Patch-Based Optimization for Image-Based Texture Mapping

SAI BI, University of California, San Diego
NIMA KHADEMI KALANTARI, University of California, San Diego
RAVI RAMAMOORTHI, University of California, San Diego

Fig. 1. The goal of our approach is to produce a high-quality texture map given the geometry of an object as well as a set of input images and their corresponding camera poses. A small subset of our input images as well as the rough geometry, obtained using the KinectFusion algorithm, are shown on the left. Since the estimated geometry and camera poses are usually inaccurate, simply projecting the input images onto the geometry and blending them produces unsatisfactory results with ghosting and blurring artifacts. We handle the inaccuracies of the capturing process by proposing a novel patch-based optimization system to synthesize aligned images. Here, we show different views of an object rendered using OpenGL with the texture map generated using our system. Our approach produces high-quality texture maps and outperforms state-of-the-art methods of Waechter et al. [2014] and Zhou and Koltun [2014].

Image-based texture mapping is a common way of producing texture maps for geometric models of real-world objects. Although a high-quality texture map can be easily computed for accurate geometry and calibrated cameras, the quality of texture map degrades significantly in the presence of inaccuracies. In this paper, we address this problem by proposing a novel global patch-based optimization system to synthesize the aligned images. Specifically, we use patch-based synthesis to reconstruct a set of photometrically-consistent aligned images by drawing information from the source images. Our optimization system is simple, flexible, and more suitable for correcting large misalignments than other techniques such as local warping. To solve the optimization, we propose a two-step approach which involves patch search and vote, and reconstruction. Experimental results show that our approach can produce high-quality texture maps better than existing techniques for objects scanned by consumer depth cameras such as Intel RealSense. Moreover, we demonstrate that our system can be used for texture editing tasks such as hole-filling and reshuffling as well as multiview camouflage.

CCS Concepts: • **Computing methodologies** → **Computational photography**;

Additional Key Words and Phrases: image-based texture mapping, patch-based synthesis

ACM Reference format:
Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. 2017. Patch-Based Optimization for Image-Based Texture Mapping. *ACM Trans. Graph.* 36, 4, Article 106 (July 2017), 11 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073610

1 INTRODUCTION

Modeling real-world objects is an important problem with a variety of applications including video games, virtual reality, and animations. Geometry reconstruction has been the subject of extensive research and many powerful algorithms have been developed [Seitz et al. 2006]. With the availability of consumer depth cameras to the public, ordinary users are now able to produce geometric models of objects using techniques like KinectFusion [Newcombe et al. 2011].

However, reproducing the full appearance of real-world objects also requires reconstructing high-quality texture maps. Image-based texture mapping is a common approach to produce a view-independent texture map from a set of images taken from different viewpoints. However, this is a challenging problem since the geometry and camera poses are usually estimated from noisy data, and thus, are inaccurate. Moreover, the RGB images from consumer depth cameras typically suffer from optical distortions which are not accounted for by the camera model. Therefore, naïvely projecting and combining the input images produces blurring and ghosting artifacts, as shown in Fig. 2.

We observe that we can overcome most of the inaccuracies by generating an aligned image for every input image. Our method builds upon the recent work by Zhou and Koltun [2014] that proposes an optimization system to correct the misalignments using local warping. Although this approach handles small inaccuracies, it fails to produce high-quality results in cases with large inaccuracies and missing geometric features because of the limited ability of local warping in correcting misalignments (see Figs. 1, 2 and 4).

Inspired by the recent success of patch-based methods in image and video editing tasks, we propose a novel global patch-based optimization system to *synthesize* aligned images. Our energy function combines our two main desirable properties for the aligned images; **1)** include *most* of the information from the original input images, and **2)** preserve the photometric consistency of the projection. By

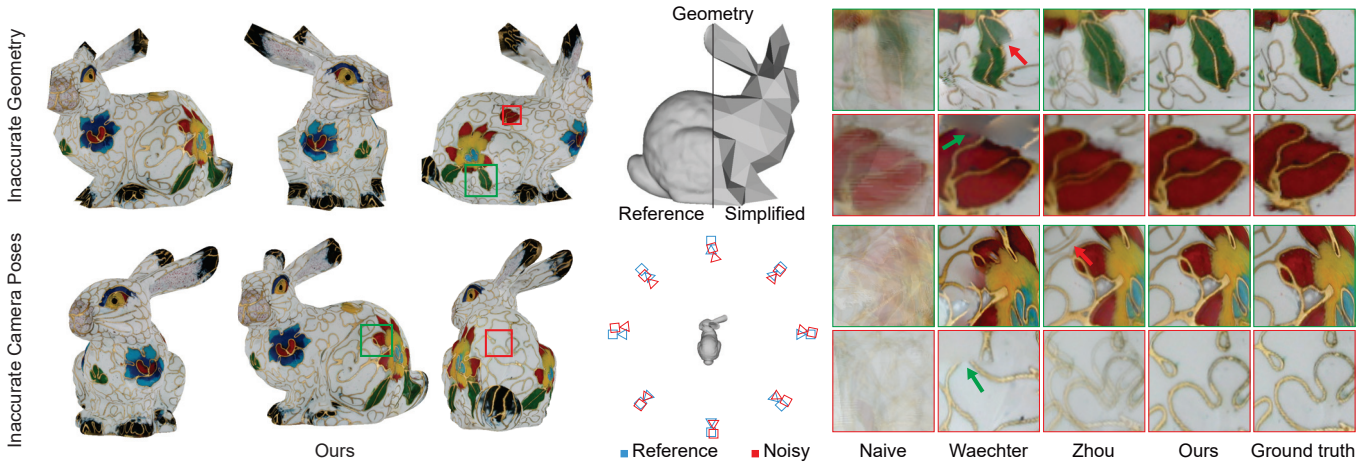

Fig. 2. We generate 24 input views by rendering a synthetic textured bunny from different viewpoints and artificially add inaccuracies by simplifying the geometry and adding noise to the camera poses. We compare our approach against state-of-the-art methods as well as naïvely projecting and combining the images. Waechter et al.'s approach [2014] selects a single view per face by solving a complex optimization system to reduce the artifacts around the face boundaries. However, their results contain visible seams because of large inaccuracies in the geometry and camera poses. Zhou and Koltun [2014] tackle the inaccuracies of the geometry by using local warping to align the input images, but fail to properly register the images and produce unsatisfactory results. Moreover, when the camera poses are significantly inaccurate, their system converges to a local minimum, and thus, their results suffer from ghosting and blurring artifacts. Our approach is more flexible and can properly synthesize aligned images that when combined produce artifact-free texture in both cases.

