

Volume Editor

Hans Weghorn
Faculty of Mechatronics
BA-University of Cooperative Education, Stuttgart
Germany

Proceedings of the 4th Annual Meeting on
Information Technology and Computer Science – ITCS, Volume 2008
Stuttgart, Germany, February 2008
Hans Weghorn (Ed.)

Copyright © 2008
All rights reserved

Printed in Stuttgart, Germany

ISSN 1614-2519

Analysis of Texture Synthesis Algorithms with Respect to Usage for Hole-Filling in 3D Geometry

Monika Kochanowski, Philipp Jenke, Wolfgang Straßer

WSI/GRIS, University of Tübingen, Germany
KoMonika@gmx.net, {pjenke, wstrasser}@gris.uni-tuebingen.de

Abstract. The development of range scanning technology has offered new possibilities to capture large datasets in computer graphics. Due to physical restrictions, captured 3D objects can contain holes that need to be filled automatically. In a different area of interest, the field of texture synthesis, research has been done on a solution for a similar 2D problem. To utilize available methods, the challenge is to find similarities and show the differences between these problems. Hereby a 3D prototype is developed on the basis of an analysis of chosen texture synthesis algorithms, projecting the problem onto an intermediate 2D algorithm. The presented solution works fast and at good quality on artificial and real-world examples.

1 Introduction

Although the technology of new range scanning devices develops quickly, physical limitations will always constrain the quality of the collected data. Holes in objects can also be introduced in the post-processing steps, for example by manual editing or combination of different point sets. Many of the proposed 3D hole-filling algorithms use a surface estimation in the filling process. As this step is computationally expensive, it is reasonable to attempt an experimental approach without it.

Another interesting research field is texture synthesis, which offers many statistical similarity-based approaches working on reconstruction of a larger sample of a smaller input texture with the goal of *perceptual similarity* [13]. In the last years many different algorithms have been proposed to solve that problem.

Here, we develop a prototype for a simple yet efficient 3D hole-filling algorithm. Therefore existing texture synthesis algorithms are analyzed to identify solutions which are applicable for the 3D problem. The results are presented on real-world examples. Finally, an outlook and a conclusion are given.

2 Related work

In the well-known approach of context-based surface completion [15] a hierarchical algorithm is used by refining a surface estimation. The algorithm in [14] uses a digital signature based on curvature and texture information for the reconstruction process.

Another algorithm developed in [2] uses a voxel-based approach, trying to copy the most similar point for each region.

A different field of interest is *image inpainting*. A famous approach [3] combines inpainting techniques with a texture synthesis approach. A different work [4] uses an iterative adaptive approach to approximate unknown regions.

The field of main interest here is texture *synthesis*. The algorithms can be classified into three categories: pixel-based, patch-based, and advanced approaches. The basis for many of the later algorithms is the *pixel-based* approach of Efros and Leung [6]. The theoretical foundation is the Markov random field (MRF) model, which mathematically describes the probability distribution for one pixel p assuming locality, therefore only considering its neighborhood. In the reconstruction process, for each missing pixel p the input is searched for the possible best neighborhoods, which are then sampled to synthesize p . An enhancement is given in [19], where a multi-resolution approach is used to improve speed and quality especially for using large pixel neighborhoods. In [1] an algorithm for synthesizing natural textures is proposed. The idea is to add information about structure in a second input picture. In [7], image analogies are used to describe the relationship of two input images to a different set of output images. All pixel-based algorithms have in common that one pixel is synthesized in one algorithm step.

In the *patch-based* approach image quilting [5], overlapping texture patches are copied before the best cut through the boundary of the overlapping region is found by dynamic programming. A similar method is used in [9], whereby here graphcuts are applied. The hybrid texture synthesis algorithm [13] uses a patch-based approach starting with large patches and subdividing these when no good match can be found. The overlapping parts are repaired pixel-based. The adoption in [12] accelerates this method by using only large patches and a pre-computed search structure in the pixel repair step.

Surveyed *advanced* algorithms contain texture optimization [8], vector field visualization [16], feature matching and deformation [20], order-independent texture synthesis [18], parallel controllable texture synthesis [11], and appearance-space texture synthesis [10].

The chosen algorithm, which serves as a basis here, has to be *general, simple, adaptable, and perform acceptably*. Therefore, neither the pixel-based approaches are used, which are inherently slow because of the neighborhood search step, nor the advanced algorithms are considered, which use complicated interacting methods to improve the search result. The patch-based algorithm hybrid texture synthesis [13] offers good results with a relatively simple approach and looks promising for 3D adoption.

