Texture Amendment: Reducing Texture Distortion in Constrained Parameterization

Yu-Wing Tai[†] Michael S. Brown[†] Chi-Keung Tang[‡] Heung-Yeung Shum[§]
National University of Singapore[†]
The Hong Kong University of Science and Technology[‡]
Microsoft Research Asia[§]

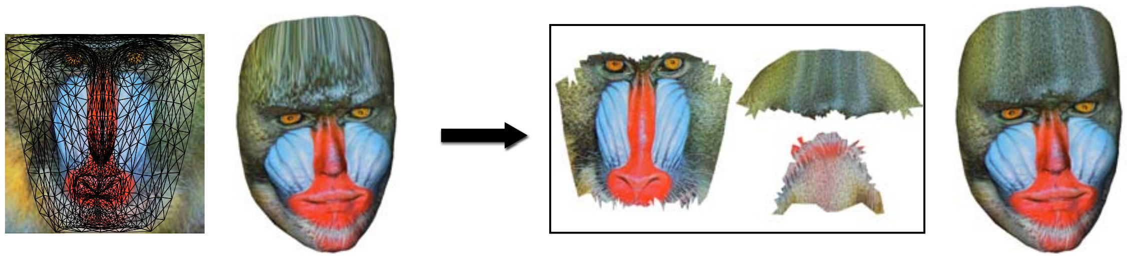

(a) Parameterization with noticeable distortion
(b) Texture is amended and distortion removed

Figure 1: *(a) Constrained parameterization resulting in texture distortion. (b) Original texture image is amended by expanding texture regions.*

Abstract

Constrained parameterization is an effective way to establish texture coordinates between a 3D surface and an existing image or photograph. A known drawback to constrained parameterization is visual distortion that arises when the 3D geometry is mismatched to highly textured image regions. This paper introduces an approach to reduce visual distortion by expanding image regions via texture synthesis to better fit the 3D geometry. The result is a new *amended texture* that maintains the essence of the input texture image but exhibits significantly less distortion when mapped onto the 3D model.

Keywords: texture-mapping, texture synthesis, image enhancement, user-assistance

1 Introduction

Texture mapping remains the *de facto* standard for simulating fine geometric detail and surface coloring on 3D models. Establishing a mapping between the 3D model and texture image is accomplished by parameterizing the 3D surface to 2D texture coordinates. There are two broad categories of parameterization for texture mapping:

Low-distortion parameterizations generate a mapping from the 3D geometry to a 2D texture space by minimizing a distortion metric between the 3D geometry and its 2D mapping (e.g. [Zhang et al. 2005; Sheffer et al. 2005; Lévy et al. 2002; Desbrun et al. 2002]). Such parameterization is performed irrespective of the texture image used.

Constrained parameterizations are used when an existing image or photograph is used as the texture image. In these approaches, the user specifies correspondences between the 3D geometry and the 2D texture image. The resulting 2D parameterization is computed such that it interpolates these user-supplied points (e.g. [Lévy 2001; Kraevoy et al. 2003; Zhou et al. 2005]).

These two parameterization approaches are at odds with one another when the 3D geometry is mismatched to the desired texture image. The constrained parameterization allows the model to fit to the texture image based on the user supplied points, but cannot avoid distortion of the image content when 3D geometry becomes compressed or stretched. This can result in unsightly distortion as shown in Figure 1. Low-distortion parameterization naturally minimizes distortion when rendered, but cannot fit to the desired texture image.

This paper proposes an approach that combines the complementary benefits of low distortion parameterization and constrained texture mapping to reduce undesirable visual distortion. Our approach determines where visual distortion will occur by examining the geometric distortion due to the constrained parameterization. In particular, we perform texture analysis of the image content: image regions where distortion occurs are expanded via texture synthesis to fit the 3D geometry. The size of this expansion is computed using low-distortion parameterization of the associated geometry. To deal with the complexity of the input texture image, simple user markup is used to assist automated texture-based segmentation and to specify texture flow. The final result is a new *amended texture* image that exhibits little or no distortion when rendered.

