Optimal Texture Map Reconstruction from Multiple Views

Lifeng Wang, Microsoft China, Beijing, P.R. China
Sing Bing Kang, Microsoft Corp., Redmond, WA, USA
Richard Szeliski, Microsoft Corp., Redmond, WA, USA
Heung-Yeung Shum, Microsoft China, Beijing, P.R. China

Abstract

The recovery of 3D models from multiple reference images involves not only the extraction of 3D shape, but also of texture. Assuming that all surfaces are Lambertian, the resulting final texture is typically computed as a linear combination of reference textures. This is, however, not the optimal means for reconstructing textures, since this does not model the anisotropy in the texture projection. Furthermore, the spatial image sampling may be quite variable within a foreshortened surface. This also has important implications for computer vision techniques that involve analysis by synthesis and the image-based rendering (IBR) technique of view-dependent texture mapping (VDTM). In this paper, starting with sampling theory, we show how weights should be spatially distributed for optimal texture construction. The local weights take into consideration the effects of anisotropy and variable spatial image sampling. We also present experimental results to verify our analysis.

1 Introduction

Texture mapping is an established rendering technique used to enhance the realism of 3D models. In computer vision, 3D models are typically constructed using multiple images (and possibly range data); their textures are recovered using combinations of appropriately extracted parts of the source images. An important issue is how these textures can be extracted as accurately as possible from multiple views, which may have radically different foreshortening effects. First, it provides insight on optimal super-resolution [1, 12]. Addressing this issue not only has implications on improving realism of the recovered 3D model, but it also impacts computer vision techniques that rely on analysis by synthesis (e.g, [6, 13]). In addition, the solution will provide hints as to how the increasingly popular image-based rendering technique (IBR) of view-dependent texture mapping (VDTM) [3] can be enhanced.

1.1 Previous work

Computer vision techniques that rely on analysis by synthesis reconstruct intermediate appearances for comparison with input images in order to refine the desired output. A typical example is the direct recovery of 3D geometry and texture from multiple reference images [6]. In another, Morris and Kanade [13] find the best triangulation for a given set of feature point correspondences across multiple images. The metric used is the reprojection error for a given hypothesized triangulation. Generation of correct textures is critical for such techniques.

There has been a significant amount of work done on generating an image with a resolution higher than its individual sources. This can also be considered as recovering an optimal texture map from multiple (smaller resolution) texture maps seen at different views. Existing super-resolution approaches can be categorized as being interpolation-based [8, 17, 9], frequency-based [18, 10, 11], and reprojection-based [2, 16]. Virtually all the work on super-resolution assume the pure translational model, which simplifies the analysis (one exception is [12]). In our work, we consider the 2D projective model, or planar patch motion in 3D.

Using a single texture map in 3D models is usually adequate. However, there is typically photometric variation across the views, due to lighting changes and non-Lambertian surfaces. View-dependent texture mapping has been proposed as an image-based means of modeling photometric variation, thus enhancing realism [3]. For a given view, reference textures are typically blended based on viewpoint proximity to the corresponding reference views (in the form of a sphere view map). Others that use the sphere view map as well include [4, 14, 15]. In the "Unstructured Lumigraph" work [1], global weights for each face texture are computed based on ray angular difference, estimates of undersampling, and field of view.

Greene and Heckbert [7] use the Elliptical Weighted Average (EWA) filter to warp a reference view into a desired view. This filter, which is space-variant, helps to reduce aliasing. In their work, only single textures are considered.

1.2 Our approach

In previous work, texture blending is done in a rather *ad hoc* fashion. Our work is the first (that we know of) that starts from basic image processing theory to derive the correct weights for blending. We follow the steps of reconstructing, warping, prefiltering, and resampling in order to warp reference textures to the desired location. We then treat this as a restoration problem and solve a linear system to compute the optimal final texture. One interesting observation (but not a surprising one) from this analysis is that the weights are

