TEXTURE MAPPING BY SUCCESSIVE REFINEMENT

Stefan Horbelt, Philippe Thévenaz and Michael Unser

Biomedical Imaging Group, IOA, DMT
Swiss Federal Institute of Technology Lausanne
CH-1015 Lausanne EPFL, Switzerland

ABSTRACT

We define texture mapping as an optimization problem for which the goal is to preserve the maximum amount of information in the mapped texture. We derive a solution that is optimal in the least-squares sense and that corresponds to the pseudo-inverse of the texture-mapping transformation. In practice, a first-order approximation of the least-squares solution is used as an initial estimate for the mapped texture. This initial solution is refined by successive approximation to yield the least-squares optimal result. In essence, the proposed multi-pass method acts like an adaptive anti-aliasing filter.

keywords: texture mapping, least-squares approximation, successive refinement, anti-aliasing

1. INTRODUCTION

In computer graphics, texture mapping refers to a warping process by which a texture image is transformed geometrically to simulate a mapping onto a given 3D surface. Texture mapping adds fine surface details on an object, like color, specularities, bumps, dirt, and so on, which leads to more realistic pictures [3]. With texture mapping, a much lower polygon count is needed to achieve results similar to a pure high-count polygon model in which each polygon would carry one value of a textured attribute. Textures are patterned images created before the time-critical rendering. Efficient techniques exist to use textures for fast rendering. Bilinear texture mapping is commonly used and is available in current graphics hardware. Here, we show the limits of the standard texture-mapping methods and propose a novel algorithm that is based on least-squares approximation and successive refinement.

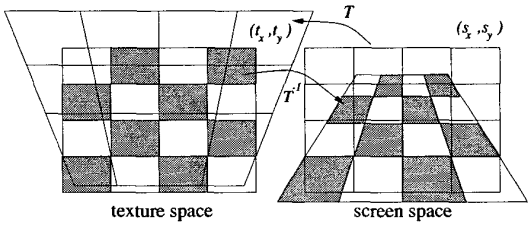

Figure 1: Transformation T between screen space and texture space.

2. STATE OF THE ART

We summarize texture-mapping methods from several articles [1],[6] and two surveys [2],[4].

Nearest neighbor is the fastest method but gives the worst quality. It maps the pixel's screen coordinates to texture coordinates and copies the value of the closest texel. **Bilinear interpolation** combined with Mipmapping pyramids is a common standard approach. **Supersampling** gives the highest quality but is the slowest method. Supersampling renders the image at (typically 16-times) higher resolution first, and then reduces the rendered image to the final resolution using standard anti-aliasing filtering and re-sampling.

The higher-quality methods generally use some kind of texture anti-aliasing algorithm. Typically, one determines the footprint of a screen pixel in texture space, and then takes the weighted sum over all texels inside the borders of this influence region. The weights are defined by an empirical filter function. The methods can be distinguished by the filter function. **Area sampling** uses a box filter, whereas elliptical weighted area (**EWA**) uses a circular Gaussian filter assuming a circular pixel support. Two acceleration methods based on tables have been proposed; they are **Mipmapping** pyramids and **summed-area tables**. Additionally, texture-mapping algorithms can be classified into **texture-space scan** and **screen-space scan** algorithms.

0-7803-6297-7/00/$10.00

3. THE TEXTURE-MAPPING PROBLEM

The texture-mapping problem is to map some input texture image f_t to produce a screen image f_s

$$f_s(s_x, s_y) \cong f_t(T(s_x, s_y)), \quad (1)$$

$$T(s_x, s_y) = (t_x, t_y), \quad (2)$$

where the geometrical transformation T, e.g., a perspective mapping on a 3D surface (as seen in fig.(2)), maps the screen coordinates (s_x, s_y) from the screen space into the texture coordinates (t_x, t_y) in the texture space. Assuming that T is invertible, the dual version of the problem is

$$f_t(t_x, t_y) \cong f_s(T^{-1}(t_x, t_y)), \quad (3)$$

$$T^{-1}(t_x, t_y) = (s_x, s_y), \quad (4)$$

where T^{-1} is the reverse mapping of T. We have used an approximate equality symbol in (1) and (3) because the mapping will not be perfect—both image and texture grids are discrete, which entails some loss of information.