2 Related Work

Texture-synthesis approaches (e.g. [Wei and Levoy 2001; Solder et al. 2002; Tong et al. 2002; Zhang et al. 2003; Wang et al. 2005; Lefebvre and Hoppe 2006]) can generate distortion-free texture over 3D surfaces, but operate from texture-patches and not from

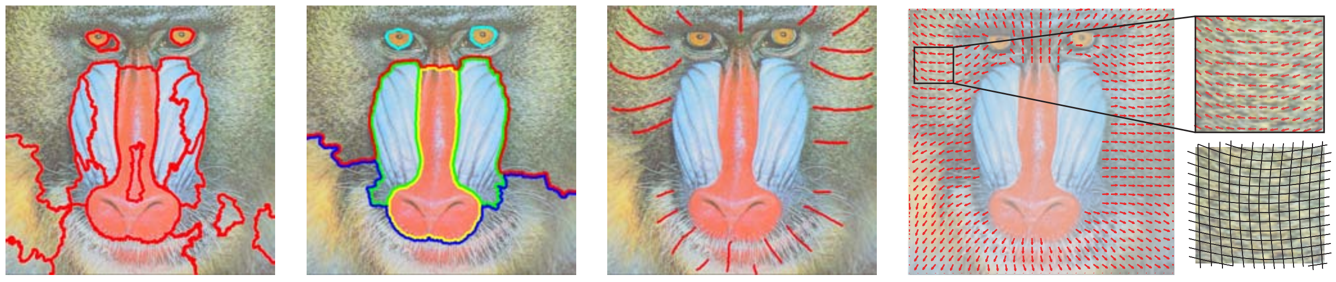


Figure 2: *(a) Initial automatic segmentation. (b) User-adjusted segmentation. (c) texture flow is specified using markup via strokes. (d) A dense flow-field is interpolated based on the markup. The texture flow is zoomed in and shown with grid lines.*

photographs or general images. Near-regular-texture (NRT) manipulation by [Liu et al. 2004] is similar in its theme of distortion driven synthesis, but is used in a different context of tiled textures and re-texturing. Image-completion techniques (e.g. [Bertalmio et al. 2000; Sun et al. 2005]) are similar in that they generate new imagery from information in the input image, but are used to interpolate missing data and not to expand imagery based on distortion. Texture image compression techniques (e.g. [Sander et al. 2002] and [Balmelli et al. 2002]) produce new texture images that optimize texture-space by compressing textureless triangles and avoid introducing distortion in the new texture image. However, these approaches cannot correct existing distortion present in the initial mapping between the 3D geometry and texture image.

When an existing photograph or image is desired as a texture for a 3D model, *constrained-parameterization* can be used to align the 3D model to the 2D input. Examples of constrained-parameterization include [Lévy 2001], MatchMaker [Kraevoy et al. 2003], and TextureMontage [Zhou et al. 2005]. For constrained-parameterizations, visual distortion may occur in the rendered 3D model when there is a mismatch between the 3D geometry and the 2D input texture image. *Low-distortion parameterizations* minimize distortion between the 3D geometry and 2D texture coordinates. These approaches often cut the 3D surface mesh into subregions, or charts, to further minimize distortion, packing the charts into an *atlas* (e.g. see [Zhang et al. 2005]). Examples include [Lévy et al. 2002; Sheffer et al. 2005] which minimizes angular distortion and intrinsic parameterization [Desbrun et al. 2002] which minimizes Dirichlet energy. Low-distortion parameterization approaches rely entirely on the model's 3D geometry and are not useful for aligning texture coordinates to an existing image or photograph.

The key idea in this paper is to use texture synthesis to expand textured image regions to better fit the 3D geometry. To do this we combine constrained-parameterization (what we *want*) with low-distortion parameterization (what we *need*). A 2D photo-editing approach [Fang and Hart 2007] similarly performed texture synthesis in lieu of image warping for 2D deformations. This allowed image regions to maintain a coherent texture appearance when expanded or distorted. While our approaches are similar in the goal of producing distortion-free output, there are three technical differences that make our problem fundamentally more challenging. First, in constrained-parameterization, the distortion measurement must be considered in both the 3D geometry and the 2D image space. Second, the specification of texture scale and orientation becomes more challenging because the synthesis domain is not on the original image. Third, as opposed to feature-aligned image retexturing, our synthesis domain is in the parameter space; as a result, spatial coherence cannot be fully employed. These issues will be considered in the next section.

3 Texture Amendment

Let $X : \mathbf{p} \rightarrow \mathbf{x}$ be a constrained parameterization (e.g., [Maillot et al. 1993; Lévy 2001; Kraevoy et al. 2003; Zhou et al. 2005]) that maps a vertex $\mathbf{p} \in R^3$ of a 3D mesh, M, to the desired feature location $\mathbf{x} = (x, y) \in R^2$ of the input image I. Our distortion metric $D(\mathbf{x})$, which measures texture and geometric distortion, is a function of both I and X. For each location $\mathbf{x}$ with significant distortion, texture synthesis, guided by texture scale and orientation at $\mathbf{x}$, will be performed in lieu of warping. Accepting as input M, I, and X, our procedure has the following components:

[Texture segmentation] Each pixel in I is labeled to belong to a region of similar texture. These regions serve as the "texture-pools" from which texture samples will come from if synthesis is necessary. User assisted splitting-and-merging of automatically determined texture regions is used to produce this segmentation.

