Copy-paste Synthesis of 3D Geometry with Repetitive Patterns

Shigeru Owada[1], Frank Nielsen[1], and Takeo Igarashi[2]

[1] Sony Computer Science Laboratories, Inc.
Gotanda, Tokyo, Japan,
{sowd|Frank.Nielsen}@acm.org
[2] The University of Tokyo, Bunkyo-ku, Tokyo, Japan
takeo@acm.org

Abstract. We propose a new copy-paste user interface for 3D geometry based on repetitive patterns. The system, guided by the user, analyzes patterns of repetition in the source geometry and then pastes the geometry while increasing or decreasing the number of repetitions using scaling and deformation, which is controlled by two freehand strokes called *handles*. The system has two main advantages over existing methods: the entire copy-paste operation is controlled by the user's stroke input and thus can be specified easily without explicitly adjusting the parameters, and splitting the shape information into source geometry and handles can not only significantly reduce the amount of data required but also quickly change a scene's appearance while keeping its structure consistent.

1 Introduction

Shape modeling is one of the central challenges in contemporary computer graphics (CG) systems. Because image-generation (rendering) technology al-

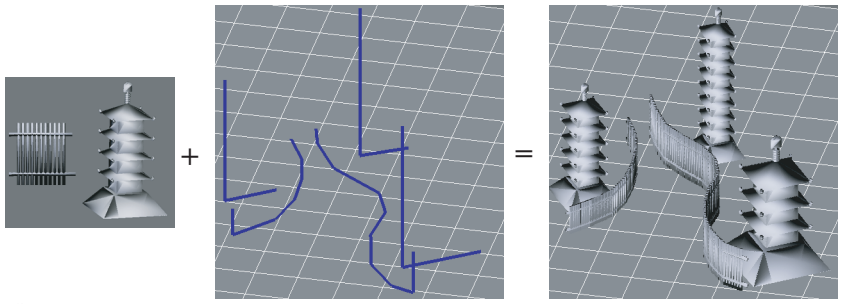

Fig. 1. Overview of the proposed copy-paste operation. Using handles, a shape can be dynamically deformed, and the number of times an element is repeated can be modified.

ready achieves excellent results, the main interest in the field is currently how to create shapes more quickly and cost-efficiently without loss of visual quality in the final output. The most time-consuming and costly processes in CG production are shape modeling and animation, because they require human input. Therefore, it is important to improve the user interface to reduce the time and cost required for CG development. The aim of this study was to enhance the process of generating stationary objects, and propose a new, more effective copy-paste tool to improve the user interface.

The copy-paste operation is one of the most important in computer-aided content creation. Its effective use can dramatically accelerate the process of shape creation, so commercial systems usually offer various types of copy-paste operation. However, most existing copy-paste functions require explicit numerical input, which is useful for regular operations but not intuitive for irregular operations such as copying with deformation.

Therefore, we propose the following copy-paste function:

- The source geometry is semi-automatically analyzed to find patterns of repetition, then split into constitutive elements.
- It is not necessary to store the entire geometry. If the original shape contains repeated elements, the system stores only a single copy of the element.
- When pasting the geometry, the user simply draws 2D strokes on the screen. These strokes are then converted to a 3D curve, which we call a *handle*. A handle represents the skeleton of the pasted geometry. The system automatically adjusts the number of times elements are repeated, and deforms the original geometry along the curve.

Some features of the copy-paste operation follow:

- The user can easily specify the information required through stroke inputs.
- The data are compressed because of the lack of repetition of constitutive elements.
- The shape acts like a *skin* on the handles, because a single set of handles can be associated with many different, stylized geometries.

2 Related work

Efficient copy-pasting is one of the key reasons that computers are so useful. Almost all commercial data-editing software supports copy-paste operations, including spreadsheets and text, image, and sound/music editors. In 3D graphics creation, excellent copy-paste skills and functions are critical for 3D CAD designers to create complex scenes quickly. Therefore, commercial 3D modeling software usually supports multiple types of copy-paste operation. For example, Lightwave 3D Ver.8 [1] supports mirror, symmetric, clone, helix, spin-it, and array copy (with the option of a grid-like or circular layout), as well as particle, path, and rail clone, and point-clone plus. However, these functions are

batched, low-level operations that require the user to carefully input parameters and select the region to be copied.

