3D Texture Mapping in Multi-view Reconstruction

Zhaolin Chen, Jun Zhou, Yisong Chen, and Guoping Wang

Graphics and Interactive Technology Lab, Peking University

Abstract. This paper proposes a novel framework for texture mapping of 3D models. Given a reconstructed 3D mesh model and a set of calibrated images, a high-quality texture mosaic of the surface can be created after the process of our method. We focus on avoiding noticeable seams, color inconsistency and ghosting artifacts, which is typically due to such facts as modeling inaccuracy, calibration error and photometric disagreement. We extend the multi-band blending technique in a principled manner and apply it to assemble texture images in different frequency domains elaborately. Meanwhile, self-occlusion and highlight problem is taken into account. Then a novel texture map creating method is employed. Experiments based on our 3D Reconstruction System show the effectiveness of our texturing framework.

1 Introduction

3D Reconstruction is a long-term and challenging problem in computer vision and computer graphics community. This kind of technology aims to reconstruct visual 3D models of real objects or large scale scenes that can be displayed in computer environment. The applications of it include virtual reality, computer animation, online exhibition, etc.

There are a lot of technological ways to perform 3D Reconstruction based on images, video or 3D scanning data. In this paper, we will address the texture mapping problem in the context of image-based modeling with a chessboard-type calibration board (see Fig. 1 below). We assume that we already have an approximate surface model of an object which is represented by a standard triangle mesh encoding. And we also assume that we have a set of images that are taken from different viewpoints and then precisely calibrated. Our focus is how to texture the reconstructed mesh surfaces so as to achieve photorealistic 3D models.

Fig. 1. With 30 images of a stone goldfish, a realistic 3D model with seamless and detailed texture is produced through our method

G. Bebis et al. (Eds.): ISVC 2012, Part I, LNCS 7431, pp. 359–371, 2012.

In general, one image cannot provide enough information to texture a 3D model. Given an image sequence, intuitively, we can bind a "best" texture patch to each triangle of the mesh model according to the viewing angle. However, this straightforward method is ineffective in the practical application environment, due to the inaccuracy of shape modeling, calibration error and the inconstancy of lighting and camera conditions. As a result, color discrepancies and lighting discontinuities appear as seams on the rendered surface.

There are mainly three directions that have been explored to remove the texture seams. The first direction is to "blur" the seams by using certain weighted averaging scheme. Second, many methods try to optimize the texture patch layout and therefore reduce the global texture inconsistencies with least details lost. The third direction focuses on the color correction or relighting in the vicinity of texture boundaries or in the global domain.

Adam Baumberg proposes an innovative texturing method [1] which employs a two-band frequency decomposition mechanism and blend images in both frequency domains to implement texture mosaicing. However, it fails to consider the significant connection between the width of transition zones and the size of features in different frequency bands. As a result, this method may produce noticeable artifacts where high frequency features such as lines and edges are broken up across regions that textured (in the high frequency band) from different misregistered images.

J. Digne et al. also make use of the Low/High frequency decomposition scheme to address the problem of high fidelity scan merging [2]. Cedric Allene et al. go a farther way on multi-band blending [3]. They extend the method to more than two frequency bands in a principled manner. Yet they fail to define a reasonable weighting function that can take model shape and viewing angle into account fully.

Victor Lempitsky et al. formulate the texture patch layout problem into a Markov Random Field energy optimization [4]. This method tries to find a balance between texture smoothness and details. Then a seam levelling procedure is applied to remove the residual seams. Ref. [5] extends this work by expanding the combinatorial search to consider local image translations when seeking optimal mosaic. The introduction of MRF indeed reduces many artifacts. However, for sophisticated surface geometries with large number of viewing images, these methods help little in practice. Besides, their seam levelling procedures both fail to achieve perfect smoothness across adjacent patches, seams are visible even in the results demonstrated in papers.

As another interesting work, Bastian Goldluecke proposes a novel approach [6] that allows to recover a high-resolutioon, high-quality texture map even from lower-resolution photographs. This method requires accurate geometry and camera calibration.

