3D Object Repair Using 2D Algorithms

Pavlos Stavrou[1], Pavlos Mavridis[1], Georgios Papaioannou[2], Georgios Passalis[1] and Theoharis Theoharis[1]

[1] National Kapodistrian University of Athens , Department of Informatics ,
{ p.stavrou , grad0751, passalis, theotheo }@di.uoa.gr
http://graphics.di.uoa.gr
[2] Athens University of Business and Economics, Department of Informatics
gepap@aueb.gr
http://www.aueb.gr/users/gepap/

Abstract. A number of three-dimensional algorithms have been proposed to solve the problem of patching surfaces to rectify and extrapolate missing information due to model problems or bad geometry visibility during data capture. On the other hand, a number of similar yet more simple and robust techniques apply to 2D image data and are used for texture restoration. In this paper we make an attempt to bring these two-dimensional techniques to the 3D domain due to their obvious advantage of simplicity and controllability. Creating a depth image with the help of a voxelisation algorithm will allow us to apply a variety of image repair algorithms in order to mend a 3D object. The use of three variations of the texture synthesis algorithm is investigated. Constrained texture synthesis and its variations using the Haar wavelet and image decomposition methods are also proposed in order to preserve patterns appearing on the object while trying to maintain its geometry intact.

1 Introduction

The repair of incomplete three-dimensional objects is a common problem in computer graphics. We subjectively regard a 3D model as incomplete when it exhibits missing surfaces and patterns. Incomplete three-dimensional objects may result from a 3D scanning procedure or any other three-dimensional data acquisition method or may be models of actually damaged real objects, in which case we seek to rectify their original shape. Data corruption or bad modelling are two other problems where shape repair may be applied to fix the model geometry.

In cases, where detail preservation is important, a more general and efficient method is required which simultaneously preserves the geometry of the object while detecting and reproducing patterns appearing on its surface. We, therefore, propose the use of 2D algorithms on 2D mapping of 3D objects to enhance performance and retain patterns and geometry.

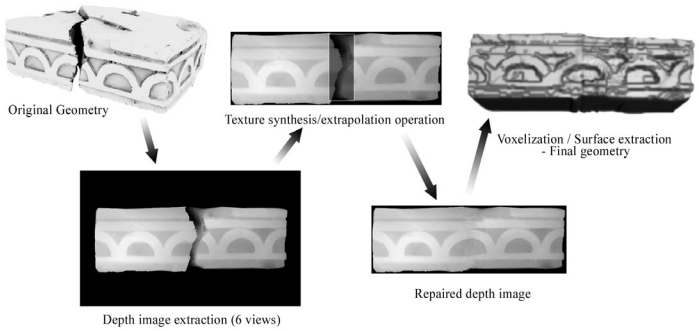


Fig 1.1 . Procedure of a 3D object repair using a 2D algorithm

Given an incomplete 3D object, a method for its repair based on two-dimensional algorithms is introduced in this paper. In order to apply 2D algorithms for the repair of a 3D object we need to represent the object in a two-dimensional form, which is capable of preserving gradient information for both high and low spatial frequency bands. We extract depth (Z-range) images for this task and then apply algorithms for image repair to them. Subsequently the depth information is transformed back into three-dimensional data via voxelisation [10, 12]. The final volumetric model represents the repaired 3D object (Fig 1.1).

1.1. Related Work in Two Dimensions

A variety of algorithms have been proposed for image repair. The *in-painting* algorithm, presented by Bertalmio et al. [1], is probably the best known. Given an initial image, which contains areas where information is corrupt, the goal of this algorithm is to produce an image that contains a fair approach of the missing information. Bertalmio et al. [2] suggests an improved method that simultaneously deals with both structure and texture. The initial image is decomposed into two coefficients for each pixel, the first representing the structure of the image and the second storing high detail texture information. The in-painting algorithm is then applied to the structure coefficient and texture synthesis is applied to the texture coefficient before the image is composed back together. Texture synthesis [3] is a common method for image reconstruction and is based on filling the missing pixels of the corrupt image by searching for similar patterns on the image.

