Image-Based Texture Replacement Using Multiview Images

Doron Tal *
Technion – Israel Inst. of Technology

Ilan Shimshoni †
University of Haifa

Ayellet Tal‡
Technion – Israel Inst. of Technology

Abstract

Augmented reality is concerned with combining real-world data, such as images, with artificial data. Texture replacement is one such task. It is the process of painting a new texture over an existing textured image patch, such that depth cues are maintained. This paper proposes a general and automatic approach for performing texture replacement, which is based on multiview stereo techniques that produce depth information at every pixel. The use of several images allows us to address the inherent limitation of previous studies, which are constrained to specific texture classes, such as textureless or near-regular textures. To be able to handle general textures, a modified dense correspondence estimation algorithm is designed and presented.

Keywords: Texture replacement, multiview stereo

1 Introduction

A basic problem in many augmented reality systems is creating new artificial images given real images [Azuma 1997; Berger et al. 1999; Debevec 1998; Keren et al. 2004]. Texture replacement is a challenging example of this type, which can be utilized in other applications, such as post-production in the entertainment industry. It is the process of painting a new texture over an existing textured image patch, in a manner that preserves the 3D appearance of the original patch. The 3D appearance can be visualized by the deformation of the texture over the 3D body in two ways. First, the texture pattern should be smaller on the more distant parts of the object. Second, as the angle between the surface normal and the viewing direction increases, the area of the texture in the image decreases (*foreshortening*).

Several texture replacement approaches have been suggested in recent years. The key distinction between them is the *shape extraction* method they use. Some of the techniques rely on *shape from texture* methods [Liu et al. 2004; Scholz and Magnor 2006], while others are based on *shape from shading*, where the luminance of the object is used, assuming a Lambertian reflectance model [Fang and Hart 2004; Zelinka et al. 2005]. In both cases a single image is given and thus the techniques are limited to specific texture types: near regular in the first case and textureless surfaces in the latter. The limitation to certain texture types severely hinders these approaches. Moreover, determining in advance whether a given texture fits the restrictions of a given method might in itself be complicated.

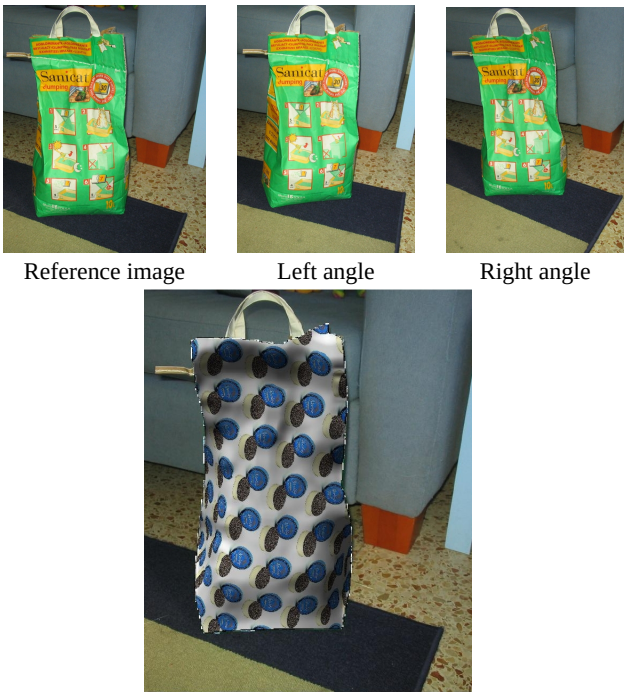


Figure 1: Texture replacement: Note that the input texture is neither smooth nor near-regular

The key challenge is to design a texture replacement approach that can handle all varieties of input textures. Since it cannot be achieved using a single image, we propose to use a few images. Given a set of multiview images of an object, one of which is the *reference* image, the goal is to replace the texture of a given patch in this image by some other texture, using the other images for shape reconstruction.

This paper makes the following contributions. First, a general, end-to-end, automatic approach for accurate texture replacement, which is based on multiview stereo for shape reconstruction, is presented. Second, a method specifically tailored for dense correspondence estimation between textured images, is described. Figure 1 gives an example of the results of our approach.

2 Related Work

In the following we describe previous texture replacement algorithms. Then, we discuss multiview stereo methods, which have not been utilized before for texture replacement.

Texture replacement: The first class of texture replacement approaches is based on shape from texture. These methods use texture deformations to extract the geometric characteristics of the surface. One of the first papers on texture replacement is [Tsin et al. 2001], which focuses on decoupling the illumination and the texture. The

*e-mail: dorontal@gmail.com
†e-mail: ishimshoni@mis.haifa.ac.il
‡e-mail: ayellet@ee.technion.ac.il

new texture is recoupled with the original illumination to create pleasing results. This method is limited to regular textured inputs and to fronto-parallel planes.

