Texture Minification using Quad-trees and Fipmaps

Alexander Bornik and Andrej Ferko

Graz University of Technology, A-8010 Graz, Austria
Comenius University, SK-842 48 Bratislava, Slovakia

Abstract
The paper extends the recently published methods for image reconstruction and texture minification using the generalized ripmap method, named fipmap, and quad-trees. Fipmap is based on the technique of partitioned iterated function systems, used in fractal image compression. The quad-tree texture reconstruction algorithm works well for many standard cases. The special cases can be solved using the fipmap minification. The approach was applied for textures from architectural image sequences and the results are very promising.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Texture Mapping

1. Motivation and Introduction

Figure 1: *Hans Holbein jr. - Ambassadors (1533) from* http://www.artchive.com. *What is the long tiny unrealistic object in the foreground?*

One of the leading trends in highly-realistic rendering is image-based rendering and/or lighting, combining real samples and modelled environments. In limited navigation applications like Quicktime-VR, the user is kept away from visual cue problems, not being allowed to go very close or too far. Ongoing virtual reality applications, like immersive surgery, interactive TV, art history distant education, or virtual archaeology cannot accept this limitation. Real life trains us in texture minification/magnification. "Human beings are apparently very good at remembering qualities of textures... computer graphics techniques are influenced by the analytical strategies of the visual system" conclude *R. M. Friedhoff and W. Benzon*[7], p. 112. Another related leading trend is perceptually-driven image synthesis, see survey by *A. McNamara*[12].

The motivation for image reconstruction and anti-aliased texture minification/magnification is given by many practical requirements. "Artifacts are extremely problematic in texture mapping and most textures produce visible artifacts unless the method is integrated with an anti-aliasing procedure" argues *A. Watt*[17], p. 256. Our application uses the textures from architectural image sequences[11], which are intended for immersive[?] virtual fly-over or walkthrough in cyber-cities[?]. For the implementation of reconstructing multi-resolution textures from image sequences we have developed an original method, based on the work of *E. Ofek et. al.*[15]. Currently, we study the further quality improvements. The advanced methods for texture manipulation in OpenGL API can be found, e. g. in *T. McReynolds*[13].

A frequently used idea in texture minification is to precalculate all the required filtering operations by so-called mipmapping[18], and - more recently - ripmapping[13]. However, the minification request may occur under conditions when both mipmapping and ripmapping fail. For instance, when both the distance and the camera orientation are very unusual, Fig. 2. We propose employing partitioned iterated function systems (PIFS)[5]. They can also control the contrast and brightness of the transformation result. This technique proved suitable in fractal image compression. Therefore, our generalization of the mipmap and ripmap approaches we call fipmap, for the sake of continuity.

The paper is structured as follows. In section 2, we discuss selected recent methods. Section 3 introduces our approach. Section 4 demonstrates the results and section 5 offers the future work and conclusions.

2. Related Work

The standard methods manipulate textures using a rectangular grid. This way of image sampling may be hostile for the image content. We deal with special textures - coming from the architecture of excavations or real buildings and interiors. The nature of them seems to be suitable for rectangular-shaped manipulations. However, some problems come with minification.

One possible solution for texture minification is to down-sample, i. e. create some subset of filtering operations. This is called mipmapping[18]. A more recent development of the idea is ripmapping[13], Figure 2.

Ripmap is intended to avoid overblurring, one of the mipmap flaws. "Imagine a pixel cell that covers a large number of texels in the u direction and only a few in the v direction. This case commonly occurs when a viewer looks along a textured surface nearly edge-on"[14], p. 113. However, the minification request may occur under conditions when both mipmapping and ripmapping fail. We have observed this phenomenon in the context of architectural outdoor scenes. The coordinate axis aligned pre-calculation fails when camera orientation - with respect to the textured surface normal - is not aligned. We will describe this formally in paragraph 3.2.

The common feature of mipmap, ripmap, and quad-tree oriented image manipulation is the use of a rectangular or square grid. This can be image hostile. Some recent research efforts take into account the image content with segmentation, edge extraction, or data dependent triangulation[4]. The image analogies[8] approach offers another prospective alternative for minification - to create filters by example. However, we have no high-resolution images for all possible camera parameters. Another texture filtering methods include summed area table and Paul Heckbert's elliptical weighted average (EWA) for anisotropic texture filtering.