optimizing our proposed energy function, we simultaneously maximize the local similarity of the aligned and input images and ensure the consistency of all the aligned images and the texture map.

Our system draws information from the source images in a patch-based manner, and thus, is flexible and able to handle large inaccuracies. Moreover, our method handles cases with missing geometric features (see Fig. 4) by synthesizing the missing content, while the existing warping-based [Zhou and Koltun 2014] and graph-cut based [Waechter et al. 2014] techniques are not able to do so. Finally, in contrast to Zhou and Koltun's approach, we perform the optimization in the image domain which makes the performance of our system independent of the complexity of the geometry. In summary, we make the following contributions:

- We introduce the first patch-based optimization system for view-independent image based texture mapping (Sec. 3.1). Our method corrects misalignments by synthesizing aligned images which can then be used to produce a single view-independent texture map.
- We propose a simple iterative two-step approach to efficiently solve our energy equation (Sec. 3.2).
- We demonstrate that our approach produces better results than existing techniques (Sec. 5). Furthermore, we show other applications of our system (e.g., texture hole-filling) which are not possible to do with the current methods.

2 RELATED WORK

Reproducing the full appearance of a real-world object from a set of images has been the subject of extensive research. Image-based rendering approaches [Buehler et al. 2001; Hedman et al. 2016] reproduce the appearance of an object by generating a view-dependent texture map [Debevec et al. 1996]. However, these methods are only able to provide the ability to navigate an object with the lighting condition of the input photographs. Therefore, they cannot be used for applications where the goal is to use the scanned object in a new environment with different lightings. Moreover, since these approaches do not produce a globally consistent texture map, they are typically not used in gaming, augmented reality, and animations.

View-independent texture mapping approaches like our own, produce a single consistent texture map from a set of images captured from different viewpoints, which can then be rendered with different lightings.[1] The main challenge of these methods is addressing the inaccuracies in the capturing process. Several methods have been presented to register the images to the geometry in a semi-automatic way [Franken et al. 2005; Ofek et al. 1997; Pighin et al. 1998] or automatically by, for example, optimizing color consistency [Bernardini et al. 2001; Pulli and Shapiro 2000], aligning image and geometric features [Lensch et al. 2001; Stamos and Allen 2002], and maximizing the mutual information between the projected images [Corsini et al. 2013, 2009]. While these methods are effective at addressing the camera calibration inaccuracies, they are not able to handle inaccurate geometry, and optical distortions in RGB images which are common problems of consumer depth cameras.

A small number of approaches have been proposed to tackle general inaccuracies. We categorize these approaches in two classes and discuss them in the following two subsections.

2.1 Single View Selection

Instead of blending the projected input images, which could generate blurry results because of misalignments, these approaches select only one view per face. To avoid visible seams between the boundaries of each face, they typically solve a discrete labeling problem [Lempitsky and Ivanov 2007; Sinha et al. 2008; Velho and Sossai Jr. 2007; Waechter et al. 2014].

For example, the state-of-the-art method of Waechter et al. [2014] solves a conditional random field energy equation, consisting of two terms: a data term which favors views that are closer to the current

[1]Note that, the final textures in this case still have the original lighting condition. However, this problem can be addressed by applying intrinsic decomposition on the source images and using albedo to generate the texture maps.

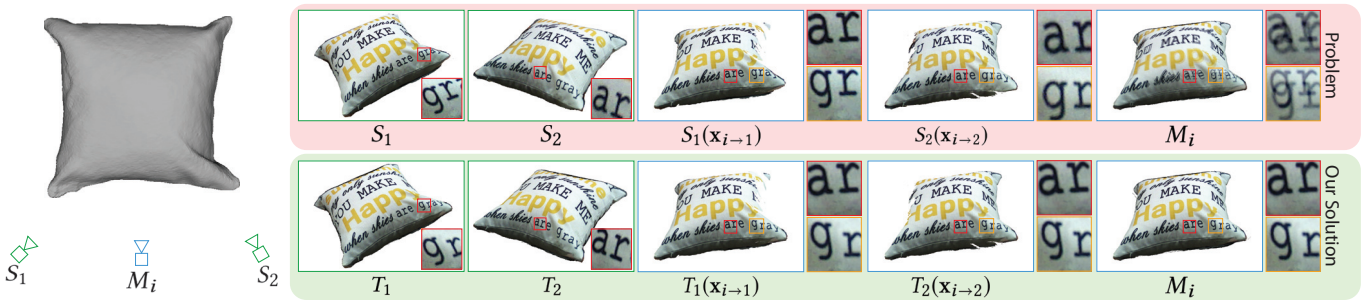

Fig. 3. Here, the goal is produce a high-quality texture map, using the rough geometry as well as two source views, shown on the left. We illustrate the texture mapping process from a novel view, shown with the blue camera. Because of the inaccuracies in the geometry, camera poses, and optical distortions of the source images, the projected source images to view i, $S_1(\mathbf{x}_{i\to 1})$ and $S_2(\mathbf{x}_{i\to 2})$, are typically misaligned. For example, the inset shows that the position of "ar" and "gr" in the two projected source images is different. Therefore, combining (averaging in this case) these projected source images produces a texture map, M_i, with blurring and ghosting artifacts. We propose to handle this misalignment problem by synthesizing a target image for every source image in a way that the projected target images are photometrically consistent. To reconstruct each target image, we keep the overall visual appearance of the source image, but move its content to correct the misalignments. Note the difference in position of "gr" and "ar" in the source and target images. In this case, since the projected images are aligned, we are able to produce a ghost-free high-quality texture map.

face and are less blurry, and a smoothness term which penalizes inconsistencies between adjacent faces. However, as shown in Fig. 2, even this approach is not able to handle the large inaccuracies in challenging cases, producing visible seams in the final texture map.