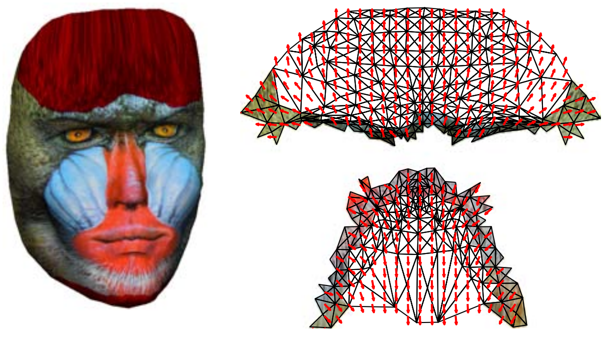
[Texture flow] Local orientation and scale of texture, collectively referred to as *texture flow*, that lie in each region are propagated from sparse user markups. This texture flow information will allow texture-patches to be scaled and oriented correctly when compared during the synthesis process.

[Scale-adaptive distortion] A distortion map $D(\mathbf{x})$ is computed that denotes regions where potential texture distortion may occur. This distortion map considers geometric distortion together with the textureness of I.

[Texture amendment] 3D geometry corresponding to the distortion map $D(\mathbf{x})$ is extracted from the original mesh M and re-parameterized using low-distortion parameterization. This new region is then "filled-in" using texture synthesis, which is guided by the texture flow information. Image-blending techniques are used to integrate this synthesized imagery with the existing image. The new synthesized regions are then added to the original texture image to produce the final amended texture image.

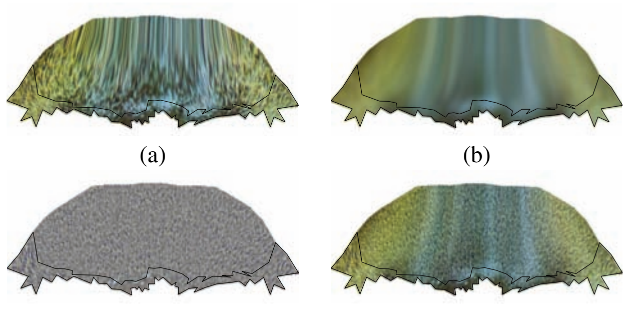
3.1 User-Assisted Texture Segmentation

An initial segmentation of the texture image is obtained automatically using the JSEG segmentation algorithm [Deng and S.Manjunath 2001] as shown in Figure 2-(a). This method typically over-segments the image. The user can refine the segmentation regions as shown in Figure 2-(b). Image-snapping tools can also be used to aid in this procedure. Note that spatially disjoint regions can be considered to form the same logical region. After segmentation, each pixel in the texture image is labeled to belong to one of



(a) (b)

Figure 3: *(a) Computed distortion map shown in red. (b) A region cut from the mesh along with its adjacent non-distorted triangles whose pixels are copied. The texture flow is warped to the distorted regions. The empty regions will be filled via texture-synthesis.*



(a) (b)

(c) (d)

Figure 4: *(a) texture image warped to the expanded component. Non-distorted pixels shown within a black-border. (b) Bi-lateral filtering applied. (b) Synthesis is performed using high-frequencies. (c) Final integrated approaches.*

the segmented regions. These segmented regions will serve as the "texture-pool" for synthesizing imagery.

3.2 Texture Flow

Texture orientation and scale variation can be present within and between the segmented regions. This variation is a problem for texture-synthesis algorithms that assume that the source and target texture are of the same scale and orientation. To address this issue, the user can draw strokes that denote orientation of the texture as shown in Figure 2-(c). Scale markup is performed by specifying a reference patch containing texture of a similar scale. This reference patch defines the default scale. Local scale change can be marked up by positioning the reference patch over an image region and stretching or compressing the patch until it visually matches the texture scale at that location (see Figure 6 and supplemental video for examples). The relative scale change between the scaled patch and reference patch is recorded as the local scale.

