Published in Proceedings EUPSICO (European Signal Processing Conference) 1073-1077, 1998
which should be used for any reference to this work

MODELING 3D TEXTURED OBJECTS BY FUSION OF MULTIPLE VIEWS

Timothée Jost, Christian Schütz, Heinz Hügli
Institute of Microtechnology, University of Neuchâtel,
Breguet 2, Neuchâtel 2000, SWITZERLAND
Tel: +41 32 7183459; fax: +41 32 7183402
e-mail: timothee.jost@imt.unine.ch

ABSTRACT

For modeling 3D objects, the object geometry is often not sufficient. Better appearance can be obtained by texture mapping which efficiently combines good appearance resolution with small geometric complexity. This paper presents a way to create such texture mapped 3D models of free-form objects, by integrating textured views obtained from range images. The three main steps consist of the following: first, texture-mapped views of the object are obtained from a 3D scanner. Then, the different views are registered, considering geometric and color information of overlapping grids. Finally, meshes can be fused by using a new overlap-removing method that keeps most of the existing triangulation and texture mapping unchanged.

1 INTRODUCTION

Object modeling is essential for the building of virtual worlds. Traditionally, creating a 3D model of an existing object relies on using standard CAD tools or manual pointing devices. This can be really hard to achieve, especially for objects of arbitrary shape. Therefore, automatic modeling methods, which operate on data obtained from range images or simple (video) camera images, have been put forward in the past few years. This paper focuses on data from range finders which can provide fast, high quality scans of an object view at low cost. Several views must be combined to describe the whole surface of the object. This reconstruction provides a geometrical model of the object.

3D models won't look realistic if only their geometrical aspect is taken into account; the appearance of an object is also important. Realistic appearance is obtained by assigning either a color or a subimage to each model mesh, leading to a colored respectively a texture-mapped object model. Of these two models, the texture-mapped model is preferred because it permits high appearance resolution also in presence of a simple, coarse geometric grid.

The automatic construction of textured 3D models from real objects has been investigated under several aspects. [3] describes a reconstruction method from camera (video) images in which the texture is directly created from the images and applied to the model, after the geometrical reconstruction. A disadvantage of this method is that some parts of the object don't appear on any silhouette and, consequently, can't be measured.

For the reconstruction from multiple range images, a proposed solution is to build a high resolution colored model and to simplify it by compression into a textured model, the texture map being created from the color of vertices [4] [5]. The major drawback in this case is the need to use high resolution meshes during all the reconstruction process. It greatly increases computation time. Finally, [2] proposes a method for registration and integration of textured 3-D data. It first builds a purely geometric model and then creates a texture map by weighted averaging of the overlapping texture images from the original data sets.

This paper presents a new reconstruction method from range images that applies to textured views. Advantages compared to other approaches are the capability to handle textured views and the lower complexity of object reconstruction. In the following, we describe main elements of this reconstruction method that extends the view integration method for pure geometric modeling presented in [7]. The model is created iteratively, each new view being added to the partial model until completion. The process can be divided in three steps: view acquisition, registration of the new view with the partial model, and fusion of both meshes. The exposed extension concerns two parts: first, the acquisition process must be changed to create textured views. Second, texture integration must be implemented into the mesh fusion step.

2 VIEW ACQUISITION

We consider data from 3D scanners that provide a color image in exact correspondence to the points of the range image. For example a stripe pattern based range finder with a color CCD camera. From this "colored range image", a triangle mesh representing the surface is built, based on a sub sampling of the range image that keeps neighborhood relations. Because of the direct correspondence between range and color images, texture is created by mapping the color image on the 3D mesh.

Note that an important aspect of this modeling method concerns the illumination of the object. Basically, illumination of the object must be the same during the acquisition of all of the views. The problem is that the angle of illumination of one point of the object changes when moving it between two acquisition, creating intensity differences between views. A first approach to solve it is to scan the color image using the acquisition projector as light source and to compensate for varying illumination depending on the surface orientation. This approach is not

easy to follow for two main reasons: first, specularity and diffusion factors are different for each material and second, light source reflections (specularity) tend to "erase" diffusion, letting no information on the object color left.

The way we choose is to create a nearly constant, diffuse illumination all over the visible part of the object. To achieve it, we place several white light sources with diffusers all over the scanning area.

3 REGISTRATION

During the registration process, a new textured view is aligned with the partial model of the object. This is done first manually, by the user who roughly estimates the alignment, then automatically, by an algorithm called ICP (iterative closest point) [1], which iteratively improves the alignment by minimization of the distance between the common parts of the surfaces.

