EUROGRAPHICS 2002 / G. Drettakis and H.-P. Seidel
(Guest Editors)

Volume 21 (*2002*), *Number 3*

Space-Optimized Texture Maps

Laurent Balmelli Gabriel Taubin Fausto Bernardini

IBM Research
T.J. Watson Center
Hawthorne, NY, USA

Abstract

Texture mapping is a common operation to increase the realism of three-dimensional meshes at low cost. We propose a new texture optimization algorithm based on the reduction of the physical space allotted to the texture image. Our algorithm optimizes the use of texture space by computing a warping function for the image and new texture coordinates. Neither the mesh geometry nor its connectivity are modified by the optimization. Our method uniformly distributes frequency content of the image in the spatial domain. In other words, the image is stretched in high frequency areas, whereas low frequency regions are shrunk. We also take into account distortions introduced by the mapping onto the model geometry in this process. The resulting image can be resampled at lower rate while preserving its original details. The unwarping is performed by the texture mapping function. Hence, the space-optimized texture is stored as-is in texture memory and is fully supported by current graphics hardware. We present several examples showing that our method significantly decreases texture memory usage without noticeable loss in visual quality.

1. Introduction

Texture images are a simple, yet effective way to increase realism of polygonal meshes at low cost. A recent review of the texture pipeline in real-time rendering systems can be found in [13]. Most current hardware graphics accelerators provide a memory cache to store texture images. Images in the cache can be accessed by the Graphics Processing Unit (GPU) through a high-bandwidth connection, improving frame rate. However, the size of the cache is limited by its cost, especially in commercial game-oriented graphics boards and game consoles. Because not all textures used by an application can reside in memory at the same time, complex memory management algorithms need to be used. In high-end applications, such as flight simulators, where larger caches are available, the amount of textures employed increases proportionally, requiring the same careful allocation of texture memory.

Image compression can alleviate the problem of limited texture memory. However, to obtain interactive frame rates, the compressed texture must be decoded on the fly during rendering, which requires specialized hardware [6, 2, 17]. To optimally use the texture resources available on today's traditional graphics boards, several authors have looked at the problem of reducing the space required to store the texture image, without resorting to complicated encodings [5, 15].

Except for simple textures containing repetitive patterns, details are typically not uniformly distributed across the image. For example, models obtained with 3D scanning systems are often textured with images that contain a large amount of background pixels. Faces of humans and characters need high resolution details in areas such as the eyes and mouth, while the appearance of other regions can be captured with relatively fewer pixels per unit area.

In this paper we describe a novel algorithm that optimizes the space used by textures. Our method works by distributing frequency content uniformly across the image. In other words, the image is stretched in high frequency areas, whereas low frequency regions are contracted. We also take into account distortions introduced by the mapping onto the model geometry in this process. The resulting image can then be resampled at lower rate (shrunk) without loosing details in areas of high frequency content. Alternatively, the image can be subsampled more aggressively with a uniform loss of visual fidelity across its regions. Only the texture image and texture coordinates are affected by the optimization. Neither the model geometry nor its connectivity change. The

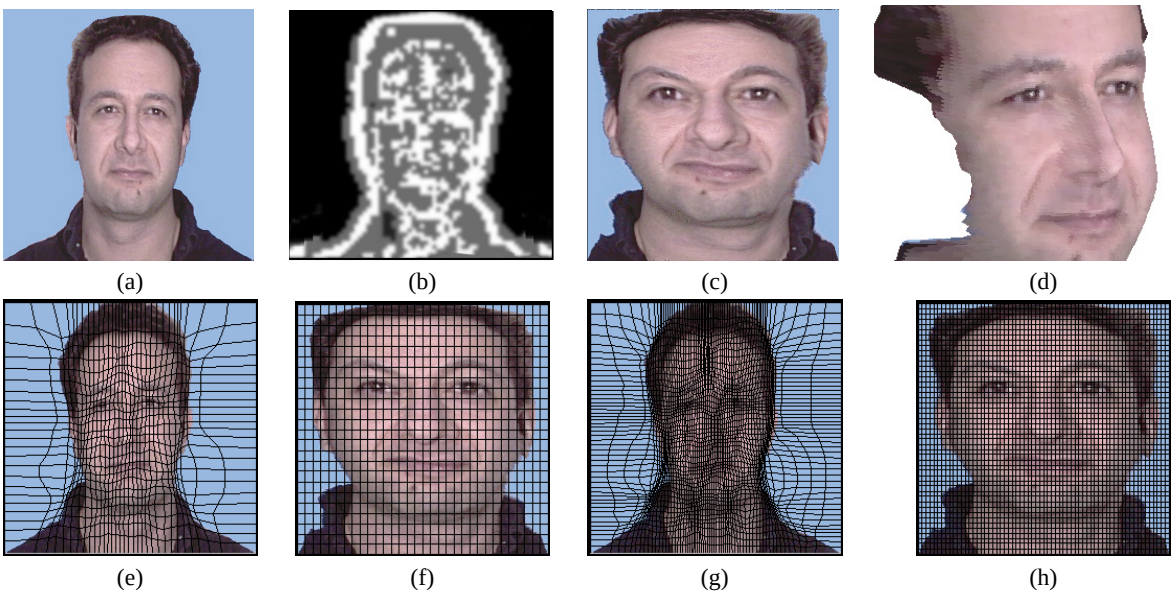

Figure 1: *Space-optimized texture maps: (a) Input texture. (b) Frequency map obtained from the multiscale analysis. (c) Warped and resampled texture. The texture is resampled at half the initial rate. Although shown here as large as the original, this image is actually only one quarter its original size. (d) Model mapped using the warped and resampled texture (see Section 6 for comparisons with the original.) (e)-(h) Multigrid warping: The figures show two pairs of images (I, I') using deformation grids of increasing density.*

smaller optimized image uses less texture memory, but does not require special hardware: Its "decompression" is performed by the warped texture mapping function computed in the optimization process.