Our fipmap idea is to employ the partitioned iterated function systems (PIFS), described in work by *Y. Fisher et al.* [5], p. 11. They extend the affine transformations by taking the third dimension, grey level, into account. In particular, the control of contrast and brightness of the transformation is enabled. This technique proved suitable in fractal image compression. Strictly speaking, a complete fractal image compression, in addition, employs masks and uses the transform for a completely different goal.

Using PIFS extended affine transform for texture minification this way has not been utilized. We describe the PIFS transform formally in the next section as a part of the fipmap algorithm. Briefly, we shear and shade ripmap sub-textures for extreme angles. Fipmap utilization is reasonable for big-

(a) mipmap

(b) ripmap

Figure 2: *Mipmapping and ripmapping. Note that the ripmap structure diagonal is formed by mipmaps*

Figure 3: *Extreme camera orientation in urban environment*

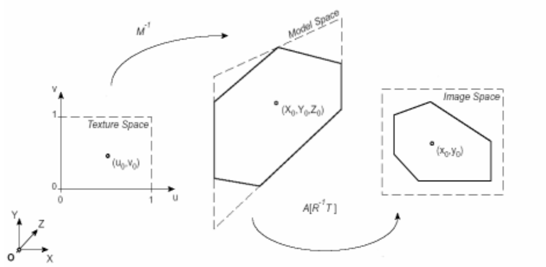


Figure 4: *Reconstruction of image acquisition: a matrix transforming from the texture coordinate system to input image coordinates; M, A, R, T are corresponding matrices*

ger non-uniform shearing, separately in both x and y. The meaning of the word bigger, will be again formalized below.

3. Our Approach

In the following, we are going to introduce texture reconstruction from multiple views and fipmap texture minification.

3.1. Texture Reconstruction from Multiple Views

Our texture reconstruction method is based on the work of *Ofek et. al.*[15] using projective texture mapping. For an arbitrary scene represented by polygons we calculate texture images for planar regions using multiple images acquired using a digital CCD camera as the images source. These images have to be registered in terms of computer vision, which is done using the method of *Z. Zhang*[19].

Using this registration information we set up a matrix that performs the transformation from a point in texture space (texture coordinates) to image coordinates in the original images.

We set up a quad-tree data structures covering each geometry part to be textured by a single polygon and fill this structure with pixel information from the original images. This is done in a recursive way covering resolution differences of texture regions that occur due to the transformation. Starting with the corner points of the whole texture region corresponding to the root node in the quad tree the size of the projection is compared with the size of a pixel in the input image. Further subdivisions are performed until the size matches. Radiometric information is stored in the corresponding node of the quad-tree in a list taking into account information from multiple images.

In contrast to *Ofek et. al.*[15] we perform object order visibility tests throughout the recursion steps to ensure that no color information from modelled occluding objects enters the quad tree data structure.

Once all images contributed their radiometric information to the quad tree structure it contains information at different resolution levels, which has to be merged in order to retrieve mip-map like texture images. We do so weighting the color information portions stored in the quad-tree preferring high resolution information over coarse information. The actual combination is performed in two steps: First values are propagated up the tree adding them to their parent's values recursively. The leaves store the difference to their parents only. In the second step this sparse Laplacian-like representation is converted once more adding the parent's value to the children recursively. After that each level of the quad tree contains texture images influenced by both high- and low resolution texture information.

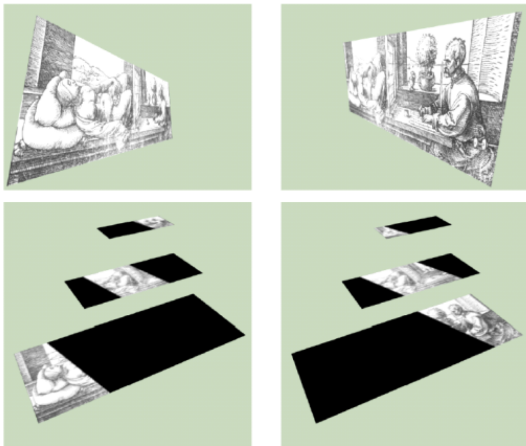


Figure 5: *Input images for two views, and quad tree level before information fusion*

In real world outdoor scenes like city models or archeological scenes images might contain non-modelled occlu-

Figure 6: *Reconstructed texture: weighting preserves high resolution information; painting by A. Duerer (1471-1528)*

sion. Such occlusion is caused by objects that have no geometrical representation in the scene graph which texture reconstruction uses. For example trees, traffic signs, or power lines could be such occluding objects.