3 Hole-filling algorithm

To ensure applicability, a 2D experimental prototype is developed as the basis for the 3D hole-filling algorithm using the main ideas of hybrid texture synthesis [13]. The splitting of the patches with the overlap strategy has shown a performance problem in some hole-filling applications, when for example a 16 x 16 pixel patch is subdivided in 1 x 1 pixels using an overlap of 2, a total of 1280 pixel reconstructions have to be done to reconstruct

256 pixels. The large patches introduce two problems: First, the quality of the best match decreases during the filling process. Secondly, one bad guess distorts the complete picture. To solve the first problem, the patches are chosen based on their generation, as older patches contain more original and less reconstructed information. Among these the patch with the highest variance is used, propagating smooth regions slower. The overlapping patch part is corrected pixel-based. To solve the second problem, a multi-resolution pyramid is created to initialize the hole pixels well. Two results of the algorithm for example images are presented in fig. 1: Linear structure is visible on the right example that is propagated well, and the natural setting in the left picture is filled nicely.



Fig. 1. Results of the 2D prototype for example images from [3] (left) and [4]

To adapt the algorithm to work on 3D objects, the new main step is *patch creation*. Using 3D points as input data, the problem has to be projected down to 2D. In a preprocessing step, the considered region of a possibly large data set can be manually selected. A certain portion is selected randomly as basis for patches. The patch is then considered to be in a sphere around the center in radius r . A principal component analysis (PCA) is employed to find the three main axes describing the chosen points. Therefore, a patch is considered as a plane, and the third dimension is the height over the patch plane. In figure 2 an example of two patches with their respective coordinate axis is shown.

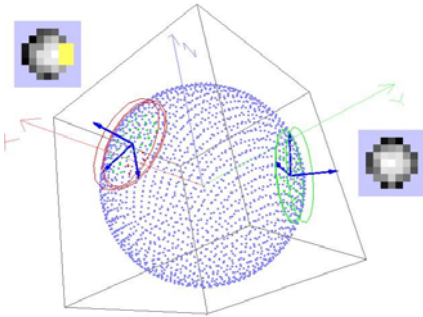


Fig. 2. Example sphere with two marked patches

The gray value of the 2D representations has been estimated by the distribution of height values over the plane: white is highest, black lowest. Squares of the patch where no points fit are considered as broken, marked as red points in the 3D picture, shown yellow in the 2D height field.

Further on, the algorithm processes patches in the same manner as the 2D experimental prototype (see figure 3): First, for the to-be-repaired broken patch, the best-matching complete patch is chosen. The missing points in the 3D data set are created using the height information from the complete patch.



Fig. 3. Completion of a broken patch, new points marked pink

In an optional pixel-based correction step, the pasted points can be smoothed to better fit into the given shape. After the patch has been fixed, all patches which can be affected by the newly pasted points need to be updated. Further on, the set is extended by using some of the pasted points as new starting points for patches in order to fill holes, which might be hit randomly only by few patches, completely.

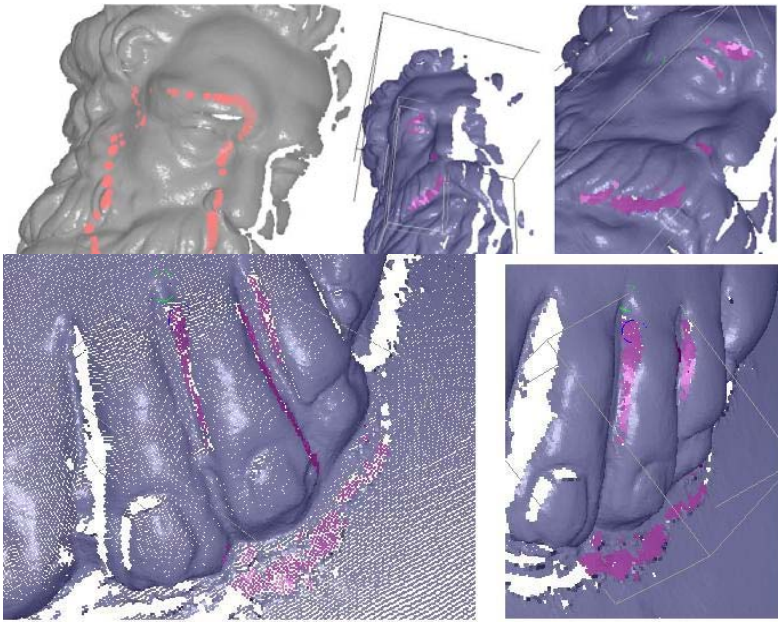


Fig. 4. Results of the 3D hole-filling algorithm (input data from [21])

4 Results

After some artificial examples including a cube and a bumpy sphere, a complete range scan of a Neptune statue [21] is chosen for reconstruction. One can see in figure 4 that

most of the marked area is reconstructed well. The small white holes in the face reconstruction have not been captured by the identification of broken patches. In the foot reconstruction the distribution of the reconstructed points is similarly dense as the given input data. One can see that most holes have been filled nicely.