Contributions In contrast with previous approaches, our method requires minimal user input. Our frequency map automatically captures the relative importance of different regions in the image. The map is efficiently and robustly computed using a wavelet packet decomposition technique and a denoising filter. We incorporate the effect of metric distortions by analyzing the Jacobian of the parametrization and combining it with the frequency map. The image warping is obtained with a simple multigrid relaxation algorithm. The final optimized image does not require special decompression hardware, and artifacts typical of compression techniques are not present.

2. Overview of the method

Our optimization algorithm consists of two steps: First, we compute a local *frequency map* for the image (Section 4). This map segments the image domain $Q = [0,1] \times [0,1]$ into regions of equal frequency. For example, Figures 1(a) and 1(b) show an input image and its frequency map. The frequency map is used to drive the relaxation algorithm in the following step.

In the second step (Section 5), we compute a warping function w for the texture, as depicted in Figure 2. We denote by $I'(u',v')$ the warped image. We use a relaxation algorithm, closely related to Laplacian smoothing, to evenly distribute frequency content across the image, stretching or shrinking regions according to the frequency map computed in the first step. We apply a multigrid scheme to deform

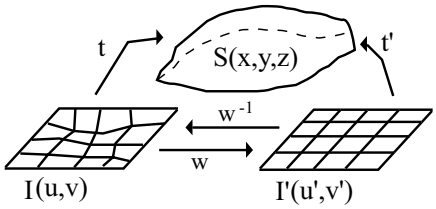

Figure 2: *Image warping: We compute a warping function $w : Q \to Q$ using a multigrid relaxation approach. The original image is mapped onto the geometry by a mapping function t. A coarse regular grid is overlaid on the original image. Its progressive deformation is driven by the frequency map of the image. Increasing grid resolutions are used in subsequent steps. The warping w is then applied to the image to create $I'(u',v')$. A new mapping function t' from the warped image onto the geometry is also computed, such that the visual appearance of the texture mapped model remains the same after the optimization.*

the image. Figures 1(e)-(h) show couples of images (I, I'), where I' is warped using grids of increasing size. Intuitively, the grid in I adaptively samples the frequency map. Hence, vertices of the grid concentrate in high frequency regions. After warping, the texture image is visually distorted (Figure 1c). However, the texture coordinates are appropriately recomputed in order to unwarp the image when it is mapped onto the surface (Figure 1d).

In the warped image, we expect the local spectrum to be uniform over the whole image on average. Since frequency content in the input image is not uniform, the target uniform frequency cannot fill the whole spectrum. Hence, the minimal sampling rate (*Nyquist frequency*) is lower on average and the image can be theoretically resampled without loss of resolution. As expected, this is not true in practice since sharp edges might not be accurately preserved. However, we show that the resampled warped image incurs much fewer distortion than its unwarped counterpart (Section 6). For example, the texture in Figure 1c is resampled at half the original rate. Hence, its size is reduced by 75%.

3. Previous work

Texture mapping optimization has been investigated during the past decade. Many researchers have studied the problem of finding a parametrization, i.e. a mapping from texture coordinates to mesh coordinates, which minimizes the distortion or equivalently *texture stretch*. Lévy *et al.* explain how to find a set of texture coordinates given an image and a surface[8]. They also propose an algorithm to compute a parametrization under constraints[7]. On the other hand, Sander *et al.* [14] propose a solution to obtain non-distorted sets of textures for a class of multiresolution meshes, known as Progressive Meshes. To minimize texture stretch, the texture coordinates are displaced in the image domain until an optimization criterion is fulfilled. These works are complementary to ours and can be applied as a preprocessing step.

Sloan *et al.* [15] and Hunter *et al.* [5] have studied a similar problem to ours from slightly different points of view. Sloan *et al.* address the problem of texture space optimization through a user-driven approach[15]. They propose to build an importance map for the texture, and suggest an automatic procedure for its computation, but give only few details. The specification by the user of an importance map in the context of a 3D painting application is discussed in more depth. The optimization technique used in this work is based on multilevel free-form deformation, and employs a metric that takes into account importance as well as metric distortion induced by the parametrization. The deformation is applied to the embedding of the 3D mesh onto the image domain defined by the texture coordinates. The authors report that a more robust integration of the different metrics is needed. An alternative approach described in the same paper uses a quadtree to partition the image in areas of uniform total importance, which are then resampled and packed as equal-size tiles into a texture atlas. This approach precludes the use of traditional mip-mapping. In contrast to this technique, our method uses a robust, wavelet-based technique to measure local frequency content. Frequency and parametric distortion metrics are easily integrated into a single frequency map. The relaxation method used to compute the warping, based on Laplacian smoothing, provides a continuous, smooth deformation that does not preclude the use of mip-mapping.

