Real-Time Spatial Surface Modeling System Using Wand Traversal Patterns of Grid Edges

Harksu KIM†, Member, Dongtaek KIM†, Jaeung LEE††, and Youngho CHAI†a, Nonmembers

SUMMARY This paper presents a grid-based, real-time surface modeling algorithm in which the generation of a precise 3D model is possible by considering the user’s intention during the course of the spatial input. In order to create the corresponding model according to the user’s input data, plausible candidates of wand traversal patterns of grid edges are defined by considering the sequential and directional characteristics of the wand input. The continuity of the connected polygonal surfaces, including the octree space partitioning, is guaranteed without the extra crack-patching algorithm and the pre-defined patterns. Furthermore, the proposed system was shown to be a suitable and effective surface generation tool for the spatial sketching system. It is not possible to implement the unusual input intention of the 3D spatial sketching system using the conventional Marching Cubes algorithm.

key words: virtual conceptual sketch, spatial surface modeling, marching cubes, wand traversal pattern

1. Introduction

With the development of computer technology, use of 3D models that utilize digital technology from the design stage to the actual production stage is widespread. 3D models are usually created with 3D modeling tools that use 2D input. Differing input according to the directions of the model, cognitive thinking, and familiarity with the graphic tools in question are needed in order to produce correct 3D models through 2D input. Since 3D models are required in a wide range of fields, demand for a method of producing 3D models in an easier and more intuitive way is increasing, leading to research in various fields. Research is being done at various levels on 3D spatial input interface systems. The research that goes beyond flat limits of the working environment makes it possible to produce models directly through 3D spatial input. In the design field that produces 3D models, the designer’s actual intentions to draw must be expressed accurately in order to be used as 3D input, so accurate tracking of position and posture of input is inevitable.

This paper presents a system that calculates the position and posture of the input device and enables spatial sketching by tracking infrared reflection markers using two infrared cameras. An algorithm that generates the surfaces corresponding to the user’s intention by using a spatial input system is needed to accomplish spatial sketching of the desired model. Moreover, the exact intended model must be drawn according to spatial input for the user to do spatial sketching freely, and surfaces must be generated in real time to make it possible to produce models intuitively.

In this system, work space is divided into a bundle of virtual grids and surfaces are generated in grid units in order to produce surfaces in real time through 3D spatial data that has been input within the spatial input system. It suggests a way to generate surfaces by retrieving the pre-defined patterns connecting the intersections created between grids and the traversal plane through spatial input data. And it proves the efficiency and precision of the suggested algorithm through comparison of a grid-based Marching Cubes algorithm. Furthermore, it applies octree space partitioning according to the complexity of input and allows the generation of a model that reflects the precise input intention in real time without an extra crack patching algorithm, making it applicable in modeling and the design sector, especially in the conceptual design stage.

This paper is composed as follows. In Chapter 2, the concept of a spatial input system, which is currently in use, the concept of a spatial grid-based algorithm, and the existing spatial grid-based surface generation algorithm are described. Chapter 3 tabulates the wand traversal patterns suggested, while Chapter 4 compares and analyzes polygonal surfaces generated by the traversal pattern table and those generated by the existing Marching Cubes algorithm. Chapter 5 presents the conclusion and the future direction of the research.

2. Spatial Input System and Grid Algorithm

2.1 Spatial Input System

A spatial input system transcends the plane limits of the existing system in which 3D models were generated through 2D input and makes it possible to express the 3D intentions of the user and enables interaction in 3D space. A tablet that makes input easier on a plane and 3D mouse which allows input of 3D data within a limited space have been developed. However, the tablet is limited in that it is a 2D plane and the 3D mouse is difficult to manipulate [1]. Since a system that allows direct input and output in 3D virtual space is lacking in accuracy of input, a multiple viewpoint visualization application according to the different viewer’s position was considered first [2].

