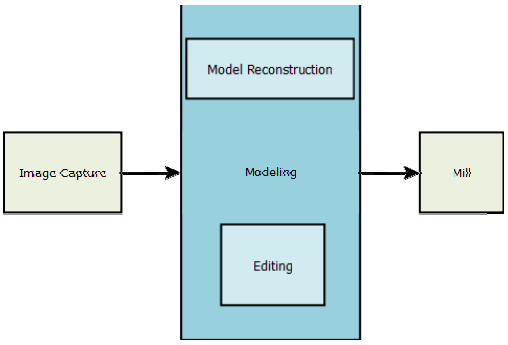

Figure 1. System architecture

The model reconstruction component reconstructs a grid-based 3D surface model of hip mould from the pixel-based point cloud. The challenge for building such a component is that, for each point in the grid-based 3D surface, we need to pick appropriate points in the pixel-based point cloud as reference points for obtaining the height of the selected points. We recursively use the nearest neighbors search method to collect those reference points in the point cloud. As the generated 3D surface model will be used for manufacturing a custom seat, we refer to 3D surface model as a seat surface model.

Model editing is another important component of the proposal system. It allows users to edit the reconstructed seat surface model. When the editing process is complete, the component saves the seat model information for the mill, i.e., cuts the edited shape out of foam. Note that the CNC mill system is an independent system, and is not in the scope of our discussion.

III. RECONSTRUCTING GRID-BASED 3D SURFACE MODELS

In this section, we first explain the pixel-based point cloud generated by a Bumblebee stereo camera, and then we define the grid-based 3D surface model. Finally, we demonstrate how to reconstruct a grid-based 3D surface model from a pixel-based point cloud.

A. Pixel-Based Point Clouds

The surface of an object is often expressed as a pixel-based point cloud for visualization on screens. There are three necessary steps to generate the pixel-based point cloud:

1) Capturing an object's surface information using a stereo vision camera. The information includes the captured images in left and right camera, focal length, and distance of left and right lenses.
2) Processing the surface information and organizing the information with appropriate data structures.
3) Displaying the organized information.

In Figure 2, the left image shows a hip mould and the right image shows the corresponding pixel-based point cloud. We use the hip mould as it is relatively convenient to be obtained compared to directly scanning a person's body. The surface information of the mould is captured by a Bumblebee stereo camera, and then the surface information is processed by the Bumblebee SDK library for generating the pixel-based point cloud. Finally, the generated pixel-based point cloud can be displayed using a small utility tool included in the Bumblebee camera.

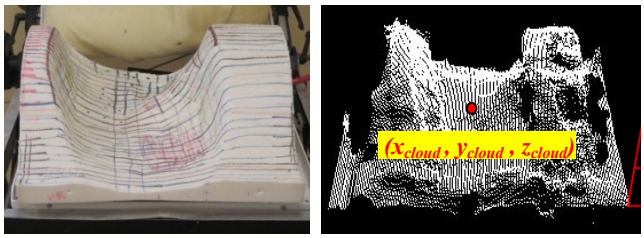

Figure 2. Hip mould (left) and the corresponding pixel-based point cloud (right)

The pixel-based point cloud is organized by a set of pixel points. Each point in the pixel-based point cloud is defined as a 3D coordinate $p(x_{cloud}, y_{cloud}, z_{cloud})$, where

x_{cloud}: the X Cartesian coordinate in the 3D world
y_{cloud}: the Y Cartesian coordinate in the 3D world
z_{cloud}: the height of pixel (x_{cloud}, y_{cloud}) in the object surface

With the object surface information captured by stereo cameras, we are able to calculate each point p by applying the following formulas:

$$x_{cloud} = \frac{d(x_l + x_r)}{2(x_l - x_r)}, \quad (1)$$

$$y_{cloud} = \frac{d(y_l + y_r)}{2(y_l - y_r)}, \text{ and} \quad (2)$$

$$z_{cloud} = \frac{df}{x_l - x_r}, \text{ where} \quad (3)$$

d: the distance of two lenses in a stereo camera
f: the focal length of both cameras
(x_l, y_l): the image pixel coordinates in the left camera
(x_r, y_r): the image pixel coordinates in the right camera

As a pixel-based point cloud mentioned in the paper contains the 3D real world positions of all the points in an object surface, we refer the pixel-based 3D point cloud simply as the *point cloud* in the rest of the paper.