2.2 Image Alignment

The approaches in this category directly handle the inaccuracies by aligning the input images. Tzur and Tal [2009] propose to estimate local camera projection for each vertex of the geometry to handle inaccuracies from calibration, geometry, etc. However, their approach requires user interaction to produce plausible results. Aganj et al. [2010] address misalignment by finding matching SIFT features in different views and warping the input images, while others [Dellepiane et al. 2012; Eisemann et al. 2008] perform warping using optical flow. These methods do not minimize the distortion globally and work on a pair of images, and thus, are sub-optimal. Gal et al. [2010] assigns each triangle to one input image and finds the optimum shift for each triangle to remove the seams, but their optimization is computationally expensive.

The recent work of Zhou and Koltun [2014], which our method builds upon, solves an optimization system to find optimum camera poses as well as non-rigid corrections of the input images, simultaneously. They use local warping to perform non-rigid alignment and propose an alternating optimization to minimize their objective function. However, the local warping is not able to correct large misalignments and it produces results with ghosting and blurring artifacts in challenging cases, as shown in Fig. 2. To avoid this problem, we propose a different optimization system with a more flexible mechanism for non-rigid alignment than local warping.

2.3 Patch-Based Synthesis

Our approach is inspired by the recent success of patch-based synthesis methods in a variety of applications such as hole-filling [Wexler et al. 2007], image retargeting and editing [Barnes et al. 2009; Simakov et al. 2008], morphing [Shechtman et al. 2010], HDR reconstruction [Kalantari et al. 2013; Sen et al. 2012], and style transfer [Bénard et al. 2013; Jamriška et al. 2015]. Patch-based synthesis has been shown to be particularly successful in applications where finding correspondences between two or multiple images (e.g., morphing and HDR reconstruction) is difficult. In our application, the synthesized aligned images need to be consistent with respect to the object's geometry, and thus, direct application of patch-based synthesis to our problem does not work. We address this challenge by proposing a novel patch-based energy equation which incorporates the geometry into the formulation.

3 ALGORITHM

The goal of most image-based texture mapping approaches is to produce a high-quality view-independent texture map using a set of N source images, $S_1, \cdots, S_N$, taken from different viewpoints. These methods usually assume that the object's approximate geometry and rough camera poses (i.e., extrinsic and intrinsic parameters) of all the source images are already estimated using existing techniques [Newcombe et al. 2011; Seitz et al. 2006]. Once the texture map is created, the object with a view-independent texture can be rendered from any novel views.

A simple way to produce a texture map is to project the source images onto the geometry and combine all the projected images. Ideally, these projected images are photometrically consistent, and thus, combining them produces a high-quality texture map. However, in practice, because of the inaccuracies, the projected images are typically misaligned. Therefore, this simple approach produces texture maps with ghosting artifacts.

We show this problem in Fig. 3 (top row) for a case with two source images S_1 and S_2. To observe the misalignment problem, we project the source images to a novel view i. Note that, projection from a source image S_j to a novel view i can be performed by remapping the source image's pixel colors, $S_j(\mathbf{y})$. Here, $\mathbf{y}$ is the projection of the pixels from image i to j. Formally, we can write this as:

$$\mathbf{y} = \mathcal{P}_j(\mathcal{G}_i(\mathbf{x})),$$

where $\mathbf{x}$ is the pixel position on image i, $\mathcal{G}_i$ projects a pixel on image i to the global 3D space, and $\mathcal{P}_j$ projects a 3D point to the image j. In this paper, for clarity and simplicity of the notation, we use $\mathbf{x}_i$ and $\mathbf{x}_{i\to j}$ to denote the pixels on image i and the pixels projected from image i to j, respectively. In this case, $\mathbf{y} = \mathbf{x}_{i\to j}$ and $S_j(\mathbf{x}_{i\to j})$ is the result of projecting source image S_j to view i. See Table 1 for the complete list of notation used in this paper.

$S_1, \ldots, S_N$	source images (input)
$T_1, \ldots, T_N$	target (aligned) images (output)
$M_1, \ldots, M_N$	texture at different views (output)
$\mathbf{x}_i$	pixel position on image i
$\mathbf{x}_{i\to j}$	pixel position projected from image i to j
$T_j(\mathbf{x}_{i\to j})$	RGB color of the j^{th} target image at pixel $\mathbf{x}_{i\to j}$, i.e., the result of projecting target j to camera i

Table 1. Notation used in the paper.

As shown in Fig. 3 (top row), because of the inaccuracies in the estimated geometry and camera poses, the projected source images, $S_1(\mathbf{x}_{i\to 1})$ and $S_2(\mathbf{x}_{i\to 2})$, are misaligned. Therefore, the texture map generated by the simple projection and blending approach contains ghosting artifacts (rightmost column). Here, M_i refers to the final globally consistent texture map, seen from camera i. Note that, M_j is reconstructed from all the source images, and thus, is different from the projected source images.