In [Liu et al. 2004] a wider set of textures is handled. This algorithm uses near-regular textures both for the source and for the target textures. The deformation field is extracted from the source texture interactively. Using the deformation field, this texture is dewarped into a regular texture. Finally, a regular texture coupled with the original illumination is deformed back into the same geometric structure.

In [Scholz and Magnor 2006] a technique for texture replacement of garments on a monocular video sequence is presented. Special coded print on the garment enables the surface coordinate system to be extracted from it with relative ease.

The other class of methods is based on *shape from shading*, where the reflection model of the surface is usually assumed to be near-Lambertian. The method proposed in [Fang and Hart 2004] is based on automatic extraction of a putative normal field and is followed by a manual correction stage. The corrected normal field is then approximated using 2D planes, and a texture is synthesized for each plane separately. Finally, the textured regions are seamed together. The method is limited to textureless input surfaces.

The method described in [Zelinka et al. 2005] extends the class of textures to include piecewise constant textures. It does so by an initial stage of image segmentation into different colors. Then, the surface normals of the most reliable color are extracted and later propagated into less reliable regions. The texture synthesis is done using jump maps [Zelinka and Garland 2004], which produce lower quality results than alternative methods but are very fast.

The advantage of the above approaches is that they are based on a single image. This comes at a hefty price of constraining the texture types, assuming an ideal reflection model. Since these assumptions are usually not precisely satisfied, user interaction is often required. To overcome these limitations, we propose to perform texture replacement using multiple images.

Multiview stereo: There are numerous stereo vision methods, where two or more images of the same object, taken from different directions, are used to reconstruct the 3D shape of the object. [Scharstein and Szeliski 2002] surveys *narrow-baseline two-frame stereo* methods, i.e., using two fairly similar images of the object. [Seitz et al. 2006] surveys *multiview stereo* algorithms. It is demonstrated that multiview algorithms present significantly better results for 3D reconstruction than the single-image approaches mentioned above.

None of these algorithms has taken the process onward to include texture replacement. Moreover, these algorithms have usually been applied to textureless surfaces, implying that with a high probability, image gradients correspond to depth discontinuities. This assumption does not hold for textured images. Our proposed algorithm handles this case.

3 General Approach

Our approach requires, as input, a set of images, one of which is regarded as the *reference image*, in which the texture patch should be replaced. The approach consists of four stages: sparse matching, bundle adjustment, dense reconstruction, and texture mapping, as described below.

3.1. Sparse matching: In this stage we acquire an initial match between two images at a time. This match is composed from a sparse set of point correspondences and the *fundamental matrix* that relates corresponding points in the two images [Hartley and Zisserman 2004].

This step consists of three stages: feature extraction from each image; creation of an initial putative list of matches; and identifying correct matches (*inliers*).

The features extracted by our algorithms are the widely used *scale-invariant feature transform (SIFT)* [Lowe 2004; Lowe]. Feature vectors obtained by SIFT are invariant to image scale and rotation, and partially invariant (i.e. robust) to changes in viewpoint and illumination.

Given two sets of feature points, one for each image, a putative match is constructed by matching each descriptor of the first set to its nearest neighbor descriptor in the other set. From this set of matches, the pairs selected are those whose distance to the closest neighbor is much smaller than the distance to the second-closest neighbor. (In our implementation, the ratio threshold is 0.6.) Many incorrect matches are thus removed.

The process described above yields a set of putative matches, many of which are still incorrect. The goal is to identify the inlier matches. This is done by using the RANSAC (Random Sample Consensus) approach [Fischler and Bolles 1981]. RANSAC estimates the parameters of a mathematical model from a set of observed data that contains outliers. In our case, the mathematical model is the fundamental matrix, the observed data is the point matches, and the outliers are the incorrect matches.

RANSAC achieves its goal by iteratively applying the following procedure. A minimal random subset of the original data points is selected. From this subset a model is computed. This model will be close to the correct model if these points are inliers. The hypothesis that this is so is then tested by checking for each of the data points its agreement with the model. If sufficiently many points have been classified as inliers relative to the estimated model, then the model is reasonably good. This iterative procedure is repeated a fixed number of times, each time producing either a rejected model because too few points are classified as inliers, or a refined model, together with a corresponding error measure.

The algorithm selects the fundamental matrix for which the largest number of matches conform. In our implementation, we use a variant of the algorithm known as MSAC, in which the score is not based only on the number of inlier matches, but also on the actual distances computed for them [Torr and Zisserman 2000].