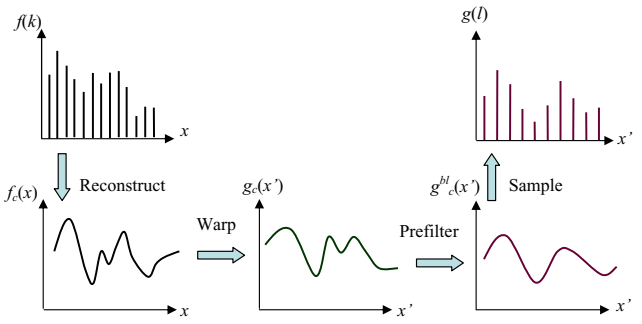

Figure 1: Steps involved in resampling. Adapted from [19].

spatially variant, and they tend to favor parts of the textures that are less foreshortened.

The remainder part is arranged as following: Section 2 shows how spatially variant weights are computed for image resampling. Section 3 describes how the final texture is reconstructed from multiple input textures using these weights. We show experiment results in Section 4, discuss the merits of our approach and future work in Section 5, and provide concluding remarks in Section 6.

2 Optimal resampling

Resampling is a common procedure used to map an image f of a certain size and shape to another g of differing size and shape. In our case, the mapping is through a homography H. To compute the spatially variant filters associated with the mapping, we apply the following steps (e.g., [19]):

- Reconstruct a continuous signal f_c using a filter r_1.
- Warp the continuous signal to the desired continuous signal g_c using the homography H.
- Prefilter g_c to anti-alias using filter r_2. This is because the warped signal g_c may contain higher frequencies than the output can handle.
- Sample g_c to produce discrete output g using an impulse function δ.

The kernels r_1 and r_2 and the impulse function δ can be combined into single kernel. These four steps are illustrated in Figure 1.

2.1 1D case

We now show how to combine r_1, r_2 and δ into a single filter in the 1D case. The 2D case is a simple extension of this analysis.

2.1.1 Reconstruction

Suppose we wish to resample a 1D bandlimited discrete signal $f(k)$ using a homography H, which maps x' in the destination signal to x in the source signal. We first construct the continuous signal $f_c(x)$:

$$f_c(x) = \sum_k f(k) r_1(x-k), \tag{1}$$

where $r_1(x) = sinc(x) = sin(\pi x)/(\pi x)$.

2.1.2 Warping

After reconstructing the continuous signal, we warp f_c to produce $g_c(x')$:

$$g_c(x') = f_c(x) = \sum_k f(k) sinc(H(x') - k) \tag{2}$$

This step is essentially a coordinate transform linking x in source texture to point x' in the destination texture, i.e.,

$$\begin{bmatrix} x \\ w \end{bmatrix} = H \begin{bmatrix} x' \\ w' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x' \\ w' \end{bmatrix}, \tag{3}$$

or $x = (h_{11}x' + h_{12})/(h_{21}x' + h_{22})$.

2.1.3 Prefilter

To prevent possible foldovers of high frequencies (causing aliasing), $g_c(x')$ is prefiltered to get *band-limited* signal $g_c^{bl}(x')$.

$$\begin{aligned} g_c^{bl}(x') &= \int g_c(t) r_2(x'-t) dt \\ &= \sum_k \int f_k sinc(H(t) - k) sinc(x' - t) dt \end{aligned} \tag{4}$$

with $r_2(x') = sinc(x')$.

2.1.4 Sampling

Finally, to produce the final discrete texture $g(l)$, $g_c(x')$ is sampled using the Dirac delta kernel $\delta(x')$.

$$\begin{aligned} g(l) &= g_c^{bl}(x')\delta(x'-l) \\ &= \sum_k \int f(k) sinc(H(t) - k) sinc(l - t) dt \end{aligned} \tag{5}$$

Unfortunately, there does not seem to be any closed-form solution to (5). To simplify our analysis and for speed considerations, we use a locally affine approximation of the function $H(t)$; this is a linearization about the peak of the $sinc(l-t)$ function in (5), i.e., at $t = l$. If the coordinate of a point in destination texture is t, the warped point $H(t)$ is approximated as:

$$H(t) = H(l + (t-l)) \approx J(l)(t-l) + C \tag{6}$$

J is the *Jacobian* and $C = H(l)$ is a constant which will not affect the reconstruction magnitude in frequency domain. So

$$\begin{aligned} g(l) &= \sum_k \int f(k) sinc(H(t) - k) sinc(l - t) dt \qquad (7) \\ &\approx \sum_k \int f(k) sinc(J(l)(t - l) - k + C) sinc(l - t) dt \end{aligned}$$

With this approximation, we obtain an expression that is basically a convolution of two $sinc$ functions, which results in a single sinc function with the *lower frequency limit*.

