Draft version. The original paper appeared in EGSR '06 proceedings.

Feature-aware texturing

Ran Gal, Olga Sorkine and Daniel Cohen-Or

School of Computer Science, Tel Aviv University, Israel

Abstract

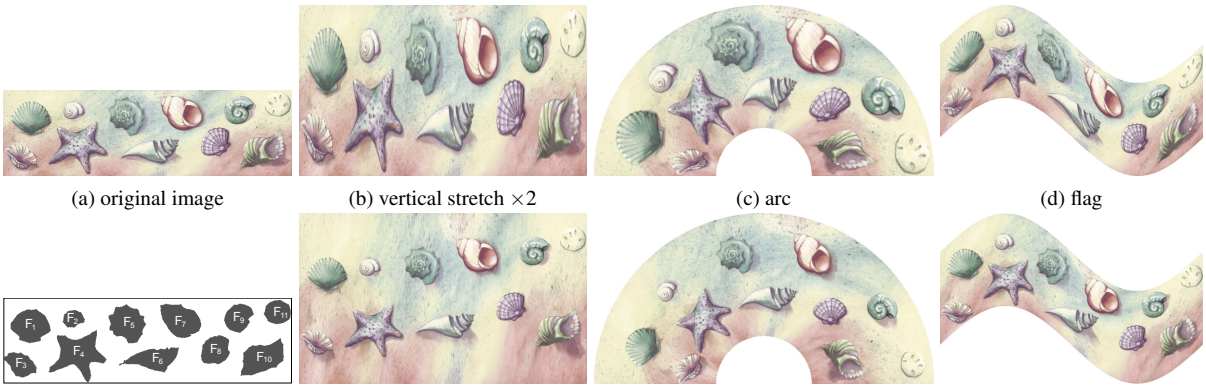
We present a method for inhomogeneous 2D texture mapping guided by a feature mask, that preserves some regions of the image, such as foreground objects or other prominent parts. The method is able to arbitrarily warp a given image while preserving the shape of its features by constraining their deformation to be a similarity transformation. In particular, our method allows global or local changes to the aspect ratio of the texture without causing undesirable shearing to the features. The algorithmic core of our method is a particular formulation of the Laplacian editing technique, suited to accommodate similarity constraints on parts of the domain. The method is useful in digital imaging, texture design and any other applications involving image warping, where parts of the image have high familiarity and should retain their shape after modification.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Color, shading, shadowing, and texture

1. Introduction

In 2D texture mapping applications, images are mapped onto arbitrary 2D shapes to create various special effects; the texture mapping is essentially a warp of the texture image, with constraints on the shape of the boundary or possibly the interior of the image as well. Such texture mapping is common in graphical design and publishing tools, as well as 2D and 3D modeling and animation applications. Commercial design tools usually provide a library of predefined warps, where the user only needs to select the desired mapping type and possibly tune a few parameters (see Figures 1, 2 for examples). Another option is to interactively design the texture map by selecting and transforming points or curves on the original image; the mapping is computed so as to accommodate such user constraints [BN92,LCS95,MJBF02]. It is also possible to apply free-form deformations with grid-based controls [SP86, LCS95, MJ96]. Texture manipulation in 2D is commonly applied by modelers when texturing 3D models: the texture map often needs to be adjusted and aligned to match particular features of the 3D surface. Constrained texture mapping methods have been developed for this purpose [ESG01, LÓ1, KSG03], where the user supplies point correspondences between the texture and the 3D model, and a suitable mapping is computed automatically.

Figure 1: *Top: standard image mapping result. Note the squeezing of the legs and the stretching of the heads. Bottom: our feature-aware mapping result; the proportions of the children are preserved.*



(a) original image (b) vertical stretch $\times 2$ (c) arc (d) flag

(e) feature map (f) feature-aware stretch (g) feature-aware arc (h) feature-aware flag

Figure 2: *Several common 2D mapping functions. The top row displays the standard mapping result; the bottom row shows the result of our method, constraining the features to retain their shape while striving to reproduce the input mapping as closely as possible. The feature map is shown in (e), where the detected connected components are numbered* $F_1, \ldots, F_{11}$.

Most image mapping and manipulation techniques treat the entire texture image homogeneously. When the deformation applied to an image introduces shearing, e.g. in the simplest situation where the aspect ratio of an image is altered by non-uniform scaling, all the image features are distorted. This may be disturbing when the image contains features with highly familiar shape, such as humans, animals, prominent geometric objects, etc. A typical example of a simple image transformation is shown in Figure 1, where the shear and stretch effects distort the images of the children in a quite unsatisfactory manner.