In our implementation, 1000 cycles of RANSAC are run. Assuming that 50% of the matcher are inliers, 1000 cycles reflect a probability of 99.999% that an error-free minimal (seven) set of matches will be chosen. Figure 2 shows a result in which the blue points indicate matched features.

3.2. Bundle Adjustment: Once stereo sparse matching has been accomplished for all image pairs, the next step is to build a consistent model of all the images. *Bundle adjustment* performs this task of visual reconstruction to produce a jointly optimal 3D structure and viewing parameter estimates (camera pose and/or calibration). Optimal refers to minimizing a cost function that quantifies the model fitting error. Jointly refers to a solution that is simultaneously optimal with respect to both structure and camera parameters [Triggs et al. 1999].

In a nutshell, the problem can be formulated as follows. Assume that n $3D$ points are seen in m views. Furthermore, assume that each camera j is parameterized by a vector $\mathbf{a}_j$ which represents the rotation and translation parameters of the camera in vector form. Let x_{ij} be the projection of the ith point on the jth image. Bundle

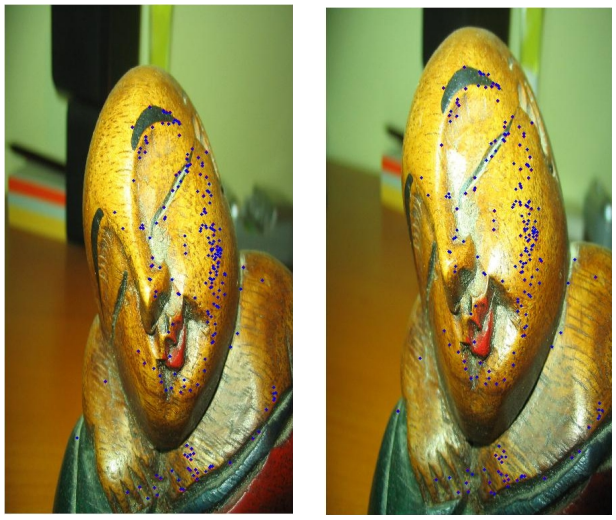


Figure 2: Sparse matching. The matched features are marked in blue.

adjustment amounts to refining the set of initial camera and structure parameter estimates in order to find the set of parameters that most accurately predicts the location of the observed n points in the set of the m available images. It minimizes the *reprojection error* with respect to all 3D points and camera parameters.

General purpose solutions of the non-linear problems are computationally demanding when employed to minimize functions that depend on many parameters. Fortunately, when solving minimization problems in bundle adjustment, the absence of interaction among parameters for different 3D points and cameras can be exploited, as done in [Lourakis and Argyros 2004].

3.3. Dense Reconstruction: Once accurate sparse matching has been accomplished and the 3D structure and the viewing parameters were found for this sparse set of points, they are used to perform the multiview dense reconstruction. This step calculates the depth of all the pixels in the reference image. Our algorithm, which is specifically tailored for textured images, is described in Section 4.

3.4. Texture Mapping: The last stage of the framework is casting a new textured image on the depth map acquired. The aim is to perform isometric-preserving texture mapping, i.e., distance-preserving mapping. There are several known algorithms for approximated isometric surface parametrization [Sorkine et al. 2002; Zigelman et al. 2002]. Our algorithm follows the general approach of the latter.

The key idea is to calculate the geodesic distances on the mesh [Kimmel and Sethian 1998; Mitchell et al. 1987; Surazhsky et al. 2005] and then find an optimal embedding on a 2D plane that preserves these distances. This is done using *Multi-Dimensional Scaling (MDS)* [Kruskal and Wish 1978; Cox and Cox 1994]. We use a simple MDS method called *classical scaling*, which uses the first two eigenvectors of the distance matrix to project the points to the plane.

After the 2D isometric embedding has been extracted, it is possible to "paint" the 3D mesh. This is done by first laying the texture image on the plane of the 2D embedding. Then, by traversing the original image, each pixel is associated with a triangle on the mesh. The sparse isometric mapping is interpolated using barycentric coordinates. Eventually, a bilinear interpolation is performed in order to find the texture color painted over the original pixel position, as illustrated in Figure 3.