A great many articles have proposed methods for efficiently copy-pasting geometry. Biermann et al. proposed a cut-and-paste tool for subdividing surfaces [4]. An implicit surface tool is an extremely useful primitive for copy-pasting that smoothes the boundary in a natural way [18]. Funkhouser et al. extended Intelligent Scissors, originally designed for 2D images, to partition 3D models [19, 8], and developed a method to combine models by connecting two open boundaries. They introduced the technique in the context of example-based modeling, which proposes combining models in databases to generate new shapes. There are some other ways to combine models [24]. For certain shapes, repetition is key to performing intelligent copy-paste operations. Procedural modeling systems usually explicitly consider patterns of repeated elements [23, 2], and techniques for identifying such repetition in the input have been extensively explored in 2D image processing for the purpose of 3D reconstruction or image segmentation based on texture. [10, 13].

Texture synthesis a way to expand textural images without introducing visible seams, while keeping the appearance of the input. A variety of algorithms achieve this goal, including pixel- [7, 14], frequency-, [9, 6], and patch-based [15, 17] synthesis, and non-periodic tiling [21, 5]. The system proposed in this paper is very similar to texture synthesis in that copy-pasting a data set is another way of synthesizing textures. There are some existing systems that synthesize 3D geometry [16, 3], but these are essentially extensions of pixel-based texture synthesis to 3D volumes and are therefore far slower than the proposed copy-paste operation.

The user interface of the proposed system was inspired by so-called sketch-based modelers. The SKETCH system utilizes a set of 2D gestural operations to input 3D shapes without changing the viewpoint to explicitly input 3D coordinates [25]. The Teddy system extends the idea of using 2D gestural inputs to define 3D geometry [11] by applying a plausible assumption of the target shape to achieve a highly intuitive response to the 2D input. The proposed system basically adopts the user interface of the SKETCH method, because it is suitable for specifying straight lines or planar curves with the help of a ground plane.

3 The proposed system

3.1 Shape representation

The input is a point cloud. If the original data are for a polygonal mesh, they are converted into a point cloud by resampling [20].

3.2 User interface

The user interface is as follows:

Fig. 2. Deformation of the loaded model. Left: before. Right: after

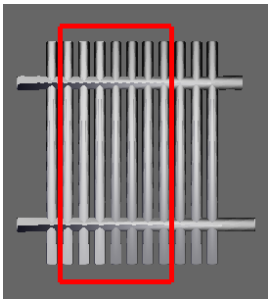

Fig. 3. Selection of region of interest

- Loading a model
  A shape model is loaded by dragging-and-dropping the source file into the main window. The system optionally finds the principal axes using principal component analysis (PCA), and rotates the model to align it with the x- or y-axis of the screen.
- Deformation of the model (optional)
  The user has the option to deform the model by drawing a single stroke on the central axis of the model, which is accomplished by pressing the left mouse button and dragging while holding down the Shift button on the keyboard (Fig. 2). The stroke is then straightened to deform the entire geometry.
- Selecting the region of interest
  The user selects the region of interest (ROI) by using the selection tool (Fig. 3) and dragging while pressing the left mouse button. In this operation (and successive operations to split the ROI into elements), all points that are rendered in the 2D region specified on the screen are considered inside the ROI.
  Once the ROI has been selected, a new window pops up that displays the selected region. This window is called the item window (Fig. 4).

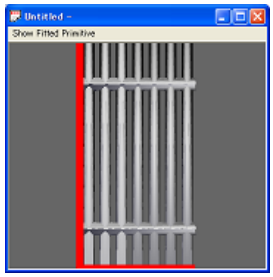

Fig. 4. Item window

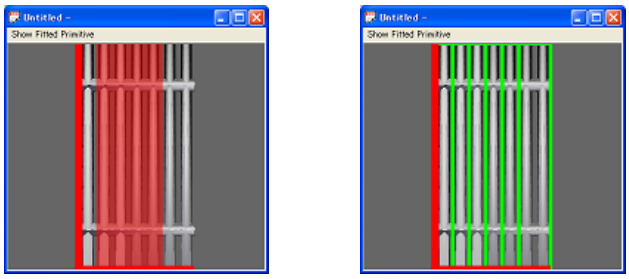

Fig. 5. Selection by the user (left) and identification of repeated elements suggested by the system (right)