Our algorithm deals with this problem by employing a median filter on the color values corresponding to a region in the quad-tree. Occluding objects, especially small ones only occur in the minority of the input images. Selection of texture values close to this middle value (median) of these values avoids these artifacts.

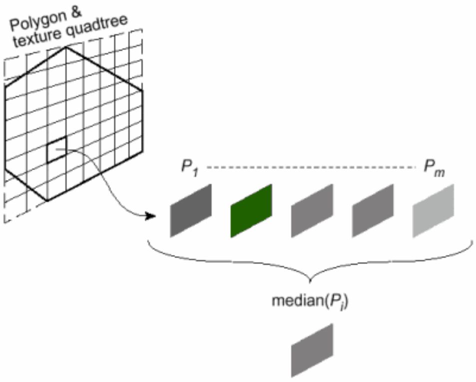


Figure 7: *Median filter: the median of multiple color values of each quadtree entry is calculated; only values close to the median are considered for texture calculation*

In addition, the median filter removes specular highlights that might be visible on highly reflective surfaces in the input images. Therefore our textures can be used together with artificial light sources in the rendering stage.

Effective use of the median filter techniques demands a sufficiently large number of input images for each texture region. In general about 5 values fulfill the criterion.

3.2. Fipmap Texture Minification

The minification is sometimes referred to as texture compression[17], p. 257. When a viewer looks along a textured surface nearly edge-on, the angle between the camera direction and the textured surface normal grows and the cosine approaches zero. In this case we have to deal with more specific texture transforms. Notice, that both mipmap and ripmap scale the texture only in the x and y directions, leaving the rest of the transformation to the final phase of texture mapping. The affine transforms in the plane include: scaling, translation, rotation, and shearing.

Our idea is to employ the partitioned iterated function system (PIFS)[5], p. 11. It extends the affine transformations by adding the third dimension z, grey level, into account. In particular, the control of contrast s and brightness o of the transformation is enabled. This helps when the so-called atmospheric perspective appears. In computer graphics, this is simplified by the light source attenuation term and depth cueing in the local illumination models[6]. Obviously, this approach cannot manage all three perspective principles: distant objects are smaller, their colors are more matte, and their contours are softer. Looking through the window at a very distant object on a sunny day can be properly solved using fipmap to control contrast and brightness. Usually, mipmap images are derived using averaging down the original image. The process creates an image pyramid by isotropic scaling, $a = d$ in (1). Again, the PIFS transform gives more and enables for anisotropy.

We describe PIFS extension of affine transformation formally:

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & s \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e \\ f \\ o \end{bmatrix} = \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{bmatrix} \quad (1)$$

Isolating the spatial part of the transformation reduces the dimension and gives the standard affine transform in the plane. The following important algebraic and geometric properties hold. The main 2/2 minor of the above matrix can be always written in polar coordinates using sine and cosine functions. Any rotation in two dimensions is a combination of scaling and shearing (true for all angles having a finite tangent) [16]. We do not use (the expensive) rotation at all. The planar affine transform, which approximately transforms one set to another is given by a triplet of function values. In fractal compression, so-called archetypes can determine the appropriate transforms *(see Y. Fisher, p. 79n)*[5]. In the fipmap method we use the PIFS transform type not iteratively. We compute, for given texture, the appropriate fipmap transform from camera parameters. When we have decided to use fipmap (only in the case of a small viewing angle and/or large observer distance when bigger non-isotropic shearing is needed) we compute the transformation and apply it. Otherwise, we use the standard method (quadtree, mipmap, ripmap).