From this sparse orientation and scale markup, a dense flow-field is interpolated through propagation using a Markov Random Field similar to that used in [Levin et al. 2004]. This dense flow-field is shown in Figure 2-(d). The zoomed region in Figure 2-(d) shows the flow field with grid lines overlayed. The texture flow allows the image patches to be oriented and scaled properly to aid texture synthesis. The texture flow will also guide the synthesis so that the synthesized imagery flows correctly with its neighboring non-distorted texture.

3.3 Scale-Adaptive Distortion

Distortion estimation is based on the geometric distortion of the mesh's 3D triangles and the amount of textureness present in the image. If a region has high geometric distortion but little or no texture it results in a low distortion response. Likewise, if a region is highly textured but undergoes little or no geometric distortion it will result in a low distortion response. We use the following equation to generate this per-pixel distortion map. For each pixel $\mathbf{x}$ in the texture image, I, a local neighborhood $\mathcal{N}(\mathbf{x})$ is defined about that pixel. The distortion metric is computed as:

$$D(\mathbf{x}) = \sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} d(f_{2D}(\mathbf{x},\mathbf{x}'), f_{3D}(\mathbf{x},\mathbf{x}')) \cdot |I(\mathbf{x}) - I(\mathbf{x}')| \quad (1)$$

where $D(\mathbf{x})$ is the distortion measure, $f_{3D}(\cdot,\cdot)$ measure the distance between $\mathbf{x}$ and $\mathbf{x}'$ in the 3D triangle's local coordinate frame, $f_{2D}(\cdot,\cdot)$ measures the distance between $\mathbf{x}$ and $\mathbf{x}'$ in the corresponding 2D triangles's local coordinate frame, and $d(\cdot,\cdot)$ is the Euclidean distance between $f_{2D}(\mathbf{x},\mathbf{x}')$ and $f_{3D}(\mathbf{x},\mathbf{x}')$. To absorb the scale difference between the 3D triangles in M and their 2D parameterization, the term $d(\cdot,\cdot)$ is normalized to be between 0 and 1 based on the minimum and maximum distances for all triangles in M. The term $|I(\mathbf{x}) - I(\mathbf{x}')|$ is the magnitude of the pixel image difference of point $\mathbf{x}$ and its neighboring pixels $\mathbf{x}'$. The size of the neighborhood $\mathcal{N}(\mathbf{x})$ can be set for different texture images (larger $\mathcal{N}$ captures larger texture). If scale markup was performed, the neighborhood size is automatically expanded or contracted locally throughout the image based on the texture scale information. This equation simultaneously considers both geometric distortion in terms of $d(\cdot,\cdot)$, and *textureness* in terms of $|I(\mathbf{x})-I(\mathbf{x}')|$. While we found this distortion metric to be effective, other approaches to determine textureness (e.g [Bae et al. 2006] and [Malik and Rosenholtz 1997]) could be used with similar results. Final thresholding of the distortion response returns a distortion map as shown in Figure 3-(a).

3.4 Texture Amendment

Our texture synthesis procedure in the distorted regions is unusual. This is because it is performed in the parameter space, where the image I can be broken into disjoint regions and therefore spatial coherence in the original input should be exploited with care. Besides the issues of texture scale and orientation, the illumination and colors of the expanded regions in the amended texture should match the image content in the input image I.

To address these issues, mesh triangles falling in distorted regions are extracted from the initial texture-map with neighboring non-distorted triangles. For the examples in this paper, we include a 4-ring neighborhood of adjacent non-distorted triangles. This neighborhood is a function of mesh density and texture-resolution and can be tuned accordingly. The extracted triangles are then re-parameterized using a low-distortion parameterization technique such as least-square conformal mapping [Lévy et al. 2002] which is used in our implementation. These new parameterized regions will serve as the amended components.

Image pixels in the non-distorted triangles are warped to their corresponding triangles in the amended component. Including these non-distorted triangles in the amended components helps the synthesized approach produce a more seamless appearance. The texture

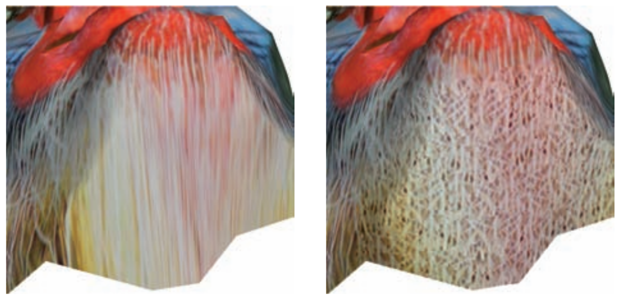


Figure 5: *Failure case shown on the mandrill's chin. Due to the long strains in the hair structure, synthesis using the corresponding texture-pool shown in (b) does not produce a good result. While the color blending approach makes the color reasonably seamless, the synthesized high-frequency component looks odd.*

flow information is also warped according to these expanded components and will be used to guide the synthesis. Figure 3 shows a diagram of the result of this procedure. Note that the low-distortion parameterization itself is not entirely distortion free and can introduce small scale changes which is incorporated into the user specified texture flow.