4. LEAST-SQUARES TEXTURE MAPPING

In this section, we introduce a novel texture-mapping algo based on least-squares approximation and successive refinement. As a quality criterion, we choose to minimize the information loss of the mapped texture. In other words, when we undo the texture mapping, we would like to reconstruct the original texture as well as possible. The loss of information is measured in the least-squares sense and is optimized in texture space.

4.1. Continuous-screen model

We define a continuous model f_s of the screen, which lives in the screen space generated by the functions φ_s:

$$f_s(s_x, s_y) = \sum_k \sum_l c_s(k, l) \cdot \varphi_s(s_x - k, s_y - l), \quad (5)$$

where (s_x, s_y) are the screen coordinates, and where $\varphi_s(s_x - k, s_y - l)$ are the screen basis functions at integer shifts (k, l). The basis functions are simply shift-version of a generating function $\varphi_s(x, y)$; for example, a separable B-spline or any other standard interpolation function.

We can attempt to recover the original texture f_t by applying the inverse mapping (3)

$$\tilde{f}_t(t_x, t_y) = f_s(T^{-1}(t_x, t_y)). \quad (6)$$

4.2. Least-squares criterion

Assume the texture model f_t is given. Then, we optimize the texture mapping on the screen f_s by minimizing the approximation error

$$\epsilon_t = \min_{c_s} \left\| f_t(t_x, t_y) - f_s(T^{-1}(t_x, t_y)) \right\|.$$

Note that we measure the approximation error in the texture space because this is where the original texture f_t is specified.

4.3. Discrete texture mapping

In the following, we assume that the input texture is specified by its texel values $f_t(\boldsymbol{m})$, $\boldsymbol{m} = (m, n)$. Together with the inverse texture mapping (3) and with the continuous-screen model (5), the transformation of the screen coefficients $c_s(\boldsymbol{k})$, $\boldsymbol{k} = (k, l)$ to the texels f_t is described as

$$\tilde{f}_t(\boldsymbol{m}) = \sum_{\mathrm{k}} c_s(\boldsymbol{k}) \cdot \varphi_s(T^{-1}\boldsymbol{m} - \boldsymbol{k}). \quad (7)$$

In order to use matrix notation, we rearrange the two-dimensional arrays (images and arrays of coefficients) in raster-scan order to one-dimensional arrays: $\tilde{\boldsymbol{f}}_t = \left[\tilde{f}_t[\boldsymbol{m}]\right] = \left(\tilde{f}_t(1,1), \cdots, \tilde{f}_t(1,N), \cdots, \tilde{f}_t(M,1), \cdots, \tilde{f}_t(M,N)\right)$, where (M, N) is the size of the texture. The same applies for $\boldsymbol{c}_s = [c_s(\boldsymbol{k})]$.

The key idea is to determine the $\boldsymbol{c}_s$ such that $\tilde{\boldsymbol{f}}_t$ is a good approximation of $\boldsymbol{f}_t$. The equation (7) in matrix form is

$$\tilde{\boldsymbol{f}}_t = \boldsymbol{A} \cdot \boldsymbol{c}_s \quad (8)$$

$$\boldsymbol{A} = [a(\boldsymbol{m}, \boldsymbol{k})] = \left[\varphi_s(T^{-1}\boldsymbol{m} - \boldsymbol{k})\right]. \quad (9)$$