We introduce a color strategy for better perception of geometry. Let W be the origin of a local coordinate system located in the center of the textured polygon. Let axes B and R are aligned with the images of texture coordinates M-1(u)

Figure 8: *Color visualisation of one octant*

and M-1(v) and G be the textured polygon normal. Let camera C be in the first octant (R+, G+, B+). The notation is inspired by the RGB cube convention with exchanged roles of white and black colors. The origin is White, the axes mean the color primaries Red, Green, and Blue. Camera C may have a Color. Given the triangle RGB we can easily find the intersection of CW line with the RGB triangle. The length of CW measures the camera distance and can be used for setting the values for brightness and contrast. The black dot is very distant and the gray one illustrates the decreased light intensity. The cosine of angle given by camera orientation and WG (normal) directions expresses the following camera cases. If the camera orientation is close to normal (green) then the mipmap works well. If the "color" of camera approaches slightly the B and R along the sides of the RGB triangle then ripmap applies. Finally, the lower part of the triangle calls for fipmap, especially the R and B corners. Expressing the camera "color" in two independent barycentric coordinates with respect to R and G gives the estimate for proportion of anisotropy. Reddish and bluish camera "color" indicates shortening of distances and bluish v (red), respectively, in the textured plane. Computationally, we can replace the barycentric camera "color" computation by cosines of camera orientation and WB resp. WG. If the camera "color" has very small amount of green then we can project the camera position to the RB plane. Denote this point by P. If the camera "color" is too red (or too blue) we can employ ripmap. The particular tuning of greenish, reddish, and bluish is done by evaluating the dotproducts (cosines, barycentric coordinates) and by thresholding. The fipmap transformation is completed by setting either b or c equal to the tangent of the angle given by WP and one of the axes R+ or B+. The detailed discussion is given with results. Note, that the exact 3D computation has to take into account the camera orientation different from CW.

The fipmap method proceeds as follows:

1. Fipmap decision:
   a. Compute camera distance and three angles of camera orientation with R, G, B axes.
   b. If camera direction is greenish then use mipmap and return.
   c. Set the contrast and brightness coefficients s, o ($e = f = 0$).
   d. Compute a, b, c, d ($a = d = 1$).
2. Texture minification:
   a. Transform the texture.
   b. Perspective texture mapping.

Three comments:

1. There were the extreme view orientations consciously used by renaissance painters for obtaining special visual effects. Anamorphosis is a special case of perspective, described but not used by Leonardo da Vinci. The famous anamorph in Ambassadors by Hans Holbein jr. (1497-1543) we use in fipmap experiment below.
2. It is subject to finer discussion when fipmap should take part. The obvious solution is to leave the decision to the user. On the other hand, as the fipmap generally gives the multidimensional family of sub-textures, the method may be very memory intensive if we wish to create the fipmap database analogously to mipmap and ripmap precomputation phase.
3. We have tested the results with multiple comparisons, including comparison with real photos, as our VRML model captures actual buildings.

4. Results

Here we show the selected results. More material on the original quad-tree algorithm can be found in[1].

4.1. Results from the Original Algorithm - Artificial Scene

We used our algorithm with an artificial scene created using 3D modelling.

It contains an L-shaped box, a red cube representing a modelled occluding object and a blue cylinder. The blue cylinder was removed from the scene for texture reconstruction and therefore is a non-modelled occluder. The texture reconstructed for the checkerboard surface by our texture reconstruction algorithm is shown below.

The reconstructed texture does not show artifacts from any of the occluders prominent in some of the input images.

(a) (b)

Figure 9: *Fipmap for Holbeint's anamorphosis: a) Holbeint's anamorphed skull in Ambassadors, a painting for two observers: the skull is visible from one third of the right margin; b) skull "original", with fipmap coefficients a=6, b=1, c=-0.45, and d=1*