1.2 Related Work in Three Dimensions

In the three-dimensional case, the proposed solutions mainly rely on two different approaches, one being hole filling and the other being surface blending.

The most popular hole-filling techniques use the implicit surface approach [4] to represent objects that are to be repaired. A local approximation of the surface is performed by fitting a function that best describes the geometry, including the missing

areas. One way to achieve this is through a sum of *Radial Basis Functions* (RBF). Once a RBF is found, the surface is automatically completed. Davis at al. [5] introduced volumetric diffusion for hole filling. Their algorithm first constructs a volumetric signed distance function around the surface samples. Then an iterative Gaussian convolution propagates adjacent distance values to fill the holes. Verdera et al. [6] propose the use of the in-painting method mentioned above to fill holes in a mesh by first representing the surface of interest in implicit form.

Surface blending techniques aim to finding a blend surface that smoothly connects two or more surfaces. One of the most interesting approaches for surface blending is the combined subdivision schemes method by Levin [7]. This particular approach allows boundary constraints to be taken into account during the generation of the blend surface points. High quality blend surfaces can be constructed using variational subdivision schemes [8]. Such surfaces minimize an energy metric of the surface relying on its curvature. Application of this method to triangular surfaces is called Discrete Fairing [9].

2 Method Overview

As mentioned in the introduction, we have opted to work with two-dimensional algorithms to repair three-dimensional objects. In order to work with such algorithms a two-dimensional representation of an object is required. The use of depth images of three-dimensional objects is adopted in this paper. The depth image creation method we selected to use is based on the setup of the voxelisation method described by Karabassi et al. [10], acquiring depth images from the 6 *z*-buffers, a pair for each of the three orthocanonical directions. Depth images encode geometric detail as intensity values, allowing a fast implementation of the selected two-dimensional algorithms. In order for the repair method to work correctly, we must acquire at least six depth images, which correspond to the faces of the bounding cube, and the object must be aligned so as to maximally expose the missing part.

After the acquisition of the depth image, the constrained texture synthesis algorithm is used for the repair of the object. We have experimented with three different variations of the method:

- Constrained texture synthesis
- Constrained texture synthesis using Haar wavelets
- Constrained texture synthesis using image decomposition

In the first and simplest variant, the algorithm is directly applied to the raw depth-image values without any pre-processing. In the second variant, the Haar wavelet decomposition [11] was applied to the depth image prior to using the texture synthesis algorithm. Texture synthesis is then performed separately on selected bands of the resulting image. The reconstructed depth image is produced by the inverse Haar wavelet transform of the synthesised sub-images. Finally, in the third variant, image decomposition transform is applied to the depth image in advance, thus acquiring the *U* and *V* coefficients of the image, where the *U* coefficient corresponds to the structure of the image and the *V* coefficient corresponds to the texture. Texture synthesis is applied to the *V* coefficient and an interpolation method to the *U*

coefficient before the image is reconstructed using the inverse image decomposition transform. Detailed descriptions of the object are presented in section 3 of the paper.

Having obtained the restored depth-images, the repaired three-dimensional object is calculated by voxelisation according to the range information stored in them, using the hardware-based voxelisation of Passalis et al. [12].

