Visual Manipulation for Grid-Based 3D Surface Models

Weifeng Xu, Ben Luebbert, Stephen Frezza, Qing Zheng
Gannon University
Erie, PA 16541
{xu001, luebbert001, frezza001, zheng003}@gannon.edu

Todd Dinner
Precision Rehab Manufacturing Inc.
North East, PA 16428
tdinner@prmrehab.com

Abstract—After constructing a grid-based 3D surface model (GSM) for a customized cushion seat using stereo visions, we have observed that the model needs to be manually edited for minimizing the ridges. This paper presents a systematical approach to develop a visual manipulation tool, called VT-GSM, for visually manipulating GSMs. To realize the approach, a 3D graphic transformation pipeline has been adopted and modified, and then, incorporated with the pipeline, a Model-View-Controller pattern has been applied for designing the architecture of the VT-GSM. After that, a variety of operations have been proposed, implemented, and demonstrated for manipulating GSMs. Finally, a polished cushion seat model has been illustrated using the VT-GSM.

Keywords- grid-based 3D surface model; customized seat, graphic transformation pipeline; visual manipulation; tool.

I. INTRODUCTION

Visual manipulation is a very powerful and effective technique for demonstrating the functionality and usability of the software systems. This paper presents a systematical approach to develop a visual manipulation tool, called VT-GSM, for visually editing grid-based 3D surface models (GSM). The VT-GSM facilitates the data interpretation and the data modification of GSM.

The tool is developed as an extension of the prior work [4]. The prior work illustrates a method to construct a GSM for a customized cushion seat using stereo visions. As have been mentioned in our prior work, individual customization, such as customization of cushion seat, is a growing movement in the manufacture of rehabilitation equipment. For example, a regular seat may cause sores and discomfort for some users who require seating for a period of time [5]. 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, the option of customized seats should be offered, as customized seats maintain low contact pressures for people they support, and thus reduce pressure in body soft tissues [4].

Creating custom cushion seats is a skill- and time-intensive process. To automate the manufacture process, a stereo vision camera was introduced for capturing the structural information of a person's hip mould (see Figure 1). Then, the captured digital information was used for constructing a GSM. A GSM consisted of a collection of point \( p(x, y, z) \) in a 3D coordination [6]. The default distance between adjacent points was consistent, e.g., half inch. The distance can be adjusted in term of the manipulation resolution. The manipulation resolution was defined as a number of row * number of columns grid. The size of the grid was critical for constructing the GSM. The grid needed to cover the desired area of the point cloud, i.e., region of interest or ROI for short (see the rectangular area in Figure 1). Any information out of the ROI boundary was cropped out. The constructed GSM is shown in Figure 2.

Figure 1. A hip mould

Figure 2. A constructed GSM from the hip mould

Although the algorithm used for constructing the GSM produces visually recognizable and tolerant results for our client, the model contains some areas needs to be polished before sending to the milling machine. We specify the reasons as follows:

1. The surface model generates ridges that need to be removed.
2. Some points (z values) should be manually adjusted.
3. The position of the surface model needs to be centered.
4. Extra edges should be removed.

The first two issues result from the inherited environment noise, such as the brightness, smoothness, and color of the object surface, the lighting of environment, and the viewing angle of the stereo camera as some surface areas cannot be viewed by both lenses of the camera from certain angles. The
third issue is caused by the operational errors. For example, the center of ROI does not match the center of the grid. The last issue is the results of a design decision that any information outside of the ROI boundary will be flattened. The flat area in the GSM needs to be cut off.

The goal of this paper is to develop a model editing tool to visually manipulate the GMSs, including viewing models from different angles and copying, shifting, and editing points in the models.

The rest of the paper is organized as follows: section II describes the graphic transformations for visualizing GMSs, section III shows the architecture design of the VT-GSM, section IV illustrates the operations of the VT-GSM and demonstrates a polished GSM by applying these operations, and section V concludes the paper.

II. GRAPHIC TRANSFORMATIONS

Essentially, the visual manipulation for GMSs is a series of 3D graphic transformation for a collection of vertices. In this section, a common approach for 3D graphic transformation is illustrated and a simplified 3D graphic transformation is proposed for the VT-GSM.

A. 3D Graphic Transformation

The common approach for visualizing 3D graphs consists of a transformation flow that causes models to appear as if viewed from a defined perspective [1]. Figure 3 displays the typical pipeline, showing a sequence of transformations that transform an input vertex \((x, y, z, h)\) to an output vertex \((x', y', z', h')\). The first three parameters correspond to \(x, y,\) and \(z\) axis, respectively, and the fourth vertex member \(h\) allows transformations to be applied that change the vertex’s position and rotate the matrix.

![Figure 3. Pipeline of 3D graphic transformation](image)

Transformations are categorized into four types: modeling transformation, view transformation, projection transformation, and viewport transformation.