To synthesize the amended regions, we modify the graph-cut texture synthesis algorithm [Kwatra et al. 2003], by taking into consideration texture flow to warp the source and output texture. In particular, in the patch matching phase we orient and scale the texture pool imagery based on the local scale and orientation specified in the user supplied markup together with the orientation and scale of the amended region. After the candidate texture patch is selected, the seams are determined as described in [Kwatra et al. 2003]. The graph-cut approach is chosen in our system because it produces reasonably seamless results where the synthesized texture patches overlap with the non-distorted boundary pixels. Alternatives such as feature-aligned texture synthesis [Wu and Yu 2004] could also be used.

To reduce illumination effects in the input texture image, the input image is converted to a high-frequency representation, by first subtracting a bi-lateral filtered version of the image from itself. Saving the bi-lateral filtered image for later use, synthesis is performed using the high-frequency component only. After the region is synthesized, Poisson image-blending [Perez et al. 2003] is used to integrate the synthesized version with the bi-lateral filtered image, which is used as the target color. This integration with the bi-lateral filtered image produces a smooth color transition between synthesized and unmodified texture. Figure 4-(a)–(d) shows this procedure step-by-step, starting with the original input texture and distortion region, the bi-lateral filtered version, the synthesized texture, and final the integrated result. The non-distorted texture regions are shown with a border drawn around them.

Performing this synthesis for each component extracted from the original texture image, the final components are then packed into a new texture along with the existing texture image to form the final amended texture as shown in Figure 1.

4 Results

Three examples produced by our approach are shown in addition to the *mandrill* texture image used as the running example in the paper. The *Graphite* software [Graphite] based on [Lévy 2001] is used to establish the constrained-parameterization. Figure 6 shows a *cow head* 3D model where a *leopard* photograph is used as a texture image. Distortion is apparent in the horns and elongated part of the face (muzzle). In this example, scale markup is also required to accommodate the changes in the leopard's spot sizes. Segmentation, markup, dense texture flow, and the final amended texture are shown. In this example the low-distortion parameterization has further cut the extracted mesh regions resulting in several amended components. In such cases, we include the boundary triangles in these cut components as described in Section 3.4. Texture synthesis is then performed one component at a time, with the imagery from a synthesized component copied to the corresponding boundary triangles of the unprocessed components. Performing synthesis in this manner allows the individual components to maintain a consistent appearance. Figure 7 shows the second example of a *Pinocchio* model mapped to a *Van Gogh* self-portrait. Distortion is noticeable around the face and hat region. Segmentation, markup, dense texture flow, and the final amended texture are shown. The last example is of a repeating *dot* pattern mapped to the 3D *cow head* shown in Figure 8. Texture flow markup is not required. In this example, we show how the user can edit the distortion map. Distortion about the eyes and horns is allowed to remain, while the elongated muzzle region is corrected. The original distortion map, edited map[1], and the final amended texture are shown. In this example, scale changes in the amended texture result from the low-distortion parameterization and not user markup. For the last example, bi-lateral filtering and image-blending is omitted.

As described in Section 3.3 there are two parameters that can be adjusted: texture neighborhood size $\mathcal{N}$ for the distortion map computation; and the distortion map's threshold value, T, which is normalized to range between $0-255$. For our experiments these are selected as follows: *mandrill* ($\mathcal{N} = 13\times13, T = 190$), *cow/leopard* ($\mathcal{N} = 15 \times 15, T = 190$), *Van Gogh* ($\mathcal{N} = 15 \times 15, T = 190$), *cow/dots* ($\mathcal{N} = 17 \times 17, T = 190$).

5 Discussion and Conclusion

Our examples do not consider that prior mesh-cutting or disjoint image regions are present in the initial 3D-to-2D mapping and texture image. Such texture-map atlas generation and image-compositing may result for certain constrained-parameterization techniques such as [Kraevoy et al. 2003; Zhou et al. 2005]. Our approach can still be used to correct distortion that arises in these techniques, however, distortion identification and our synthesis procedure would need to be modified to consider this broken topology.