In this paper, we introduce an *inhomogeneous* 2D texture mapping method that is capable of preserving the shape of masked regions of the texture while warping the image according to the user specifications. This feature-aware texture mapping is guided by a feature mask defined by a rough selection of the features; in the mapping result, these features will undergo solely a similarity transformation, possibly at the expense of the background regions in the texture that are allowed to deform more. Our work is related to the texture optimization techniques of Balmelli et al. [BTB02] and Sander et al. [SGSH02], where the texture map is warped to allow higher pixel budget for the high-frequency details of the texture image.

At a first glance, it seems that a feature-preserving mapping could be achieved by cutting out the features, warping the rest of the image as desired and then pasting the features back and adjusting their orientation and scale. However, this poses several difficulties: *(i)* precise segmentation of the features with correct alpha-mattes for subsequent seamless compositing is required; *(ii)* it is not clear how to prescribe the similarity transformation of the features; *(iii)* texture synthesis needs to be applied for the holes that are likely to form around the features; alternatively, the pasted features could overlap with parts of the warped texture, causing information loss. The above tasks are quite complex; moreover, the tuning of such an algorithm would require significant amount of user interaction. In contrast, our method does not require a highly accurate matte but rather a loose selection of the features, which can be done using standard selection tools. Our technique produces coherent, smooth image warps by drawing upon the recent machinery of differential representations and deformation techniques [ACOL00, IMH05, SMW06, Sor06].

2. Feature-aware mapping

We will first describe our feature-preserving texture mapping technique assuming that an input warping function $W : \mathscr{R}^2 \rightarrow \mathscr{R}^2$ is given. Assume that the input image is represented by a regular pixel grid of dimensions $m \times n$. We denote the grid of the input image by $G = (V, E, K)$, where $V = \{v_1, v_2, \ldots, v_N\}$ is the set of node positions ($N = mn$), $E = \{(i, j)\}$ is the set of directed edges between the nodes and K is the set of quad faces of the grid. Throughout the discussion we assume that G is a 4-connected quad grid, although the algorithm can be easily extended to any general meshing of the image. We assume that we know the values of the input mapping W on all the grid nodes v_i.

The user provides a feature mask that marks the parts of the image whose shape should be preserved. We denote the mask by $M = \{m_1, \ldots, m_N\}$, such that $m_i = 1$ if pixel i belongs to a feature and $m_i = 0$ otherwise. The feature nodes indices are thus $F = \{i \text{ s.t. } m_i = 1\}$. We partition F into its connected components: $F = F_1 \cup F_2 \cup \ldots \cup F_d$ (see Figure 2(e)). Our goal is to find a mapping of the original grid G that is as close as possible to the input warp W and respects the shape of the features specified by the mask M. We would like to preserve the shape of all the quads contained in the features, meaning that they should undergo solely a similarity or rigid transformation. Rigid transformation implies that the size of the features will be preserved, whereas

input image and its feature mask | vertical stretch × 2 | underlying grid

Figure 3: *Feature-preserving stretching. The mapping preserves the shape of the features at the expense of the background.*

a similarity transformation allows varying the size according to the warping function W. We leave the choice between rigid and similarity behavior up to the user.

We first devise a proper shape preserving transformation for each quad $Q = (v_{i_1}, v_{i_2}, v_{i_3}, v_{i_4})$ that has at least one node in F. We approximate $W(Q)$ with a rotation/similarity transformation, by taking the linear component of W and extracting the rotation from it by means of the polar decomposition. Specifically, denote $W(Q) = (v'_{i_1}, v'_{i_2}, v'_{i_3}, v'_{i_4})$; denote by $v = \frac{1}{4}\sum_{k=1}^{4} v_{i_k}$ the centroid of Q; the centered vertices are then $u_{i_k} = v_{i_k} - v$ (and similarly, u'_{i_k} for $W(Q)$). We can linearly approximate the homogeneous part of W on Q by

$$T_{W,Q} = [u'_{i_1}\ u'_{i_2}\ u'_{i_3}\ u'_{i_4}] \cdot [u_{i_1}\ u_{i_2}\ u_{i_3}\ u_{i_4}]^*, \qquad (1)$$