Note that in the right image of Figure 2, the point cloud contains black areas. These black areas indicate invalid z values due to 1) the brightness, smoothness, and color of the object surface; 2) the lighting of environment; and 3) the viewing angle of the stereo camera as some surface areas cannot be viewed by both lenses of the camera at certain angles.

B. Grid-Based 3D Surface Models

A grid-based 3D surface model is based on a two-dimensional grid, i.e., a 2D grid. The 2D grid is defined in terms of a required resolution and represented by a $r*c$ matrix, where r is the number of rows and c is the number of

columns in the 2D grid (see Figure 3). Thus, the 2D grid consisted of *(r-1)*(c-1)* squares. Each point (i.e., the corner of the square) in the 2D grid is evenly distributed and denoted as *p(m, n),* where *1<=m<=r*, *1<=n<=c*. We use *p(m', n', h)* to indicate the reference point picked up from the point cloud for constructing the grid-based 3D surface model. The values of *m', n',* and *h* are calculated based on the formulas *1, 2* and *3*.

How to determine which reference point *p(m', n')* needs to be selected? The 2D reference point *p(m', n')* is corresponding to a grid position *p(m, n)* in the 2D grid. Ideally, the point set that constructs the grid-based 3D surface model should be a subset of the point cloud.

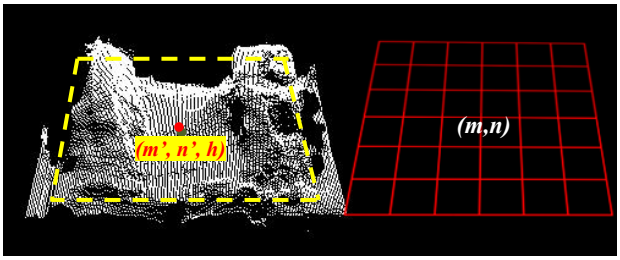

Figure 3. Point cloud (left image) and 2D grid for grid-based 3D surface model (right image)

Let's formally define the one-to-one relation in 2D between the point cloud and the grid-based 3D surface model as follows:

(One-to-one relation in 2D) *Given two sets A and B, a relation R⊆A×B, A is the subset of a point cloud and B is the point set of a grid-based 3D surface, the cardinality of relation R is one-to-one, that is <a, b> ϵ R, where a=(m', n') and b=(m, n).*

C. Constructing Grid-Based 3D Surface Models

Constructing a grid-based 3D surface model requires a generated point cloud and a desired resolution for displaying and manipulating a grid-based 3D surface model.

The challenges pertaining to the process of reconstructing grid-based 3D surface models include:

1. How to select *A*, the subset of a point cloud, so that *A* and *B* is a one-to-one relation? i.e., each position point *(m, n)* in the 2D grid can be mapped to a real world position *(m', n')* in the point cloud?
2. How to deal with the missing information, i.e., black areas, in a point cloud? In other words, the information at the position *(m', n')* may not exist due to the black areas in the point cloud. The one-to-one relation in 2D stands under the two assumptions: (a) all the areas of the object surface are needed to be visible, and (b) the object should be placed in an ideal environment. Failing to meet such needs results in black areas. In such a case, the *h*, height of the point *(m, n)*, won't be available.

Our approach to reconstruct a grid-based 3D surface model consists of five steps. We will address these challenging issues during the discussion of these steps. These steps include:

1. Designing a 2D grid based on a desired resolution.
2. Computing the related position for each point in the 2D grid.
3. Covering the desired area in the point cloud with the 2D grid.
4. Computing the real world coordinate for each point in the 2D grid. Those points are the reference points in the point cloud.
5. Estimating the height of each point in the 2D Grid.

Designing a 2D grid is the first step for constructing the grid-based 3D surface model. The purpose of the grid is to provide desired resolution for displaying, editing, and milling surface models. The points in the grid will be used for calculating the reference points in the point cloud. The size of the 2D grid is critical for constructing the grid-based 3D surface model. The grid needs to cover the desired area of the point cloud, i.e., region of interest or ROI for short (see dashed area in Figure 3). Any information out of the ROI boundary will be cropped out.