Figure 2 illustrates this in the frequency domain. The intuition is that for downsampling (highest input frequency exceeds the Nyquist output sampling rate), removing higher frequency before sampling is required to avoid aliasing. On the other hand, for upsampling (highest input frequency is below the Nyquist output sampling rate), this is not necessary.

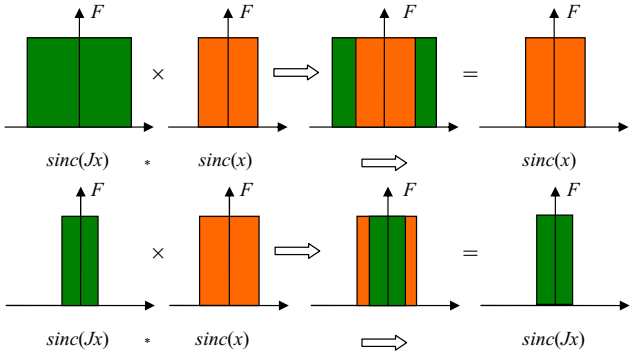

Figure 2: Explanation of prefiltering with two sinc functions in frequency domain. $sinc(Jx)$ is associated with the input warped signal while $sinc(x)$ is associated with the output sampling rate.

For a point l in the destination texture, the kernel $r(x)$ can be written as:

$$r(x) = \begin{cases} sinc(x) & \text{if } J(l) \leq 1 \\ sinc(J(l)x) & \text{otherwise} \end{cases} \qquad (8)$$

While the $Jacobian$-based spatial variant filter is an approximation, in our experiments, the deviation from the numerically approximated full solution is small (below an intensity level). This may not be true for very highly foreshortened textures, which we did not look at in our experiments.

2.1.5 Weight matrix

From (3), we get

$$dx = J(x')dx' = \frac{h_{11}h_{22} - h_{12}h_{21}}{(h_{21}x' + h_{22})^2} dx' \qquad (9)$$

$J(x')$ yields the instantaneous frequency rate change between the source and warped signals. We can write the discrete texture mapping as:

$$g(l) = f(x) * r(x)|_{x=H(l)} = \sum_{l=1}^{M} w_{l,k} f(k), \qquad (10)$$

It can be written as:

$$\begin{bmatrix} g(1) \\ \vdots \\ g(l) \\ \vdots \\ g(N) \end{bmatrix} = \begin{bmatrix} w_{1,1} & .. & w_{1,k} & .. & w_{1,M} \\ .. & .. & .. & .. & .. \\ w_{l,1} & .. & w_{l,k} & .. & w_{l,M} \\ .. & .. & .. & .. & .. \\ w_{N,1} & .. & w_{N,k} & .. & w_{N,M} \end{bmatrix} \begin{bmatrix} f(1) \\ \vdots \\ f(k) \\ \vdots \\ f(M) \end{bmatrix} \qquad (11)$$

This equation yields the local weight matrix of dimension $M \times N$, where M is the length of the input 1D signal and N is the length of the output 1D signal. The local weight $w_{l,k}$ can be written as:

$$w_{l,k} = \begin{cases} sinc(H(l) - k) & \text{if } J(l) \leq 1 \\ sinc(J(l)H(l) - k) & \text{otherwise} \end{cases} . \qquad (12)$$