3 Texture Synthesis for Image Repair

The first algorithm proposed for texture synthesis is *texture synthesis by non-parametric sampling* by Efros and Leung [3]. This algorithm composes one pixel at a time. For each pixel *p*, it extracts its neighbourhood and searches in the texture sample for a similar pattern. The texture sample is defined as a part of the initial image or the whole image itself and the neighbourhood of a pixel is defined as a square window, which contains all the known neighbouring pixels of *p*. The search is exhaustive as all neighbourhoods in the image are compared and a distance factor is calculated. Next, the algorithm randomly selects one of the minimum-distance neighbourhoods and copies the corresponding pixel into pixel *p*. The distance factor uses a number of metrics. One of the most popular is the *mean square distance* metric:

$$d = \frac{1}{N}\sum \left\| x_{ij} - y_{ij} \right\|^2 \quad \textbf{(1)}$$

where x_{ij} (y_{ij}) is the value of the pixel at position (i,j), in neighbourhood x(y) and N is the number of (i,j) positions in which both pixels x_{ij} and y_{ij} are not missing from the image.

A further improvement of the mean square distance metric is the weighted mean square distance metric, which weights the differences in Eq. 1 according to the distance of the elements from the central pixel. Therefore, pixels that are closest to the centre have greater contribution than the ones further away. As a weighting function we use the Gaussian distribution function. This is the distance metric adopted in this paper.

3.1 Constrained Texture Synthesis

Constrained texture synthesis is an improvement to the texture synthesis method by non-parametric sampling. Given the sample of a texture, which contains gaps, we attempt to fill these gaps in such a manner that the produced texture and resulting image do not exhibit any discontinuities, especially when dealing with well-defined shapes as patterns. The main idea is to apply texture synthesis as above, introducing a limit to the values that certain pixels may acquire. The algorithm processes the whole texture sample and composes only the missing pixels. For better results, the pixel with the largest number of neighbours is composed first. In this paper, a new constraint is introduced, which permits only tileable new textures to be produced. This means that a uniform texture without discontinuities can be formed by repetition of the

reconstructed image. Instead of repeating a square-window pattern search at the current composed pixel, a symmetric search around the current location is performed up to a pre-defined distance in order to detect and bridge textural details across the gap. Results of using this method are presented in figure 3.1.

3.2 Constrained Texture Synthesis Using Haar Wavelets

Let *H* be the Harr wavelet transform [11]. For four adjacent pixels in the image aligned as $[\mathbf{L},\mathbf{R}]:\mathbf{L}=[a,c]^T,\mathbf{R}=[b,d]^T$, a vector $[a,b,c,d]^T$ is constructed and is multiplied with matrix *H* to produce a vector $[ll,lh,hl,hh]^T$. Tilling and applying this square window across the image produces four bands (images) LL, LH, HL and HH each of which has the resolution of one quarter of the original image. This is one step of the Haar wavelet transform. We can choose to continue applying the transform recursively either only to the LL band or to all resulting bands (Fig 3.2a).

The HH, HL and LH bands of the image now only contain repeated patterns of the image and are free of smooth (slow) intensity transitions where the texture synthesis algorithm would fail. The LL band works as a box filter and contains only the low frequencies of the image. Increasing the number of steps of the transform will result in the LL band containing very few details of the original image and only the color gradient will appear, allowing us to use a simple interpolation method for the missing pixels. We apply the constrained texture synthesis algorithm to the other three bands to fill the missing textures.

Finally we apply the inverse Haar wavelet transform to the synthesized image to acquire the reconstructed depth image (Fig 3.2b).

3.3 Constrained Texture Synthesis Using Image Decomposition

The image decomposition technique allows us to decompose the original image (*f*) into two coefficients. The two coefficients represent different attributes of the original image. The *v* coefficient holds the noise and object texture of the image, while the *u* coefficient represents the shape and large intensity variations, which correspond to the main geometric shape in our case (Fig 3.3). Combined, the two coefficients give the original image:

$$f = u + v \tag{2}$$

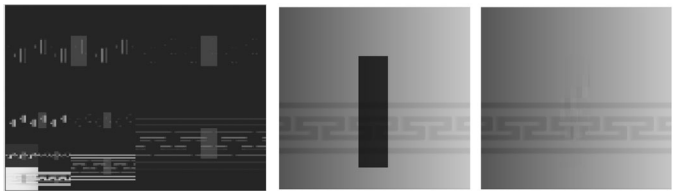
Rubin, Osher and Fatemi proposed [13] the minimization of the following equation for the construction of the *u* and *v* coefficients, where λ is a free parameter which corresponds to the desired coherency between the input image and the *u* image.