In this paper, we develop a more practical, effective and complete texturing framework for 3D modeling. Multi-band blending idea is adopted, clarified and improved. A visibility preprocessing step for solving self-occlusion problem are proposed. Especially, highlight problem, which is always overlooked by previous 3D mapping methods, is handled in a artful way by utilizing the features of our texturing framework. Finally, we describe a novel texture map creating method, which can fully represent the texture data at a low space cost without information loss.

2 Multi-band Blending

A common problem to all applications of photomosaics is how to combine several relative images into a panorama or image mosaic. Due to varying illumination conditions and perspective differences, using some weighted averaging scheme in the overlapping regions to eliminate visible seams is advisable. Apparently, the width of the transition zone plays a crucial role in the blending procedure. If the transition width is small compared to the image features, seams will still be visible. Conversely if it is larger than the image features, features from all images may appear as ghosting within the transition zone. Under some circumstances, deciding an appropriate transition width to achieve color and lighting continuity without introducing ghosting artifacts is very difficult, or even impossible. For instance, if the input images have very large and very small features at the same time, it would be hard to choose an appropriate size as the transition width.

From the viewpoint of the frequency theory, an image can be regarded as a collection of different frequency signals and the sizes of features are proportional to the corresponding wave lengths. If we can decompose the images into a set of band-pass filtered component images and blend them in different frequency domains, then the final image mosaic can be obtained by simply sum all the image mosaics of different bands. This is the basic idea of [7]. Adam Baumberg extends this technique to texturing 3D models by using a two-band decomposition and then fusing the low frequencies while keeping intact the high frequency content [1]. The work is notable, but not enough satisfactory for complex scenes whose features cover a large range of scale.

To remove seams, the transition width should be comparable to the largest feature, and to avoid ghosting, the transition width should not be much larger than the smallest feature. We should "slice" the frequency of the image into more bands to restrict the sizes of features in each band. Then a proper transition width would be easier to achieve.

Following [3, 7], we firstly construct Gaussian pyramids by using a cascade filtering approach. We denote the *b*th level of the Gaussian pyramid of image I as $G_b(I)$. The first level is just a copy of the original image. Any higher level is generated by convoluting the previous level image with a kernel-constant Gaussian function $g(\sigma)$:

$$G_b(I) = G_{b-1}(I) * g(\sigma) \ , \tag{1}$$

where σ is proportional to the length of the diagonal of the object's bounding box.

Then we construct Laplacian pyramids by subtracting the adjacent levels of Gaussian pyramids. We denote the *x*th level of the Laplacian pyramid of image I as $L_b(I)$. The highest level is the same as that of Gaussian pyramid. And the lower levels are calculated as

$$L_b(I) = G_b(I) - G_{b+1}(I) \ . \tag{2}$$

Apparently, one level of Laplacian pyramid is just one component image of the original one that dominates a limited frequency band. By summing all these levels, the original image can be recovered reversely.

Fig. 2 below shows an example of the pyramid construction procedure.

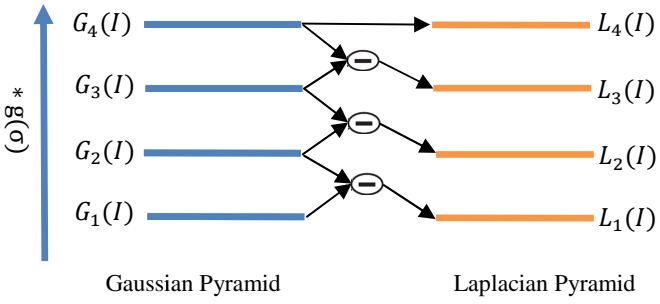


Fig. 2. Constructing a Laplacian Pyramid with 4 levels

As mentioned above, we need to blend texture in the overlapping regions in different frequency bands. Suppose the Laplacian pyramids have B levels, that is, we have decomposed the images into B bands. The texture patch of each triangle of the mesh surface can be calculated as

$$T(x) = \sum_{b=1}^{B} \sum_{i=1}^{|I|} \frac{W_{b,i}(x) * L_b(I_i)(\Pi_i(x))}{\sum_{i=1}^{|I|} W_{b,i}(x)} \quad , \tag{3}$$

where I is the viewing image set, x is a triangle of the mesh surface, Π_i is the projection from 3D space to image I_i and W is a weighting function.