Cohen *et al.* propose an *image optimization method* based on frequency distribution. They compute a frequency map for the image using Fourier analysis and spectrum integration. They partition the image using a k-d-tree data structure, and resize each partition to equalize their frequency content. While the technique is shown to be effective in reducing image size while retaining good overall visual quality, the resulting optimized image contains discontinuities that introduce artifacts in rendering. The new texture also needs to be augmented by a hierarchy of quadrilaterals used to undo the warping during rendering. Although the unwarping uses only traditional texture mapping operation, it introduces extra computations in the pipeline. In contrast, our frequency analysis method is based on multi-scale, wavelet packets approach allowing us to control resolution both in space and frequency. Such a flexibility cannot be achieved using Fourier analysis. Our optimized texture image does not require any additional data structures or extra computations during rendering. The warping is continuous and smooth, minimizing distortion artifacts.

Our algorithm optimizes the use of texture space and allows for reduced memory usage. Hence, it can be related to compression methods. Texture compression techniques using new graphics hardware architectures have been proposed[2, 17]. Simple compressed ecodings have been proposed to reduce the cost of such additional hardware[6], but they have been shown to introduce visible quantization artifacts in some applications. In contrast, our approach does not need any particular hardware and produces gracefully degrading image quality for reduced image sizes.

4. Local frequency map

In this section, we explain how we compute a local frequency map for the texture image. We first explain how the frequency content is accurately measured (Section 4.1). Then, we show how the deformation induced by the parametrization is taken into account in order to obtain the frequency map (Section 4.2). Finally in Section 4.3, we give the algorithm to obtain a frequency map taking into account both the measured characteristics of the texture image and the deformation induced by the parametrization.

4.1. Wavelet packet decomposition

In this section, we use standard notations in discrete image processing: The texture image is denoted by $I(e^{j\omega})$ in Fourier domain. Note that $\omega = (\omega_1, \omega_2)$ since the Fourier domain is two-dimensional.

We use a wavelet packet (WP) expansion[18] in order to analyze the frequency content in the input image. This transform is a standard tool in image processing and decomposes an image into a set of coefficients characterizing frequency subbands. It is usually implemented using an iterated filter bank like the one represented in Figure 3. In the figure, H_0

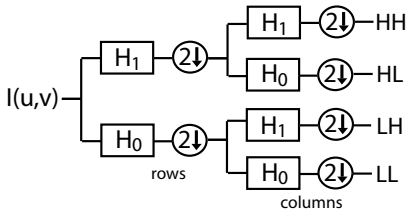

Figure 3: *Image filter bank: The filter bank takes an image $I(u,v)$ as input and outputs four matrices of coefficients HH, HL, LH and LL. The boxes H_0 and H_1 denote convolution using low-pass and high-pass filters, respectively. Finally, $2\downarrow$ denotes the downsampling operator (by two). All these notations are standard in the filter banks literature*[18].

denotes the low-pass filter, whereas H_1 denotes the high-pass filter. First, each filter is convolved with the rows of the image and the result is downsampled by two. Then, the filters are applied to the image columns and the outputs are again downsampled by two. The two steps above represent one iteration on the image. After one iteration, we obtain four sets of coefficients denoted by HL, LH, HH and LL. The set LL contains low-pass coefficients, whereas HL, LH and HH contain detail coefficients. Sets HL and LH correspond to horizontal and vertical high frequencies. The set HH is a combination of details in both dimensions. Consider the input image in Figure 1: after two iterations each coefficient matrix is four times smaller than the original image. Hence, it is common to group them in a matrix having the same size as the input image (Figure 4).

The filters H_0 and H_1 are chosen to be half-band filters. We use a pair of 17-tap biorthogonal (i.e. FIR) filters providing good frequency and space localization[18]. Hence the image spectrum $I(e^{j\omega})$ is split between coefficients in subband $[0,\pi/2]$ (LL) and subband $[\pi/2,\pi]$ (HH, HL and LH). An octave-band decomposition is obtained by iterating on the LL subband. In the WP expansion, the operation is iterated on all subbands. Hence, the image spectrum $I(e^{j\omega})$ is split linearly (a one-dimensional analogy is depicted in Figure 5).

In order to obtain an accurate measurement of the local frequency, we use a three-level WP decomposition. Hence,

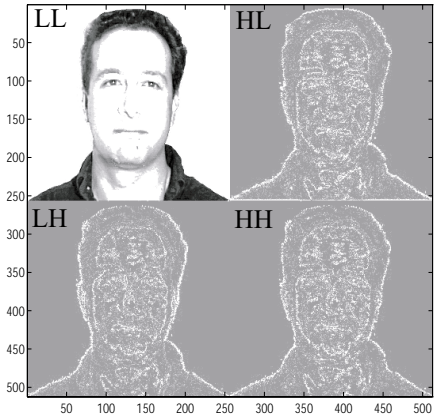

Figure 4: *One filter bank iteration: After applying the filter bank depicted in Figure 3 to the input image in Figure 1a, four matrices of coefficients (LL, HL, LH and HH) are obtained. These matrices are grouped in a matrix having the same size as the original image.*

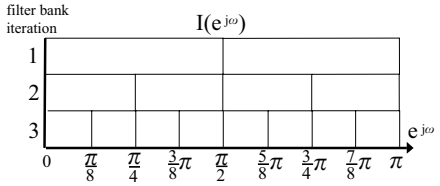

Figure 5: *Spectrum subdivision after a three-level packet transform: The linear subdivision is suggested for a one-dimensional signal $I(e^{j\omega})$ using ideal filters. A three-level packet transform yields 8 subbands.*

the image spectrum is split into 8 subbands. Since our image is two-dimensional, we have actually 64 matrices of coefficients since three detail coefficient matrices per subband are obtained after each iteration. In the next section, we explain how the measurements obtained using the WP decomposition are scaled by taking into account the deformation of the parametrization, i.e. by taking into account the model geometry.