2.2 2D case

The analysis for the 2D case is the same as for 1D. Here we show only the results. The analogy of (8) in 2D is

$$r(x, y) = sinc(X(x', y')x, Y(x', y')y), \qquad (13)$$

where

$$\begin{cases} X(x', y') = \begin{cases} 1 & \text{if } J_1(x', y') \leq 1 \\ J_1(x', y') & \text{otherwise} \end{cases} \\ Y(x', y') = \begin{cases} 1 & \text{if } J_2(x', y') \leq 1 \\ J_2(x', y') & \text{otherwise} \end{cases} \end{cases} \qquad (14)$$

$J_1(x', y')$ and $J_2(x', y')$ indicate the instantaneous frequency rate change along the x direction and y direction, respectively. If the sizes of the source and destination textures are $M \times N$ and $P \times Q$, respectively, then the local weight matrix will have a dimension of $(MN) \times (PQ)$.

3 Optimal construction from multiple textures

In this section, we present our approach for constructing a texture from multiple input textures. One can also view this as recovering a super-resolution image from lower-resolution inputs.

Given N measured textures $\{Y_k\}_{k=1}^N$ of size $M_k \times N_k$, we can compute N weight matrices $\{W_k\}_{k=1}^N$, each of which projects the respective texture to the desired view. All the matrices are obtained using the method described in Section 2. We assume that these textures are different views

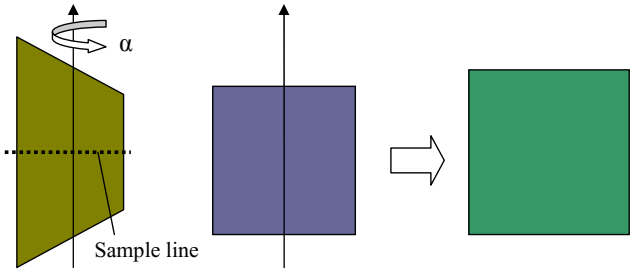

Figure 3: Reconstruction from two textures, with one rotating and other still.

of a single texture X of size $M \times N$. More specifically, each measured texture is the result of an arbitrary geometric warping performed on the ideal high-resolution image. We rewrite a 2D texture as a 1D vector by stacking columns of pixels onto a single column. Hence,

$$Y_k = W_k X \quad for \;\; 1 \leq k \leq N, \tag{15}$$

where W_k is the precomputed kth weight matrix of size $(MN) \times (M_k N_k)$.

With N textures, we have

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_N \end{bmatrix} = \begin{bmatrix} W_1 \\ \vdots \\ W_N \end{bmatrix} X = QX \tag{16}$$

The obtained model equation $Y = WX$ is a classic restoration problem model. We can use the Moore-Penrose pseudo-inverse to solve the equations.

$$X = (Q^T Q)^{-1} Q^T Y = Q^+ Y \tag{17}$$

We implemented the algorithm using Matlab^{TM}. Various iterative methods for solving this large set of linear equations could have been used as well [20].

4 Results

In this section, we show an example of the effect of foreshortening on local weights for blending as well as results on constructing textures from multiple views.

4.1 Effect of foreshortening on weights

When a texture is foreshortened, it is instructive to show how the weights change. As shown in Figure 3, we use two textures to construct a new texture. One texture is rotated from 0° to 50° about the central vertical axis while the other is kept still. We then take the weight profile of the weight matrix corresponding to the rotating texture. Figure 4 shows the evolution of weights with rotation angle (which indicates the degree of foreshortening). As expected, as the angle increases, the change in weights within the texture becomes more dramatic. Using a single global weight or kernel (such as linear or cubic) in such cases would clearly be suboptimal.

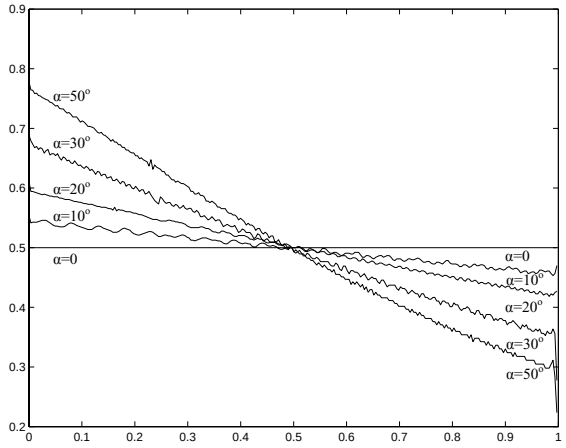

Figure 4: Profile of weight distribution with changing rotation of the first texture (see Figure 3).