As computing power has been enhanced, a spatial input
system maximizes the visual immersive sense of the user, while simultaneously making an interaction for the user possible in various ways. Used with haptic devices and other similar devices, this increases immersive sense, even with imprecise spatial input [3]. 3D spatial input was tried on a desktop computer when the 3D output device was not running smoothly [4]. As stereoscopic images and immersive projection environment have become more common, creating geometries through spatial curves [5], gesture input [6] and 3D input wand has been attempted in various ways. Generally, using a 3D spatial input system in a 3D environment with a haptic device enhances approachability to virtual space and alleviates the awareness of the difference inherent in the 3D input interface, enabling more precise input [8].

The spatial input devices used in this paper were the two infrared cameras in Fig. 1 (a) and an input wand with three reflection markers, as shown in Fig. 1 (b). OptiTrack V100R2 [9] from NaturalPoint is the camera used in the system. Twenty-six infrared LEDs illuminate infrared rays and the cameras use infrared filters to capture images at 100 frames/sec. The images are converted into files and used to find the 3D location of the three reflection markers and consequently calculate the spatial position and posture of the input wand. By using the wand’s characteristics, such as position, direction, and speed within the scope of the two infrared cameras, figures can be drawn or spatial geometries can be modeled.

2.2 Spatial Grid-Based Modeling Algorithm

An important factor in enhancing the sense of immersion of virtual environment modeling is the real time interaction between the graphic representation and the input devices. Tracking of the input device must be done in real time by using the camera and the generation of surfaces should also be done in real time. In addition, this must be carried out within a permissible margin of error to be used in the modeling system [10]. Using dynamic spatial grids [11] to input or map a 2D tablet, which enables precise input in 3D space [12], can reduce the margin of error. This paper aims to generate surface models in real time by using 3D data received from an infrared tracking system. Creating surfaces from strokes is close to a surface reconstruction problem from a point set. But the surface generation in this paper uses the ordered input point set (triangles), whereas the points set in the general reconstruction problem do not have the ordered sequence (point cloud) so the reconstruction algorithm has to find or formulate the order to connect points.

In the case of modeling, such as spatial sketching, similar surfaces are input repeatedly in the same space in search of the exact surface that the user wants to create, so surfaces overlap and ultimately lead to a point where generating surfaces in real time is no longer possible. In order to solve this problem, 3D input space was first divided into grids, polygons were generated within grid units, and unit polygons were connected to generate the desired surface model. The number of basic grids in a spatial grid was fixed, so the amount of data that could be contained changed in accordance with the number of grids through the over-sketching algorithm, regardless of the input surface numbers, and real time surface modeling was made possible by applying the over-sketching algorithm only to the applicable grids according to the input location.

The algorithm that generates surfaces according to such grid units mostly involve Marching Cubes [13] used in volume modeling, while research on expanding the range, limitations, and properties of calculation of various figures is being carried out [14]. Creating sharp edges that cannot be created through the Marching Cubes algorithm can be implemented by using internal vertices within the specific grids in the Expended Marching Cubes [15] and the intersection points and normal vectors between the grids and the input isosurfaces in the Dual Contouring [16]. Furthermore, there is a method by which to organize the intersection patterns and convert the boundary [17] and 3D data [18] into grid internal data. When the complexity of the geometry differs in parts, the grids are subdivided into octrees to maintain the form, while the crack patching algorithm can be used to connect the cracks resulting from octree partitioning [19]. This paper suggests a way to include the surface data which has been input in real time into the prepared grids in real time, rather than redistributing the given geometry in a space that has been divided into grids, thus making it possible to construct surface in real time during octree partitioning using a smaller number of patterns than with [17] or [18] without an additional crack patching algorithm.

3. Wand Traversal Pattern Algorithm of Grid Edges

3.1 Decision of Wand Traversal Patterns

The spatial grids used in this system are a bundle of cubes with 12 edges in each cube created by connecting 8 vertexes. The 8 vertexes have position data and the intersection points in edges include normal vector as well as position data.

In this spatial input system, a wand with three infrared reflection markers receives spatial input, as shown in Fig. 2. The position of the wand shows the coordinates and movements of the wand and a polygonal surface is formed by connecting the intersection of the edges in each consecutive traversing grid according to movement.