4.2. Surface parametrization

Given a set of coordinates $(u,v) \in \Omega \subset Q$, where Q is the unit square $[0,1] \times [0,1]$, and a surface given by a set of points $(x,y,z) \in S \subset \mathbb{R}^3$, a parametrization t is the continous function:

$$(u,v) \in \Omega \mapsto t(u,v) = (x(u,v), y(u,v), z(u,v)) \in S. \quad (1)$$

The mapping $t(u,v)$ is a one-to-one transform. It is common to choose (u,v) such that their ranges span the texture domain. Hence, the texture image is used as the parametriza-

tion domain for the surface (or a surface *patch* in the case of multiple parametric domains.)

The parametrization function (1) maps a set of points in $\mathbb{R}^2$ to set of points in $\mathbb{R}^3$. The mapping distortion is measured using the 3×2 Jacobian matrix

$$J = \left[\frac{\partial t}{\partial u} \quad \frac{\partial t}{\partial v} \right]. \tag{2}$$

The minimum and maximum distortions of the parametrization are given by the singular values of J. These values are equal to the singular values of $J^T J$, where ^{T} is the transpose operator. Moreover, they can be written in closed-form.

Until now we assumed t to be a continous parametrization function. In practice, the mapping between texture domain and surface domain is defined using discrete sets of values: the texture and mesh coordinates. The surface defined by the polygonal mesh is piecewise linear. Hence, we use an approximation for t returning one set of singular values per mesh polygon (see [14] for example.) We can then compute one value per vertex by averaging values of incident faces and obtain a piecewise continous field by bilinear interpolation.

If the parametrization is not distorted by the mapping, then both singular values equal to 1. When any singular value is larger than 1, then the parametrization is stretched. In contrast, if any singular value is smaller than 1, then the parametrization is contracted. We are interested in the latter case. Let us call ϕ the minimum singular value. If $\phi < 1$ then the sampling frequency on the surface is decreased, i.e. spatial resolution is reduced. A well-known consequence is aliasing caused by undersampling of the texture image. The *uncertainty principle* states that sharpness in space is traded off for sharpness in frequency. Hence, whenever the parametrization is shrunk, the mapping increases the local frequency in the texture image.

4.3. Algorithm for local frequency measurement

In this section, we explain how a frequency map for the texture is computed using a hierarchical approach. Recall that we must also take into account the distortion introduced by the parametrization to correctly compute the frequency map. This will be explained at the end of this section.

Typically, high frequency regions contain edges and small details, whereas low frequency zones contain low variations in intensity. In order to detect frequency changes accurately, we perform a multiscale analysis of the input image. Mallat *et al.* have shown that the local image regularity is characterized by the decay of the wavelet transform amplitude across scales[11]. Hence, edges can be efficiently detected by following local maxima in detail coefficient matrices. These values are known as *wavelet maxima*[11].

The algorithm is based on the following general idea: For each pixel $I(u,v)$, we test the existence of a wavelet maximum within an area around the corresponding wavelet coefficient. If such a maximum exists, the existence is recorded in the frequency map, denoted by $M(u,v)$. The WP transform allows maxima to be found at multiple scales (i.e. frequencies), leading to a classification of the image into frequency region. We explain the algorithm in more details below. Note that the classification is performed only in regions covered by the mesh, i.e. used for texturing. Unused regions are set to the smallest classification value.

Recall that at each step, the WP expansion splits the spectrum of the input coefficients into two subbands (Figure 5). After the first iteration, the image is transformed into a set of low-pass coefficients (describing frequencies between 0 and $\pi/2$), and three sets of detail coefficients (representing frequencies between $\pi/2$ and π). For each set of detail coefficients (i.e. HL, LH and HH), wavelet maxima are measured using a threshold value estimated from the variance in the corresponding subband[10]. Hence at step 1, pixels are sorted into two sets and the results are stored in a matrix $M^1(u,v)$.

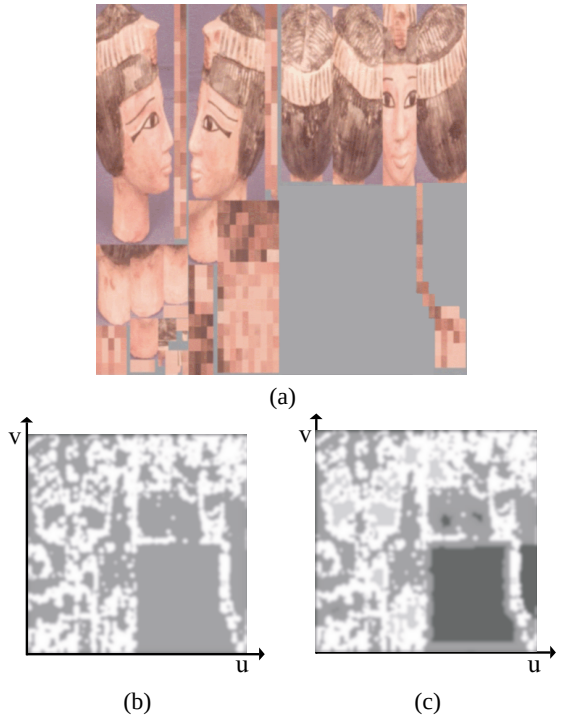

Figure 6: *Local frequency content measurement: (a) The input image is a texture atlas. (b) The map $M^1(u,v)$ (Eq. 3) segregates the pixels $I(u,v)$ into two classes of frequencies. The matrix $M^2(u,v)$ (Eq. 4) is computed by refining $M^1(u,v)$. In order to do this, the previous classification results are sorted again into four subclasses. Results of applying the optimization to this image are discussed in Section 6.*