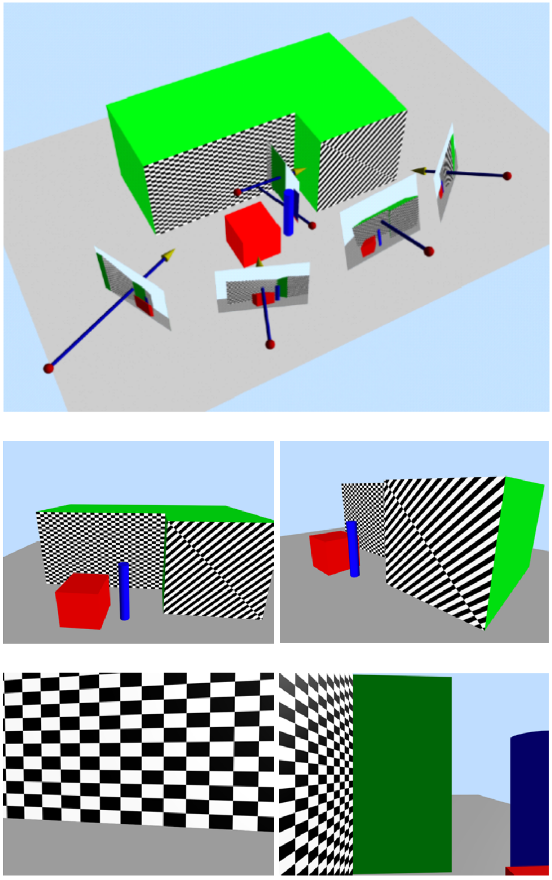


Figure 10: *Artificial scene: artificial scene used for texture reconstruction, arrows show camera orientations of the input images, four input images can be seen on the right*

4.2. Results from the Original Algorithm - Outdoor Scene

Texture reconstruction for the real world scenes is the main purpose of our algorithm. We show the results of our method for a building of the Graz University of Technology, which can be seen in Figure 12 including some of the images used for reconstruction.

As can be seen above in the Figure 12 the dozen of input images used contain occlusion by cars, traffic signs, trees and other objects. Nevertheless the output does not contain major occlusion artifacts or reflection artifacts in the regions supported by a sufficiently large number of input images. Artifacts are mainly caused by the cars close to the facade, so they are occluders in all of the images.

Figure 14 shows the successful removal of an occluding tree using our method. The remaining seams are due to differing illumination level of the input images. The color tone of the occluding tree (brown) was completely removed.

4.3. Results from the Original Algorithm - Indoor Scene

Indoor scenes can be far more complex compared to outdoor scenes in the context of the lighting situation. Global light interaction occurs at a higher level, so these scenes pose more challenging input data for texture reconstruction.

Again our algorithm is able to avoid occlusion artifacts. It can remove occluder texture information, namely information of the red box (modelled) and the non- modelled fig tree on the black cupboard. However low frequency intensity changes may be noticed on the yellow pin board. These are caused by global illumination phenomena and can't be detected/removed by our algorithm.

4.4. Fipmap Results

For the fipmap work illustration, we have selected the well-known test image for LPPM measurements[16], p. 11. We show only the results of the first phase of the algorithm. The test image for measurements of line pairs per milimeter (LPPM) contains several affine copies of the same image content: the black square and the sets of parallel lines resp. filled rectangles. This is repeated with different orientations and scaling factors. The axially aligned composition is created. We recommend to observe transforming the black squares or the decimal digits. In figures 16 and 17 there are pairs of images showing the increasing texture minification which can be observed on the transformed black squares. They are modified into the parallelograms with increasing

Figure 11: *Reconstructed texture for artificial scene, removed the non-modeled occluder*

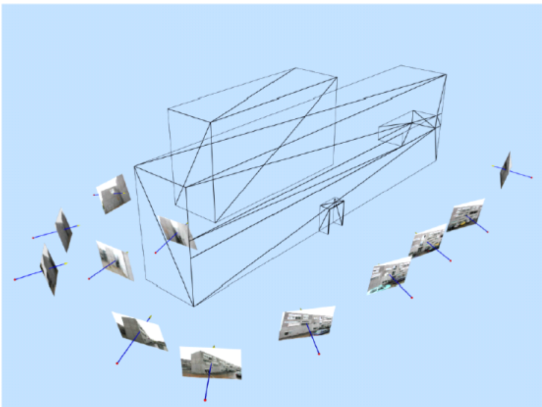


Figure 12: *Real world outdoor scene: Geometrical model used for texture reconstruction and two of the input images*

Figure 13: *Scene rendered using textures reconstructed using our method with artificial lighting*