4.2 Texture reconstruction

Figures 5 and 6 show results of image resampling (for a test pattern and photo, respectively). As can be seen, using the simple linear or cubic kernels results in artifacts, while using the proper spatially variant filter yields good results.

Figure 7 shows an example of texture reconstruction from multiple textures. The input textures are Figure 7(a-c), while Figure 7(d-f) shows the reconstructed texture at different poses using global weights (computed based on proximity to reference camera viewpoints). Figure 7(g-i) are the corresponding outputs of our algorithm.

An example of texture reconstruction from real images is shown in Figure 8. Figure 8(a,b) are the input textures taken with the camera at an angle approximately 60° on each side of the surface normal. Figure 8(d) shows the final reconstructed 400×300 texture using our method. (The homographies were extracted using a standard iterative full image registration technique.) In comparison, the texture reconstructed using a simple blending approach shown in Figure 8(c) appears more degraded.

In our approach, the weight matrices for the input textures are computed in order to construct the final texture. Figure 8(e,f) shows the distribution of weights corresponding to the real input textures (a,b) respectively. Each point P' on the destination texture is a local linear combination of different points from the input textures, i.e., $P' = \sum_j \sum_i w_{ij} P_{ij}$, where w_{ij} is the weight corresponding to P_{ij}, i.e., the ith point in the jth texture. The weight displayed at each point of the destination texture corresponding to the jth texture is simply $\sum_i w_{ij}$. The weights are coded by intensity; the higher the intensity, the larger the weights. It is clear that the weights are higher at the less foreshortened ends.

5 Discussion and future work

Our method for reconstructing a texture from multiple views has interesting implications. It verifies that spatially-variant filters should be used, especially in the case of highly foreshortened input textures. This is not surprising, as foreshortened textures may contain significant anisotropy and sampling variation. Using a global weight (albeit view-dependent) for blending is suboptimal. However, the use of global weights is adequate in cases where the apparent size of textures is small or the foreshortening is insignificant. While there is also speed considerations, our analysis may be used to compute more optimal global weights. Parametric approximations of the spatially-variant weights may also be used for speedups. Using this approximation for speedups is an item for future work.

Our analysis may also be applied to view-dependent texture mapping (VDTM). Even though our analysis makes the strong Lambertian assumption, it suggests that using global weights may not be appropriate in many cases. It may be possible to locally modulate the global weights for blending despite photometric variation. While this is done to a certain extent in [1], our approach has better theoretical justications.

One of extensions that we are considering is the application of our analysis to non-planar texture. Here the change in sampling will be affected by both foreshortening and depth variation. We will also investigate the effect of photometric variation on the appropriate use of local weights.

6 Conclusion

In this paper, we described an optimal means for reconstructing a single texture map from multiple views. In previous work, texture blending is based on heuristics, typically done using view-dependent global weights. Our work is the first (that we know of) that starts from basic image processing theory to derive the correct weights for blending. Our analysis has shown that the proper approach is to compute spatially-variant weights for optimal blending. These weights take into consideration the anisotropy in the texture projection and changes in sampling frequency due to foreshortening.

Our analysis, while simple, is effective, as results have shown. This approach can be incorporated in a variety of applications, such as analysis by synthesis methods, super-resolution techniques, and view-dependent texture mapping.

References

[1] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. *ACM SIGGraph to appear*, 2001.

[2] M.C. Chiang and T.E. Boult. Local blur estimation and super resolution. *Proc. of CVPR'97*, pages 821–826, 1997.

[3] P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. *Computer Graphics proceeding, Annual Conf. Series, ACM SIGGraph*, pages 11–20, 1996.

[4] P.E. Debevec, Y. Yu, and G. D. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. *Eurographics Rendering Workshop*, pages 105–116, 1998.

[5] M. Elad and A. Feuer. Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images. *IEEE Trans. on Image Processing*, 6(12):1646–1658, 1997.