As shown in Fig. 3, when the input wand enters at the
top plane of the grid and exits at the bottom plane, the sequence of four intersections is determined according to the order that the wand meets each edge. Following the sequence of each intersection, the plane where the wand enters the grid and the plane where it exits the grid can be located. If the intersections at which the edges are crossed are connected, a polygonal surface can be created. The input data is structured as polygons (triangles) which are generated by moving the input wand. The intersection points are calculated by comparing the plane equation of these triangles with the edges of each grid. Eventually, the intersection points are used to define the output surface.

When there are three intersections, connecting the consecutively formed intersections results in a triangular surface. When there are four or more intersections, various types of polygonal surfaces are formed according to the sequences in which the intersections are connected. As in the case of Fig. 4, even a basic input of crossing the middle of the grid can result in six different patterns, depending on the sequence of the intersections. There can be four 5-sided polygon patterns according to the moving direction of the input wand in addition to the square pattern showing the basic polygonal surface.

Figure 5 shows that different polygons can be formed depending on the moving direction of the wand, even when the crossing order and intersections are the same. In Fig. 5 (a), the wand crosses the grid without any turns, while Fig. 5 (b) shows the wand turning as it crosses the grid. The number of the intersections in Fig. 5 (b) is the same as Fig. 5 (a), but the input movement is different, so the surface must be drawn in a different way. This can be taken care of by including the normal vector of the wand to each intersection at the time of input.

Using the existing Marching Cubes algorithm to draw the trajectory of traversal requires determining the distance between each vertex and the input plane of traversal as the vertex value, then using this value to retrieve the Marching Cubes table and draw the appropriate traversal polygons. If the same vertex value is calculated so that there is no difference in the distance, but the different wand traversal movement is applied in the input process, drawing a traversal plane that is different from the input is possible. In this system, we created a polygon by determining the sequence and direction of the intersections following the input wand’s movements, then connecting them according to the pre-defined intersection patterns. In this way, we were able to produce a wand traversal plane directly within the grid by following the movement of the input wand without assigning the vertex value and the reverse calculation of the isosurface.

3.2 Wand Traversal Pattern Table

By connecting the intersections that include the sequence and direction of the wand’s movements, various types of polygons can be generated as shown in Fig. 6. A maximum of six intersections with meaningful movement of the wand can be produced, and they make it possible to form each of the polygons by arranging patterns of plausible input and
creating a table according to the number of intersections. When looking for a pattern, we made the retrieval possible only when the wand’s movements had crossed through the grid, and we defined patterns that took all the turns and twists within the grid into account.

In order to create one surface, there must be at least three intersections, as shown in Fig. 7. The three intersections formed by one wand traversal are connected to construct a triangular surface. There is one connecting pattern and eight forms can be made for each of the corners of the grid.

When there are four intersections, the side that the wand enters and the side that the wand exits are determined by the intersections formed consecutively. In Fig. 8, the entering side is designated as IN and the exiting side as OUT. When there are four intersections in a grid, the polygonal surface of traversal formed resulting from the location of the intersections on IN and OUT were divided into eight types. The 8 traversal types can be formed into 42 plausible polygon patterns, including the turns and twists within the grid, making it possible to produce a total of 336 polygons for each of the sides of the grid. These patterns make possible all forms created by the Marching Cubes algorithm, as well as cases where the turns and twists within the grid are included.

When five intersection points are formed according to the wand’s traversal in this system, three patterns can be formed with the polygons, as shown in Fig. 9. Five intersection points are formed with one movement of the wand when the input enters at the IN side, curves at one edge, and then exits through the OUT side.

There are two cases in which six intersection points are formed through the wand’s traversal: when the intersection edges in the IN side are adjacent and 2 edges are passed during the traversal with the intersection edges in the OUT side are also adjacent and when the intersection edges in the IN side are opposite and two edges are passed during the traversal with intersection edges in the OUT side are also opposite. The six intersection points formed create polyg-
onal surfaces in succession according to the input sequence and direction. Figure 10 shows that when six intersection points are formed, the IN side and the intersection points of two edges first create a rectangle and then the intersection points of the edges and the intersection points of the OUT side together create a spatial hexagon.