$$\inf_u F(u) = \int \|\nabla u\| + \lambda \int \|f - u\|^2 \, dxdy \tag{3}$$

Vese et al. [14] proposed an improvement of the above model, which achieves better separation of texture and noise:

(a) (b) (c)

Fig 3.1. Constrained texture synthesis. (a) Input and repaired image. Black area represents the missing pixels. (b) Repair of an ancient artifact. One set of input and output depth images is shown. The object is voxelised from the rectified depth images. (c) Repair of a curved wall depth image. The gradient prevents the algorithm from detecting similar pixel regions.



(a) (b)

Fig 3.2. Constrained texture synthesis using Haar wavelets. (a) Three-step Haar wavelet transform of an image. (b) Image reconstruction using Haar wavelet method.

$$\inf_{u,g_2,g_2} F(u,g_1,g_2) = \int \|\nabla u\| + \lambda \int \left\| f - u - \partial_x g_1 - \partial_y g_2 \right\|^2 dxdy + \mu \int \|\nabla u\| \sqrt{g_1^{\,2} + g_2^{\,2}}\, dxdy \qquad (4)$$

where $g_1 = \frac{1}{2\lambda} \frac{u_x}{\|\nabla u\|}$ and $g_2 = \frac{1}{2\lambda} \frac{u_y}{\|\nabla u\|}$.

The minimization of equation (4) with respect to u , g_1 and g_2 results in an Euler-Lagrange differential equation system.

Having now decomposed the depth image into the structure (u) and texture (v) coefficients, we apply the constrained texture synthesis algorithm to the texture coefficient in order to fill out the missing detail information. As far as the structure coefficient is concerned, since it contains only slow variations, a simple interpolation method is adequate for the restoration of the missing information, provided the gaps are not disproportionately large compared to the object. For this task, we can use either the RBFs or the image in-painting method [4][2]. After the two coefficients have been repaired, the image is re-composed (Eq. 2) to form the restored depth information (Fig 3.3).

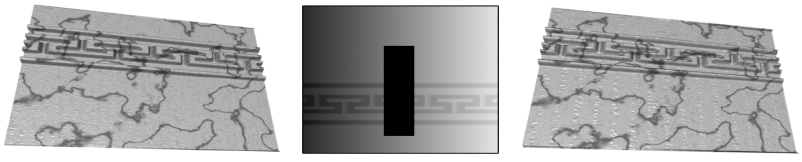
4 Results

The constrained texture synthesis algorithm produces perfect results when applied to images with repeated structural patterns, which conform to the Markov random fields model. However when applied to depth images (Fig 3.1b), the resulting image exhibits discontinuities at the depth values. These are of course translated as slope discontinuities on the reconstructed three-dimensional surface, after voxelisation. This is an inherent problem of the texture synthesis algorithm, which blindly copies pixel values to complete the missing information and disregards changes in the overall structure. In Fig 3.1c, where the depth image of a curved wall is shown, we observe another failure of the algorithm. Following a vertical path to complete missing pixels and following a horizontal path (both confront with the rule of the missing pixel with the most neighbours to be completed first) can produce different results due to the gradient direction of the image, which is a result of depth value variance. This issue during the testing of the method was addressed by simply using both approaches and selecting the most suitable one each time.

When applied to the same test case, the constrained texture synthesis using wavelets achieves far better results due to the isolation of the patterns in the high-frequency Haar coefficients. However, we can still observe a small amount of noise in the reconstructed image (Fig 3.3). Texture synthesis tends to perform worse in high levels of wavelet decomposition due to the decrease of the spatial resolution of the wavelet bands. At the same time, for the proper application of the interpolation algorithm to repair the LL, many iterations of the wavelet analysis are required. The problem of automatically selecting the level of the wavelet decomposition, so that both texture synthesis and interpolation methods work, needs further investigation.