Compared to a classic ICP matching process, where only vertex-to-vertex links are made, the algorithm used here is modified in order to register textured views. First, it couples vertices from one surface with points on the triangles of the other mesh. Second, it also uses surface orientation and color information to compute the matching error. These modifications result in a more accurate matching, especially when meshes contain a small number of vertices, which is the case of textured views. A detailed description of this multi-feature matching algorithm can be found in [6].

4 FUSION OF TEXTURED VIEWS

The basic principle of the fusion of two views is to remove the overlapping part of one of the views, then to fill the resulting gap with new triangles and map them with the texture of the removed part.

The originally proposed fusion algorithm [7] takes advantage of the closest point relationships, established during the registration process, to detect the overlapping surfaces. Therefore, removing overlapping triangles and detecting gap frontiers is straightforward. A modified algorithm, which keeps the advantages of the original one, is proposed here. It keeps most of the triangulation and textures of the basic meshes. Furthermore, texture integration also benefits from closest-point relationships.

4.1 Algorithm Overview

A short step-by-step description of the algorithm is presented below: P and X are the meshes to be fused. Practically, X is the partial model and P represents the view to add.

1)* *Overlap detection:* Thanks to closest-point relationships.

2) *New texture coordinates calculus:* The registration algorithm delivers, for the overlapping vertices of X, the closest points in the triangles of P. Each redundant vertex of X is then assigned the corresponding texture coordinates in the texture of P. These coordinates will then be used in the gap filling process (5) and for texture extension (6).

3)* *Overlap removal:* The redundant triangles of P are eliminated. No texture mapping changes are made at this point.

4)* *Gap frontier detection:* Frontier of the gap(s) are found for filling.

5)* *Gap(s) Filling:* The texture of the new gap triangles is taken from the overlapping texture of P.

6) *Texture extension:* Texture of P is extended on X from gap border.

7) *Texture filtering:* Texture boundaries between P and X are filtered.

8) *Texture image creation:* Elimination of useless texture and creation of new image.

4.2 Texture Processing

After filling the gaps between X and P, all gap triangles contain texture from P and all adjacent triangles of X are still textured with X image. All adjacent edges are defined as texture boundaries. At this point, these boundaries are much visible on the object model because of discontinuities:

1) Texture resolution (number of texels per surface unit) discontinuity, since the angle between surface normal and viewing direction is different from one view to another for the same part of the object (Fig. 1).

2) Texels color discontinuity, due to a difference of intensities between views (illumination) or a small shift introduced in the registration process.

Both discontinuities must be reduced to obtain a continuous, unified texture. Texture extension reduces the resolution discontinuity and texture filtering permits a smooth color transition between both textures.

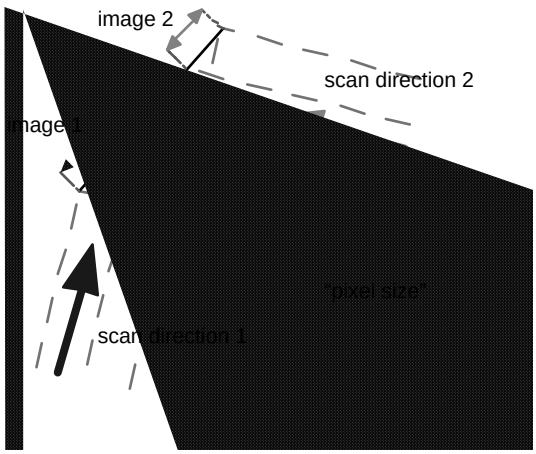


Figure 1. Scanning the surface S from different angles produces different texel sizes.

* for further explanations see [6]

4.2.1 Texture Extension

Around texture boundaries, X texture often has a lower resolution than the corresponding P texture. This comes from the fact that we keep all the X mesh and erode P mesh and resolution tends to be lower near the edges of a view, the angle between scanning direction and surface normal being bigger (Fig. 1). The resolution step between textures can be reduced by extending P texture onto X mesh, using the corresponding texture coordinates given in step 2 of the fusion algorithm. A simple algorithm iteratively moves the boundaries one row of triangles further into X mesh. T_i, the list of the next row of triangle, is first computed,

$$T_i = \left\{ t = \{v_0, v_1, v_2\} \middle| t \notin T_k \text{ and } \left| \{v_0, v_1, v_2\} \cap B_{i-1} \right| = 1 \text{ or } 2 \right\}$$

and B_i, the list of the new texture boundaries points, is obtained from T_i.

$$B_i = \left\{ v_j \middle| v_j \in t \text{ with } t \in T_i \text{ and } v_j \notin B_k \right\}$$

$$i = 1,..,N \qquad k = 0,..,i-1$$

For the initial conditions, T_0 contains the gap triangles and B_0 contains the gap border vertices on X.