- Specifying a region that includes repeated elements (optional)
  If the selected ROI contains repeated elements, the user can add a stroke to the item window to explicitly specify that the region contains repetitions (Fig. 5 left). After doing so, the system automatically identifies the repeated elements and splits the model accordingly, separating the repeated elements from their surroundings (Fig. 5 right). Because it is assumed that the number of repetitions is a natural number, the boundary is also improved to avoid overlapping elements.
  Note that this procedure is optional. If no region is designated as containing repetitions, the model is stretched when it is pasted.
- Pasting
  The user draws a handle in the target domain, and the 2D stroke is converted into 3D by projecting it onto the Z-buffer (Fig. 6 left). Then the user draws another stroke touching one of the two ends of the first stroke. This second stroke should be drawn in the vertical direction (Fig. 6 middle). The system again converts the stroke into 3D by projecting it onto the plane that is vertical to the floor and passes through the endpoint of the first stroke.
  The system measures the length of the 3D strokes, and pastes the geometry, the shape of which is deformed by the first stroke. The scale of the pasted geometry is determined by the length of the second stroke, and the number of repetitions is determined by the length of the first stroke (Fig. 6 right).

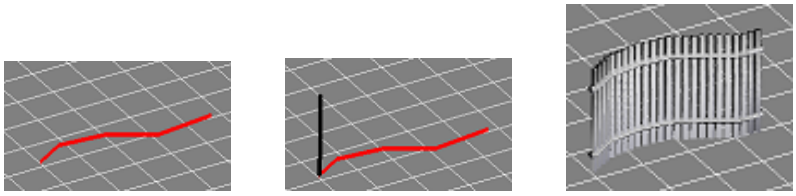

Fig. 6. Paste operation

The system stores the original geometry and the handles separately, so it is possible to apply different geometries to the same handles by importing a new geometry set file that includes the same repetitive elements as the original.

3.3 Implementation

Deformation by a single stroke Many algorithms are available for deforming geometry using user stroke input. We adopted the 2D 'as rigid as possible' (ARAP) technique [12]. When the user specifies the deformation stroke for the source project (Fig. 7 left), the system first determines whether the shape is aligned along the vertical or horizontal direction, using the difference between the x and y coordinates of the two endpoints of the input stroke. To be concise, the model is assumed to be aligned along the vertical direction (y axis). Then a coarse triangular mesh is constructed that covers the rendered object, using the Z-Buffer information. In our case, the grid size was 40 pixels. Then some vertices in the mesh are snapped to the stroke drawn by the user (See Fig. 7 center). These snapped vertices function as constraints, which are aligned along the vertical axis and retain the same distances between their points. Controlled by these constrained vertices, 2D ARAP deformation is applied to the mesh, and then each point is relocated using the deformation matrix of the triangle that contains the point.

The original 2D ARAP technique [12] consists of two steps: scale-free deformation and scale adjustment. However, applying only the first step returns a more intuitive result than applying both steps.

Item window and repetition analysis The selected ROI is displayed in the item window, and the user selects a region with repeated elements. The direction of the stroke is either vertical or horizontal, assuming that the structure is aligned along one of the axes. Once the region containing repeated elements has been specified, the frequency of the pattern is computed from the current Z-buffer. The algorithm is as follows, assuming that the stroke is drawn along the x axis:

– Each horizontal raster of the current Z-buffer is analyzed by convolving trigonometric functions at various frequencies. This convolution operation is the same as discrete Fourier transform (DFT) but differs in that the wavelength of the convolved trigonometric functions is not the pixel length to