The goal of our work is to remove texture distortion by expanding distorted texture regions. Our approach is best suited for stochastic and near-regular texture. As the texture regions become more structure-like, texture distortion will become less noticeable. Noticeable distortion of structure would most likely be a concern if there is a great discrepancy between the 3D geometry and 2D texture image, in which the choice of texture image for the particular 3D geometry may not be suitable in the first place.

We also note that our approach can fail when unsuitable texture is available in the input image for synthesis. For example in Figure 5 we shows an example on the chin of the *mandrill*. Due to the long hair structure in that region there is unsuitable texture in the associate texture pool to produce satisfactory results (see Figure 2-(a)). While our color blending approach makes the color seamless, the synthesized high-frequency looks visually different. Whether

[1]We expect the user will often want to make slight changes to the distortion map using a simple painting interface. This gives the user the discretion to add or remove regions. This can be particularly useful when there are regions for which the distortion is desired to be kept.

or not the result is more acceptable than the texture distortion is a subjective decision left to the user.

In conclusion, we have presented an approach to reduce undesired texture distortion that commonly arises when 3D geometry does not sufficiently match to its texture image. Our approach amends the original texture image by expanding regions where distortion occurs. Our method uses simple markup of the input texture image together with the 3D-to-2D mapping to effectively determine distorted regions and correct them using texture-synthesis. Our approach complements the relative advantages of constrained and low-distortion parameterizations, and provides a valuable tool in generating visually appealing texture images for 3D models, a cornerstone of many graphics applications.

Acknowledgments

We thank the SIGGRAPH-Asia reviewers for their constructive comments and Wai-Shun Tong and Kun Zhou for their fruitful discussions. We also thank Bruno Levy for his publicly available Graphite and LSCM software. This work was supported by Singapore grant R-252-00-333-133 and Hong Kong grant RGC-CERG 620207.

References

BAE, S., PARIS, S., AND DURAND, F. 2006. Two-scale tone management for photographic look. *ACM Trans. Graph. 25*, 3.

BALMELLI, L., TAUBIN, G., AND BERNARDINI, F. 2002. Space-optimized texture maps. *Computer Graphics Forum 21*, 3.

BERTALMIO, M., SAPIRO, G., BALLESTER, C., AND CASELLES, V. 2000. Image inpainting. In *SIGGRAPH 2000*, 417–424.

DENG, Y., AND S.MANJUNATH, B. 2001. Unsupervised segmentation of color-texture regions in images and video. *PAMI 23*, 8 (Aug), 800–810.

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic parameterizations of surface meshes. In *Eurographics*.

FANG, H., AND HART, J. C. 2007. Detail preserving shape deformation in image editing. *ACM Trans. Graph. 26*, 3.

GRAPHITE. http://alice.loria.fr/index.php/software/3-platform/22-graphite.html.

KRAEVOY, V., SHEFFER, A., AND GOTSMAN, C. 2003. Matchmaker: constructing constrained texture maps. *ACM Trans. Graph. 22*, 3.

KWATRA, V., SCHODL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph. 22*, 3.

LEFEBVRE, S., AND HOPPE, H. 2006. Appearance-space texture synthesis. *ACM Trans. Graph. 25*, 3.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2004. Colorization using optimization. *ACM Trans. Graph. 23*, 3.

LÉVY, B., SYLVAIN, P., NICOLAS, R., AND JEROME, M. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3.

LÉVY, B. 2001. Constrained texture mapping for polygonal meshes. In *SIGGRAPH*.

LIU, Y., LIN, W., AND HAYS, J. 2004. Near-regular texture analysis and manipulation. *ACM Trans. Graph. 23*, 3.

MAILLOT, J., YAHIA, H., AND VERROUST, A. 1993. Interactive texture mapping. In *SIGGRAPH*.

MALIK, J., AND ROSENHOLTZ, R. 1997. Computing local surface orientation and shape from texture for curved surfaces. *International Journal of Computer Vision 23*, 2 (June), 149–168.

PEREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. 22*, 3.

SANDER, P., GORTLER, S., SNYDER, J., AND HOPPE, H. 2002. Signal-specialized parameterization. In *Eurographics Workshop on Rendering*.

SHEFFER, A., LÉVY, B., MOGILNITSKY, M., AND BOGOMYAKOV, A. 2005. Abf++: fast and robust angle based flattening. *ACM Trans. Graph. 24*, 2, 311–330.