The matrix is defined as

$$M^1(u,v) = \begin{cases} 1, & \text{if a local maxima was found,} \\ 0.5, & \text{otherwise.} \end{cases} \quad (3)$$

In Figure 6b, we show the two-color classfication for the input image in Figure 6a. After the second iteration, each subband has been again split into two half-bands. Hence $M^1(u,v)$ can now be refined: Each previous entry is sorted by looking for wavelets maxima in the corresponding detail matrices. The new matrix $M^2(u,v)$ is then defined as

$$M^2(u,v) = \begin{cases} 1, & \text{when } M^1(u,v) = 1, \text{ local maxima,} \\ 0.75, & \text{when } M^1(u,v) = 1, \text{ no local maxima,} \\ 0.5, & \text{when } M^1(u,v) = 0.5, \text{ local maxima,} \\ 0.25, & \text{when } M^1(u,v) = 0.5, \text{ no local maxima.} \end{cases} \quad (4)$$

In Figure 6c, we show the four-color refinement of the map $M^1(u,v)$. Recall that we use a three-level WP expansion. Therefore, our final frequency map $M(u,v) = M^3(u,v)$ has 8 different intensities.

The frequency map $M(u,v)$ segments the input image into frequency region. However due to the parametrization (1), the frequency in the texture image may vary as explained in Section 4.2. To compute a frequency map for the texture, we construct a piecewise continuous field as described in Section 4.2 using the minimal singular value ϕ. Call $\phi(u,v)$ this field, then the modulated frequency map is given by the ratio $\sigma(u,v) = M(u,v) \cdot \phi(u,v)$. Moreover, we have

$$\sigma(u,v) = \begin{cases} > 1, & \text{if the texture image is undersampled,} \\ < 1, & \text{if the texture image is oversampled.} \end{cases} \quad (5)$$

5. Relaxation algorithm

Let $Q = [0,1]^N$ be the N-dimensional unit cube. In this paper we are interested in the case $N = 2$, but the formulation and algorithm are the same independently of the dimension. Our decision to treat the general case despite the additional complexity is based on the many potential applications that we envision for this algorithm beyond the needs of this paper.

In this section a *warping function* means a C^1-continuous function $w : Q \to Q$, with *inverse warping function* $\eta = w^{-1} : Q \to Q$ of the same class. In particular both w and η are one-to-one and onto. When restricted to the boundary ∂Q of cube, they both define one-to-one and onto continuous functions $\partial Q \to \partial Q$ as well. The points in the domain of w will be denoted ν, and those in the domain of η will be denoted ν'.

Very often not all the texture image is used to texture the surface. That is, the texture coordinates are defined on a smaller (closed) subset of the interior of the unit square $\Omega \subset Q^\circ$, the border of the texture image is not used for texturing the surface, and so, larger distortions are acceptable close to the boundaries (the image of the set Ω through the warping function w will be denoted Ω'). This is why in our current implementation we impose an additional restriction: the warping and inverse warping functions keep the corners of the square fixed. This restriction does not seem to create major artifacts, and simplifies the implementation. In general, our warping functions satisfy Neuman boundary conditions

$$\left.\frac{\partial w}{\partial n}\right|_{\partial Q} = 0\,,$$

i.e., at boundary points the partial derivatives are nonzero only along directions aligned with the boundary. Note that this constraint removes only one degree of freedom for most boundary points, but some boundary points are more restricted, because the boundary is only piecewise smooth. For example, the corners of the cube have zero degrees of freedom (i.e., they cannot move).

Given a continuous frequency map $\sigma : \Omega \to \mathbb{R}_+$, the problem we are facing is to construct a warping function w so that $\sigma(\nu)$ measures the *stretching* of w at the point ν. That is, a point at distance d from ν in Ω should be mapped by w to a point at distance $d \cdot \sigma\nu$ in Ω'. The stretching will cause regions where $\sigma < 1$ to contract, while regions where $\sigma > 1$ will expand. Since this problem can only be solved approximately, we introduce a variational formulation, and a discrete relaxation algorithm to solve it.

The warping functions that we use in our current implementation are defined by a rectangular grid and bilinear interpolation within the quadrilateral faces of the grid. We initialize the algorithm with a regular grid G with vertices $V = \{\nu_1, \ldots, \nu_K\}$ covering the unit square Q. We can choose the number of samples along each axis independently. Choosing rows and columns approximately proportional to the width and height of the input texture image produces approximately square faces on the texture image, and works well in practice.