where A^* denotes the pseudoinverse of matrix A. In fact, $T_{W,Q}$ is an approximation of the Jacobian of W on Q; if given the analytical expression of W, we can replace $T_{W,Q}$ by the Jacobian of W at, say, v_{i_1}. To extract the rigid component of $T_{W,Q}$ we perform its singular value decomposition: $T_{W,Q} = U\Sigma V^T$; the rigid component of $T_{W,Q}$ is then

$$R_{W,Q} = VU^T. \qquad (2)$$

To devise the feature-preserving mapping, we formulate the following optimization problem: we would like all the elements outside of F to undergo a transformation as close as possible to W, and all the elements in F should undergo solely the rigid (or similarity) component of W. It is convenient to formulate the requirements of this optimization per quad. If quad $Q = (v_{i_1}, v_{i_2}, v_{i_3}, v_{i_4})$ belongs to a feature (i.e. it has at least one node in F), we define the following four equations related to its four edges:

$$\widetilde{v}_{i_{k+1}} - \widetilde{v}_{i_k} = R_{W,Q}(v_{i_{k+1}}) - R_{W,Q}(v_{i_k}), \quad k = 1,\ldots,4 \text{ cyclically} \quad (3)$$

where $\widetilde{v}_{i_k}$ are the unknown deformed grid nodes. Similarly, if Q does not belong to a feature, we add the following four equations for its edges:

$$\widetilde{v}_{i_{k+1}} - \widetilde{v}_{i_k} = W(v_{i_{k+1}}) - W(v_{i_k}), \qquad k = 1,\ldots,4 \text{ cyclically} \quad (4)$$

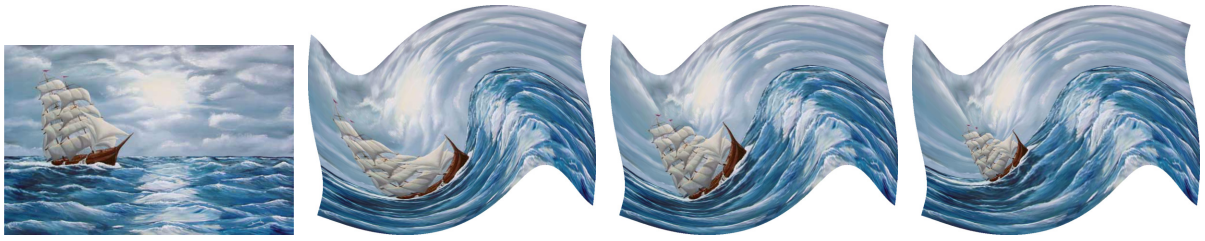
Overall, we obtain an over-determined system of $4|K|$ equations in $2N$ unknowns, which can be solved in the least squares sense. Note that the system is separable in the two coordinates, thus we can solve for x and y separately, with the system matrix containing N columns. We constrain the boundary nodes to their positions under W to make the optimization problem well-posed:

$$\widetilde{v}_i = W(v_i), \quad \forall i \in \partial G. \qquad (5)$$

Solving for $\widetilde{v}_1,\ldots,\widetilde{v}_N$ will provide us with a mapping that rigidly preserves the features, including their size. To obtain a shape-preserving mapping that allows appropriate scaling of the features, we modify the local transformations $R_{W,Q}$ as follows. We estimate the average scaling of each connected feature component F_i under W by observing the singular values of the transformations $T_{W,Q}$. For each element $Q \in F_i$, we take the smaller singular value of $T_{W,Q}$, and average those values over all $Q \in F_i$, obtaining the average scale factor λ_i. We chose to average the smaller singular values, because intuitively, if we stretch the image in one direction, the feature size should remain constant. The target local transformations of the quads in each F_i are thus updated to be $\lambda_i R_{W,Q}$, and Eq. (3) is modified accordingly.

2.1. Smoothing the mapping

When the input warp W is largely deforming the geometry of G, feature shape preservation may be compromised. To compensate for such situations, we found it useful to apply weights to Eq. (3) that is responsible for feature preservation: each side of those equations is multiplied by weight w_F (we use $w_F = 10$). Since we are solving a least-squares system, this multiplication results in w_F^2-magnification of the corresponding error terms in the minimization functional, forcing the optimization to respect the features more, at the expense of larger deformation of other areas. However, since the weights are abruptly discontinuous at the feature boundaries (weighting of 1 outside the feature and $w_f \gg 1$ inside),



input image | standard swirl | feature-aware swirl (rigid) | feature-aware swirl (similarity)