The modeling transformation applies to the model itself. It changes the actual data in the model, and also positions the model in the context of all objects in its scene. Points or vertices that compose the model can be translated, scaled, and rotated.

The view transformation designates how the model appears because of the chosen perspective coordinate and line of sight. Points in visible objects can be transformed to give the appearance of different angles and positions of view.

The projection transformation changes the size of the displayed model based on its depth and distance away from the current perspective.

The viewport transformation maps the model's viewable points to coordinate on the monitor. More specifically, 3D shapes are mapped to two-dimensional window coordinating with a viewpoint transformation. This transformation maps the 3D scene to a 2D rectangle.

B. VT-GSM 3D Graphic Transformation

The VT-GSM 3D graphic transformation (see Figure 4) is a modified version of the common 3D graphic transformation. To improve the performance of the VT-GSM, it combines view, projection, and viewport transformation into a single transformation process. The combined transformation is grouped in a dashed frame in the figure. The VT-GSM 3D graphic transformation also adds a vertex manipulation component for editing vertices.

![Figure 4. Pipeline of VT-GSM 3D graphic transformation](image)

To combine three transformation into one process, we 1) choose a fixed perspective coordinate for simplifying the view transformation, 2) maintain a consistent scale to view the 3D grid model for simplifying the project transformation, and 3) pick up the same points to be mapped in a monitor for simplifying the viewport transformation.

The vertex manipulation component is responsible for altering the point \(z\) position in GMSs. Whenever a \(z\) value has been changed, the system generates an action change event, and the event will be handled by an event handler. The event handler pushes the changed vertex value to the VT-GSM pipeline so that the modified values can be visualized.

The only transformation of the VT-GSM kept is the model transformation. The model transformation mainly includes model rotations and translation.
A rotation presents a new orientation of the same surface by turning its points around to \( x \), \( y \), or \( z \) axis. This is accomplished by iterating through each vertex in a GSM, and multiplying the desired rotation matrix. These three rotation matrices are shown in Figure 5, respectively. Each rotation matrix is based on the sine and cosine of the desired angle of rotation [2].

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos(x) & -\sin(x) & 0 \\
0 & \sin(x) & \cos(x) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
\cos(y) & 0 & \sin(y) & 0 \\
0 & 1 & 0 & 0 \\
-\sin(y) & 0 & \cos(y) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
\cos(z) & -\sin(z) & 0 & 0 \\
\sin(z) & \cos(z) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Figure 5. \( x \), \( y \), and \( z \) rotation matrices

Translating the surface moves all its vertices through 3D space by the same amount in certain direction. Simple addition of the \( x \), \( y \), and \( z \) components by a constant translation in each dimension will move the entire surface. This can be horizontal, vertical, or depth movement. The angles of any line in the surface are unchanged. The translation matrix is shown in Figure 6.

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
tx & ty & tz & 1
\end{bmatrix}
\]

Figure 6. Translation matrix

III. ARCHITECTURE DESIGN

The architecture of the VT-GSM simply follows the Model-View-Controller (MVC) pattern. MVC is an architecture that separates an object's data from its rendering and allows the applications to encapsulate responses to inputs [2]. The architecture structures a system into model, view, and controller components. The model contains the actual data of the object of modification. The controller is the component that changes the model's data. The view displays the model and outputs results to the user. The MVC follows a general control flow, i.e., the controller acts once input is received, and changes the model's state accordingly. Then the view gets data from the model and generates a display for the user. This is looped until the user exits.

The MVC pattern implemented in the VT-GSM (Figure 7) with a set of components: Surface, ShapeView, and ShapeControl. Each component corresponds to a modular in MVC, i.e., model, review, and control, respectively. The control component ShapeControl changes the model, i.e., Surface's data. The Surface holds the vertices that compose the GSM. Once the Surface has been modified, the ShapeControl pushes the surface data to ShapeView for rendering. The graphic transformation starts working right after the pipeline receives the data.

The Surface class, designated as the model component, contains the data of the GSM, such as the size of the GSM, the number of points in \( x \) and \( y \) dimensions, the angles of the shape, and the coordinates of currently selected points. The Surface object's actual points are stored in a two-dimensional array. Each entry is a vertex with four homogenous coordinates. Four-dimensional matrices are used to implement the transformations [2]. Each vertex in array has four coordinates, \( x \), \( y \), \( z \), and \( h \). The \( x \) and \( y \) positions are pre-calculated based on the required display resolution. The value \( z \) is the height of the surface we want to manipulate.