In order to correctly reproduce a constant, we require that φ_s satisfies the partition of unity:

$$\sum_{k \in Z} \varphi_s(x - k) = 1. \quad (10)$$

A direct implication is that the rows of $\boldsymbol{A}$ sum up to one:

$$\sum_{\mathrm{k}} a(\boldsymbol{m}, \boldsymbol{k}) = 1.$$

4.4. Least-squares solution

Given the original texture f_t, we want to minimize

$$\min_{\mathrm{c}_s} \left\| f_t(m, n) - \tilde{f}_t(m, n) \right\|_{l_2}^2 = \min_{\mathrm{c}_s} \left\| \boldsymbol{f}_t - \boldsymbol{A} \cdot \boldsymbol{c}_s \right\|^2 \quad (11)$$

to get the least-squares solution $\boldsymbol{c}_s$. The solution can be expressed explicitly in terms of $\boldsymbol{A}^+$, the pseudo-inverse of $\boldsymbol{A}$:

$$\boldsymbol{c}_s = (\boldsymbol{A}^\top \boldsymbol{A})^{-1} \boldsymbol{A}^\top \cdot \boldsymbol{f}_t = \boldsymbol{A}^+ \cdot \boldsymbol{f}_t.$$

Because of the large size of the involved matrices, the direct computation of the inverse $(\boldsymbol{A}^\top \boldsymbol{A})^{-1}$ is too expensive to be carried out in practice. In first approximation, we will assume that it is close to a diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1 \ldots \lambda_k) \cong (\boldsymbol{A}^\top \boldsymbol{A})^{-1}$ and we write

$$\boldsymbol{A}^+ \cong \tilde{\boldsymbol{A}}^+ = \Lambda \cdot \boldsymbol{A}^\top = \left[\lambda_k \cdot \varphi_s(T^{-1}\boldsymbol{m} - \boldsymbol{k})\right]. \quad (12)$$

The partition of unity (10) must again be fulfilled. This time, all columns of $\boldsymbol{A}^+$ have to sum up to one, due to the transposition of $\boldsymbol{A}$. We set the elements of Λ to be

$$\lambda_k = \frac{1}{\sum_m \tilde{a}(\boldsymbol{m}, \boldsymbol{k})},$$

which ensures that the partition of unity is satisfied by the inverse approximation as well.

4.5. Successive approximation

The first-order approximation of the texture mapping is

$$\boldsymbol{c}_s = \boldsymbol{A}^+ \cdot \boldsymbol{f}_t \cong \Lambda \boldsymbol{A}^\top \cdot \boldsymbol{f}_t. \quad (13)$$

This solution can be refined by successive approximation, where the error correction term is $\boldsymbol{B} = (\boldsymbol{I} - \tilde{\boldsymbol{A}}^+ \boldsymbol{A})$. We give the successive least-squares texture-mapping algorithm in pseudo-code language:

$\boldsymbol{c}_s^0 \cong \tilde{\boldsymbol{A}}^+ \cdot \boldsymbol{f}_t, \quad i = 0$ // initial approximation
DO $\quad i = i + 1$
$\quad \Delta \boldsymbol{f}_t = \boldsymbol{f}_t - \boldsymbol{A}\boldsymbol{c}_s^{i-1}$ // map back to get loss
$\quad \Delta \boldsymbol{c}_s = \tilde{\boldsymbol{A}}^+ \Delta \boldsymbol{f}_t$ // map the loss to screen
$\quad \boldsymbol{c}_s^i = \boldsymbol{c}_s^{i-1} + \Delta \boldsymbol{c}_s$ // correct the screen
WHILE ($\|\Delta \boldsymbol{f}_t\| > \epsilon$ and $\|\Delta \boldsymbol{f}_t\|$ decreasing)

Rearranging the formulas, we get the recursive form

$$\boldsymbol{c}_s^i = \boldsymbol{B} \cdot \boldsymbol{c}_s^{i-1} + \tilde{\boldsymbol{A}}^+ \cdot \boldsymbol{f}_t \quad (14)$$

and from it the close-form solution

$$\boldsymbol{c}_s^i = \boldsymbol{B}^i \cdot \boldsymbol{c}_s^0 + \sum_{k=0}^{i-1} \boldsymbol{B}^k \cdot \tilde{\boldsymbol{A}}^+ \cdot \boldsymbol{f}_t. \quad (15)$$