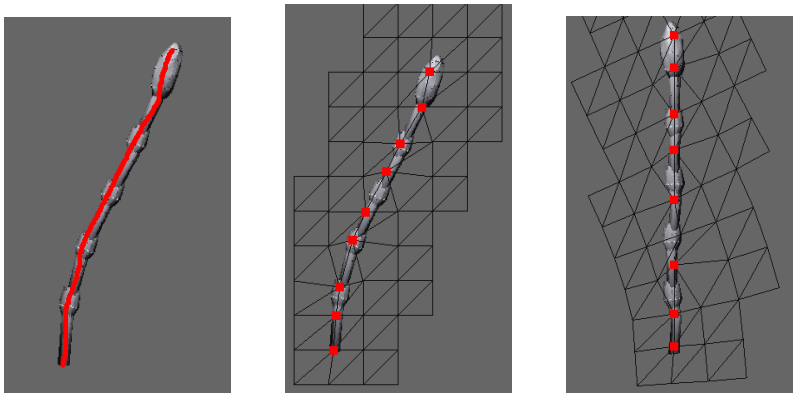

Fig. 7. Deformation by 2D ARAP mesh

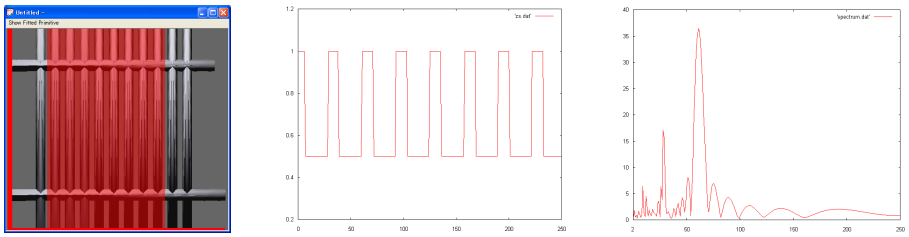

Fig. 8. Frequency analysis of the Z-buffer. Left: input. Center: sampled Z-buffer of one raster. Right: the spectrum of the raster

the power of two. The magnitude of the frequency component is measured for all bands, from the two-pixel wavelength to the half-image-width wavelength, using a step-size wavelength of one pixel (see Fig. 8). For example, the raster with trigonometric functions is convolved with wavelengths of two pixels, three pixels, four pixels,..., w/2 pixels, where w is the width of the ROI.

– Find the peak of the frequency histogram for all rasters.

Once the frequency of elements has been identified, the object is split by its wavelength. If the width of the object does not match the integer multiplication of the wavelength, the surplus points are removed from the region with repeated elements and added to a neighboring non-repetitive region.

Pasting with repetition and deformation The user draws two handles comprising one horizontal 3D curve and one straight, vertical line. This user interface is similar to the one proposed in the SKETCH system [25]. As explained in Section 3.2, the handles consist of two 3D curves. The vertical stroke defines the scale of the pasted model, and the horizontal stroke defines the number of repetitions and deformation.

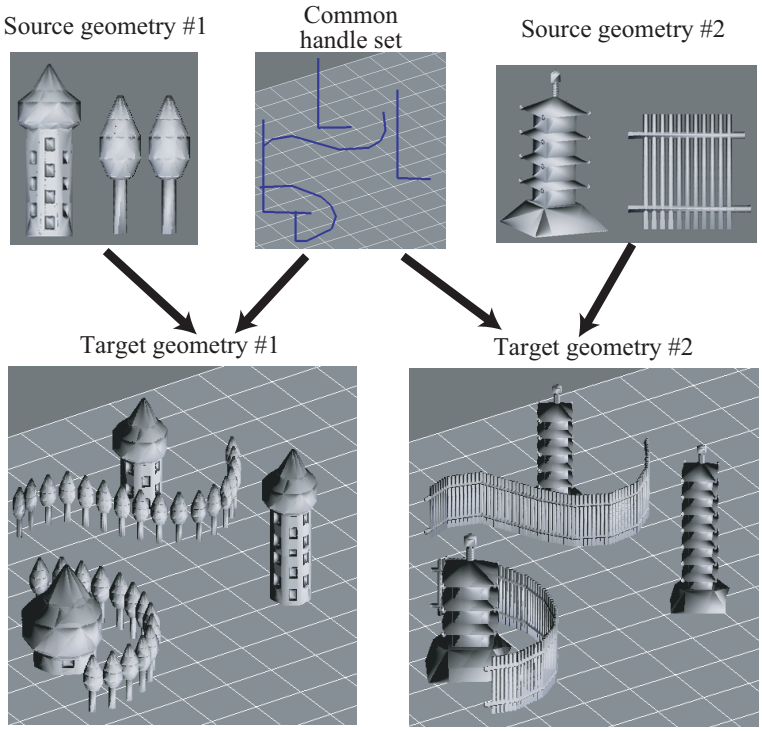

Fig. 9. Different source geometries with a common handle set

Again, deformation is performed using the 2D ARAP technique [12]. First, the number of repetitions in the source geometry is fixed and an un-deformed shape is created in the object space. Then, the boundary box of the object is calculated. The bottom face of the boundary box is tessellated by triangles to define the deformation field, and it is assumed that the un-deformed horizontal handle is defined as at the central line of the tessellated field. This line (or more precisely the vertices on this line) is used as the constraint of the 2D ARAP technique and deforms the tessellated bottom face to the target domain. All points in the original shape are then relocated according to the mesh, assuming that the barycentric coordinate of each point with respect to the enclosing triangle is maintained.

4 Results

The proposed system can create a model like that shown in Fig. 1 within a few minutes. One interesting feature is that different geometries can be applied to one set of handles (Fig. 9).