Figure 4: *End of a storm. Comparison between the standard swirl mapping function and our feature-aware mapping. The rigid version constrains the size of the features to remain the same while the similarity version allows uniform scaling of the feature.*

such solution damages the smoothness of the mapping near the feature boundary. This can be easily corrected by assigning a more smooth weighting function: we compute a local distance field to the feature and assign smoothly decreasing weights for the quads in the vicinity of the feature as functions of the distance field. The equations associated with those "transition-quads" are of type (3). We chose to use the following polynomial as the decay function:

$$f(x) = \frac{2}{\rho^3}x^3 - \frac{3}{\rho^2}x^2 + 1, \tag{6}$$

where the constant $\rho > 0$ controls the extent of the decay; the weights in the intermediate region around the feature boundaries are thus defined as

$$w(Q) = w_F \cdot f(D(Q)) + 1 \cdot (1 - f(D(Q))), \tag{7}$$

where $D(Q)$ is the value of the distance to the feature at the center of Q. We set the decay radius ρ to be the the width of two grid cells; outside of this radius the weights are set to 1. Observe the effect of the weighting scheme in Figure 3.

3. Interactive texture mapping

We distinguish between two possible modes of our texturing application: input-warp mode (described in the previous section) and interactive mode. In both modes, the feature regions of the input image are first specified by a feature mask. In the interactive mode, the user designs the mapping using the standard controls of image boundary editing and/or prescription of inner curve transformations. The mapping is computed taking into account these user-defined constraints and the feature mask, using a deformation technique based on differential coordinates. These user's manipulations are interpreted by our system as positional constraints on the grid nodes, i.e. simply

$$\widetilde{v}_i = c_i, \quad i \in U, \tag{8}$$

where U is the set of the nodes constrained by the user and c_i are the new positions for those nodes. The mapping of the free grid nodes is decided by applying the Laplacian editing optimization [SLCO*04, IMH05]. The goal of this optimization is to create a smooth and as-rigid-as-possible mapping of the grid shape that respects the user constraints (8). "As-rigid-as-possible" means that if the user-constraints imply solely a rigid (or similarity) transformation of the grid shape, the optimization technique indeed delivers such transformation; otherwise, the optimization finds a mapping that is locally as close as possible to being rigid, which is perceived as an intuitive result [ACOL00]. The optimization involves solving a sparse linear system of size $2N \times 2N$.

Once the mapping function W is established in the above manner, we create its feature-preserving approximation according to the feature mask, as described in Section 2.

4. Implementation details

The algorithmic core of our feature-sensitive texture mapping is the solution of the least-squares optimization expressed by Eqs. (3-4) and (5). When put together, these equations form an over-determined linear system of the form:

$$A(x\ y) = (b_x\ b_y), \tag{9}$$

where $x = (\widetilde{x}_1, \ldots, \widetilde{x}_N)^T$ are the x coordinates of the deformed grid and $y = (\widetilde{y}_1, \ldots, \widetilde{y}_N)^T$ are the y coordinates. The system is separable in the two coordinates, so the system matrix A has N columns. The matrix is very sparse since there are only two non-zero coefficients in each row. We solve the system by factoring the normal equations:

$$A^T A(x\ y) = A^T(b_x\ b_y). \tag{10}$$

We use the TAUCS library [Tol03] for efficient sparse matrix solvers. Cholesky factorization provides a sparse lower-triangular matrix L such that

$$A^T A = LL^T. \tag{11}$$

Then, the equations can solved by double back substitution:

$$Lx_{temp} = A^T b_x \tag{12}$$

$$L^T x = x_{temp}, \tag{13}$$

and in the same fashion for the y component. Thus, a single factorization serves solving for multiple right-hand sides. We attribute the construction of the A matrix, the normal equations matrix and the factorization to the pre-process, since they only depend on the grid and the feature map