The size of the 2D grid and the size of the ROI should match the size of the seat surface, i.e., 20"×24". We pick up 81 rows and 97 columns as our desired resolution. In other words, the distance between each point in the 2D grid is 0.25 inches. Each point in the 2D grid is defined as *p= {(m, n): 1<=m<=81 and 1<=n<=97}*. The center of the 2D grid is *p(41, 49)*. We use *p(gridCenterRow, gridCenterCol)* to indicate the center of the 2D grid.

Given a real world distance in inches, *distance*, between each point in the 2D grid, we are able to calculate a related position for each point in the 2D grid in terms of the center of the 2D grid. Let's assume the center of the 2D grid *p(gridCenterRow, gridCenterCol)* corresponds to a position *p(0,0)* in inches, then each point in the 2D grid is corresponding to a related position *p(x, y)* in inches, where

$$x_{grid} = (m - gridCenterRow) * distance \quad (4)$$

$$y_{grid} = (n - gridCenterCol) * distance \quad (5)$$

After obtaining the grid, we need to cover the ROI with the 2D grid. The center of the 2D grid overlaps the center of the ROI (see Figure 4).

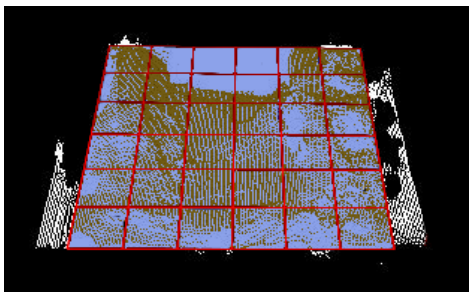

Figure 4. Cover ROI with 2D grid

We use *p(ROI_x, ROI_y)* to indicate the center of ROI. The center of the ROI is in a real-world coordinate. It is calculated using the Bumblebee SDK library. Thus, we can adjust the formulas (4) and (5) to (6) and (7), respectively, for calculating the real world coordinate for each point in the 2D grid in terms of the center of ROI. Those points in the 2D grid are identical to the reference points, *p(m', n')*, in the point cloud we have mentioned earlier.

$$m' = (m - gridCenterCol) * distance + ROI\_x \quad (6)$$

$$n' = (n - gridCenterRow) * distance + ROI\_y \quad (7)$$

The final step is to calculate the height of each reference point in the point cloud. If each reference point position *p(m', n')* matches a corresponding point in the pixel-based point cloud, $p(x_{cloud}, y_{cloud}, z_{cloud})$, we can simply find the height of each reference point. Formally, the height *h* of a reference point *p(m', n')* can be obtained as follows:

(Matching height) *Let A be the point cloud and B be a set of reference points calculated based on a 2D grid,* $p(x_{cloud}, y_{cloud}, z_{cloud}) \in A$ *and* $p(m', n', h) \in B$. *Then for each p(m', n'), we can match* $h = z_{cloud}$ *if* $m' = x$ *and* $n' = y$.

However, the matching method is an infeasible approach as the black areas exist in the point cloud in Figure 2. In other words, some of the points $p(x_{cloud}, y_{cloud}, z_{cloud})$ may not exist.

We use the estimated information to cover this missing information. More specifically, we estimate the height of each point based on the nearest neighbors in the point cloud. As shown in Figure 5, the nearest neighbors of a point include all the points within a defined distance. The distance is adjusted related to the size of black areas. We refer to distance as a smoothing distance since the parameter determines how smooth the reconstructed seat surface is.

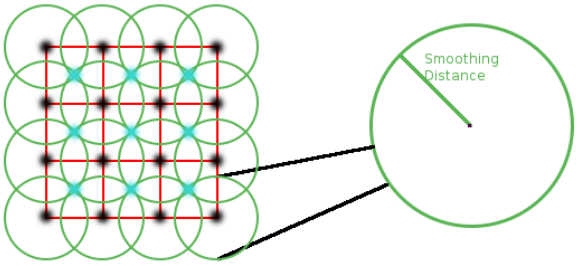

Figure 5. Every captured point (in blue), is within the radius of every grid point (in black)