To overcome this misalignment problem, we propose to *synthesize* an aligned (target) image, T_i, for every source image, S_i. As shown in Fig. 3, the targets are reconstructed by moving the content of the source images to correct the misalignment. As a result, all the target images are photometrically consistent, and thus, projecting them onto the geometry and combining them produces a high-quality result. In the next section, we explain our patch-based optimization system to synthesize these target images.

3.1 Patch-Based Energy Function

Our main observation is that to produce a high-quality texture map, the target images should have two main properties: **1)** each target image should be similar to its corresponding source image, and **2)** the projected target images should be photometrically consistent. Our goal is to propose a global energy function which codifies these two main properties.

To satisfy the first property we ensure that each target image contains *most* of the information from its corresponding source image in a visually coherent way. To do so, we use bidirectional similarity (BDS) as proposed by Simakov et al. [2008]. This is a patch-based energy function which is defined as:

$$E_{\text{BDS}}(S,T) = \frac{1}{L}\Big(\underbrace{\sum_{s\subset S}\min_{t\subset T} D(s,t)}_{\text{completeness}} + \alpha \underbrace{\sum_{t\subset T}\min_{s\subset S} D(s,t)}_{\text{coherence}}\Big), \tag{1}$$

where α is a parameter defining the ratio of these two terms, s and t are patches from the source S and target T images respectively, and D is the sum of squared differences of all the pixel values of the patches s and t in RGB color space. Moreover, L is the number of pixels in each patch, e.g., $L = 49$ for a 7×7 patch.

Here, the first term (completeness) ensures that every source patch has a similar patch in the target and vice versa for the second term (coherence). The completeness term measures how much information from the source is included in the target, while the coherence term measures if there are any new visual structures (artifacts) in the target image. Minimizing this energy function ensures that most of the information from the source is included in the target image in a visually coherent way. In our implementation, we set $\alpha = 2$ to give more importance to the coherence term.

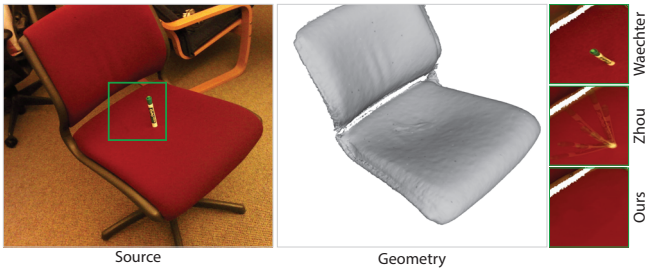

Fig. 4. This scene demonstrates a chair and a green marker on top. Because of the inaccuracies of the consumer depth camera, the marker's geometry is not reconstructed. In this case, the texture from the marker should not appear in the final texture map. The method of Waechter et al. [2014] selects one of the source images for each triangle in the geometry. Since the marker exists in all the source images, this method incorrectly places it in the final texture. Zhou and Koltun [2014] align the source images by locally warping them. Therefore, they are not able to remove the marker from the aligned images, resulting in ghosting artifacts. Our patch-based approach synthesizes the target images, and thus, only includes valid information from the source images. Therefore, we are able to remove the marker from the source images and produce an artifact-free result.

Note that, Eq. 1 is defined for a single pair of source and target images. To enforce the similarity property for all the images, we extend this equation as:

$$E_1 = \sum_{i=1}^{N} E_{\text{BDS}}(S_i, T_i). \tag{2}$$

Patch-based synthesis is more flexible than local warping [Zhou and Koltun 2014], and thus, is more suitable to handle large inaccuracies in the geometry and the camera poses. Furthermore, while local warping inherently preserves the visual coherency, it includes *all* the information from the source in the aligned (target) image which is not desirable in our application. If the geometric model does not contain specific features, the regions corresponding to these features should not be included from the source images in the texture map. Therefore, this method produces results with blurring and ghosting artifacts in these regions, as shown in Fig. 4. Waechter et al.'s method [2014] selects one view per face and can avoid ghosting artifacts in this case. However, this approach is not able to remove the texture corresponding to the missing feature, since it exists in all the source images. Note that, missing geometric features occur in most cases with significantly inaccurate geometry (Fig. 9), which is why the existing techniques poorly handle these challenging cases.

Although the similarity of the target and source images is a necessary condition for generating a high-quality texture map, it is not sufficient, as shown in Fig. 5. Therefore, we need to enforce the second property by ensuring the consistency of the target images. This constraint can be implemented in several ways. For example, we can enforce the consistency by ensuring that the projected target images are close to the current target, i.e., $T_j(\mathbf{x}_{i\to j}) = T_i(\mathbf{x}_i)$. This constraint can be formally written as the ℓ^2 distance between $T_j(\mathbf{x}_{i\to j})$ and $T_i(\mathbf{x}_i)$ and be minimized in a least square sense.

Alternatively, the constraint can be enforced by ensuring the consistency of the current target and average of all the projected targets, i.e., $1/N \sum_{j=1}^{N} T_j(\mathbf{x}_{i\to j}) = T_i(\mathbf{x}_i)$. Similarly, we can enforce the texture at view i to be consistent with the projected target images, i.e., $T_j(\mathbf{x}_{i\to j}) = M_i(\mathbf{x}_i)$, to enforce the constraint. Since all the target images will be consistent with each other and the final texture map after optimization, these different approaches