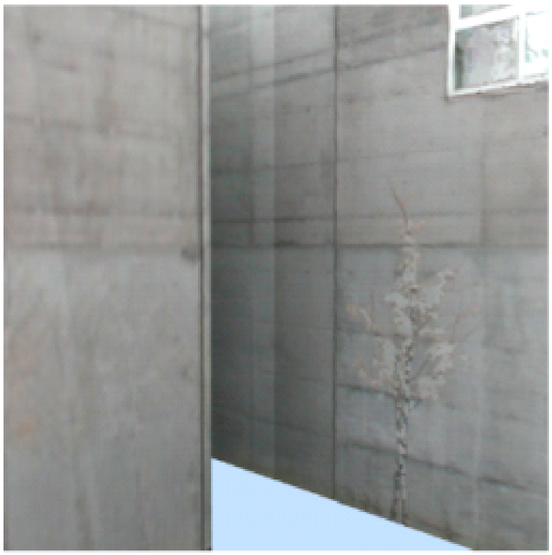


Figure 14: *Occlusion removal: the occluding tree was successfully removed by our algorithm*

maximum angle. This angle along with the scaling of the square side displays the texture minification. In other words, we can visually compare the image quality in Figure 3 where the single texels produce the perceptually wrong texturing with the smooth fipmap appearance in figures 16, 17, 18, 19. Thus, we can immediately see that the texturing using fipmap higher quality of the imagery. Our experiments show

The affine coefficients modify the texture shearing intuitively enough. The texture orientation causes no problems. The higher differences of values cause the following appearance. Notice the similarity of set up with the Figure 3.

The correspondence with camera position and the superiority over axially aligned approach is obvious. The disadvantage is the large multidimensional database. In the above experiments we did not assume the gray levels modification. In Figure 19, there one can observe another kind of aliases - with the set of longer parallel line segments.

5. Conclusions and Future Work

In our previous work, we implemented the texture extraction from multiple image sequences following *Ofek et al.*[15] and provided a new better method[11]. We did not consider view-dependent texture mapping and to view scenes we needed a special application. Recently developed methods gave us the inspiration for the current research.

Figure 15: *Indoor office scene rendered using textures calculated by our method, input images are shown together with camera positions*

Figure 16: *Schematic illustration of one octant and competence of methods.*

We have introduced a new algorithm for texture minification. The fipmap minification is computationally intensive and the highest precision improvements might be imperceptible. The study of the feasibility and perceptual quality trade off is in progress.

To quote the concluding statement from *Friedhoff, p. 131*[7]: "The process of evaluating a texture is rooted in the feverish activities of preconscious visual analysis... one kind of texture can appear realistic while another, closely related by algorithm, seems unrealistic". Our future work will address texture preprocessing of arbitrary meshes, projection onto non-planar surfaces, eventually on implicit surfaces[?], parallel-processing support, hardware acceleration and view planning - both for cyber cities[9] and virtual archaeology installations[3]. If the movement trajectory is known in advance the fipmap database is not necessary to precompute. In this case the transformation coefficients can be computed directly and eventually interactively tuned for obtaining the high quality perceptual realism.

Figure 17: *Practical fipmap use requires to process a few camera directions. Six alternatives are shown in figures 15, 16, and 17.*

This method is better, too. We have illustrated this in five concluding figures using extremely well structured and well known image. Fipmap produces no artifacts and even can control the grey level.

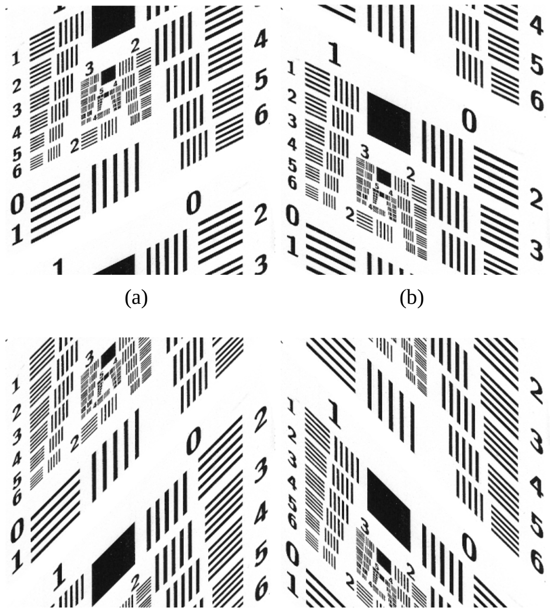
5.1. Preprocessing of Arbitrary Meshes

Geometry data calculated from images or even geometry data modelled using a CAD-tool may contain many coplanar surfaces not represented by a single indexed faceset. On the other hand there might be non-planar surfaces represented by a single faceset. Our current texture reconstruction framework requires each texture entity to be represented by a single indexed faceset, which up till now has to be done manually in a pre-processing step.