Note that we use the threshold value, i.e., the number of the nearest neighbors, as a parameter for smoothing the object surface. We count the number of the nearest neighbors dynamically. If the number of the nearest neighbors is less than our expectation, we need to gradually increase the search radius to include more neighbors. To measure if a point $p_1(x_1, y_1)$ in a point cloud is one of the nearest neighbors of a reference point $p_2(x_2, y_2)$, we need to calculate if the defined radius, *r*, is greater than the distance *d* between the p_1 and p_2, e.g., we need to determine if $r - d >= 0$, or

$$r - \sqrt{\Delta x^2 + \Delta y^2} >= 0 \quad (8)$$

The p_1 is one of the nearest neighbors of the reference p_2 if the predicate (8) is true. After obtaining the list of nearest neighbors, we average the heights of those neighbors for the reference point *p(x, y)*.

The pseudocode of the height estimation method is listed as follows:

```
1.  Define the threshold for the nearest neighbors
2.  Initialize the increment search distance
3.  Define a result list
4.  For each point in the 2D grid
5.     Use the point as a reference point
6      Count the number of neighbors for the
            reference point within the initial distance
7.    If the number of the neighbors less than the threshold
8.        Increase the search distance
9.        Goto step 6
10.    Else
11.       Compute the average height of the neighbors
12.       Assign the height to reference point
13.       Insert the new reference point into the result list
14.   Endif
15. Endfor
16. Return the result list
```

We have implemented the approach using a recursive function, and we list the pseudo-code as follows:

```
Point constructGridBasedModelg( Point gridPoints,
                                Point pixelCloudPoints,
                                float searchRadius,
                                int neighborsThreshold){
    List grid3DModel =empty;
    for each gridPoint in gridPoints {
      List  neighborList=empty;
      if  (distanceOf (gridPoint, each pixelPoint in
                                pixelCloudPoints)
                    < searchRadius){
          add pixelPoint to neighborList;
      }
      if (the size of list > neighborsThreshold){
          computer average height of neighborList;
          assign the height to gridPoint to form
                        one of the 3D points for grid 3D model;
          insert the new 3D point to grid3DModel
      }
      else{
          constructGridBasedModelg (gridPoints,
                        pixelCloudPoints,
                        searchRadius + a small increment,
                        neighborsThreshold);
    }
    return grid3DModel;
}
```

The complexity of the recursive function depends on the size of the point cloud and the grid-based 3D model, search radius, and the threshold of the number of neighbors.

In the best case scenario, without calling the function twice, there are enough neighbors captured within the search radius of each reference point. It requires the method traversing both points listed for computing the distance and generating the grid-based 3D model. Let's assume that the number of the points in the 2D grid is n, the complexity of the best scenario is $O(n^2)$.

In the worst case scenario, the number of the points captured in the point cloud equals the *neighborsThreshold*. The recursive function needs to call itself several times. The number of repetitions associates with the size of the point cloud (in inch) and size of the increment.

$$Repititions = \frac{size\ of\ the\ point\ cloud}{increment}$$

As the increment is redefined in terms of the size of the point cloud, the additional repetitions can be computed as n. Therefore the complexity for the worst case is $O(n^3)$.

IV. EXPERIMENTS

The accuracy of the surface model depends on several factors: the accuracy of the point cloud, the search radius for the nearest neighbors, and the threshold of the number of the nearest neighbors. In our experiments, we pre-define the threshold as a fixed number, and thus for a given point cloud, the search radius, i.e., smooth distance, is the key factor for determining the surface of the grid-based model.

The purpose of the experiments is to compare the reconstructed grid-based surface models in terms of different smoothing distances. Therefore, we are able to pick an appropriate smoothing distance for generating an acceptable grid-based 3D surface model. The acceptance criteria include both the smoothness and the accuracy of the surface model. The criteria are verified by observing the difference between the measured height and the height produced by the given smoothing distance at every 1/4" (the resolution in this experiment).

More specifically, in our experiments, we first define the desired resolution as 81 × 97. The size of squares in the grid is 1/4"×1/4". Then, for a generated point cloud, we reconstruct grid-based 3D surface models by applying different smoothing distances, including 0", 1/4", 1/2", and 3/4". To demonstrate, we list four screenshots of grid-based 3D surface models from Figure 6 to Figure 9, one for each smoothing distance. For analysis, we show one slice of a surface model for each smoothing distance in Figure 10.