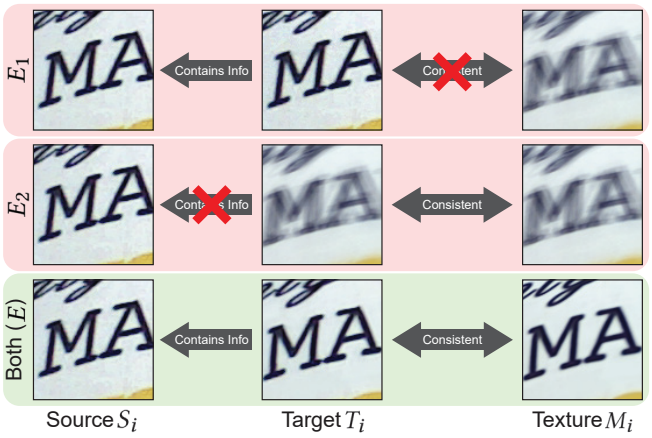

Fig. 5. We evaluate the effect of enforcing our main properties with the two terms in Eq. 5. Only optimizing the first term ensures that the target image contains most of the information of the source image. However, since the consistency constraint is not enforced, the final texture is not consistent with the target image. On the other hand, by only optimizing the second term, the consistency constraint is enforced, and thus, the target image and the final texture are photometrically consistent. However, the target image has ghosted content which does not appear in the source image. Our full approach optimizes both terms and ensures that the targets contain source contents and are consistent. Therefore, only our full approach is able to produce a high-quality texture map.

result in similar optimum target images. However, to be able to utilize alternating optimization (see Sec. 3.2), we use the last strategy ($T_j(\mathbf{x}_{i\to j}) = M_i(\mathbf{x}_i)$) and write our consistency energy equation as:

$$E_C(\{T_j\}_{j=1}^N, M_i) = \frac{1}{N}\sum_{\mathbf{x}_i}\sum_{j=1}^{N} w_j(\mathbf{x}_{i\to j})\left(T_j(\mathbf{x}_{i\to j}) - M_i(\mathbf{x}_i)\right)^2, \quad (3)$$

where the first summation is over all the pixel positions $\mathbf{x}_i$ on image i. Here, the weight w_j enforces the constraint to be proportional to the contribution of the j^{th} projected target image. In our implementation, $w_j = cos(\theta)^2/d^2$, where θ is the angle between the surface normal and the viewing direction at image j and d denotes the distance between the camera and the surface.[2] This weight basically gives smaller weight to the cameras that look at the surface at a grazing angle and are further away from the object. Minimizing this energy function ensures that all the target images are consistent with the final texture map viewed from camera i. We extend this equation to enforce the consistency constraint for all the images as:

$$E_2 = \sum_{i=1}^{N} E_C(\{T_j\}_{j=1}^N, M_i) \quad (4)$$

To satisfy our two properties, we propose the complete objective function to be the weighted summation of E_1 and E_2:

$$E = E_1 + \lambda E_2, \quad (5)$$

where λ defines the weight of the consistency term and we set it to 0.1 in our implementation. Optimizing our proposed patch-based energy function produces target images that contain most of the information from the source images, are visually coherent, and preserve the consistency of the projection. Once the optimum target images, T_i, are obtained, they can be used to produce a single consistent texture in different ways. For example, this can be done

[2]We use the interpolated normal and vertex from the fragment shader.

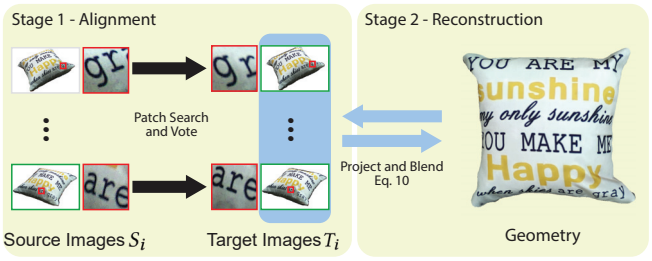

Fig. 6. Our approach synthesizes aligned images (target) by optimizing Eq. 5. We propose to optimize this energy function with a two-step approach. During alignment, patch search and vote is performed between the source and target images to obtain new targets. Note that, while the source and target images are similar, the target image is reconstructed by moving the source content to ensure alignment. In the reconstruction stage, the target images are projected on the geometry and combined (Eq. 10) to produce the texture at different views. The two steps of alignment and reconstruction are continued iteratively and in multiple scales until convergence.

by first projecting all the target images to the geometry. After this process, each vertex receives a set of color samples from different target images. The final color of each vertex can then be obtained by computing the weighted average of these color samples.[3]

We evaluate the effect of each term in our optimization system in Fig. 5. Optimizing the first term alone produces aligned images that have the same visual appearance as the source images, but are not consistent. Optimizing the second term produces consistent target images, but they contain information that does not exist in the source images. Optimizing our proposed full energy function produces a high-quality texture map by enforcing both properties.

3.2 Optimization

To efficiently optimize our energy function in Eq. 5, we propose an alternating optimization approach which simultaneously solves for the target images, $T_1, \cdots, T_N$, and the texture at different views, $M_1, \cdots, M_N$. Specifically, we minimize our energy function by alternating between optimizing our two sets of variables. We initialize the targets and textures with their corresponding source images, i.e., $T_i = S_i$ and $M_i = S_i$. We then iteratively perform our two steps of alignment and reconstruction until convergence. The overview of our algorithm is given in Fig. 6. Below, we explain our two steps:

1) Alignment. In this stage, we fix $M_1, \cdots, M_N$ and minimize Eq. 5 by finding optimum $T_1, \cdots, T_N$. This is done using an iterative search and vote process similar to Simakov et al. [2008]. In the first step, we perform a patch search process, as proposed by Simakov et al., to find the patches with minimum $D(s,t)$ (see Eq. 1), where D is the sum of squared differences. In the next step, we perform the voting process to obtain $T_1, \cdots, T_N$ that minimize Eq. 5 given the calculated patches in the previous step. Note that, as we will discuss next, there is a key difference between our and the original voting [Simakov et al. 2008] which is because of our additional consistency constraint, E_C.

For the sake of clarity, we explain our voting by first discussing each term of Eq. 5 separately.

[3]We can generate the global texture with either the target images, T_i, or the textures, M_i, as they are very similar after optimization.