Finally, the constrained texture synthesis method using image decomposition presents the best results so far, as the texture of the *v* coefficient is properly synthesized and the patterns are correctly pieced together. Slight discontinuities appear when repairing the *u* coefficient using our interpolation, indicating that a more accurate interpolation method is needed to produce more consistent results.

It must be noted that in all cases the produced results were symmetrically correct to the original object. Patterns exhibited on the input object were coherently reproduced, during the repair procedure, to form an object whose texture was smoothly reconstructed (Figures 3.1b, 3.2b, 3.3).



(a) (b) (c)

Fig. 3.3. Constrained depth image synthesis using image decomposition. (a) The original 3D surface. (b) A damaged depth image of (a). (c) The final reconstructed surface after the voxelisation of the repaired and re-composed depth images.

5 Conclusion and Future Work

We presented a method that uses texture synthesis on depth images to repair a 3D object. We utilized three methods of texture synthesis which worked well on specific cases. None of the three produced very high quality results for all cases of 3D objects. However, texture synthesis results were impressive, considering their simplicity and speed of execution. The combined use of these methods, based on the characteristics of each object, will greatly increase the efficiency of 3D object repair using 2D algorithms. In addition, one should consider extensions of the presented algorithms, such as more accurate interpolation techniques, to increase the accuracy of the results.

References

1. Marcelo Bertalmio, Guillermo Sapiro, Vicent Caselles, and Coloma Ballester. Image inpainting. In Kurt Akeley, editor, *SIGGRAPH 2000, Computer Graphics Proceedings*, pages 417.424. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.
2. M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. UCLA CAM Report, 02(47), 2002.
3. Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *ICCV (2)*, pages 1033.1038, 1999.
4. Jonathan C. Carr, Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. Reconstruction and representation of 3D objects with radial basis functions. *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 67.76. ACM Press / ACM SIGGRAPH, 2001.
5. Davis, J., Marschner, S. R., Garr, M., and LevoyY, M. 2002. Filling holes in complex surfaces using volumetric diffusion. In Proceedings of the 1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT-02), IEEE Computer Society, Los Alamitos, CA, G. M. Cortelazzo and C. Guerra, Eds., 428.438.
6. Verdera, J., Caselles, V., Bertalmio, M., and Sapiro, G. 2003. Inpainting surface holes. In 2003 International Conference on Image Processing 2003 ICIP.
7. Adi Levin. Combined subdivision schemes - an introduction. In *SIGGRAPH course notes*, Subdivision for Modeling and Animation, pages 129.132, 2000.
8. Leif Kobbelt. Variational subdivision schemes. In *SIGGRAPH course notes*, Subdivision for Modeling and Animation, pages 159.164, 2000.
9. Leif Kobbelt. Discrete fairing and variational subdivision for freeform surface design. The Visual Computer, 16(3-4):142.158, 2000.
10. Karabassi E.A., Papaioannou G., and T. Theoharis. A fast depth-buffer-based voxelisation algorithm. ACM Journal of Graphics Tools, pages 5.10, 1999.
11. E. Stollnitz, T. DeRose and D. Salesin. Wavelets for Computer Graphics : Theory and Applications. Morgan Kaufmann Publishers Inc., 1996
12. G. Passalis, I. A. Kakadiaris, and T. Theoharis. Efficient Hardware Voxelisation. In *Proceedings of Computer Graphics International*, pp. 374-377, Crete, Greece, June 2004.
13. L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. Phys.~D, 60 (1992), pp. 259--268.
14. L. Vese and S. Osher. Modeling textures with total variation minimization and oscillating patterns in image processing, 2002. UCLA CAM Report, 02(20), 2002.