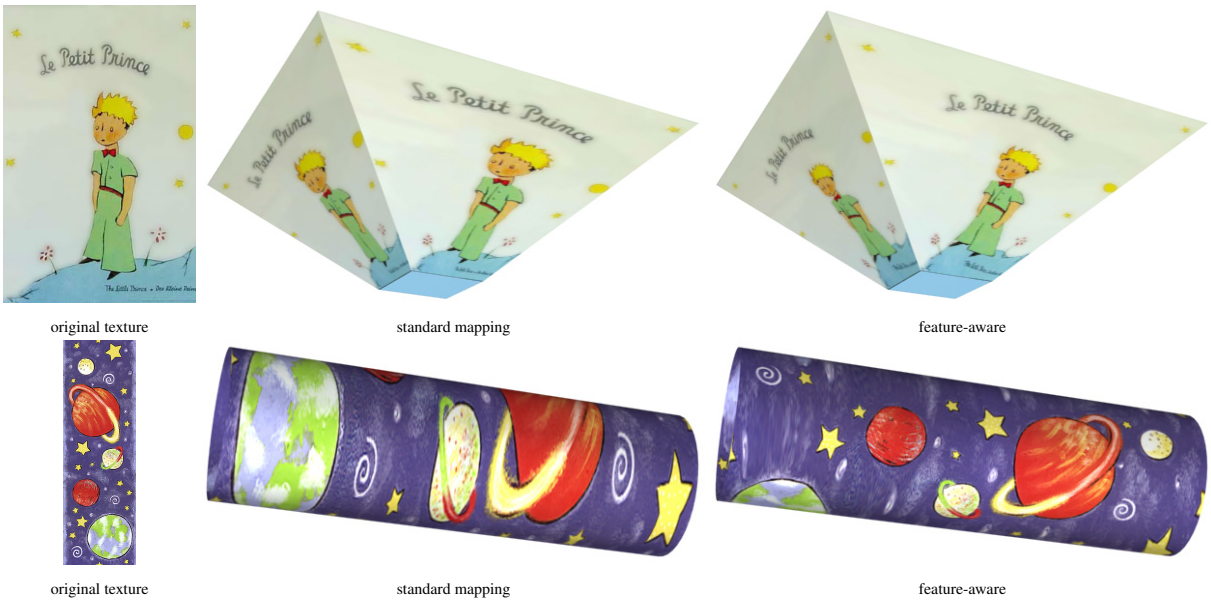


Figure 5: *Texture mapping of simple 3D objects using standard mapping and our feature-aware technique.*

of the input image; the matrix factorization is the most computationally-intensive part, taking a few seconds for grids with several tens of thousands of quads. Once the factorization is computed, back substitution is extremely fast (see Table 1). When varying the input warp function W, we only need to update the right-hand side of the system (the b_x, b_y vectors) and perform back-substitution, so the user can experiment with various mappings in real time. Of course, manipulation of very large images may slow down due to the large dimensions of the system matrix; to maintain interactive response in this case we define the grid to be slightly coarser than the pixel grid of the input image, so that the size of the system remains in the order of $20000 - 50000$ variables. For example, we can efficiently handle an image of 1000×1000 pixels by defining the size of the grid cells to be 5×5 pixels.

Computing the initial mapping by interactively-placed user constrains (Section 3) also requires solving a sparse linear system of size $2N \times 2N$ (see [SLCO*04, IMH05] for details). We do this in the same manner: pre-factoring the system matrix and solely varying the right-hand side of the system when the user manipulates the boundary constraints. Since the back-substitution is fast, the manipulation is interactive, as demonstrated in the accompanying video.

5. Results and discussion

We have implemented our feature-sensitive texturing system on a Pentium 4 3.2GHz computer with 2GB RAM. We assume that the feature mask comes together with the input image, defined in some external image editing software.

Size	Setup	Factor	Rhs setup	Solve
50×100	0.156	0.110	0.015	0
100×100	0.375	0.250	0.031	0.015
100×200	1.141	0.562	0.047	0.031
200×200	2.171	1.407	0.109	0.063

Table 1: *Timing statistics (in seconds) for the different parts of the mapping algorithm.* Setup *stands for the setup of the normal equations matrix;* Rhs setup *denotes the building the right-hand side of the normal equations and* Solve *stands for the back-substitution. Note that the system setup and matrix factorization is done in a pre-process, once per given image grid.*

For our experiments, we created the feature maps in Photoshop [Pho06] using the standard selection tools (Magic Wand, Lasso and Magnetic Lasso). The process of feature selection is quite easy since our feature-aware texturing needs only a rough binary matte.

We have experimented with various input warping functions that are commonly available in most image editing packages. We compare the results of unconstrained mapping with our feature-preserving mapping in Figures 1, 2, 4 and 6. It can be clearly seen in all the examples that our mapping preserves the shape of the features while gracefully mimicking the input mapping function. The similarity-preserving mapping allows uniform scaling of the features, and thus it has more freedom to approximate the input mapping. For instance, when the input mapping implies enlargement of the image, the similarity-preserving mapping will allow uniform scaling of the features, whereas the rigid mapping will

Figure 6: *Comparison between standard arc warping and our feature-aware mapping. Top: original image; next is the standard mapping, feature-aware mapping with rigid constraints and similarity constraints.*

constrain the features to remain in their original size, thus introducing more stretch to the background areas. Figure 5 shows several textured 3D shapes using our technique and compares them to standard homogeneous texture mapping.