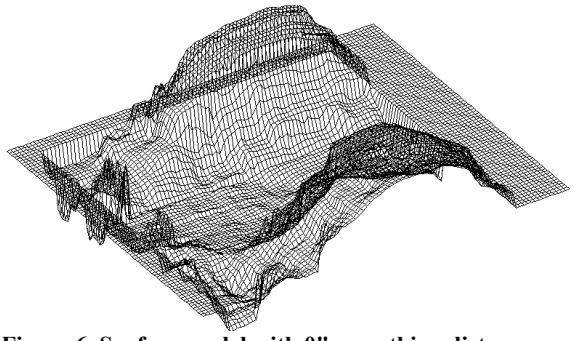

Figure 6. Surface model with 0" smoothing distance

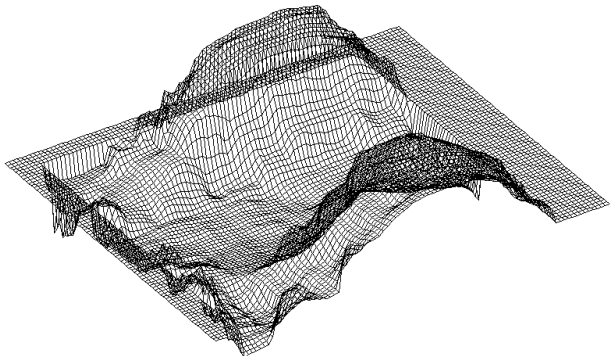

Figure 7. Surface model with 1/4" smoothing distance

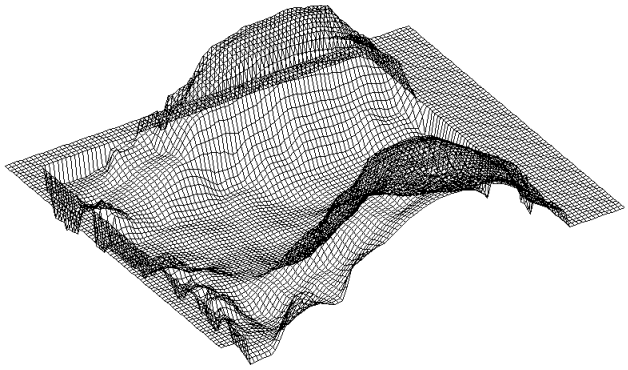

Figure 8. Surface model with 1/2" smoothing distance

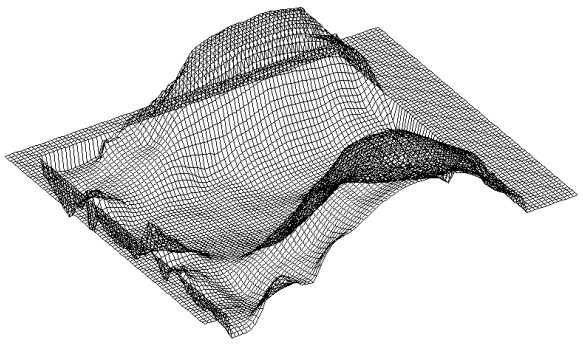

Figure 9. Surface model with 3/4" smoothing distance

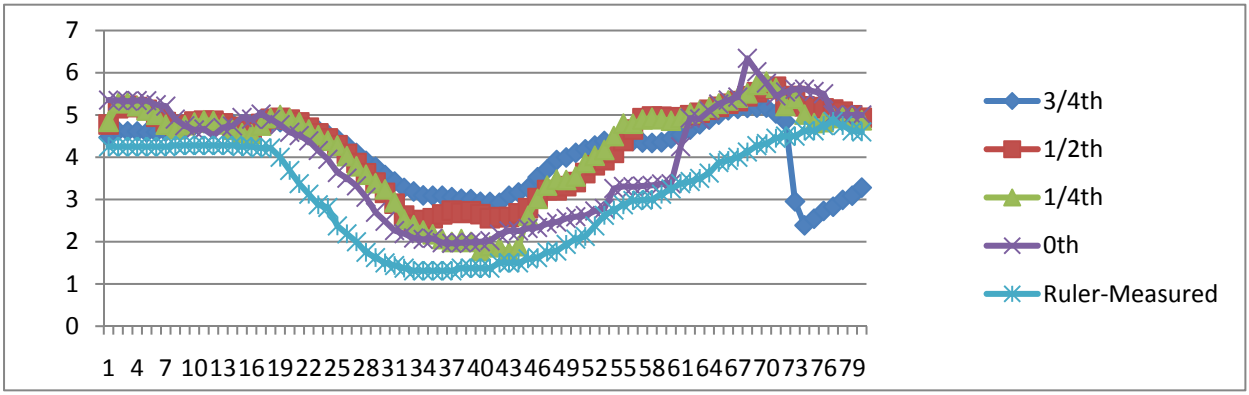

Figure 10. Slices of surface models for different smooth distances

Figure 6 to Figure 10 indicate that the 0" smoothing distance has the most accurate height compared to ruler-measured results. The 0" smoothing distance implies that for each reference point, we need to measure the distance to the point in the point cloud at least twice. We need to have the second distance measurement because the first calculation based on formula (8) cannot meet the threshold we have pre-defined. With an appropriate search distance increment, we can guarantee to obtain the closest height in the point cloud for the reference point. However, it does not guarantee the accuracy of the height, as there may be the inherited noise in the point cloud. The inherited noise leads to unsmooth surfaces which are useless for milling seat models.

On average, the smoothing distancing 0" results in the lowest standard deviation. It also produces some of the largest differences, near the end of the edge. This is due to an exceptionally skewed approximation that needed to take place for the last members of the row; it is a common problem that a grid point reaches far for a nearest captured point and acquires one that more correctly represents another part of the shape.

Constructing grid-based 3D surface models with smoothing distances 1/4" and 1/2" both result in smoother surfaces than 0" smoothing distance does. The model with the 1/4" smoothing distance is more accurate than the 1/2" smoothing distance, as the former distance results in a smaller standard deviation. However, the latter smooth distance constructs a smoother 3D surface.