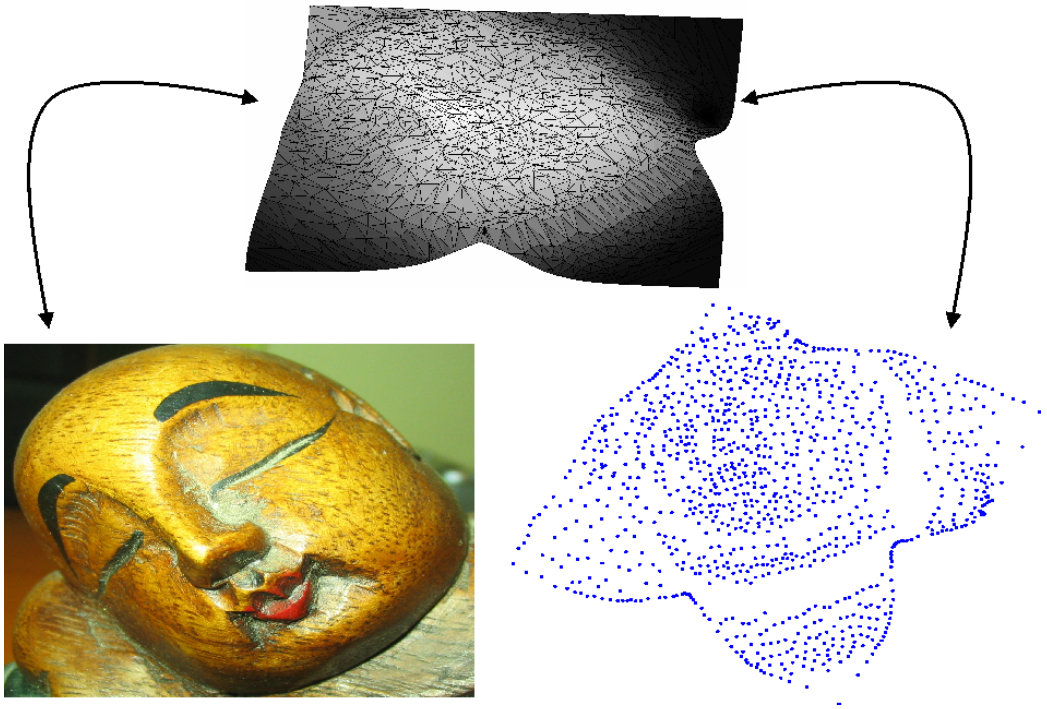


Figure 3: Texture mapping. To paint a new texture over the original image (bottom left), each pixel in the image is mapped to the 2D mesh embedding (bottom right). The triangle that includes the pixel is found in the simplified mesh (up). Then using barycentric coordinates, the mapping to the 2D mesh embedding is computed.

Since the depth map is large and complex (usually more than 100,000 pixels), texture mapping is preceded by simplification [Garland and Heckbert 1997; Garland].

4 Dense Reconstruction

Dense reconstruction is a process that calculates the depths of all the pixels. Assume that we are given m images $I_i, i = 1 \ldots m$, which associate each 2D-coordinate $\mathbf{x}$ with intensity values $I_i(\mathbf{x})$. The images are taken with a set of cameras, for which the internal and external calibrations are known, as a result of an internal calibration procedure [Strobl et al.] and bundle adjustment. Our aim is to estimate the depth map D_1 that assigns a depth-value $D_1(\mathbf{x})$ to all pixel locations in the reference image I_1. In this multiview stereo problem, the information from all the images will contribute to the computation of each pixel of the map D_1.

To enable free image acquisition using a hand-held digital camera, the algorithm should handle *wide baselines*. However, inherent to the wide-baseline setting, is the problem of *occlusions*, in which not all parts of the scene visible in a particular image, are also visible in the other images. Moreover, because of the large difference in viewpoint, pixels in different images, which are projections of the same point in the scene, might have different color values.

We are seeking a multiview reconstruction algorithm that satisfies the following requirements: (1) *Occlusion handling.* Occlusion is a difficult problem because, when a pixel in the reference image is occluded in one of the other images, their photometric values will usually be different. The algorithm should distinguish between this case and the case where the pixels differ because there is an error in the depth estimation (i.e., the pixels do not match). (2) *Textured-surface handling – reconstructing objects regardless of their texture.* In a given image of a textured surface, the algorithm may detect texture edges or it may detect edges that appear as a result of depth discontinuities. An algorithm that uses the edge cue in the 3D reconstruction as a depth discontinuity cue has to be able to distinguish between both types of edges.

We base our algorithm on [Strecha et al. 2004], which is a probability-based optimization approach for multiview reconstruction. Though this algorithm generally produces accurate results, it has a couple of drawbacks. First, the algorithm's performance depends largely on a good initialization, otherwise it might get stuck in local minima. Textured images are, by their very nature, not photometrically smooth, because a small error in the depth of a pixel can cause a non-smooth change in its color. Thus, this algorithm often fails to converge for our examples. Second, the assumption made that the photometric properties of all occluded pixels are similar, is beneficial for the case of specularities, but is insufficient for dealing with real occlusions. In particular, not only occluded pixels need not have the same color, but also in textured images, their colors vary significantly.