[6] P. Fua and Y. G. Leclerc. Taking advantage of image-based and geometry-based constraints to recover 3-D surfaces. *Computer Vision and Image Understanding*, 64(1):111–127, July 1996. Also available as Tech Note 536, Artificial Intelligence Center, SRI International.

[7] N. Greene and P.S. Heckbert. Creating raster ominimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, pages 21–27, 1986.

[8] M. Irani and S. Peleg. Improving resolution by image registration. *CVGIP:Graphical Models and Image Processing*, 53:231–239, 1991.

[9] G. Jacquemod, C. Odet, and R. Goutte. Image resolution enhancement using subpixel camera displacement. *Signal Processing*, 12:139–146, 1992.

[10] S.P. Kim, N.K. Bose, and H.M. Valenzuela. Recursive reconstruction of high resolution from noisy undesampled multiframes. *IEEE Trans. Acoustics. Speech Sign. Proc.*, 38:1013–1027, June 1990.

[11] S.P. Kim and W.Y. Su. Recursive high resolution reconstruction of blurred multiframes. *IEEE Trans. on Image Processing*, 12:534–539, October 1993.

[12] S. Mann and R.W. Picard. Virtual bellows: Constructing high quality stills from video. *IEEE Int'l Conf. on Image Processing*, pages 363–367, 1994.

[13] D.D. Morris and T. Kanade. Image-consistant surface triangulation. *CVPR*, 1:332–338, June 2000.

[14] K. Pulli, M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, and W. Stuetzle. View-based rendering. *Eurographics Workshop on Rendering*, pages 23–34, 1997.

[15] P. Rademacher. View-dependent geometry. *Computer Graphics proceeding, Annual Conf. Series, ACM SIGGraph'99*, pages 436–446, 1999.

[16] H. Strack and P. Oskoui. High resolution image recovery from image plane arrays, using convex projection. *Journal of Optical. Soc. Am A*, 16:1715–1726, 1989.

[17] A.M. Tekalp, M.K. Ozkan, and M.I. Sezan. High resolution reconstruction from low resolution image sequence and space-varying image restoration. *In IEEE Int'l Conf, Acoustics, Speech and Signal Proc*, pages III–169–172, 1992.

[18] R.Y. Tsai and T.S. Huang. Multiframe image restoration and register. *Advances in Computer Vision and Image Processing, JAL Press Inc.*, 1984.

[19] G. Wolberg. *Digital Image Warping.* IEEE Computer Society Press, Los Alamitos, California, 1990.

[20] D. M. Young. Iterative solution of large linear systems. *New York:Academic*, 1971.