We relax the grid by minimizing an energy function. This relaxation procedure should move the vertices to new locations so that the distance $|\nu_i - \nu_j|$ between two grid vertices ν_i and ν_j connected by a grid edge $e = (i,j)$ is approximately inversely proportional to the value of σ at the edge midpoint. Since we assume σ to be continuous and sufficiently smooth, we approximate this value by the geometric average $(\sigma(\nu_i)\sigma(\nu_j))^{1/2}$, and we minimize this energy function

$$E_D = \sum_{e=(i,j)} \sigma(\nu_i)\sigma(\nu_j)\,|\nu_i - \nu_j|^2\,. \quad (6)$$

We want to emphasize the fact that this is a nonlinear function of the grid vertex positions. The simplest case is when the function σ is constant. In this case E_D is a quadratic function of the grid vertex positions, and easy to minimize because the boundary constraints are linear: this is a classical least-squares problem. But for the general case of arbitrary σ we first need to analyze existence and uniqueness of solu-

Figure 7: *Results of the relaxation algorithm applied to two different texture images. Note how background pixels are reduced to a very small area in the final optimized image. Also, areas of the images with high frequency content are expanded relatively more than regions with smooth color variations. The warping preserves continuity and smoothness in the image. The updated texture coordinates undo the warping. A traditional texturing pipeline can be used with these images (see also Color Plates.)*

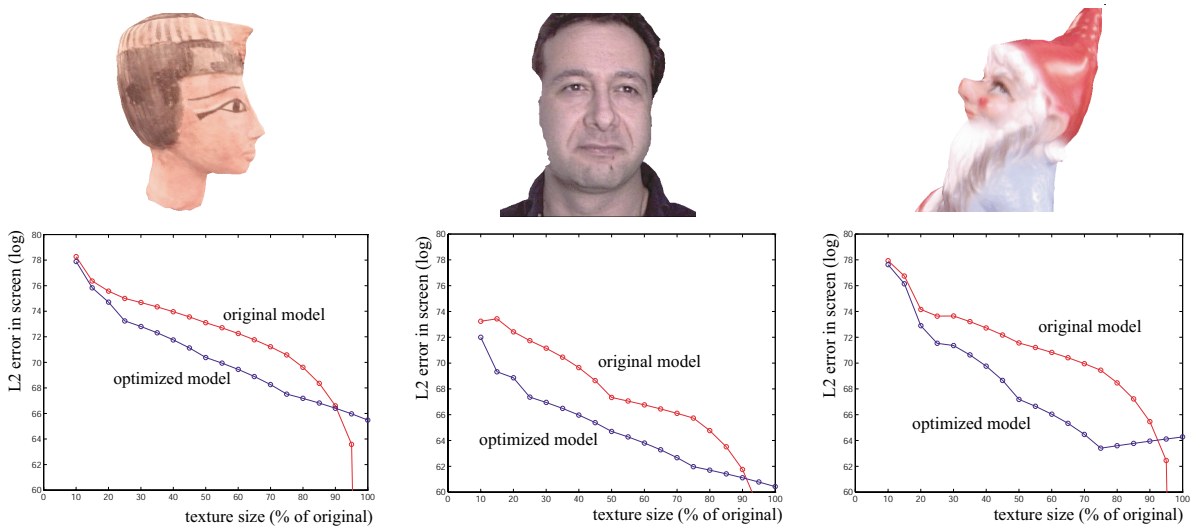

Figure 8: *These rate-distortion curves compare the error introduced by subsampling the images with and without our optimization for the three models shown in the top row. Details are discussed in Section 6.*

tions, and then develop an algorithm to efficiently compute the minimizer.

This is why we have chosen this energy function. Since we relax the grid covering the domain Ω of the warping function w, our relaxation procedure in fact estimates the inverse warping function η, whose domain is covered by the initial regular grid ($v_i = \eta(v'_i)$). Because of the regularity of the grid with vertices v'_i, in this case the distance between two warped vertices connected by an edge $|v_i - v_j|$ is equal to the absolute value of a finite difference approximation of a partial derivative of the inverse warping function with respect to one of the N coordinate axes $|\eta(v'_i) - \eta(v'_j)|$. It turns out that the energy E_D (6) is a discretization of the integral

$$E_C(\eta) = \int_Q \sigma(\eta)^2 \, ||D\eta||^2 \, dv' \quad , \tag{7}$$

where $D\eta$ is the Jacobian matrix of η, and $||D\eta||$ is the Frobenius norm (square root of sum of squares elements) of this matrix, i.e.,

$$\|D\eta\|^2 = \sum_{i,j} \left(\frac{\partial \eta_i}{\partial v_j} \right)^2 .$$

It is not a very difficult problem in calculus of variations to prove that the minimizer $\hat{\eta}$ of E_C for $N = 1$ satisfies

$$\sigma(\hat{\eta}(v')) \, \|D\hat{\eta}(v')\| = \text{constant}.$$

This is not necessarily true for $N > 1$, but the behavior of the minimizer is similar. In addition, the minimizer exists, and is unique, but we postpone a detail analysis of the nonlinear PDE problem associated with the minimization of (7) and our discretization to a forthcoming paper.

Our relaxation algorithm is a successive approximation algorithm where the new position of a vertex v_i^{n+1} is deter-

mined by adding a displacement to the corresponding current position v_i^n. A related algorithm was proposed by Li [9] in the context of adaptive mesh generation for finite elements computations. The displacements are computed collectively as a continuous function of the current positions $\delta(v)$. In our case, we define the displacements by equating the partial derivatives to zero, i.e., formulating it as a descent algorithm.