First (Similarity) Term: We start by rewriting the BDS energy function (E_1) using the obtained patches during search as done in Simakov et al. [2008]:

$$E_1(i, \mathbf{x}_i) = \frac{1}{L}\left[\sum_{u=1}^{U}\left(s_u(\mathbf{y}_u) - T_i(\mathbf{x}_i)\right)^2 + \alpha \sum_{v=1}^{V}\left(s_v(\mathbf{y}_v) - T_i(\mathbf{x}_i)\right)^2\right]. \tag{6}$$

where $E_1(i, \mathbf{x}_i)$ refers to the error E_1 for a specific camera i and pixel $\mathbf{x}_i$. Here, s_u and s_v are the source patches overlapping with pixel $\mathbf{x}_i$ of the target for the completeness and coherence terms, respectively. Moreover, $\mathbf{y}_u$ and $\mathbf{y}_v$ refer to a single pixel in s_u and s_v, respectively, corresponding to the $\mathbf{x}_i^{\text{th}}$ pixel of the target image. Finally, U and V refer to the number of patches for the completeness and coherence terms, respectively. Note that, most of these variables are a function of the current pixel, $\mathbf{x}_i$, but we omit this dependence for simplicity of the notation. See the original paper by Simakov et al. [2008] for the derivation of this equation. To obtain T_i's that minimize the above equation, we need to differentiate the error with respect to the unknown color $T_i(\mathbf{x}_i)$ and set it equal to zero which results in:

$$T_i(\mathbf{x}_i) = \frac{\frac{1}{L}\sum_{u=1}^{U} s_u(\mathbf{y}_u) + \frac{\alpha}{L}\sum_{v=1}^{V} s_v(\mathbf{y}_v)}{\frac{U}{L} + \frac{\alpha V}{L}}. \tag{7}$$

Here, the target is obtained by computing a weighted average of the pixel colors of a set of source patches, overlapping with the $\mathbf{x}_i^{\text{th}}$ pixel of the target image. Note that, although the normalization terms, $1/L$, cancel out, we keep them here to be able to easily combine this equation with the next term (Eq. 8) in Eq. 9.

Second (Consistency) Term: The first term is the standard voting process, as proposed by Simakov et al., and basically draws information from the source image to reconstruct the targets. Our key difference lies in the second term which enforces the consistency constraint by ensuring that the target images are close to the textures. As shown in the Appendix, the targets minimizing the second term in Eq. 5 can be calculated as:

$$T_i(\mathbf{x}_i) = \frac{\frac{1}{N} w_i(\mathbf{x}_i) \sum_{k=1}^{N} M_k(\mathbf{x}_{i\to k})}{w_i(\mathbf{x}_i)}. \tag{8}$$

Again, although the weights, $w_i(\mathbf{x}_i)$, cancel out, we keep them in this equation for clarity, when combining the two terms in Eq. 9. Here, each target is computed by averaging the current texture maps from different views. This is intuitive as the constraint basically enforces the aligned image to be as close as possible to the textures.

Combined Terms: Intuitively, the targets solving the combined terms should be reconstructed by drawing information from the source images, while staying similar to the textures. Since the two terms are combined with a λ factor (see Eq. 5), the combined solution can be computed by separately adding the numerator and denominator of the terms in Eqs. 7 and 8 as:

$$\boxed{T_i(\mathbf{x}_i) = \frac{\frac{1}{L}\sum_{u=1}^{U} s_u(\mathbf{y}_u) + \frac{\alpha}{L}\sum_{v=1}^{V} s_v(\mathbf{y}_v) + \frac{\lambda}{N} w_i(\mathbf{x}_i)\sum_{k=1}^{N} M_k(\mathbf{x}_{i\to k})}{\frac{U}{L} + \frac{\alpha V}{L} + \lambda w_i(\mathbf{x}_i)}}. \tag{9}$$

As can be seen, the final updated target is a weighted average of the result of regular voting (Eq. 7) and the average of all the current texture maps (Eq. 8). This means that the consistency term basically enforces our updated targets to remain close to the current textures.

Single iteration Multiple iterations

Fig. 7. We show that a single iteration of search and vote produces results that are very similar to those with multiple iterations.

This energy function is minimized by iteratively performing the search and vote process until convergence. These iterations work by using the updated targets after voting as the input to the search process in the next iteration. We empirically found that only one iteration of search and vote is sufficient to obtain high-quality results, as shown in Fig. 7.

2) Reconstruction. In this step, we fix $T_1, \cdots, T_N$ and produce optimum texture at different views, $M_1, \cdots, M_N$, to minimize Eq. 5. Since the textures only appear in the second term (E_C), which is quadratic, the optimal textures can be easily obtained as follows:

$$\boxed{M_i(\mathbf{x}_i) = \frac{\sum_{j=1}^{N} w_j(\mathbf{x}_{i\to j}) T_j(\mathbf{x}_{i\to j})}{\sum_{j=1}^{N} w_j(\mathbf{x}_{i\to j})}}. \tag{10}$$

This is our texture generation equation which basically states that the optimum texture is obtained by computing a weighted average of all the projected targets. In case the targets are misaligned, which is usually the case at the beginning of the optimization, this process produces textures with ghosting and blurring. The next iteration of the alignment process will then try to reduce the misalignment between the targets, which consequently results in a texture map with fewer artifacts after reconstruction.

We continue this process of alignment and reconstruction iteratively until convergence. As is common with the patch-based approaches [Barnes et al. 2009; Wexler et al. 2007], we perform this process at multiple scales to avoid local minima and speed up the convergence (see Sec. 4). Note that the iterations here are done between our two main stages of alignment and reconstruction. We also have an inner iteration between the search and vote process at every alignment stage. However, as discussed, we found that only one iteration of search and vote is sufficient during alignment.