To compute $W_{b,i}(x)$, we firstly calculate the angular deviation (denoted as $\alpha_i(x)$) of the viewing vector of source image I_i from the normal vector of mesh triangle x. Then we calculate $W_{b,i}(x)$ as

$$W_{b,i}(\mathrm{x}) = \begin{cases} \left(\alpha_\mathrm{i}(\mathrm{x}) < \frac{\pi}{2} \cdot \frac{\mathrm{b}-1}{\mathrm{B}-1}\right) ? \left(\frac{\pi}{2} - \alpha_\mathrm{i}(\mathrm{x})\right) : 0, & 2 \le \mathrm{b} \le \mathrm{B} \\ \left(\alpha_\mathrm{i}(\mathrm{x}) == \max\left\{\alpha_\mathrm{j}(\mathrm{x}) \middle| 1 \le \mathrm{j} \le |\mathrm{I}|\right\}\right) ? \left(\frac{\pi}{2} - \alpha_\mathrm{i}(\mathrm{x})\right) : 0, & \mathrm{b} = 1 \end{cases} \tag{4}$$

This strategy suggests that the higher the frequency band is, the less viewing images of one triangle would be used to blend texture. As a result, the transition zones would be proportional in size to the features represented in each band.

Fig. 3 illustrates the steps of adding the blended texture in each band to a 3D model. In practice, we assign B as 3 or 4 in order to take a balance between algorithm effects and computational cost.

Fig. 3. The steps of adding the blended texture in each band from low to high to a 3D model

3 Visibility Processing

So far, we have not taken self-occlusion and visibility issue into account yet. In practice, quite a lot of objects have self-occlusion problem which gives rise to erroneous texture binding in shadow regions. Obviously, we need a preprocessing step to address it.

The overview of our method is as follows. All triangles of the mesh surface are projected into its viewing images via the corresponding calibration information. Among all projected triangles that can cover it, each pixel of the images records the index of the closest one, while setting the others invisible from this image (see Fig. 4).

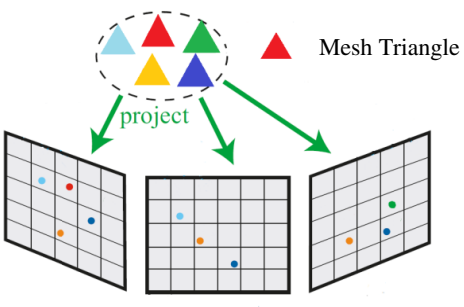


Fig. 4. Each pixel of the images records the nearest triangle that can cover it through projection for now

The details are as follows:

for $i = 1:|I|$

Build array T and V. $T[m, n]$ records the closest triangle to pixel$[m, n]$. $V[x, i] ==$ true means that triangle x is visible from image I_i, not vice versa.

Initiate elements of T and V as -1 and true respectively.

Project each triangle to the current image I_i and then check every pixel that is covered by it.

Suppose the index of current triangle is x.

Suppose current pixel is pixel$[m, n]$.

```
if (T[m, n] == -1)
    T[m, n] = x;
else
    if(dist(x, pixel[m, n]) < dist(T[m, n], pixel[m, n]))
        V[T[m, n], i] = false;
        T[m, n] = x;
    else
        V[x, i] = false;
```

4 Highlight Removal

Due to the surface inhomogeneity of real objects, purely Lambertian relection is not possible. In practice, directional illumination often leads to highlights which cause significant variations in the appearances of an object. As presented in Sect. 2, our texture mapping method can handle varying overall illumination conditions very well, however, highlight problem still need one more specific treatment due to its troublesome distinctiveness.

Previous methods for highlight removal can be roughly separated into two categories by the number of images used. The first category methods [8, 9, 10] uses only one single image to remove the undesirable effects of highlights. The key idea of these methods is that the highlight and neighbor pixels contain useful information for guiding the estimation of the underlying diffuse color. However, they only work well with simple colored images. For more general cases, such as multicolored or complex textured images, obtained information may be insufficient to recover shading and texture in highlight regions. To make the problem more tractable, the second category methods utilize more data from multiple images captured under changing conditions, such as different viewpoints [11] or illumination conditions [12, 13]. This type of methods are moderately practical, since their requirements limit their applicability.