We first compute unconstrained displacement vectors

$$\delta_U(v)_i = \sum_{j \in i^\star} \mu_{ij}(v_j - v_i) - \left\{ \sum_{j \in i^\star} \mu_{ij} |v_i - v_j|^2 \right\} \frac{\nabla\sigma(v_i)}{2\sigma(v_i)},$$

where $i^\star$ is the set of grid vertex indices connected to i by a grid edge, and

$$\begin{cases} \mu_i &= \sum_{j \in i^\star} \sigma(v_j), \\ \mu_{ij} &= \sigma(v_j)/\mu_i \,. \end{cases}$$

These displacements are independent of scaling of the frequency map: If we replace $\sigma(v)$ by $\lambda\sigma(v)$, for some positive λ, we obtain the same displacements, which is highly desirable. Another way of looking at this property is by observing that $\nabla\sigma/\sigma$ is equal to $\nabla \log(\sigma)$. But affine scaling of σ, or equivalently the ratio q between minimum and maximum values of σ in Q, affects the quality of the result. In our implementation, the frequency map is represented as an 8-bit gray level image, and the user can affinely scale the frequency map to the interval $[q, 1]$, with q accessible to the user through the program's user interface. Small values of q produce less grid distortion, and larger values produce more distortion. The target value for q is determined in the frequency map estimation phase.

Note that the weights μ_{ij} are non-negative, they add up to one in the neighborhood of each vertex $\sum_{j \in i^\star} \mu_{ij} = 1$, and are not necessarily symmetric $\mu_{ij} \neq \mu_{ji}$. Also observe that the first sum on the right hand side is nothing but the well known Laplacian operator from mesh signal processing [12, 16]. Laplacian smoothing tends to move each vertex to the weighted average of its neighbors, with weights inversely proportional to the target sample rate.

The second term is equal to zero if the vertex v_i is an extremum of σ, and by continuity, is very small in a neighborhood. In general, this term pulls the vertex v_i along a descent direction for σ. The ratio involving the gradient of $\sigma(v_i)$ is discretized as a finite differences approximation in the direction of the neighbor h of i in which the ratio varies fastest. That is

$$\frac{\nabla\sigma(v_i)}{2\sigma(v_i)} \approx \frac{(\sigma(v_h) - \sigma(v_i))}{(\sigma(v_h) + \sigma(v_i))} \frac{v_h - v_i}{|v_h - v_i|^2},$$

where $h \in i^\star$ is the neighbor of i corresponding to the maximizer of

$$\max \left\{ \frac{|\sigma(v_j) - \sigma(v_i)|}{(\sigma(v_j) + \sigma(v_i))} \frac{1}{|v_j - v_i|} : j \in i^\star \right\}.$$

Note that this discretization produces an appropriate scaling, because

$$|\sigma(v_h) - \sigma(v_i)| \leq |\sigma(v_h) + \sigma(v_i)| \,,$$

and the following two expressions

$$\begin{cases} \sum_{j \in i^\star} \mu_{ij} |v_i - v_j|^2 / |v_h - v_i|, \\ \sum_{j \in i^\star} \mu_{ij} |v_i - v_j|, \end{cases}$$

are of the same order of magnitude.

Then we apply the (linear) boundary constraints. The four corners of the unit square (which correspond to four vertices of the grid) do not move, and other boundary vertices are constrained to only move along their supporting straight boundary segments. The results are constrained displacement vectors $\delta(v_n)_i$, which are subsequently applied to the current vertex positions

$$v_i^{n+1} = v_i^n + \delta(v_n)_i \;\; i = 1, \ldots, N \,.$$

Let $\Xi = Q^K$, i.e., the cartesian product of K copies of the unit cube. And let $f : \Xi \to \Xi$ be the continuous function defined by the relaxation rules just introduced. Since Ξ is compact and convex, by Brouwer fix point theorem[4], f has a fixed point. If f is a contractor, then by Banach fix point theorem [1] it has a unique fixed point, and our successive approximation algorithm converges to it from any starting point (the function f is a contractor, or contraction map, if there exists a constant $K \in [0, 1)$ such that $|f(v) - f(v') \leq K|v - v'|$ for all v and v' in Ξ). Unfortunately, f does not seem to be a contractor in the general case. The relation between the initial grid sampling rate and the frequency content of the frequency map σ play a crucial role. But it can be shown that each local minimum has a neighborhood where f is a contractor [3].

All of this means that to ensure convergence to the global minimum, the initial grid has to be very fine. Again, the sampling rate of the initial grid has to be finer than the rate determined by the Nyquist frequency of the map. If the shortest wavelength of the frequency map σ is shorter than the distance between two vertices v_i and v_j connected by a grid edge, it is clear than there is no guarantee of convergence to any meaningful result. But a very fine grid implies slow convergence. So, we follow a multigrid approach with a hierarchy of grids, where the coarse iteration is performed on a coarser grid and an appropriately downsampled frequency map. In our current implementation both the grid and the frequency map can be upsampled and downsampled interactively. Finally, in Figure 7 we show the results of the relaxation algorithm when applied to two texture images.

6. Experimental results

We demonstrate the advantages of using our texture space optimization technique on several models. We compare the results of using progressively smaller, space-optimized

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

Figure 9: *A comparison of the visual quality obtained at reduced texture image sizes without (top row) and with (bottom row) space optimization. (a) Original texture size. In (b), (c) and (d), the texture is reduced at 30%, 20% and 10% of the original size, repectively (See also Color Plates.)*

textures versus using non-optimized textures of the same reduced size. We report quantitative results using rate-distortion curves (Figure 8), and illustrate the improved visual fidelity by comparing a model rendered with various texture sizes with and without optimization (Figure 9.)