Once converged, our algorithm produces the aligned images, $T_1, \cdots, T_N$, as well as the optimum texture at different views, $M_1, \cdots, M_N$, which will be very similar. Since our target images are consistent, a single global texture can be obtained by projecting all the target images on the geometry and averaging their color samples to obtain the final color at each vertex.

4 IMPLEMENTATION DETAILS

Capturing input data. We use an Intel RealSense R200 camera to capture our input RGB-D sequences. This camera records depth and color sequences with a resolution of 628×468 and 1920×1080, respectively, both at 30 fps. To minimize the color variations, we use fixed exposure and white balancing. We estimate the geometry and the camera poses of each frame using the KinectFusion algorithm [Izadi et al. 2011]. Note that, this approach estimates the

camera pose of the depth frames and we also assign these estimated camera poses to the corresponding color frames.[4]

Keyframe Selection. To reduce the number of our input images, we select a subset of images with a greedy approach similar to Zhou and Koltun's method [2014]. Specifically, given a set of already selected key frames, we use the method of Crete et al. [2007] to find a frame with the lowest blurriness in the interval of $(t, 2t)$ after the last selected key frame. In our implementation, t varies between 30 to 60 frames depending on the scene.

Alignment. To accelerate the search process, we use the PatchMatch algorithm of Barnes et al. [2009] with the default parameters and patch size of 7. Moreover, to avoid the target images deviating significantly from the source images, we limit the search to a small window of size $0.1\sqrt{w \times h}$, where w and h are the width and height of the source.

Multiscale Optimization. We solve our energy function in Eq. 5 by performing the optimization in multiple scales. Specifically, we start by downsampling all the source images to the coarsest scale. We first initialize the targets, $T_1, \cdots, T_N$, and the textures, $M_1, \cdots, M_N$, with the low resolution source images and perform the alignment and reconstruction stages iteratively until convergence. We then upsample all the targets and textures to the resolution of the next scale and perform our two stages iteratively at this new scale. Note that, instead of upsampling the sources from the coarser scale, we directly downsample the original high resolution source images to the current scale. This allows the system to inject high frequency details into the targets and textures. We continue this process for all the finer scales to obtain the final targets at the finest scale. In the coarsest scale, the input image has 64 pixels in the smaller dimension and we have a total of 10 scales with scaling factor of $\sqrt[9]{x/64}$, where x is the smaller dimension of the original source images. We perform 50 iterations of alignment and reconstruction at the coarsest scale and decrease it by 5 at each finer scale.

As shown in Fig. 8, this multiscale approach is necessary to avoid local minima, and consequently, produce high-quality results. Intuitively, our optimization system aligns the global structures in the coarser scales and recovers the details in the finer scales. A video demonstrating the convergence of our algorithm at multiple scales can be found in the supplementary video.

5 RESULTS

We implemented our framework in MATLAB/C++ and compared against the state-of-the-art approaches by Eisemann et al. [2008], Waechter et al. [2014] and Zhou and Koltun [2014]. We used the authors' code for Waechter et al. and Eisemann et al.'s approaches, but implemented the method of Zhou and Koltun ourself since their source code is not available online. Note that, for Eisemann et al.'s approach, we use the implementation for static scenes and generate view-independent textures to have a fair comparison. We demonstrate the results by showing one or two views of each object, and videos showing the texture mapped objects from different views can be found in the supplementary video. Note that, our scenes are

[4]One may obtain the color camera poses by applying a rigid transformation to the depth camera poses, but this strategy would not significantly help for two reasons: 1) the shutters of the depth and color cameras are not perfectly synchronized, and 2) our depth and color cameras are close to each other, and thus, they have similar poses.

Fig. 8. The energy function in Eq. 5 has a large number of local minima. By minimizing this energy function at the finest scale, there is a significant possibility of getting trapped in one of these local minima. Similar to other patch-based approaches, we perform the optimization at multiple scales to produce high-quality results, as shown on the right.

generally more challenging than Zhou and Koltun's scenes. This is mainly because of the fact that we casually capture our scenes under typical lighting conditions, and thus, our geometries have lower accuracy. We have tested our method on the Fountain scene from Zhou and Koltun's paper and are able to produce comparable results, as shown in Fig. 14 (Aligned Target).

Figure 9 compares our approach against other methods on six challenging objects, and the estimated geometry for these objects is shown in Fig. 10. The Truck is a challenging scene with a complex geometry which cannot be accurately captured with consumer depth cameras. Eisemann et al. [2008] works on a pair of images and corrects misalignments using optical flow without optimizing a global energy function, which is suboptimal. Therefore, their method produces blurry textures as their warped images typically contain residual misalignments. Waechter et al. [2014] select one view per face by solving an optimization system to hide the seams between adjacent faces. However, their method is not able to produce satisfactory results in this case, since they assign inconsistent textures to some of the adjacent faces because of significant inaccuracies. Note the tearing artifacts at the top inset and the distorted bear face at the bottom inset. Moreover, the local warping in Zhou and Koltun's approach [2014] is not able to correct significant misalignment in this case, caused by inaccurate geometry (see Fig. 10). Therefore, their results suffer from ghosting and blurring artifacts. Our method synthesizes aligned target images and is able to produce high-quality texture maps with minimal artifacts.

None of the other approaches are able to handle the Gun scene. Specifically, note that only our approach is able to reconstruct the thin black structure at the bottom inset. Because of inaccuracies in optical flow estimation, Eisemann et al.'s approach produces results with tearing artifacts. It is worth noting that the method of Waechter et al. performs color correction to fix the color variations between adjacent faces. Since in this case the images are significantly misaligned, adjacent faces may have inconsistent textures. Therefore, the color correction introduces discoloration which is visible in the two insets. Next, we examine the House scene, which has a complex geometry. Waechter et al. produce tearing artifacts, while Eisemann et al. and Zhou and Koltun's results demonstrate ghosting artifacts. This is mainly due to the complexity of this scene and the inaccuracy of the geometry (see Fig. 10). On the other hand, our method is able to produce high-quality results on this challenging scene.