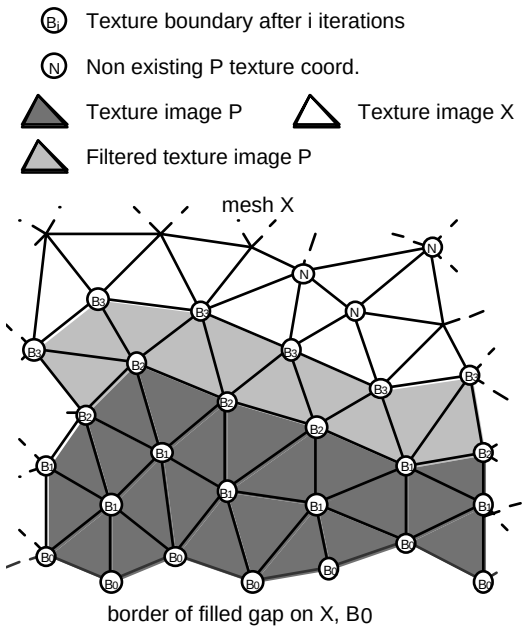


Figure 2. Extension of texture P on mesh X, from the gap border and filtering at boundaries

Practically, some vertices may not have a corresponding P texture coordinate. This has to be taken into account when extending texture. A simple test has been added in order to stop extension one step before reaching such a "bad" vertex, to permit texture filtering, which is done during the last iteration.

A small change can be applied to this algorithm to take texture resolution into account, extension being made only for triangles with a higher P resolution.

4.2.2 Texture Filtering

To insure a smooth color transition, the texture of the last row of triangle on the P side of the boundaries is filtered. This is achieved thank to a weighted averaging of the overlapping texture. The texture weight varies with the distance r between the considered pixel and the texture boundary.

Practically, the filter is applied to each triangle $t \in T_N$, during the last iteration of texture extension as shown in Figure 2.

So, for each pixel of texture of t we have:

$$C_p = \alpha(r) C_p + (1 - \alpha(r)) C_x \ , \quad \alpha(r) = \frac{r}{|r|}$$