We need a specific approach to handle the highlight problem in our texture mapping pipeline. In fact, the context of multi-view reconstruction gives us much convenience to circumvent some limiting assumptions made in previous works. First, each part of an object can be saw in several images in the given sequence. Such redundancy guarantees that the highlight regions in one image can always have highlight-free counterparts in other reference images. Second, the correspondence relationship between pixels of different images are already prepared. Finally, the Multi-band Blending alleviate the difficulty of highlight region replacement.

We develop a simple but very effective approach to remove highlight effects. The highlight regions are detected on the mesh triangle level (remeshing is employed beforehand to guarantee the mesh regularity.), based on the color differences between the corresponding areas in image sequence. After visibility processing, each mesh triangle can be projected into its "visible" images via calibration information. Then we calculate the average colors of these projected regions respectively and find the median. If the average color of some region deviates much from the median value, this region will be identified as a highlight region. Concerning color value deviation, "much" is 20 percent of the R, G, B values in our implementation. Then for each image, we adopt BFS to group the highlight triangle regions and use proportionable circles to cover these groups. So far highlight detection is completed.

A key observance that we can take advantage of is that, due to the relative movement of viewing direction and object, a highlight spot in some image is often out of highlight areas in other images. With calibration information, highlight pixels can be resampled by blending their counterparts in other reference images. After that, Poisson Image Editing [14, 15] is introduced to seamlessly integrate the resampled regions into target images. This measure can change the overall illumination of resampled highlight regions and reduce the lighting inconsistency along the boundaries. However, as mentioned in Sect. 2, any blending strategy (e.g., weighted averaging according to the areas of the projected highlight regions in each image.) inevitably leads to ghosting artifacts. In the context of

our texture mapping pipeline, this problem can be circumvented in a very natural and effective way. The key insight is, in the process of Multi-band Blending, we only extract low level frequency information from these resampled highlight regions. For these regions, benefited from the overlapping of image sequence, we can always obtain high level frequency information from other reference images. Fig. 5&6 shows some illustrations of our highlight removal approach.

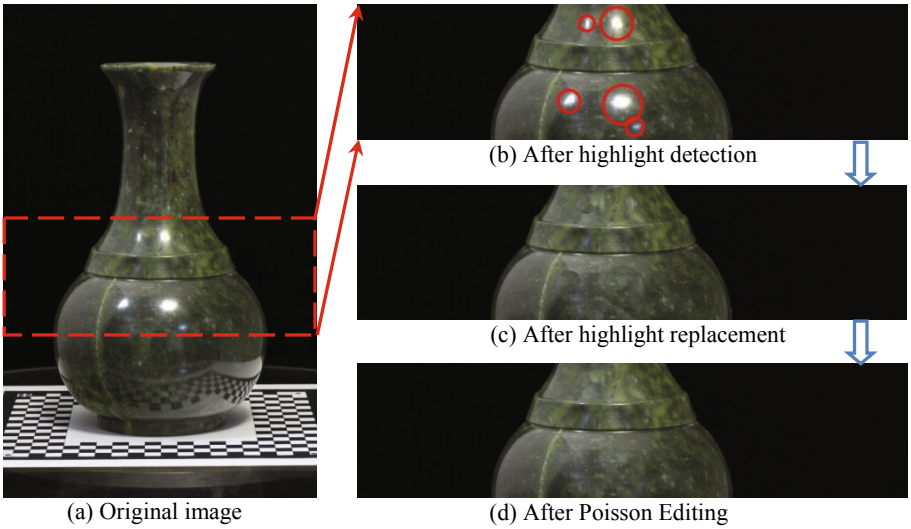


Fig. 5. The procedure of highlight removal in one image. (b), (c) & (d) show only the ROI.