4. Comparison of Marching Cubes and the Wand Traversal Pattern Algorithm

4.1 Range and Method of Creating a Surface

Figure 11 (a) is a surface formed by connecting the intersection points from the wand’s input according to the pattern defined in this system. Connecting the intersection points according to the pattern means connecting all of the intersection points formed by the user’s movement in a 3D virtual space consisting of grids, so a surface is formed in the same position as the actual input. Figure 11 (b) is a surface formed using the Marching Cubes algorithm. The reversely calculated isosurfaces for each grid are connected and are similar to the wand’s actual input, but a different surface is formed.

The two algorithms are different in their range and method of drawing the surface. The wand traversal pattern forms a surface only where the entire wand has crossed the grid. With the Marching Cubes algorithm, however, the value of vertexes are determined even when there is a partial input within the grid, so this vertex values determine the pattern and then is calculated in reverse to form isosurfaces. Because of this difference in range, the wand traversal pattern forms a surface in a much smaller area than the actual input, while the Marching Cubes algorithm forms a larger surface than the actual input and may produce a model that is different from what the user intends.

When the wand crosses the grid without any significant change, as shown in Figs. 12 (a) and 12 (b), done with Marching Cubes, and 12 (c), done with a wand traversal pattern, a surface similar to the wand’s movement is shown.

However, Fig. 13 (a) shows that when the wand enters the grid from the upper side, turns, and then exits through the front side, the result of using Marching Cubes, as shown in Fig. 13 (b), may produce a pattern that is quite different from the actual input. In Fig. 13 (c), the polygonal surface created with the wand traversal pattern has intersection points that include sequence and direction, so the resulting polygon formed by the intersection points is a similar shape to the actual input, regardless of where the wand twists or any changes in the input.

Figure 14 (a) is a case in which the frequency of input is high due to rapid change within the grid. Figure 14 (b) shows that using the Marching Cubes algorithm may result in a type of pattern that is different from the user’s actual movements when the frequency of input is inconsistent due to faults in the isosurface calculation. The surface in Fig. 14 (c) is formed by connecting the intersection points, but since it is only created through intersection points regardless of the movements within the grid, it produces a surface similar to that of input and therefore raises the possibility of producing a surface that is even more similar to actual movement by controlling the size of the grid.

The input in Fig. 15 (a) enters diagonally to the upper side, passes an edge within the grid, and then exits through the side. In this case, the Marching Cubes algorithm may
present polygons in all 15 patterns depending on the reverse isosurface calculation. This may produce surfaces similar to that of input depending on the pattern, but it may also produce patterns that are unintended, as shown in Fig. 15 (b). The wand traversal pattern, however, forms polygons by searching for the appropriate patterns according to the 36 intersection points that can be produced by movement, so it makes it possible to produce the desired surface regardless of the complexity of input. Figure 15 (c) is a polygon formed with 5 intersection points.

4.2 Crack-Patching, According to Octree Partitioning

Using a grid-based algorithm produces models according to pre-defined patterns, so the resulting shapes are similar to actual input. Figure 16 (a) is a target surface that the user wishes to draw. 16 (b) shows limitations in producing a surface similar to the intended surface. 16 (c) is a surface formed by smaller grids in order to produce a surface that is even more similar to the intended surface and is consequently inefficient, for it requires more calculation in accordance with the number of grids. Figure 16 (d) uses octree portioning of grids depending on the complexity of the surface and produces a surface efficiently.

When octree portioning is applied to the grids in a grid-based algorithm, surfaces produced by the difference in size between adjoining grids sometimes cannot be connected. An empty space appears when a surface presented as a single plane in a large grid is presented as several planes in the adjoining small grids. In a wand traversal pattern algorithm, a method similar to the crack patching in the Marching Cubes algorithm is used to solve the problem of adjoining grids with a different size. However, cracks that appear due to octree partitioning from large grids to small grids can be patched easily by inheriting the intersection points of the parent grid to the child grid, without having to reset the vertex value by using the reverse calculation process of the partitioned surface.