In our approach, the first problem is overcome by extracting many feature points in the previous stages. A key observation is that this can be achieved by taking several narrow-baseline image pairs, such that for each image pair many matches can be easily found (e.g., Figure 5(a)-(f)). Occlusion and foreshortening are handled through the use of a few pairs, rather than a single stereo pair, and are stitched together via bundle adjustment. The depth of these points, which have been estimated accurately by bundle adjustment, is kept fixed throughout the algorithm. These points are used as anchor points, and the depth has to be estimated only in the regions between them.

The second problem is solved by changing the way occlusion is handled. Intuitively, rather than assuming that all occluded pixels have a similar color, we base the visibility update on the previous pixel likelihood estimation. Namely, we assume that pixels which are very similar to their counterparts in the reference image are probably visible, as explained below. The rest of this section outlines the algorithm.

Given the camera calibrations and a depth value $D_1(\mathbf{x}_1)$ for a position $\mathbf{x}_1$ in I_1, the corresponding pixel location in the i^{th} image can be computed by:

$$\lambda_i \mathbf{x}_i = D_1(\mathbf{x}_1) K_i R_i^T R_1^{-1} \mathbf{x}_1 + K_i R_i^T (t_1 - t_i), \tag{1}$$

where K_i, R_i and t_i are the internal calibration matrix, rotation, and translation of the i^{th} camera, respectively. The overall mapping is denoted as $\mathbf{x}_i = l_i(\mathbf{x}_1, D_1(\mathbf{x}_i^1))$ or $\mathbf{x}_i = l_i(\mathbf{x}_1)$.

Each input image I_i, $i = 2, \ldots, m$ is regarded as a noisy measurement of the reference image I_1:

$$I_i(l_i(\mathbf{x}_1)) = I_1(\mathbf{x}_1) + \varepsilon, \tag{2}$$

where $\varepsilon \sim N(0, \Sigma)$ is the image noise, assumed to be normally distributed with zero mean and covariance matrix Σ.

Occlusion is modeled by a set of visibility maps $V_i(\mathbf{x}_1)$, which specify whether a scene point X that projects onto $\mathbf{x}_1$ in I_1 is also visible in image I_i. Every element of $V_i(\mathbf{x}_1)$ is a binary random variable, which is either 1 or 0, corresponding to visibility or occlusion, respectively. The set V_i contains the *hidden variables*, and their values must be inferred from the input images. Note that, as the first image was chosen as the reference image, all pixels in the first image are visible and therefore, $V_1(\mathbf{x}_1) = 1$.

Under the assumption that the image noise is independent and identically distributed for all pixels in all views, the data likelihood L can be written as the product of all individual pixel probabilities, where the product is:

$$L = \prod_{i=1}^{m} \prod_{\mathbf{x} \in I_1} N(I_1(\mathbf{x}) - I_i(l_i(\mathbf{x})); 0, \Sigma). \tag{3}$$

The algorithm assumes that the surface is smooth nearly everywhere (*smoothness prior*). The smoothness prior is modeled as an exponential density distribution of the form $\exp(-R(I_1, D_1)/\lambda)$. Here, λ is a parameter that reflects our prior belief as to how smooth D_1 is and $R(I_1, D_1)$ is a data-driven regularizer that relates image gradients to depth discontinuities. A high image gradient existing at a particular point $\mathbf{x}_1$ causes the algorithm to assume that a large depth discontinuity at that point and in that direction is more likely a-priori. The following regularizer is used:

$$R(I_1, D_1) = \nabla D_1^T T(\nabla I_1) \nabla D_1. \tag{4}$$

Here, $T(\nabla I_1)$ is a diffusion tensor defined by:

$$T(\nabla I_1) = \frac{1}{|\nabla I_1|^2 + 2\nu^2} (\nabla I_1^{\perp} \nabla I_1^{\perp T} + \nu^2 \mathbf{1}), \tag{5}$$

where $\mathbf{1}$ is the identity matrix, ν is a parameter controlling the degree of anisotropy, namely it controls the level of alignment between D_1's and I_1's gradients.

The downside of this data-driven regularizer is when attempting to recover the depth map of textured scene. Textures usually do not have depth discontinuities however, they do contain significant intensity gradients. Using the data-driven regularizer may lead to increased noise within the texture elements. In our algorithm, we adjust the anisotropy parameter ν according to the scene at hand.

(a) Depth map (b) Textured depth map of the reconstructed scene from different views

Figure 4: Reconstruction results

Minimizing the negative logarithm leads to the following energy formulation:

$$E(D_1,V) = \frac{1}{2}\sum_{i=2}^{m}\sum_{\mathbf{x}\in I_1} V_i(\mathbf{x})\left(d_i(\mathbf{x})^T\Sigma^{-1}d_i(\mathbf{x}) + \log(\sqrt{2\pi}|\Sigma|)\right) + \frac{1}{\lambda}R(I_1,D_1), \tag{6}$$

where $d_i(\mathbf{x}) = I_1(\mathbf{x}) - I_i(l_i(\mathbf{x}))$. Each term in the sum is weighted by the probability of the pixel to be visible.