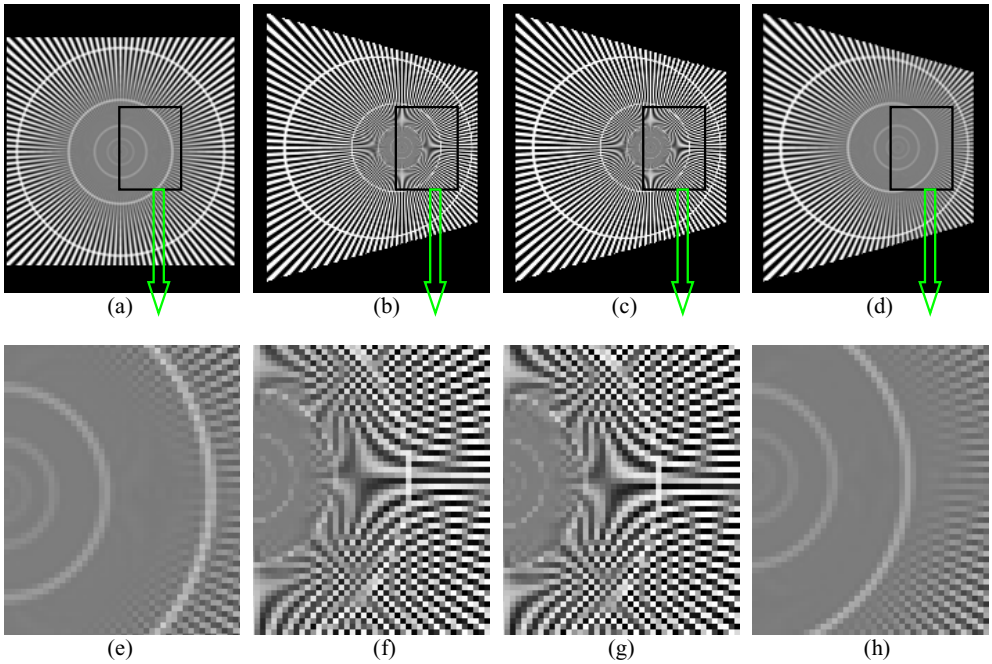

Figure 5: Image resampling example (test pattern): (a) Source texture, (b) Result of using linear interpolation, (c) Result of using cubic interpolation, (d) Result of using our spatially variant kernel. (e-h) are zoomed views. Notice the highly aliased effect in (f) and (g).

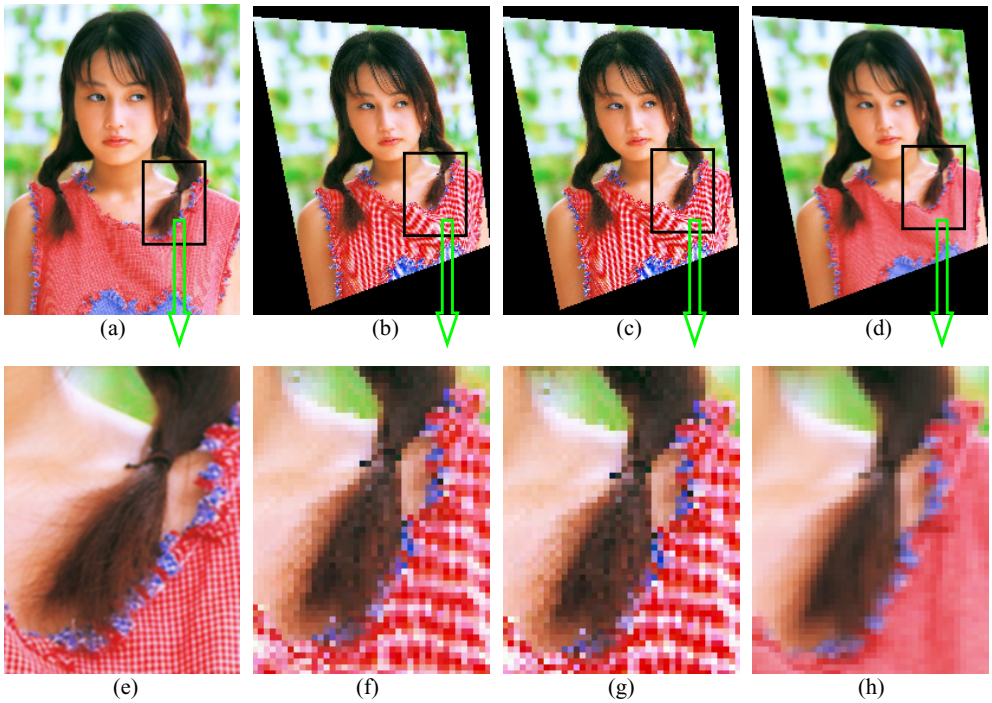

Figure 6: Image resampling example (photo): (a) Source texture, (b) Result of using linear interpolation, (c) Result of using cubic interpolation, (d) Result of using our spatially variant kernel. (e-h) are zoomed views.