Figure 8 shows rate-distortion curves for three different textured models. The curves illustrate the error introduced by subsampling the images with and without our optimization (respectively blue and red curves). The horizontal axis measures the size of the subsampled image as a percentage of the original. Going from right to left, the texture is progressively shrunk. The vertical axis measures error (on a logarithmic scale) according to the following simple metric: The image is applied as texture to the model and rendered from a fixed viewpoint (a sample rendering is shown in the top row for the three models.) The viewpoint is chosen to have all the significant texture details on screen. A pixel-by-pixel difference image and its L_2-norm are computed and plotted in the diagrams. The optimized image produces smaller errors for basically all image sizes, except for very small (less than 10%) shrinkage factors, where small artifacts introduced by the warping outpower the better use of space.

A comparison of the visual quality obtained at reduced texture image sizes, with and without space optimization, is shown in Figure 9. The top row shows renderings of a head model. From left to right, the rendered image is produced using a texture of the original size, and then reduced at 30%, 20% and 10% of the original size. The bottom row shows the same model rendered using a space-optimized texture. The images are rendered at a screen resolution that approximately matches the original texture size. It is evident that pixelization artifacts are much more visible with the unoptimized textures, especially at 30% and 20% of the original size. Even at 10% of the original size, the optimized image preserves some high-frequency details that are absent in unoptimized version.

7. Conclusion

We have tested our texture optimization algorithm on a variety of datasets, taken from collections of textured 3D models, or scanned by the authors. We have obtained very good results in most cases. As in previous approaches, and for obvious reasons, this technique does not lead to spectacular results with small textures containing repetitive patterns, such as grass or bricks. In contrast, significant savings in space can be obtained by optimizing textures produced by many 3D scanning systems, which contain large areas of background pixels. We have also obtained very good results on textured models of heads and characters, where details are

concentrated in a few localized areas. The main advantages of our technique with respect to prior art can be summarized as follows:

- The image is analyzed automatically, with a robust, efficient technique based on wavelets.
- Metric distortion introduced by the parametrization can be integrated by simply combining the measurement of frequency content with simple properties of the Jacobian of the parametrization.
- Image warping is implemented as a relaxation algorithm with nice convergence and stability characteristics.
- The warping preserves image continuity and smoothness, allowing the use of a traditional texturing pipeline, including mip-mapping.

More work remains to be done to quantitatively characterize the smaller error introduced with respect to unoptimized images. It would be interesting to employ metrics that take into account human perception. Finally, to better preserve sharp edges, we plan to investigate the use of anisotropic filters for image warping and resampling.

References

1. S. Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 2:133–181, 1922.
2. A. C. Beers, M. Agrawala, and N. Chaddha. Rendering from compressed textures. *Proc. of SIGGRAPH*, pages 373–378, August 1996.
3. I. N. Bronshtein and K. A. Semendyayev. Handbook of mathematics. *Van Nostrand Reinhold, New York*, 1985.
4. L.E.J. Brouwer. Über abbildung von mannigfaltigkeiten. *Mathematische Annalen*, 71:71–97, 1910.
5. A. Hunter and J. D. Cohen. Uniform frequency images: Adding geometry to images to produce space-efficient textures. *Proc. of IEEE Visualization*, pages 243–250, 2000.
6. S3 Inc. S3TC DirectX 6.0 standard texture compression. `http://www.s3.com/savage3d/s3tc.html`, 1998.
7. B. Levy. Constrained texture mapping for polygonal meshes. *Proc. of SIGGRAPH*, pages 417–424, August 2001.
8. B. Levy and J-L. Mallet. Non-distorted texture mapping for sheared triangulated meshes. *Proc. of SIGGRAPH*, July 1998.
9. S.Z. Li. Adaptive sampling and mesh generation. *Computer Aided Design*, 27(3):235–240, March 1995.
10. S. Mallat. *A wavelet tour of signal processing*. Academic Press, 24-28 Oval Road, London NW1 7DX, UK, 1999.
11. S. Mallat and S. Zhong. Characterization of signals from multiscale edges. *IEEE trans. on Pattern Analysis and Machine Intelligence*, 14(7):710–732, July 1992.
12. J.L. Mallet. Discrete smooth interpolation in geometric modeling. *ACM-Transactions on Graphics*, 8(2):121–144, 1989.
13. T. Möller and E. Haines. *Real-Time Rendering*. A. K. Peters, 1999.
14. P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. *Proc. of SIGGRAPH*, pages 409–416, 2001.
15. P. J. Sloan, D. Weinstein, and J. Brederson. Importance driven texture coordinate optimization. *Proc. of Eurographics*, 17(3), 1998.
16. G. Taubin. A signal processing approach to fair surface design. *Proc. of SIGGRAPH*, pages 351–358, March 1995.
17. J. Torborg and J. T. Kajiya. Talisman: Commodity realtime 3D graphics for the pc. *Proc. of SIGGRAPH*, pages 353–363, August 1996.
18. M. Vetterli and J. Kovačević. *Wavelets and subband coding*. Prentice Hall PTR, Englewood Cliffs, New Jersey 07632, 1995.