To minimize the energy, an *expectation maximization (EM)* procedure is used. This procedure iterates between the estimation of V (the visibility maps) and the minimization of $E(D_1,V)$ (Equation 6), as follows.

E-step: In the $(k+1)^{th}$ iteration, the hidden variables $V_i(\mathbf{x}_1)$, are replaced by their conditional expectation, given the current estimate $D_1^{(k)}$ for D_1.

The question is how to update V_i. Basing it on a *global* estimation of the probability of a pixel of a certain color to be occluded, as done in [Strecha et al. 2004], is insufficient for producing accurate visibility maps. This is so since it focuses mostly on photometric problems such as specular reflections, where all specular pixels have the color of the light source. Occluded pixels, however, are usually not of the same color.

Instead, we base the visibility update on the previous pixel likelihood estimation, assuming that the pixel in image i have a similar color to its corresponding pixel in the reference. We use the following update step:

$$V_i(\mathbf{x}) \leftarrow \frac{N(d_i(\mathbf{x});0,\Sigma)}{\max_{\mathbf{y}\in I_1} N(d_i(\mathbf{y});0,\Sigma)}. \tag{7}$$

M-step: At the M-step the intent is to compute values for D_1 that minimize Equation 6, given the current estimates of V_i. To minimize E w.r.t D_1, we follow a gradient descent approach. By applying the Euler-Lagrange formalism, we get:

$$\frac{\partial E}{\partial D_1} = \sum_{i=2}^{m} -2V_i(I_1 - I_i(l_i))^T\Sigma^{-1}\nabla I_i(l_i)\partial l_i + \frac{1}{\lambda}\mathrm{div}(T(\nabla I_1)\nabla D_1). \tag{8}$$

Image I_1 is excluded from the sum, because $l_1(\mathbf{x}_1)$ is the identity transformation. The derivative ∂l_i is a 2-vector, whose expression is derived from Equation 1.

Figure 4 illustrates the 3D reconstruction results of our running example.

5 Results

The input images were taken using an off-the-shelf standard Canon A70 with 3MP. As a prepossessing stage, the radial distortion and internal calibration of the camera are extracted [Strobl et al.]. Then, the images are dewarped, to eliminate the camera radial distortion.

We have built an interactive utility, in which the user can select the scale, position, and rotation of the new texture. This is done during the final step of the algorithm, by placing the new texture on the flattened surface (i.e., the result of the MDS step described above). This utility provides immediate feedback on how the final result would look.

Figures 1,5,6 illustrate our results. Note that these results cannot be compared to previous results, since previous work assumed that the input is a single image and the texture is constrained, while in our case the input consists of a few images and the texture is arbitrary.