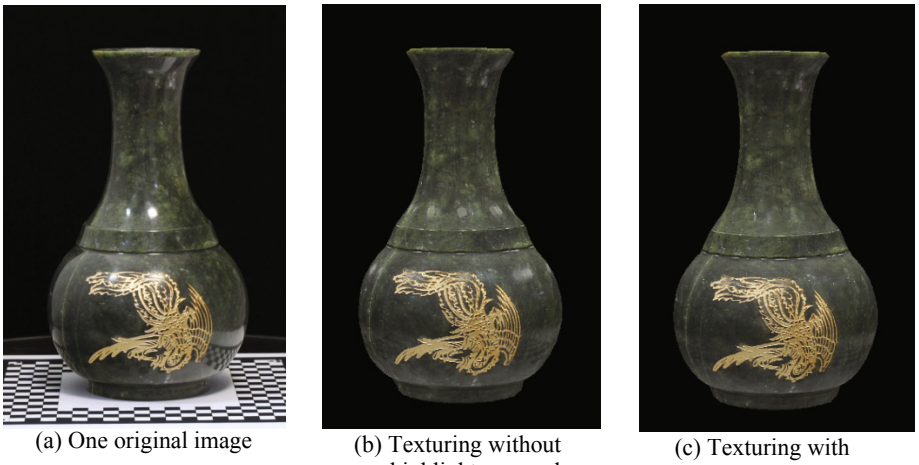


Fig. 6. Comparison between texturing without and with highlight removal

5 Texture Map Creation

Finally, we need to pack texture data generated into a single texture map for two reasons. First, blending different (band) images during rendering is inefficient. It's more practical to pre-compute and store the blended texture, which allows the object could be rendered within just one pass. Second, the reconstructed models should be encoded into standard formats in order to be displayed in virtual environments.

The "box" scheme proposed in [1] suffers from two problems. First, it cannot guarantee that every triangle is completely visible in one of the 6 canonical views (top view, front view etc.), which means that the texture data might not be fully represented in the texture map. For example, obviously, the inwall of a cup placed levelly cannot be visible in any canonical view. Second, the combination of the 6 canonical views is not guaranteed to be optimal. There might be another combination of views that can preserve texture data better.

Another reference method comes from [16], which places texture patches of triangles randomly with a surrounding auxiliary area. This method can preserve the full texture of an object, no matter how complex its shape is. However, the randomness of texture patch placement leads to a texture map that is difficult to read for human. In addition, the uniform size of texture triangles somehow causes information loss. Furthermore, this method also suffers from the problem of image space wasting due to a lot of "padding" between texture triangles.

Considering all the advantages and disadvantages of these two strategies, we implement an improved approach of packing texture map.

A preliminary version is as follows: According to the viewing angle, every triangle has a "best viewing image". Based on this "best viewing image" we can easily and naturally partition the surface into several areas. For each area, we can project it to the corresponding "best viewing image", and save its texture data using exactly the same shape of the projected area. Then we sort them in ascending order of the polar angle of area center and pack them together. Finally, for the sake of mipmapping while rendering, we need to pad a three or four pixel width band to enclose each area.

The method presented above works well when the target object has regular shape and the input image number is small. However, if the object is so sophisticated that twenty or more images are needed to cover it, the texture surface would be cut into so many fragmentary areas that it becomes impossible to pack them together without much waste of space (we would need more "padding"). This situation also undermines our intention to make the texture map human-readable, although the texture data is fully represented. So it is necessary to pick out a subset of images which can "see" the object with fairly good viewing angle as a whole.

Let's begin with some definitions:

$\alpha_{i,t}$: the angle between the view line of image *i* and the face normal of triangle *t*.
α_{Thres}: a threshold of α, such as $\pi/10$.
I: the input image set.
T: the mesh triangle set.
V_t: a conditioned viewing image set of triangle *t*.
V_t is calculated as: $V_t = \{\, i \mid \alpha_{i,t} < \alpha_{Thres} || \alpha_{i,t} == min\,\{\alpha_{j,t} | 0 \le j \le |I|\}\,\}$

We transform the problem of selecting an appropriate subset of the input images into a Minimal Hitting Set problem: finding out the minimal subset of I that contains at least one element of each V_t $(0 \le t \le |T|)$. As we all know, the Minimal Hitting Problem is NP-Complete. However, the particularity of our context enables us to perform an efficient algorithm which can give an exact solution within tolerable time in most cases. As for other cases in which the generation of an exact solution would cost too much time, an approximate solution is given.