SOLDER, C., CANI, M., AND ANGELIDIA, A. 2002. Hierarchical pattern mapping. *ACM Trans. Graph. 21*, 3.

SUN, J., YUAN, L., JIA, J., AND SHUM, H.-Y. 2005. Image completion with structure propagation. *ACM Trans. Graph. 24*, 3, 861–868.

TONG, X., ZHANG, J., LIU, L., WANG, X., GUO, B., AND SHUM, H.-Y. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Trans. Graph. 21*, 3.

WANG, L., GU, X., MUELLER, K., AND YAU, S.-T. 2005. Uniform texture synthesis and texture mapping using global parameterization. *The Visual Computer 21*, 8, 801–810.

WEI, L.-Y., AND LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. In *SIGGRAPH*.

WU, Q., AND YU, Y. 2004. Feature matching and deformation for texture synthesis. *ACM Trans. Graph. 23*, 3.

ZHANG, J., ZHOU, K., BELHO, L., GUO, B., AND SHUM, H.-Y. 2003. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Trans. Graph. 22*, 3.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2005. Feature-based surface parameterization and texture mapping. *ACM Trans. Graph. 24*, 1, 1–27.

ZHOU, K., WANG, X., TONG, Y., DESBRUN, M., GUO, B., AND SHUM, H.-Y. 2005. Texture montage: Seamlessly texturing of arbitrary surfaces from multiple images. *ACM Trans. Graph. 24*, 3.

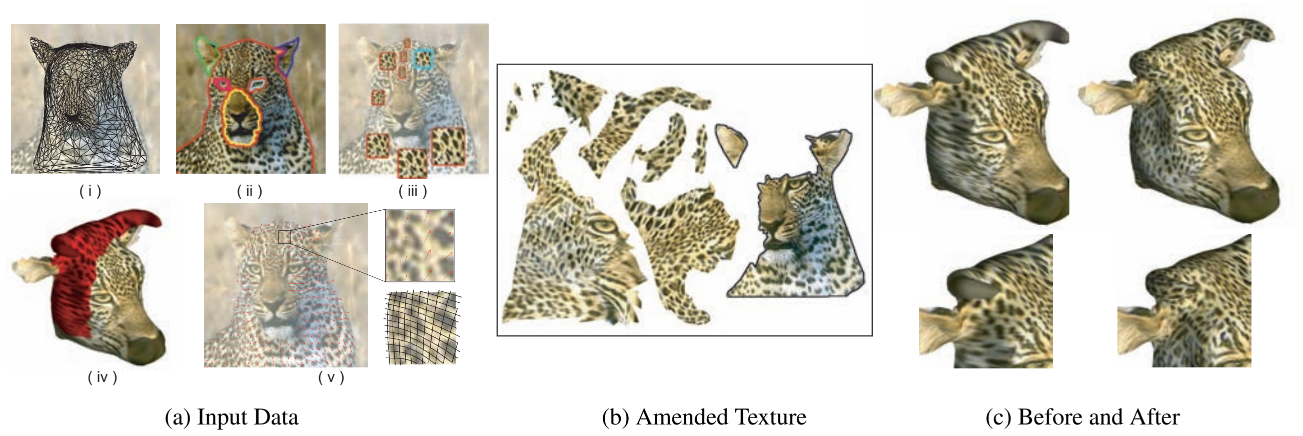


Figure 6: *(a) (i) texture image with constrained parameterization; (ii) segmentation; (iii) flow markup; (iv) distortion map; (v) dense-flow. In this example scale is marked up. The reference patch is shown in cyan. Orientation markup is omitted for clarity (see supplemental video). (b) The amended texture is shown. The original non-distorted imagery is denoted with a black border. (c) Before-and-after comparisons with zoomed-in regions are shown. Note distortion differences about the cow's muzzle and horns.*