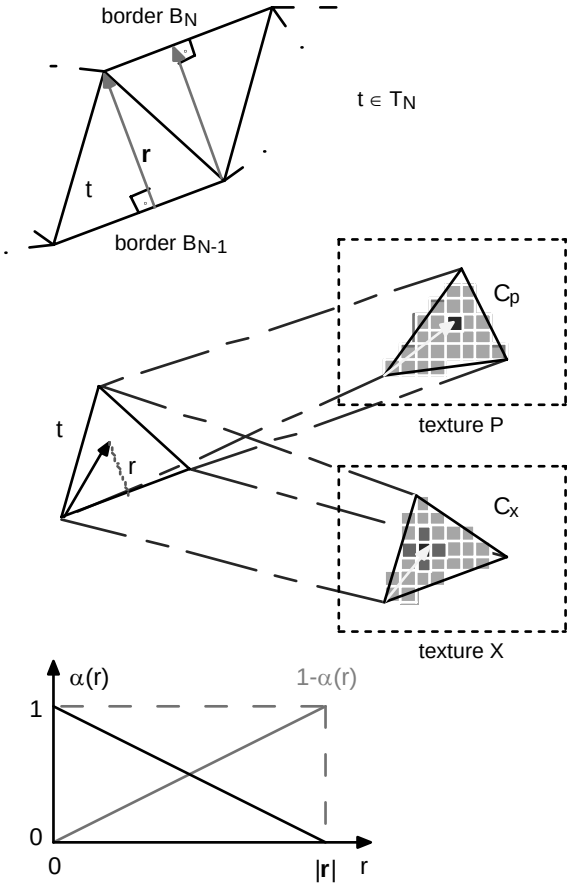


Figure 3. Texture filtering at boundaries

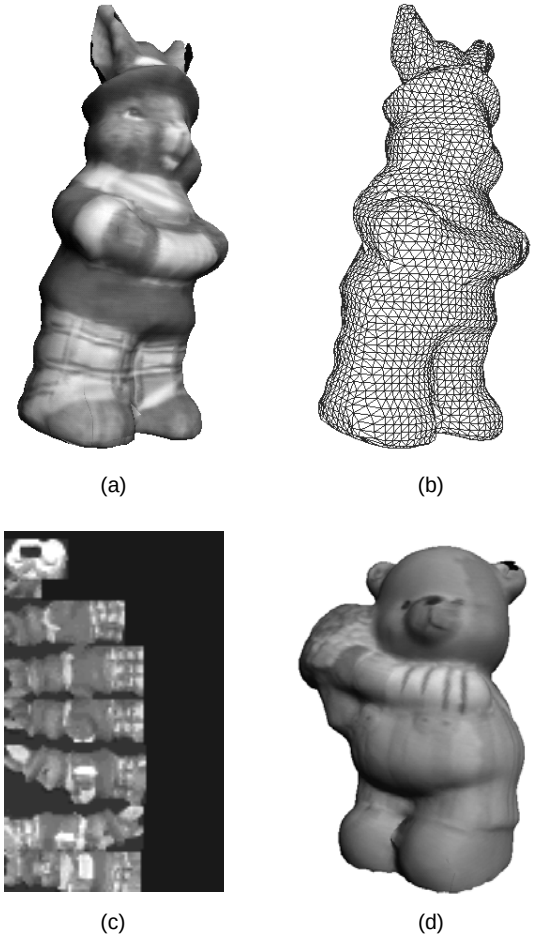
4.2.3 Creation and Reduction of Texture Image

The resulting texture image is built as follow: The texture images of each view are placed side by side in the global image, from left to right. These "subimages" are called texture patches. After adding a view to the incomplete model, overlap removing and texture extension leave some of the existing texture image useless. One can find the bounding box of useful texture in each patches, looking at the texture coordinates position in the images. In order to do that, the number, position and size of each patch are coded into the image. After that, the new texture image is created, keeping only the useful part of each patch and adding the patch from P image on the right. An example of final texture image is shown in Fig. 4c.

5 RESULTS

Figure 4 presents two different reconstructed models. The rabbit model is made of 8 views. It contains 4000 vertices and 8000 triangles. Both textured and wire frame representation are shown. The equivalent colored model is made of 70000 vertices and 135000 triangles.

The bear model shows the necessity of having a constant, diffuse illumination during acquisition. The acquisition illumination was not optimal and color discontinuities can be observed, especially on the head.



(a) (b)

(c) (d)

Figure 4. (a) textured rabbit model, (b) wire frame representation of the rabbit model, (c) texture image of the rabbit model, (d) textured bear model

An example of the usefulness of texture extension and filtering is shown in Fig. 5. Color and resolution discontinuities are perceptible before the process (a). Good improvement can be seen in (b): texture boundaries between patches become nearly invisible after extension and filtering.

(a) (b)

Figure 5. (a) texture boundary before process, (b) texture boundary after extension and filtering

6 CONCLUSION

A new algorithm to integrate textured meshes is presented in this paper. One of the key features of this method is that it keeps most of the original triangulation and texture mapping of the original meshes. It was successfully implemented as an extension to a previous system [7], adding possibility to use texture in the basic overlap-removing and gap-filling method. Typical reconstructed objects were presented. These results show that integration of textured view is possible. Presented method opens the perspective of a more general and less complex way to reconstruct 3D objects from range images.

Future investigations include:

a) A mesh reduction method to decrease the geometrical complexity of "texture patched" models.
b) An optimal way to place patches in the texture image to reduce its size.

7 ACKNOWLEDGMENTS

This research has been partially funded by the Swiss national science foundation under project number 2100-43530.

REFERENCES

[1] P.J. Besl, N.D. McKay, "*A Method for Registration of 3-D Shapes*", Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 14, pp. 239-256, 1992.

[2] A.E. Johnson, S.B. Kang, "*Registration and Integration of Textured 3-D Data*", IEEE International Conference on Recent Advances in 3-D Digital Imaging and Modeling, Ottawa, pp. 234-241, 1997.

[3] W. Niem, J. Wingbermühle, "*Automatic Reconstruction of 3D Objects Using a Mobile Monoscopic Camera*", IEEE International Conference on Recent Advances in 3-D Digital Imaging and Modeling, Ottawa, pp. 173-180, 1997.

[4] M. Soucy, G. Godin, M. Rioux, "*A Texture-Mapping Approach for the Compression of Colored 3D Triangulations*", The Visual Computer, vol. 12, pp. 503-514, 1996.

[5] M. Soucy, G. Godin, R. Baribeau, F. Blais, M. Rioux, "*Sensors and Algorithms for the Construction of Digital 3-D Colour Models of Real Objects*", Proc. IEEE Int. Conf. on Image Proc., ICIP'96, Lausanne, vol. 2, pp. 409–412, 1996.

[6] C. Schütz, T. Jost, H. Hügli, "*Multi-feature Matching Algorithm for Free-Form 3D Surface Registration*", International Conference on Pattern Recognition, ICPR'98, Brisbane, 1998.

[7] C. Schütz, T. Jost, H. Hügli, "*Semi-Automatic 3D Object Digitizing System Using Range Images*", Proc. Asian Conference on Computer Vision, ACCV'98, Hong Kong, vol. 1, pp 490-497, 1998.