Auto-detection of co-planar surfaces in arbitrary geometry data and modification of that data in an adequate way could be developed. This would greatly enhance the usability of the method for other than hand-made models.

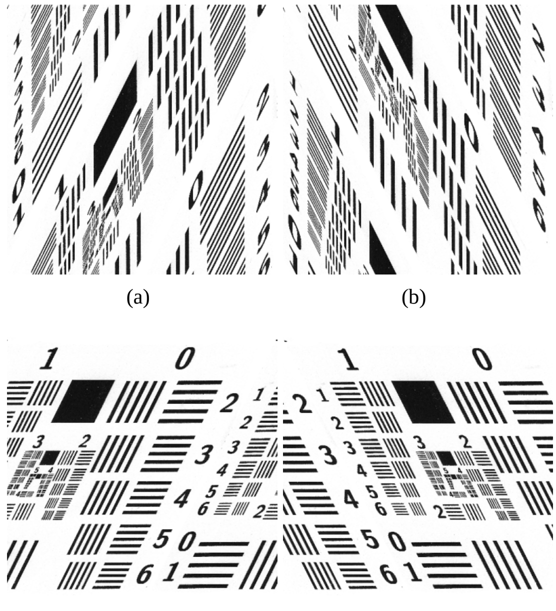
5.2. Projection onto Non-Planar Surfaces

Currently texture calculation using our method is limited to planar surfaces. *Ofek et al.*[15] has already suggested texture



(a) (b) (c) (d)

Figure 18: *Fipmap: Coefficients* (a,b,c,d) *equal to* $(1,0,0.5,1)$, $(1,0,-0.5,1$, $(1,0,1,1)$ *and* $(1,0,-1,1)$ *respectively*



(a) (b) (c) (d)

Figure 19: *Fipmap: Coefficients* (a,b,c,d) *equal to* $(1,0,2,1)$, $(1,0,-2,1)$, $(1,0.5,0,1)$, *and* $(1,-0.5,0,1)$ *respectively*

calculation based quad-trees build over cylindric surface. One might also consider other primitives like basic shapes, spheres or even implicit surfaces like free form surfaces.

5.3. Parallel-Processing Support

If textures for a whole scene have to be calculated it would be useful to be able to do it in parallel. This is possible, because the quad-tree data structures used for each surface are independent. Currently, parallel calculation can in principle be done by storing the input data in directories shared among multiple computers and assigning tasks to each machine individually and manually. Future implementations could include methods for dynamic work- and load distribution among a number of machines connected by some sort of network. This would supersede the necessity to assign a task to a machine manually and would maximize the utilization of the available resources.

5.4. Hardware accelerated visibility

Recent graphics adapters allow high resolution rendering, some even off screen rendering. Such hardware could be used for rendering the whole scene textures are calculated using a single color for each single texture entity. Afterwards visibility tests can be performed by lookups in these precalculated images instead of object order visibility tests.

5.5. View-Planning

Our approach delivers information about the number of images that contribute to different texture regions. Currently we don't take advantage of this information. This information could be used for calculation of additional viewpoints that, added to the input data, could eliminate regions covered by too few images. An even more sophisticated version could automatically calculate viewpoints that result in a, preferably equal, user specified number of images contributing to each node of the quad-trees.

Such extensions could be very useful when reconstructing textures for large urban models with high geometrical complexity where an optimal viewpoint distribution can't otherwise be estimated.

6. Acknowledgements

This work has in part been funded by the European Union under contract No. IST-1999-20273. We wish to thank Dipl. Ing. Markus Grabner, Dr. Konrad Karner, Dipl. Ing. Rainer Kalliany, and Prof. Horst Bischof for fruitful discussions.