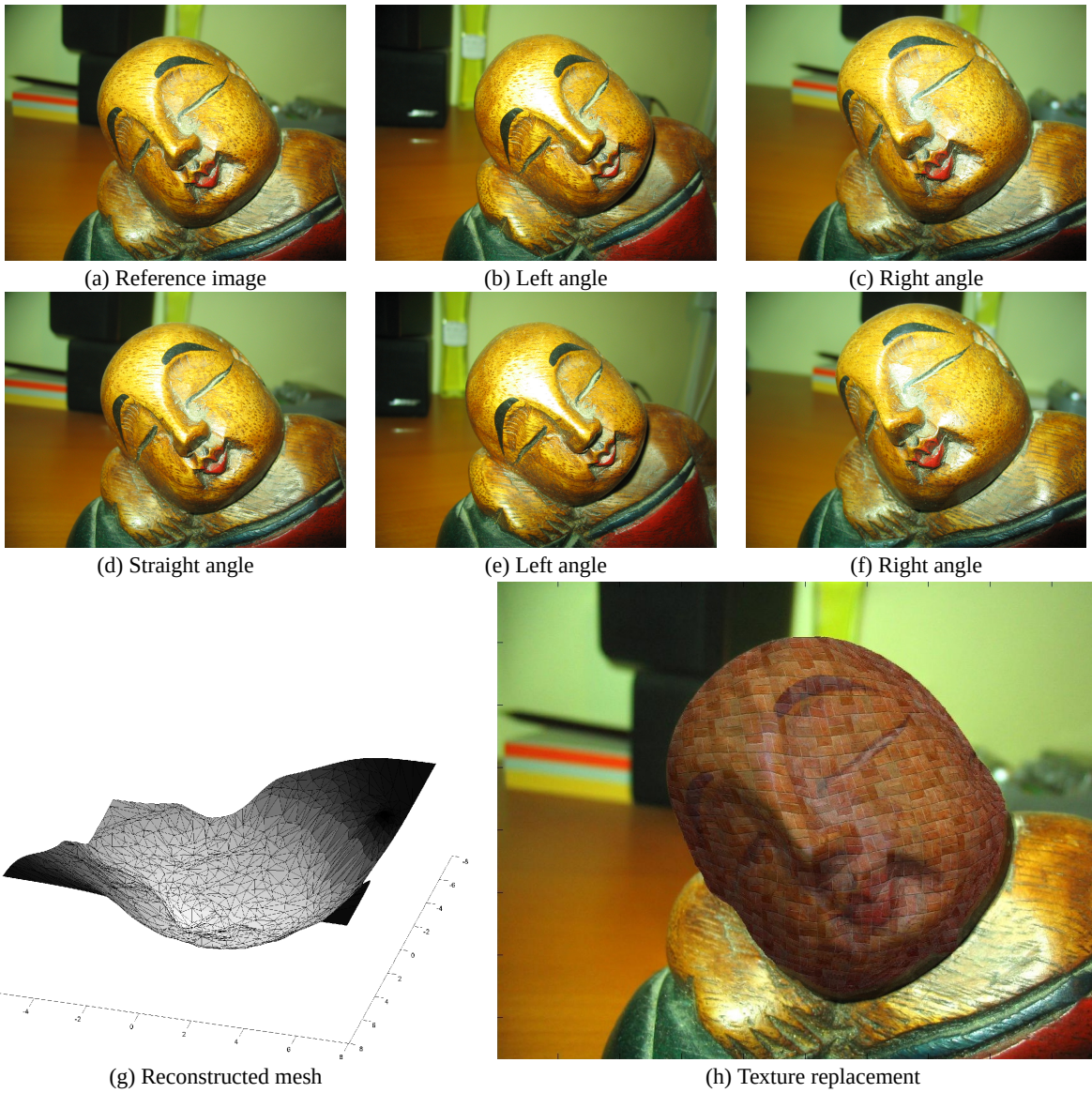
To generate the result shown in Figure 1 three images are used and a total of 1996 correspondences are automatically extracted. For Figure 5, six images are used and a total of 3765 correspondences are extracted. There are three pairs of close images, each is regarded as a narrow-baseline stereo pair. The advantage of this image configuration is that it is able to solve the main problem in the dense reconstruction – not having enough sparse points. The narrow-baseline image pair produces many correspondences for 3D points whose normals point roughly towards the camera. In order to extract correspondences of points that point in other directions, two additional narrow-baseline image pairs are used.

A similar configuration is used in Figure 6 (top), for replacing a bread texture using 880 correspondences. For obtaining the results shown in Figure 6 (bottom), 3 1.5MP images are used and 1269 correspondences are extracted.

The algorithm was implemented in unoptimized Matlab and ran on a 1.86GHz Intel Core2. Sparse matching takes 6.5 minutes using [Lowe], which can be accelerated to under 10 seconds using more efficient data structures [Lowe 2004; Goshen and Shimshoni 2006]. Bundle adjustment and dense reconstruction take together 3 minutes. Texture mapping takes 1.5 minutes, mostly devoted to fast marching.

6 Conclusions

This paper presents an automatic end-to-end approach for texture replacement, which is based on state-of-the-art techniques for shape reconstruction from multiview images. The dense reconstruction stage of our approach is modified to deal with any kind of textured surfaces.



(a) Reference image (b) Left angle (c) Right angle

(d) Straight angle (e) Left angle (f) Right angle

(g) Reconstructed mesh (h) Texture replacement

Figure 5: Chubby statue result. Input images that consist of three narrow-baseline image pairs (a-f), reconstructed mesh (g), and texture replacement results (h).

As the results show, the major advantage of this method over previous methods, is its ability to paint a new texture on an image containing any type of texture. The drawback of the approach is that more than a single image has to be used. The benefits of taking several images outweigh the drawbacks of the single-image techniques, which are inaccurate, and thus require user interaction, and are constrained to special classes of textures.

In the future we wish to design a method that can handle video streams of deforming textures, rather than utilizing a small set of images of a static surface. We would also like to develop methods for illumination extraction for general textured images, which would complement this work.

Acknowledgments: This work was partially supported by the European FP6 NoE grant 506766 (AIM@SHAPE), by the Israeli Ministry of Science, Culture & Sports, grant 3-3421, by Dr. I. Libling Fund, by the S.and N. Grand Research Fund, and by the A.M.N Foundation.