In extreme deformation cases, the feature-aware mapping may introduce fold-overs, which may result in texture discontinuity. Preventing self-intersections within the least-squares optimization is quite difficult; in future work we plan to explore post-processing relaxations to fix the fold-overs.

We have summarized the timing information in Table 1. It is evident that all the required operations can be performed in real time. Some interactive sessions are recorded in the accompanying video.

6. Conclusion

We have presented a method that allows performing non-homogeneous texture mapping. Our method is guided by a feature map that roughly masks the important features in the

texture image; the mapping produced by our algorithm then preserves the shape of those features. Our framework is able to rectify any given texture mapping to preserve the features' shape, and it can also produce a feature-preserving mapping that obeys user-defined boundary constraints. The technique is useful in any application involving texture mapping, image re-scaling and warping; it is generic and can be applied to any image warping mechanism.

Acknowledgements

The oil painting "End of a storm" is courtesy of Samuel Simoes. The stars and shells wallpapers are courtesy of *Wallpaper Borders R Us*. This work was supported in part by a grant from the Israeli Ministry of Science.

References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In *Proceedings of SIGGRAPH* (2000), ACM Press, pp. 157–164.

[BN92] BEIER T., NEELY S.: Feature-based image metamorphosis. In *Proceedings of SIGGRAPH* (1992), ACM Press, pp. 35–42.

[BTB02] BALMELLI L., TAUBIN G., BERNARDINI F.: Space-optimized texture maps. *Computer Graphics Forum (Proceedings of Eurographics '02) 21*, 3 (2002), 411–420.

[ESG01] ECKSTEIN I., SURAZHSKY V., GOTSMAN C.: Texture mapping with hard constraints. *Computer Graphics Forum (Proceedings of Eurographics '01) 20*, 3 (2001), 95–104.

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005) 24*, 3 (2005), 1134–1141.

[KSG03] KRAEVOY V., SHEFFER A., GOTSMAN C.: Matchmaker: constructing constrained texture maps. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3 (2003), 326–333.

[LÓ1] LÉVY B.: Constrained texture mapping for polygonal meshes. In *Proceedings of SIGGRAPH* (2001), ACM Press, pp. 417–424.

[LCS95] LEE S.-Y., CHWA K.-Y., SHIN S. Y.: Image metamorphosis using snakes and free-form deformations. In *Proceedings of SIGGRAPH* (1995), ACM Press, pp. 439–448.

[MJ96] MACCRACKEN R., JOY K. I.: Free-form deformations with lattices of arbitrary topology. In *Proceedings of SIGGRAPH* (1996), ACM Press, pp. 181–188.

[MJBF02] MILLIRON T., JENSEN R. J., BARZEL R., FINKELSTEIN A.: A framework for geometric warps and deformations. *ACM Transactions on Graphics 21*, 1 (2002), 20–51.

[Pho06] PHOTOSHOP:, 2006. http://www.adobe.com/.

[SGSH02] SANDER P. V., GORTLER S. J., SNYDER J., HOPPE H.: Signal-specialized parametrization. In *Proceedings of the 13th Eurographics Workshop on Rendering* (2002), Eurographics Association, pp. 87–98.

[SLCO*04] SORKINE O., LIPMAN Y., COHEN-OR D., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proceedings of Symposium on Geometry Processing* (2004), pp. 179–188.

[SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. In *Proceedings of SIGGRAPH* (2006), ACM Press. Accepted for publication.

[Sor06] SORKINE O.: Differential representations for mesh processing. *Computer Graphics Forum 25*, 4 (2006). Presented at Eurographics 2005 as "State-of-the-art report: Laplacian mesh processing".

[SP86] SEDERBERG T. W., PARRY S. R.: Free-form deformation of solid geometric models. In *Proceedings of SIGGRAPH* (1986), ACM Press, pp. 151–160.

[Tol03] TOLEDO S.: TAUCS*: A Library of Sparse Linear Solvers, version 2.2*. Tel-Aviv University, Available online at http://www.tau.ac.il/~stoledo/taucs/, Sept. 2003.