References

1. A. Bornik. Textures from architectural image sequences. Master's thesis, Graz University of Technology, May 2001. 6

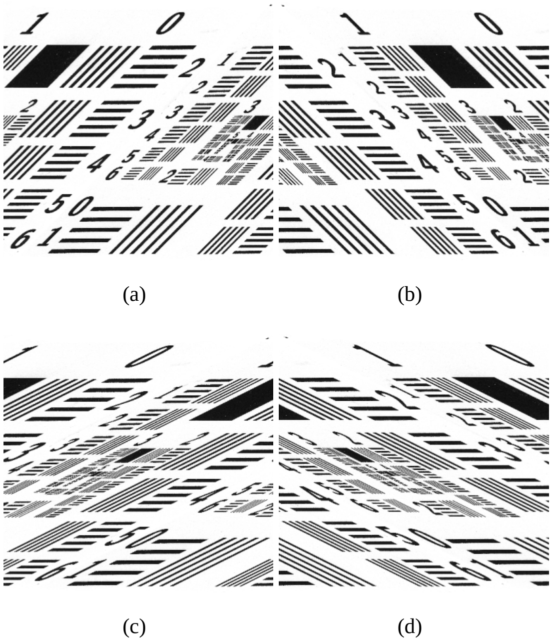


Figure 20: *Fipmap: Coefficients* (a,b,c,d) *equal to* $(1,1,0,1)$, $(1,-1,0,1)$, $(1,2,0,1)$, *and* $(1,-2,0,1)$ *respectively*

Figure 21: *Fipmap: Nearly edge-on view,* (a,b,c,d) *equal to* $(1,-2,0.5,3)$

2. A. Cole. *Eyewitness Art - Perspective*. Dorling Kindersley, 1992. 5

3. J. Cosmas, J. Itagaki, D. Green, E. Grabczewski, L. Van Gool, A. Zalesny, D. Vanrintel, F. Leberl, M. Grabner, K. Schindler, K. Karner, M. Gervautz, S. Hynst, M. Waelkens, M. Pollefeys, R. DeGeest, R. Sablatnig, and M. Kampel. 3d murale: A multimedia system for archaeology. In *Proceedings of the International Symposium on Virtual Reality, Archaeology and Cultural Heritage 2001*, November 2001. 8

4. N. Dyn, D. Levin, and S. Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA Journal of Numerical Analysis*, 10:137–139, 1990. 2

5. Y. Fisher. *Fractal Image Compression*. Springer, 1995. 2, 4

6. J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics, Principles and Practice*. Addison Wesley, second edition, 1990. 4

7. R. M. Friedhoff and W. Benzon. *The second computer revolution: Visualization*. Harry N. Abrams, 1989. 1, 8

8. A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *SIGGRAPH 2001 Conference Proceedings*. ACM SIGGRAPH, 2001. 2

9. K. Karner, J. Bauer, A. Klaus, F. Leberl, and M. Grabner. Virtual habitat: Models of the urban outdoors. In E. Baltsavias, editor, *Proceedings of the Third International Workshop on Automatic Extraction of Man-Made Objects from Aerial and Space Images*, pages 393–40. A.A. Balkema Publishers, 2001. 8

10. S. Kent. *Eyewitness Art - Composition*. Dorling Kindersley, 1995. 5

11. H. Mayer, A. Bornik, J. Bauer, K. Karner, and F. Leberl. Multiresolution texture for photorealistic rendering. In T. L. Kunii, editor, *Proceedings of Spring Conference on Computer Graphics 2001*, pages 174–183. Comenius University Bratislava, 2001. 1, 8

12. A. McNamara. Visual perception in realistic image synthesis. *Computer Graphic Forum*, 20(4):201–210, 2001. 1

13. T. McReynolds. Programming with opengl, advanced techniques, 1998. 1, 2

14. T. Möller and E. Haines. *Real-Time Rendering*. A. K. Peters, 1999. 2

15. E. Ofek. Multiresolution textures from image sequences. *IEEE Computer Graphics and Applications*, 17(2):18–29, 1997. 1, 3, 8, 9

16. D. Salomon. *Computer Graphics and Geometric Modeling*. Springer, 1999. 4, 7

17. Alan Watt. *Three-Dimensional Computer Graphics*. Addison-Wesley, third edition, 2000. 1, 4

18. L. Williams. Pyramidal parametrics. *Computer Graphics*, 17(3):1–11, 1983. 1, 2

19. Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the International Conference on Computer Vision (ICCV'99)*, pages 666–673, 1999. 3