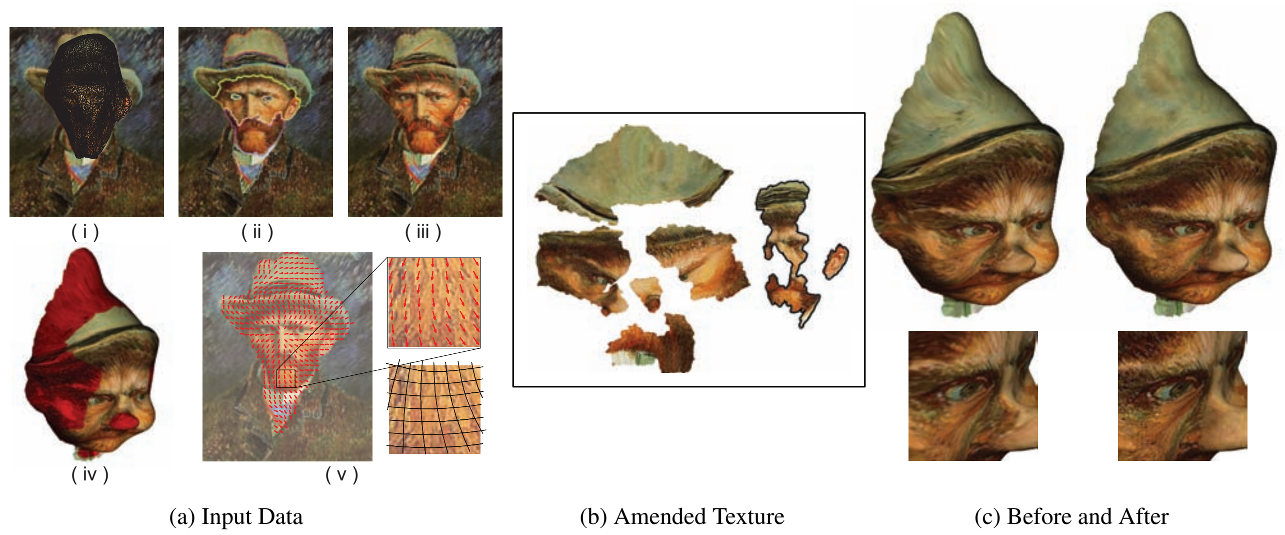


Figure 7: *(a) (i) texture image with constrained parameterization; (ii) segmentation; (iii) orientation; (iv) distortion map; (v) dense-flow. (b) The amended texture is shown. The original non-distorted imagery is denoted with a black border. (c) Before-and-after comparisons with zoomed-in regions are shown. Note distortion differences about the hat and face.*

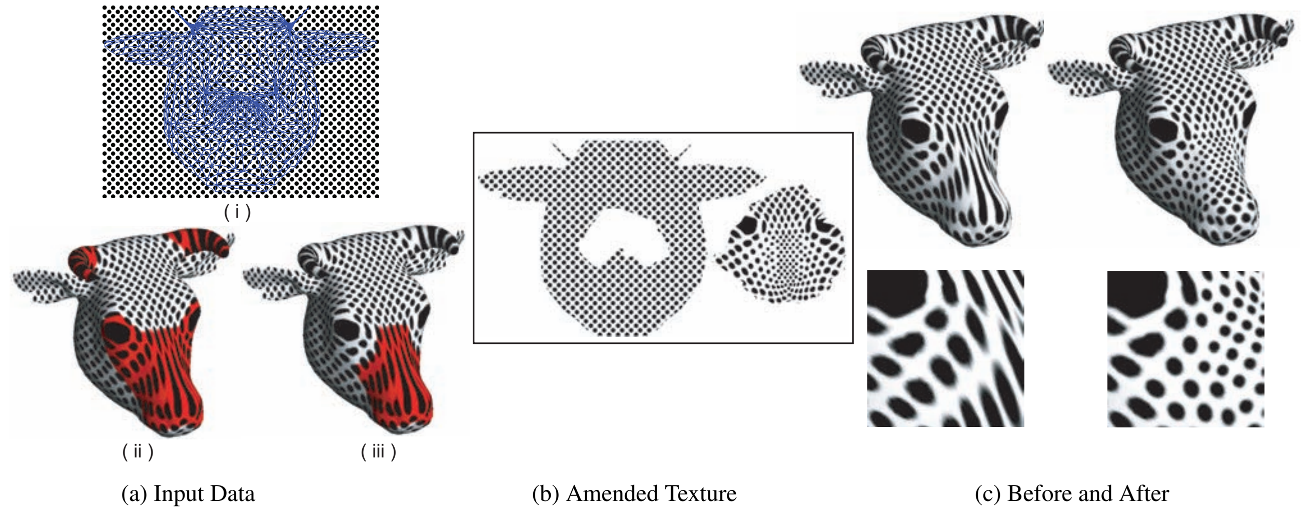


Figure 8: *(a) (i) texture image with constrained parameterization. No segmentation or flow is needed. The bottom (left-to-right) shows the (ii) original distortion map and the (iii) edited map. (b) The amended texture is shown. (c) Before-and-after comparisons with zoomed-in regions are shown.*