The ShapeView is responsible for visualizing the GSM. The ShapeView acquires positions of each vertex in the GSM. After applying matrix transformations to present a particular rotation and translation of every point, the ShapeView plots them by drawing a line from each point to the point in the next row and the next column. This continues until the plot reaches the last row. The ShapeView is also responsible for displaying the correct colors, and indicating user input. The input is highlighted to the user with the use of special hues and a near-real time response in position. It is worth mentioning the ShapeView includes a repaint method. The repaint method implements all the pipeline transformation. It will be invoked whenever a value change event is generated. For a selected area, it removes old visual data and repaints new meaningful frames.

The ShapeControl coordinates the Surface and ShapeView. It listens to the event generated by user actions and determines when and how to change the GSM and when to push the changed model for visualization. The accepted inputs are related to mouse events, keyboard events, and window events. Based on different events and combinations of different events, the user of the VT-GSM can specify different editing modes that allow them to edit individual points or a collection of points.
IV. EXPERIMENTS

The VT-GSM provides various visual manipulation functionalities. These functionalities mainly include:
1. Edit: move a point(s) in a GSM up and down
2. Shift: shift a point(s) in a GSM to left and right
3. Copy: copy row/column in a GSM over the rows/columns
4. Undo: restore previous operation

As those operations must apply to a point or a collection of points, we further organize the operations in terms of points, areas, and shapes (Table 1). The user interface of the VT-GSM is shown in Figure 8. In the experiments, the GSM in Figure 8 is used for a baseline GSM on which all the editing and visualization are based. For comparison purpose, all the modified portions of the GMSs shown below are repainted in a different color. For the limited space, only a few operations are demonstrated in the section, including point editing, shifting, area editing, shape editing, and undo.

Table 1. VT-GSM operations

<table>
<thead>
<tr>
<th></th>
<th>Point</th>
<th>Area</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit (↑↓)</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Shift (←→)</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Copy</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Undo</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Figure 8. User interface of VT-GSM

A. Point editing

The point editing function is for changing the $z$ value of a point. It allows a mouse to click, select, and drag a point to the required height. When a point is selected for editing, the point is highlighted. The mouse movement changes the value of the $z$, and then the point that the mouse pointed to will be repainted. Figure 9 demonstrates the point editing operation.

B. Shift

The shift operation causes each point in the shape to copy itself into the point to its left or right. In Figure 10, the original shape is altered by shifting to left.

Figure 9. Point editing operation

Figure 10. Shift the whole area to left

C. Area editing

The area editing operation is used to uniformly change the heights, i.e., the value on $z$ coordinates, of each point in a sub-area in the GSM. The user may specify an absolute or relative height increase. An absolute increase will raise each point to the specified height, resulting in a cube-like plateau as shown in Figure 11. A relative increase will raise each point by the specified height, preserving the layout of the shape, but raising the selected points to a new elevation.

Figure 11. Area editing operation

D. Shape editing

The shape operation is similar to area operation. It bows the selected area, changing the heights closest to the center at the greatest rate. The result is a nearly spherical shape change that changes small malformations into smooth grooves more natural for seats than a sharp right-angle
change. Figure 12 shows an exaggerated instance of this for demonstrative purposes.

Figure 12. Shape editing for a selected area

E. Copy rows/coloums

Touching up a seat shape is usually made easier with the ability to copy a row to other rows in a selected area, or to copy the values from one column into other columns. This is usually to smooth out areas near the edge that have no effect on the custom form, and are leveled for the ease of manufacture. This function requires the MVC components to accept a certain point from the user. For a forward-back copy, each point in the input row is copied to each other row. For a left-right copy, each point in the input column is copied to each other column. In Figure 13, a small area was selected along the left (sitter’s right) ridge. The copied column was chosen within this area, and each other column changed its vertices’ values to mimic it.

Figure 13. Copy a row operation

F. Undo

The undo operation removes all the changes applied to the GSMs. After the GSMs are loaded into memories, all the operations applied to the GSMs will be saved in a list. It is worth mentioning that the system saves the list of the operations, not the list of modified GSMs that the users have seen in each modification. It improves the performance of the system as each GSM contains 81*69 vertices, saving a sequence of GSMs requires more memory than saving a sequence of operations. When the undo operation is executed, the ShapeControl component picks up the previous operation, reverses it, and reverts to the previous GSM.

G. Result of User Editing

To illustrate the visual manipulation results, Figure 14 displays an edited GSM based on Figure 2 using the operations described above.

Figure 14. GSM after editing

V. CONCLUSION AND FUTURE WORK

We have presented a systematical approach for developing a visual manipulation tool, called VT-GSM, for visually editing GSMs. A 3D graphic transformation pipeline has been adopted and modified for visually manipulating GSMs. The experiments demonstrate manipulation functions necessary for visually manipulating the GSMs. Finally, a polished cushion seat model is illustrated using the VT-GSM.

The VT-GSM is necessary for manually manipulating the GSM. The future work includes developing an algorithm to automatically smooth the GSM.

REFERENCES