A crack appears between a large grid and a small grid during octree partitioning, as shown in Fig. 17 (b), because the small grid forms a surface closer to the actual input. Figure 17 (b) shows the empty space having been filled up by connecting it to a large grid adjoining it.

This system implemented octree partitioning according to changes in normal vectors of the input. When the normal vectors of the intersection points formed on the entering surface of the grid and of those formed on the exiting surface are compared and found to change over a certain limit, octree partitioning is implemented in that particular grid, as shown in Fig. 18. A polygon with a pattern similar to that of input is produced through the inclusion of the direction of input, but there are limitations in producing the same model as the actual input using the limited pre-defined patterns. Therefore, octree partitioning was used according to changes in input direction to produce a complex polygon model even closer to the input.

Figures 19 (a), (d), (g), and (j) are the input wand’s spatial movements. Figures 19 (b), (e), (h), and (k) are models produced with the Marching Cubes algorithm with octree partitioning. Figure 19 (c), (f), (i), and (l) are models by applying octree partitioning individually according to the wand’s input and then connecting the wand intersection points in each grid. In this case, several virtual wand shapes are applied instead of using the shape of the physical wand itself. Unlike a model that is formed similar to input by connecting the input points to locate the intersection points, a surface produced with the Marching Cubes algorithm is continuous when the input is consistent, but shows inconsistent cracks in the parts where the frequency of input changes.
Compound model consisting of many surfaces can be generated by combining two or more different strokes as shown in Fig. 20. At first, boundary grids which contain two or more surfaces are defined and an additional surface construction algorithm in this boundary grid is implemented to merge or zip two or more surfaces into one continuous surface. It is not possible to handle sharp features, since the discrete grid structure is used to represent polygons in the proposed algorithm. Special attention is required to handle sharp features in grid-based modeling, such as dual contouring or extended Marching Cubes.

The proposed algorithm is designed to generate the polygons in real-time as shown in Fig. 21. The Marching Cubes implementation in this paper is also fast enough to generate the polygons interactively, due to the surface input instead of volumetric input. In case of surface input, assigning the vertex value with the distance to the input surface is straightforward. But in volumetric input, assigning the vertex value requires some kind of interpolation process. So the speed of the process is better in the Marching Cubes implementation, but there are some holes which are fatal errors in Marching Cubes with the discontinuous input. In case of using the intersection points by the wand traversal patterns, holes between grids are not generated.

5. Conclusions

This system aims to enhance the accuracy of the user’s intended surface through real-time spatial input. We propose connecting the wand traversal intersection points that form between the input and the grids for actual input and accurate surface formation. We compared the surface produced by referring to the pre-defined patterns in accordance with the position and direction of the intersection points formed in order by the wand’s input and produced using the existing Marching Cubes algorithm. We divided the wand’s simple input and the complex input containing the turns and twist and then compared the results of each algorithm and found that forming a surface close to what the user had intended was possible through the wand traversal pattern, even when the Marching Cubes algorithm was unable to do so. Furthermore, we made it possible to efficiently produce a model by applying octree partitioning to the grid, as well as solving the problem of filling cracks in a lot simpler way than the crack-patching method of the Marching Cubes by connecting the intersection points formed between each grid. This new grid-based algorithm made a more direct formation of models possible through real-time spatial input, and we expect it to be used in fields such as product design and 3D authoring tools.

Acknowledgments

This work is funded by NRF (R01-2007-000-20283-0) and Seoul R&B Program (PA100017).

References

[1] S. Zhai, “User performance in relation to 3D input device design,” Proc. ACM SIGGRAPH, vol.32, no.4, pp.50–54, 1998.
[2] S. DiVerdi, I. Rakkolainen, T. Hollerer, and A. Olwal, “A novel walk-through 3D display,” SPIE Electronic Imaging, Stereoscopic Displays and Applications XVII, vol.6055, pp.1–10, 2006.
[3] B. Baxter, V. Scheib, M.C. Lin, and D. Manocha, “DAB: Interactive haptic painting with 3D virtual brushes,” Proc. ACM SIGGRAPH, pp.461–468, 2001.