This procedure of splitting a source into basic shapes and handles has some interesting potential applications. One is the compression of data. Because han-

dles usually require very little data, a shape can be efficiently represented even if the target geometry contains many repetitions. For example, the handles in Fig. 9 amount to only about 1 kb of data. Another interesting application is skin geometry. Imagine if 3D desktops are developed in the future. We may want to use different desktop styles, each customized to some special purpose, but still need common functional support of the operating system. Splitting styles and skeletons should be strongly advantageous in such a case.

5 Conclusions and future work

We proposed a new copy-paste technique that is based on pattern analysis. A region containing repeated elements of the object is specified manually, but the frequency is analyzed automatically. Then, based on the analysis, the object is split into elements. We also introduced the notion of handles to ease the pasting operation. Handles specify the scaling factor, number of repetitions, and the deformation path using two 3D paths.

In future work, we hope to improve the model so that it can analyze the structure of repetition (related to the texture-based segmentation of 2D images) automatically, and be applied to scanned data or more generic shapes. This is a more challenging task, because general shapes may contain very complicated structures.

Another possible future direction would be to mix multiple input shapes to create an entirely new model, similar to texture mixture (texture fusion), which combines multiple textures to synthesize a single image [22]. The synthesized texture has an appearance that is intermediate to the input textures. This technique can be directly applied to geometry synthesis, but may require more elaborate processing, such as decomposition of shapes into textures, fine geometry, and base geometry. We believe that a technique that flexibly mixes input models would be a significant contribution to the field of example-based modeling.

References

1. Lightwave 3D. *NewTek http://www.newtek.com/.*
2. Daniel Bekins and Daniel G. Aliaga. Build-by-number: Rearranging the real world to visualize novel architectural spaces. In *IEEE Visualization*, page 19, 2005.
3. Pravin Bhat, Stephen Ingram, and Greg Turk. Geometric texture synthesis by example. In *Eurographics Symposium on Geometry Processing*, pages 43–46.
4. Henning Biermann, Ioana Martin, Fausto Bernardini, and Denis Zorin. Cut-and-paste editing of multiresolution surfaces. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 312–321, New York, NY, USA, 2002. ACM Press.
5. Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. *ACM Transtions on Graphics (SIGGRAPH 2003 Proceedings)*, 22(3):287–294, 2003.
6. Jean-Michel Dischler, Djamchid Ghazanfarpour, and R. Freydier. Anisotropic solid texture synthesis using orthogonal 2d views. *Comput. Graph. Forum*, 17(3):87–96, 1998.

7. Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the International Conference on Computer Vision-Volume 2*, page 1033. IEEE Computer Society, 1999.
8. Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. *ACM Trans. Graph.*, 23(3):652–663, 2004.
9. David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. ACM Press, 1995.
10. Ken ichi Kanatani and Tsai-Chia Chou. Shape from texture: general principle. *Artif. Intell.*, 38(1):1–48, 1989.
11. Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 409–416. ACM Press/Addison-Wesley Publishing Co., 1999.
12. Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141, 2005.
13. John Krumm and Steven Shafer. Texture segmentation and shape in the same image. In *The Fifth International Conference on Computer Vision*, pages 121–127, June 1995.
14. Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. *ACM Trans. Graph.*, 24(3):795–802, 2005.
15. Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics (Proc. Siggraph 2003)*, 22(3):277–286, 2003.
16. Ares Lagae, Olivier Dumont, and Philip Dutré. Geometry synthesis. In *Siggraph 2004 technical sketch*, 2004.
17. Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. *ACM Trans. Graph.*, 24(3):777–786, 2005.
18. Kevin T. McDonnell, Yu-Sung Chang, and Hong Qin. Digitalsculpture: a subdivision-based approach to interactive implicit surface modeling. *Graph. Models*, 67(4):347–369, 2005.
19. Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 191–198, New York, NY, USA, 1995. ACM Press.
20. Diego Nehab and Philip Shilane. Stratified point sampling of 3d models. In *Eurographics Symposium on Point-Based Graphics*, pages 49–56, June 2004.
21. Jos Stam. Aperiodic texture mapping. In *Technical Report R046, European Research Consortium for Informatics and Mathematics (ERCIM)*, 1997.
22. Li-Yi Wei. *Texture Synthesis by Fixed Neighborhood Searching. Ph.D. Thesis*. Stanford University, 2001.
23. Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. Instant architecture. *ACM Trans. Graph.*, 22(3):669–677, 2003.
24. Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.
25. Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: an interface for sketching 3d scenes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 163–170. ACM Press, 1996.