Let we refer the target subset of I as S. Considering that many triangles are visible from only one image, we process them first:

```
for t = 1: |T|
    if (|V_t| == 1)
        S = S ∪ V_t
```

The next step depends on the number of images not in S now. If $|I| - |S| < n$ (in practice, we assign n = 10), we perform an exhaustive search to find an exact solution:

```
for num = 1: (|I| − |S|)
    for ∀ I′ ⊆ (I − S) && |I′| == num
        if I′ hit all remaining traingles
            S = S ∪ I′; break;
```

If $|I| - |S| \ge n$, by controlling α_{Thres}, we can ensure that $|V_t| \le k$ $(1 \le t \le |T|)$, where k is a constant. Then we use the k-approximation of k-hitting algorithm to get an approximate solution:

```
while any un-hit triangle exists
    suppose the current triangle is t
        S = S ∪ V_t
```

α_{Thres} indeed limits the lost of texture information. And S guarantees that most neighborhood relations of the surface triangles are preserved, which means less "padding".

After obtaining S, every triangle has a "best viewing image" among S. The subsequent texture map creating steps are similar to that of the preliminary version described earlier. Fig. 7 shows a result.

In contrast with the reference methods, our texture map creating scheme is obviously outstanding. It not only guarantees that every triangle visible from some image can have its texture data fully represented in the texture map, but also ensures the texture map to be human-readable to some extent.

Fig. 7. Left - a reconstructed model. Right – corresponding texture map

6 Experiments

We demonstrate the effectiveness of our texturing method by comparing it with other four methods:

A. *Best Viewing Image*: For each triangle, choose texture from best viewing image.
B. *Global Blending*: Weighted averaging used all visible images.
C. *Boundary Blending*: Blend texture alone boundaries.
D. *Two-band Blending*: The method proposed in [1].

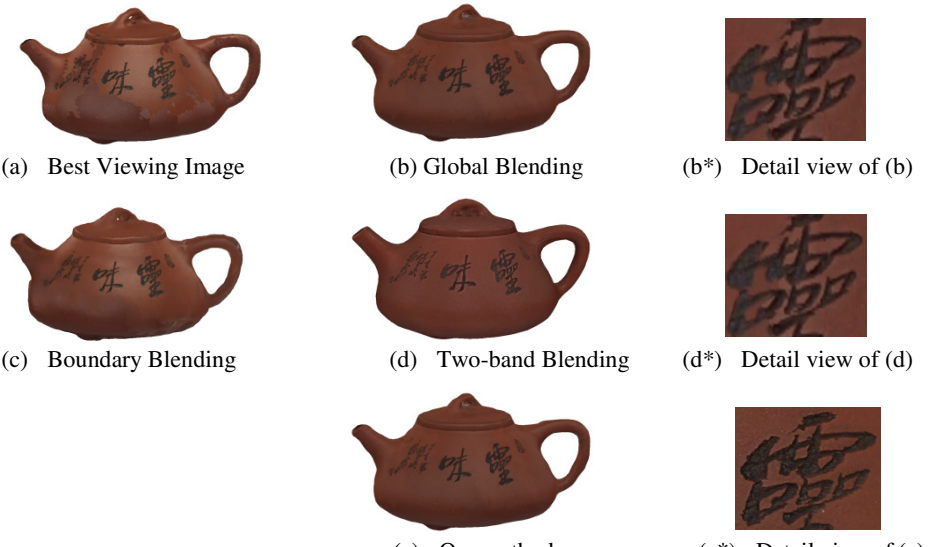
We conduct hundreds of experiments with different objects. Fig. 8 gives a representative example. This example uses an image sequence consisting of 16 images (3888×2592) of a teapot to test these methods. Note that, due to different surface reconstruction approaches, the mesh models may have minor differences.

(A) Although the Best Viewing Image method can preserve fine texture details very well, it produces noticeable seams due to varied lighting and camera conditions. (B) The Global Blending method can obtain seamless texture with no jumps in color and lighting appearance, however, at the cost of introducing ghosting artifacts. (C) The Boundary Blending method likes a balance or compromise of the two methods above. It tries to relieve the color discrepancies and lighting discontinuities while avoiding ghosting artifacts. Unfortunately, the effort is not enough satisfactory. (D) The overall performance of the Two-band Blending method is quite good. However, due to information loss in the texture map creation step, it fails to remain the features sharp. Besides, some feature lines appear broken up. (E) In contrast, our method not only achieves global texture smoothness but also preserves details very well.