The generated 3D surface model with a smoothing distance of 3/4" has the least accurate results and the smoothest surface, as well as largest the standard deviation. As smoothing distance increases indefinitely, the model becomes less accurate, leveling off to one height that is the average of all values on an infinite plane.

Note that those smoothing distance are the initialized value before the recursive search method calls itself. The smoothing distance will be increased if there are not enough neighbors found within the search radius. Our future work includes investigating a Gaussian smoothing filter to compare which smoothing algorithm, e.g., the nearest neighbor or Gaussian filter, fits better for smoothing the surface of seat models.

REFERENCES

[1] http://www.ptgrey.com/products/stereo.asp

[2] F. Huang, Integrating a Stereo Camera System into CoViS, Engineering College of Copenhagen, 2006.

[3] G. Jin, S. J. Lee, J. K. Hahn, S. Bielamowics, M. Rajat, and W. Raymond, "3D surface reconstruction and registration for image guided medialization laryngoplasty," in Advances in Visual Computing. Heidelberg: Springer Berlin, 2006, pp. 761-770.

[4] M. A. Brault, Americans with disabilities: 2005, current population reports, Washington, DC: US Census Bureau, 2008, pp. 70-117.

[5] R. A. Cooper and R. Cooper, "Trends and issues in wheeled mobility technologies," Retrieved Dec 2009, http://www.ap.buffalo.edu/ideaproto/Space%20Workshop/Papers/WEB%20-%20Trends_Iss_WC%20(Cooper).htm

[6] S. Se and P. Jasiobedzki, "Stereo-vision based 3D modeling and localization for unmanned vehicles," International Journal of Intelligent Control and Systems, Special Issue on Field Robotics and Intelligent Systems, Vol. 13, March 2008, pp. 47-58.

[7] S. Sheldon and N. A. Jacobs, Report of a Consensus Conference on Wheelchairs for Developing Countries, Bengaluru, India, Nov, 2006.

[8] S.Y. Zheng, R.R. Wang, C.J. Chen, and Z.X. Zheng, "3D measurement and modeling based on stereo-camera," The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVII, Part B5, 2008.

[9] Y. Li, A. Richard, D. Brenza, and J. Dansereau, "Determination of generic body–seat interface shapes by cluster analysis," IEEE Transactions on Rehabilitation Engineering vol. 8, Dec. 2000, pp. 481-489.