[4] E. Sachs, A. Roberts, and D. Stoops, “3-Draw: A tool for designing 3D shapes,” IEEE Comput. Graph. Appl., vol.11, no.6, pp.18–26, 1991.

[5] G. Wesche and H. Seidel, “FreeDrawer-A free-form sketching system on the responsive workbench,” Proc. VRST 2001, pp.167–174, 2001.

[6] S. Schkolne, M. Pruett, and P. Schroder, “Surface drawing: Creating organic 3D shapes with the hand and tangible tools,” Proc. SIGCHI, pp.261–268, 2001.

[7] M. Fiorentino, R. Amicis, G. Monno, and A. Stork, “Spacedesign: A mixed reality workspace for aesthetic industrial design,” Proc. International Symposium on Mixed and Augmented Reality, pp.86–94, 2002.

[8] M. Farber, T. Dahmke, C.A. Bohn, and H. Handels, “Needle bending in a VR-puncture training system using a 6DOF haptic device,” Proc. Medicine Meets Virtual Reality17, pp.91–93, 2009.

[9] www.NatualPoint.com, OptiTrack V100R2, NatualPoint Inc., 2006.

[10] M. Fiorentino, G. Monno, P.A. Renzulli, and A.E. Uva, “3D pointing in virtual reality: Experimental study,” Proc. International Conference on Tools and Methods Evolution in Engineering Design, 2003.

[11] S.H. Han, “A study on the effectiveness of 3D input interface by using dynamic grid,” Master Thesis, Graduate School of Advanced Imaging Science, Multimedia and Film, Chung-Ang University, Seoul, Korea, 2005.

[12] M. Masry, D. Kang, and H. Lipson, “A freehand sketching interface for progressive construction of 3D objects,” Comput. Graph., vol.29, pp.563–575, 2005.

[13] W. Lorenzen and E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” Proc. ACM SIGGRAPH, pp.163–169, 1987.

[14] T.S. Newman and H. Yi, “A survey of the marching cubes algorithm,” Comput. Graph., vol.30, pp.854–879, 2006.

[15] L.P. Kobbelt, M. Botsch, U. Schwanecke, and H.P. Seidel, “Feature sensitive surface extraction from volume data,” Proc. ACM SIGGRAPH, pp.57–66, 2001.

[16] T. Ju, F. Losasso, S. Schaefer, and J. Warren, “Dual contouring of hermit data,” Proc. ACM SIGGRAPH, pp.339–346, 2002.

[17] Y. Teshima, K. Kase, S. Usami, and A. Makinouchi, “Method and program for converting boundary data into cell inner shape data,” US patent 7,321,366 B2, 2008.

[18] K. Kase, Y. Teshima, S. Yamazaki, S. Usami, and A. Makinouchi, “Method and program of converting three-dimensional shape data into cell internal data,” US patent 7,333,104 B2, 2008.

[19] R. Shekhar, E. Fayyad, R. Yagel, and J.F. Cornhill, “Octree-based decimation of Marching Cubes surfaces,” Proc. IEEE Visualization ’96, pp.335–342, 1996.

Dongtaek Kim has a M.S. in the graduate school of advanced imaging science of Chung-Ang University, Seoul, Korea. His research interests include virtual sketch and WEB applications.

Jaeeung Lee is a professor in the mechanical engineering department of Chung-Ang University, Seoul, Korea. His research interests include noise filtering of spatial wands, vibration control. Lee has a Ph.D. in mechanical engineering from University of Michigan.

Youngho Chai is a professor in the graduate school of advanced imaging science of Chung-Ang University, Seoul, Korea. His research interests include spatial sketching, virtual collaborative design and computational geometries. Chai has a Ph.D. in mechanical engineering from Iowa State University.

Harksu Kim is a Ph.D. candidate in the graduate school of advanced imaging science of Chung-Ang University, Seoul, Korea. His research interests include object tracking, multiple view geometry and virtual sketch applications. Kim has a M.S. in computer engineering from Keon-Yang University.