Fig. 9-11 show more example results of our method. Note that the source images are not all displayed.

7 Conclusion

We have presented a novel and practical framework for texture mapping of 3D models in a multi-view setting, where several calibrated views of a object with known surface geometry are available. The basic technique we employ is multi-band blending. We extend it in several respects while taking self-occlusion and highlight problem into



(a) Best Viewing Image (b) Global Blending (b*) Detail view of (b)

(c) Boundary Blending (d) Two-band Blending (d*) Detail view of (d)

(e) Our method (e*) Detail view of (e)

Fig. 8. Comparison of different texturing methods

Fig. 9. Signet

Fig. 10. Vase

Fig. 11. Crocodile

account. At last, we propose an efficient mechanism to create a single texture map that provides full representation of the texture data.

Our work relies on a modest prerequisite that the images are enough to cover the full range of target object and have relatively wide overlapping regions. Except this, we don't require strong assumption on lighting condition. And we don't need an exact geometry model or perfect calibration data as well. Our method can achieve color continuity and detail preservation while avoiding noticeable seams and ghosting artifacts. On account of the idea's generality, this framework not only can handle indoor objects, but also can be extended to deal with large-scale scenes.

Acknowledgements. This work was supported by National Basic Research Program of China (2010CB328002), National High-tech R&D Program of China (2011AA120301) and National Natural Science Foundation of China (60973052, 61121002, 61173080).

References

1. Baumberg, A.: Blending Images for Texturing 3D Models. In: BMVC (2002)
2. Digne, J., Morel, J.-M., Audfray, N., Lartigue, C.: High Fidelity Scan Merging. Computer Graphics Forum 29(5) (2010)
3. Allene, C., Pons, J.-P., Keriven, R.: Seamless Image-Based Texture Atlases using Multi-band Blending. In: ICPR (2008)
4. Lempitsky, V., Ivanov, D.: Seamless Mosaicing of Image-Based Texture Maps. In: CVPR (2007)
5. Gal, R., Wexler, Y., Ofek, E., Hoppe, H., Cohen-Or, D.: Seamless Montage for texturing Models. In: EUROGRAPHICS (2010)

6. Goldluecke, B., Cremers, D.: Superresolution Texture Maps for Multiview Reconstruction. In: ICCV (2009)
7. Burt, P.J., Adelson, E.H.: A Multiresolution Spline with Application to Image Mosaics. ACM Trans. on Graphics 2(4) (1983)
8. Tan, R.T., Ikeuchi, K.: Separating Reflection Components of Textured Surfaces Using a Single Image. In: ICCV (2003)
9. Tan, P., Lin, S., Quan, L.: Separation of Highlight Reflections on Textured Surfaces. In: CVPR (2006)
10. Yang, Q., Wang, S., Ahuja, N.: Real-Time Specular Highlight Removal Using Bilateral Filtering. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 87–100. Springer, Heidelberg (2010)
11. Lin, S., Li, Y., Kang, S.B., Tong, X., Shum, H.-Y.: Diffuse-Specular Separation and Depth Recovery from Image Sequences. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2352, pp. 210–224. Springer, Heidelberg (2002)
12. Petschnigg, G., Agrawala, M., Hoppe, H., Szeliski, R., Cohen, M., Toyama, K.: Digital Photography with Flash and No-Flash Image Pairs. In: SIGGRAPH (2004)
13. Agrawal, A., Raskar, R., Nayar, S.K., Li, Y.: Removing Photography Artifacts using Gradient Projection and Flash-Exposure Sampling. In: SIGGRAPH (2005)
14. Perez, P., Gangnet, M., Blake, A.: Poisson Image Editing. In: SIGGRAPH (2003)
15. Feng, J., Zhou, B.: Automatic Highlight Removal in Visual Hull Rendering by Poisson Editing. In: VRCAI (2008)
16. Maruya, M.: Generating a Texture Map from Object-Surface Texture Data. In: EUROGRAPHICS, vol. 14(3) (1995)