The term with the initial approximation $\boldsymbol{c}_s^0$ of the screen in (15) vanishes with increasing iterations i. The conditions for convergence are that the matrix $\tilde{\boldsymbol{A}}^+$is well conditioned, and that the eigenvalues of $\boldsymbol{B} = (\boldsymbol{I} - \tilde{\boldsymbol{A}}^+ \boldsymbol{A})$ are smaller than 1. Since the matrices $\boldsymbol{A}$ and $\boldsymbol{A}^+$ strongly depend on the geometric mapping T, we cannot guarantee that the algorithm will converge for all possible configurations.

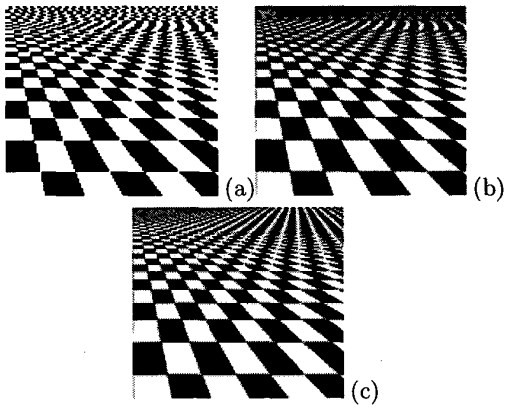

Figure 2: Mapped textures: (a) by nearest-neighbor method, (b) by trilinear mipmapping, (c) by successive refinement using linear-spline model and multiresolution version of the algorithm.

The inversion of the matrix $\boldsymbol{A}$ makes sense only in areas where the texture is shrunk while mapped to the screen. Otherwise, the texture mapping problem is ill-posed and the solution is manifold. In the underdetermined case, the texture is mapped by interpolation with the classical method instead (e.g., bilinear mipmapping). In the underdetermined area the normalisation weights λ_k are smaller than 1, because too few texels contribute to one screen pixel, typically less than 4 for the bilinear case ($N = 1$). Based on these two rules, we determine the area on which to successively refine the screen. A pixel that is close to the horizon maps to a large texture area. For such large texture reduction factor, we switch to a lower texture resolution (mipmapping) which reduces the computation cost.

5. RESULTS

To demonstrate our algorithm, we map a checkerboard onto an infinite plane. The results of the simple nearest-neighbor texture-mapping algorithm are shown in Fig. 2a. This image exhibits severe aliasing artifacts. Trilinear mipmapping—a good standard method—reduces these effects (Fig. 2b). The proposed method (Fig. 2c) avoids the aliasing effects because it implicitly acts like an optimal anti-aliasing filter. Here, the underlying continuous model for the successive-refinement method is a bilinear spline.

Area sampling corresponds to the first (and only) iteration of the proposed method, using a spline of degree $N = 0$ (nearest neighbor). For non-trivial degrees ($N > 0$), the successive-refinement method gets close to the optimal biorthogonal prefilter after several iterations. This filter resembles the filter of the EWA