The short reconstruction times underneath one minute for all examples show the good performance of the algorithm. The reason therefore is the restriction of the sample set to a well-defined region, the random sampling procedure which uses only a portion of the available points as patch, and the 2D height field, which does not store 3D information. The implemented prototype has been embedded in an existing library [17] and therefore is integrated with various existing functionality.

Areas of improvement are a sharp-edge problem, as these patches are sometimes considered as broken. This could be improved by using an additional density metric for bad patch identification. Further on, an overlap strategy for smoother integration could be used as in the 2D prototype. Finally, the pixel correction step could be performance enhanced.

5 Conclusion

In this paper a 3D hole-filling prototype has been developed on the basis of a survey of existing texture synthesis techniques. The results presented show that the simple algorithm is applicable for a variety of settings and offers good performance considering the size of the input data.

References

1. Ashikhmin, M.: Synthesizing natural textures. In: *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, New York, USA, ACM Press (2001) 217–226
2. Berner, A.: *Ergänzung dreidimensionaler Objekte durch Selbstähnlichkeit*. Master's thesis, WSI/GRIS University of Tübingen (2007)
3. Criminisi, A., Perez, P., Toyama, K.: Object removal by exemplar-based inpainting. In: *CVPR '03: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 02. Los Alamitos, CA, USA, IEEE Computer Society (2003) 721–728
4. Drori, I., Cohen-Or, D., Yeshurun, H.: Fragment-based image completion. In: *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, New York, USA, ACM Press (2003) 303–312
5. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, New York, USA, ACM Press (2001) 341–346
6. Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: *ICCV'99: Seventh International Conference on Computer Vision*. Volume 02., Los Alamitos, CA, USA, IEEE Computer Society (1999) 1033–1038
7. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, New York, USA, ACM Press (2001) 327–340

8. Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture optimization for example-based synthesis. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, New York, USA, ACM Press (2005) 795–802
9. Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: image and video synthesis using graph cuts. In: SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, New York, USA, ACM Press (2003) 277–286
10. Lefebvre, S., Hoppe, H.: Appearance-space texture synthesis. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers, New York, USA, ACM Press (2006) 541–548
11. Lefebvre, S., Hoppe, H.: Parallel controllable texture synthesis. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, New York, USA, ACM Press (2005) 777–786
12. Nealen, A., Alexa, M.: Fast and high quality overlap repair for patchbased texture synthesis. In: CGI '04: Proceedings of the Computer Graphics International (CGI'04), Washington DC, USA, IEEE Computer Society (2004) 582–585
13. Nealen, A., Alexa, M.: Hybrid texture synthesis. In: EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering, Aire-la-Ville, Switzerland, Eurographics Association (2003) 97–105
14. Park, S., Guo, X., Shin, H., Qin, H.: Surface completion for shape and appearance. *Vis. Comput.* 22 (2006) 168–180
15. Sharf, A., Alexa, M., Cohen-Or, D.: Context-based surface completion. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, New York, USA, ACM Press (2004) 878–887
16. Taponecco, F., Alexa, M.: Vector field visualization using markov random field texture synthesis. In: VISSYM '03: Proceedings of the symposium on Data visualisation 2003, Aire-la-Ville, Switzerland, Eurographics Association (2003) 195–202
17. Wand, M., Berner, A., Bokeloh, M., Fleck, A., Hoffmann, M., Jenke, P., Maier, B., Staneker, D., Schilling, A.: Interactive Editing of Large Point Clouds. In: PGB '07: Proceedings Symposium on Point-Based Graphics (2007)
18. Wei, L.Y., Levoy, M.: Order-independent texture synthesis. Technical Report 01, Computer Science Department, Stanford University (2002)
19. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, New York, USA, ACM Press/Addison-Wesley Publishing Co. (2000) 479–488
20. Wu, Q., Yu, Y.: Feature matching and deformation for texture synthesis. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, New York, USA, ACM Press (2004) 364–367
21. AIM@SHAPE Shape Repository V4.0. <http://shapes.aimatshape.net/> (date: 03/29/2007)