References

AZUMA, R. 1997. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments 6*, 4, 355–385.

BERGER, M.-O., WROBEL-DAUTCOURT, B., PETITJEAN, S., AND SIMON, G. 1999. Mixing synthetic and video images of an outdoor urban environment. *Machine Vision Applications 11*, 3, 145–159.

COX, M. A. A., AND COX, T. 1994. *Multidimensional Scaling*. Chapman and Hall.

DEBEVEC, P. 1998. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH*, 189–198.

(a) Reference image

(b) Texture replacement

Figure 6: Texture replacement results of bread (top) and sofa (bottom) images.

FANG, H., AND HART, J. C. 2004. Textureshop: texture synthesis as a photograph editing tool. *ACM Trans. Graph 23*, 3, 354–359.

FISCHLER, M. A., AND BOLLES, R. C. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6, 381–395.

GARLAND, M. Qslim software package. Website. http://graphics.cs.uiuc.edu/∼garland/software/qslim.html.

GARLAND, M., AND HECKBERT, P. 1997. Surface simplification using quadric error metrics. *Proceedings of SIGGRAPH'97*, 209–215.

GOSHEN, L., AND SHIMSHONI, I. 2006. Balanced exploration and exploitation model search for efficient epipolar geometry estimation. In *European Conference on Computer Vision*, II: 151–164.

HARTLEY, R. I., AND ZISSERMAN, A. 2004. *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press.

KEREN, S., SHIMSHONI, I., AND TAL, A. 2004. Placing three-dimensional models in an uncalibrated single image of an architectural scene. *PRESENCE 13*, 6, 692–707.

KIMMEL, R., AND SETHIAN, J. 1998. Computing Geodesic Paths on Manifolds. *Proceedings of the National Academy of Sciences of the United States of America 95*, 15, 8431–8435.

KRUSKAL, J. B., AND WISH, M. 1978. *Multidimensional Scaling*. Sage.

LIU, Y., LIN, W.-C., AND HAYS, J. H. 2004. Near regular texture analysis and manipulation. *ACM Transactions on Graphics 23*, 3, 368 – 376.

LOURAKIS, M., AND ARGYROS, A. 2004. The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg-Marquardt algorithm. Tech. Rep. 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug.

LOWE, D. G. Demo software: Sift keypoint detector. Website. http://www.cs.ubc.ca/∼lowe/keypoints/.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (Nov.), 91–110.

MITCHELL, J., MOUNT, D., AND PAPADIMITRIOU, C. 1987. The discrete geodesic problem. *SIAM Journal on Computing 16*, 4, 647–668.

SCHARSTEIN, D., AND SZELISKI, R. 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision 47*, 1-3, 7–42.

SCHOLZ, V., AND MAGNOR, M. 2006. Texture replacement of garments in monocular video sequences. In *Eurographics Symposium on Rendering*, 305–312.

SEITZ, S. M., CURLESS, B., DIEBEL, J., SCHARSTEIN, D., AND SZELISKI, R. 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, I: 519–528.

SORKINE, O., COHEN-OR, D., GOLDENTHAL, R., AND LISCHINSKI, D. 2002. Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization*, 355–362.

STRECHA, C., FRANSENS, R., AND VAN GOOL, L. J. 2004. Wide-baseline stereo from multiple views: A probabilistic account. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, I: 552–559.

STROBL, K., SEPP, W., FUCHS, S., PAREDES, C., AND ARBTER, K. Camera calibration toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/.

SURAZHSKY, V., SURAZHSKY, T., KIRSANOV, D., GORTLER, S., AND HOPPE, H. 2005. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph 24*, 3, 553–560.

TORR, P. H. S., AND ZISSERMAN, A. 2000. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding 78*, 1 (Apr.), 138–156.

TRIGGS, B., MCLAUCHLAN, P., HARTLEY, R. I., AND FITZGIBBON, A. W. 1999. Bundle adjustment: A modern synthesis. In *Vision Algorithms Workshop: Theory and Practice*, 298–372.

TSIN, Y. H., LIU, Y., AND RAMESH, V. 2001. Texture replacement in real images. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, II:539–544.

ZELINKA, S., AND GARLAND, M. 2004. Jump map-based interactive texture synthesis. *ACM Transactions on Graphics 23*, 4, 930–962.

ZELINKA, S., FANG, H., GARLAND, M., AND HART, J. C. 2005. Interactive material replacement in photographs. In *Graphics Interface*, K. Inkpen and M. van de Panne, Eds., 227–232.

ZIGELMAN, G., KIMMEL, R., AND KIRYATI, N. 2002. Texture mapping using surface flattening via multidimensional scaling. *IEEE Transactions on Visualization and Computer Graphics 8*, 2, 198–207.