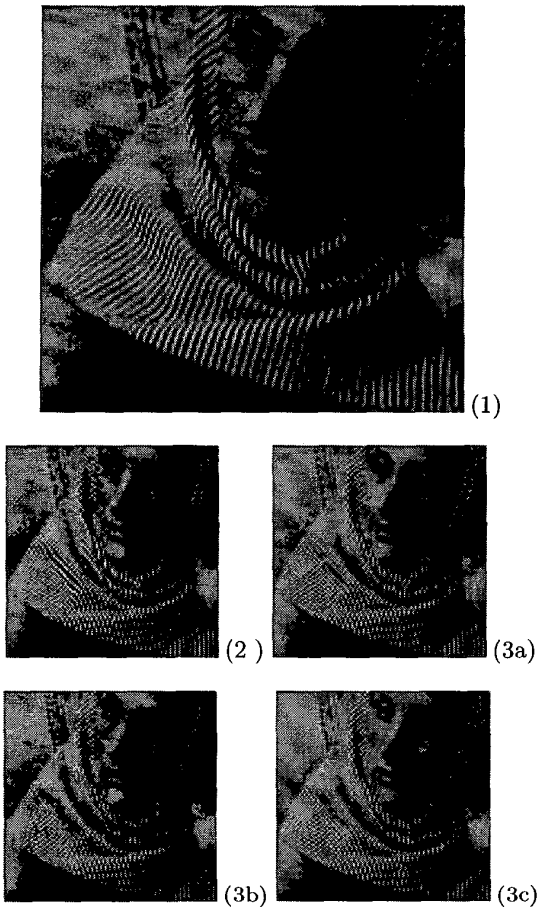

Figure 3: Reduction-by-two mapping: (1) Original texture, (2) Simple resampling (aliasing artifacts), (3) Successive refinement with: (3a) bilinear spline, (3b) bicubic spline, (3c) cubic Keys kernel.

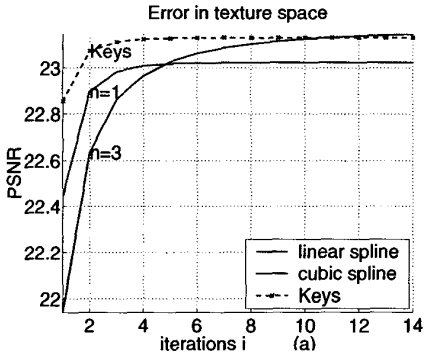

Figure 4: The reconstruction quality measured between backprojected texture and original texture increases with the number of iterations.

method, but tends to be much sharper. In addition, our method preserves the partition of unity.

To verify our method, we consider the simple reduction-by-two case because the l_2-optimal solution is known and can be obtained by alternative means [5]. Experimental results for different interpolation methods are shown in Figure 3. The cubic interpolators (Keys and B-spline) tend to give better results than the linear one. The PSNR in texture space is higher and the visual appearance is slightly better. The graph in Figure 4 gives the evolution of the error in texture space—the criterion optimized by the algorithm—and illustrates the convergence behavior. We note that the basis functions that are more localized (linear spline and Keys) converge in less iterations. The cubic spline needs more iterations but eventually outperforms the other methods. We verified that the algorithm converges to the l_2-optimal solution obtained by [5].

6. CONCLUSION

We have defined a novel way to solve the texture-mapping problem. The solution minimizes the loss of information and is optimal in the least-squares sense. The practical successive texture-mapping algorithm computes a first approximation of the pseudo inverse of the interpolation matrix $\boldsymbol{A}$ and then uses successive refinement to yield the optimal solution. Our method requires the geometric mapping T to be invertible. The images obtained are more detailed and reveal no aliasing artifacts. Moreover, if our screen model is piecewise constant, then the first iteration of our method is equivalent to area sampling.

7. REFERENCES

[1] D. Cohen-Or, "Exact anti-aliasing of textured terrain models", *Int. Journal of Computer Graphics*, vol. 13, no. 4, pp. 184-198, 1997.

[2] P.S. Heckbert, "Survey of texture mapping," *IEEE Computer Graphics and Applications*, vol. 6, no. 11, pp. 56-67, 1986.

[3] F. Jordan, S. Horbelt, T. Ebrahimi, "Streaming of photo-realistic texture mapped on 3D surface", *ICIP'97*, vol. 2, pp. 390-393, 1997.

[4] D.F. Rogers, *Procedural elements for computer graphics*, 2nd. Ed., McGraw-Hill, 1998.

[5] M. Unser, A. Aldroubi and M. Eden, "B-spline signal processing: Part II — Efficient design", *IEEE Trans. Signal Proc.*, vol.41, no.2, pp.821-848, 1993.

[6] L. Williams, "Pyramidal parametrics", *Siggraph'83*, vol. 17, pp. 1-11, 1983.