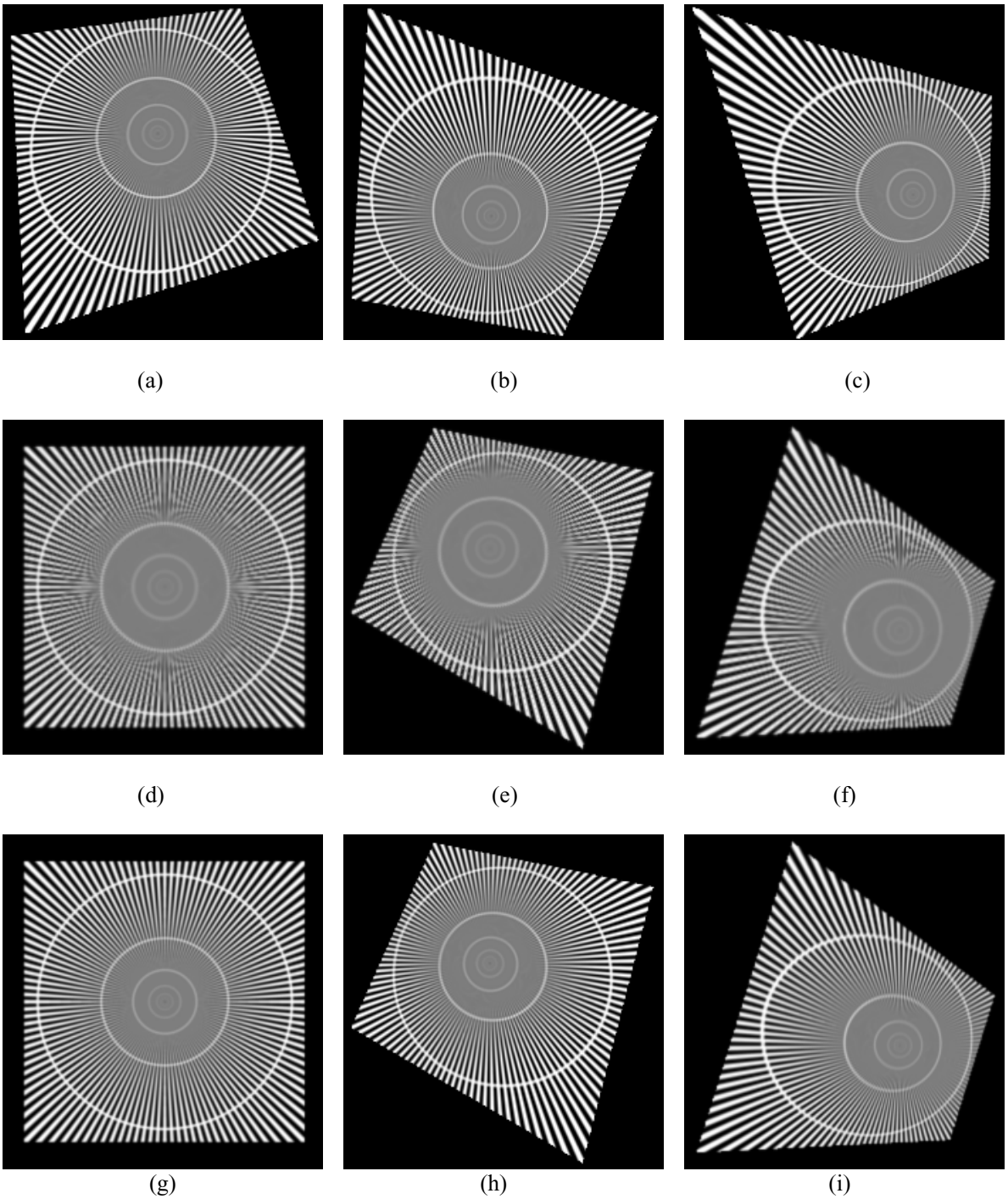

Figure 7: Texture reconstruction from multiple inputs (synthetic): (a-c) Three textures captured from different views, (d-f) Three textures reconstructed using a simple blending algorithm with global weights on (a-c), (g-i) Three textures reconstructed using our algorithm on (a-c). The textures generated using our method are sharper and less aliased.

Figure 8: Texture reconstruction using real images: (a,b) Textures captured from two different views, (c) Reconstructed texture using a simple blending method with global weights (with part of it zoomed), (d) Reconstructed texture using our method (with part of it zoomed), (e,f) Corresponding weight distribution used in texture reconstruction for inputs (a,b) respectively. Lighter pixels indicate higher local weights. All weights have been normalized to sum to 1.