The top inset of the Backpack scene shows a region with a fairly smooth geometry. However, Eisemann et al.'s method is still not able to properly align the images and generates blurry textures. Moreover, Waechter et al.'s method generates results with tearing artifacts due

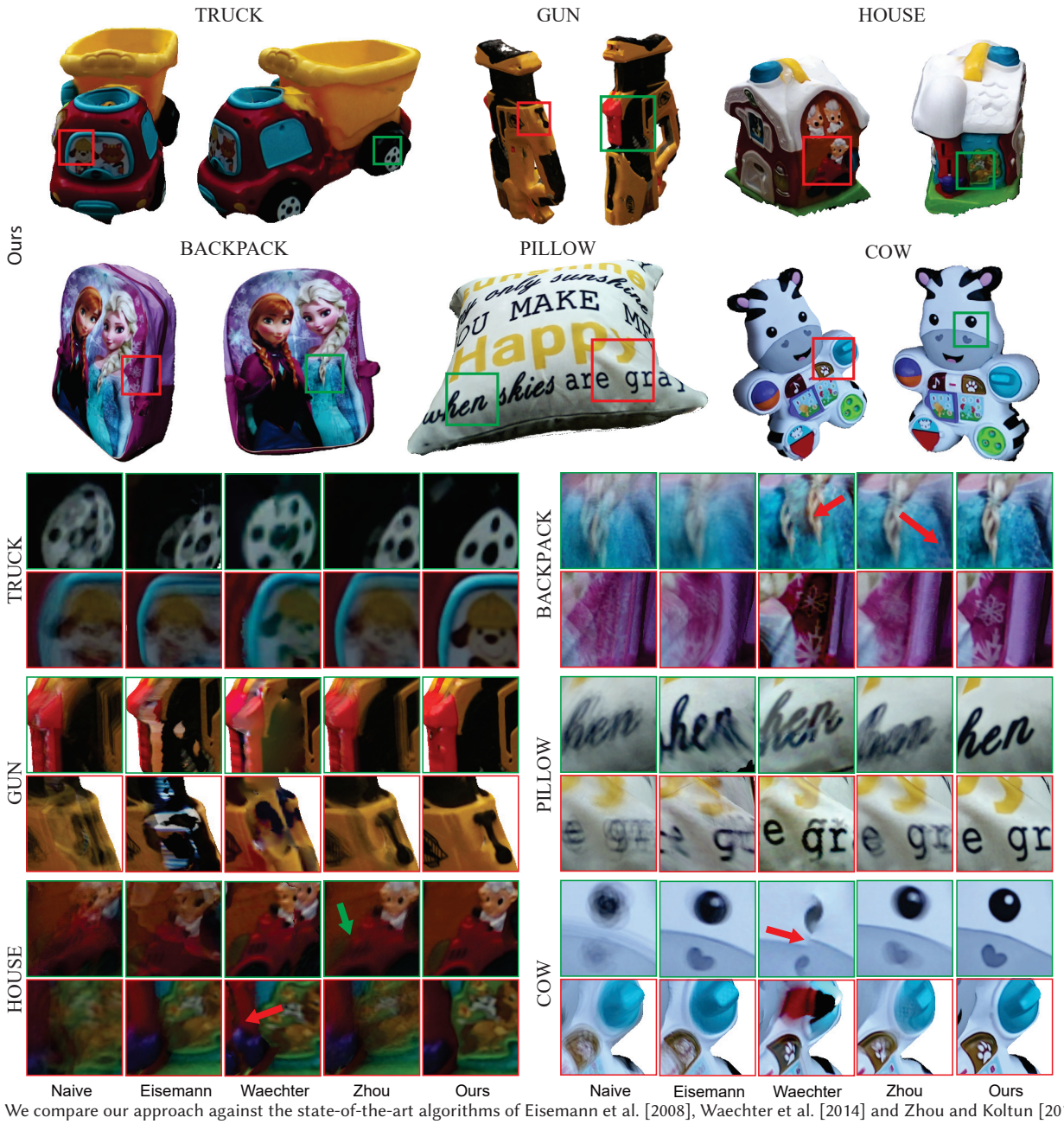

Fig. 9. We compare our approach against the state-of-the-art algorithms of Eisemann et al. [2008], Waechter et al. [2014] and Zhou and Koltun [2014]. We also demonstrate the result of naïvely projecting all the images and averaging them for comparison. Other approaches are not able to handle these challenging scenes and produce results with tearing, discoloration, blurring, and ghosting artifacts. On the other hand, we generate artifact-free high-quality results.

to incorrect camera poses. Although Zhou and Koltun's method corrects most of the misalignments in this case, their result is slightly blurrier than ours. The bottom inset shows a region from the side of the backpack with a complex geometry. In this region, Waechter et al.'s method demonstrates discoloration artifacts, while Zhou and Koltun and Eisemann et al.'s approaches produce results with ghosting artifacts. Similarly, none of the other methods are able to properly reconstruct the textures on the sides of the Pillow, a region with complex geometry. It is worth mentioning that Waechter et al.'s approach also produces discoloration artifacts in the underside of the pillow (see supplementary video). Finally, only our method properly reconstructs the eye and heart at the top inset and the blue and brown structures at the bottom insets of the Cow scene.

We compare our method against other approaches on the Human scene in Fig. 11. This scene is particularly challenging for all the methods since the subject was moving during the capturing process. While all the other approaches produce results with ghosting and blurring artifacts, our method properly handles all the inaccuracies and generates a high-quality texture.

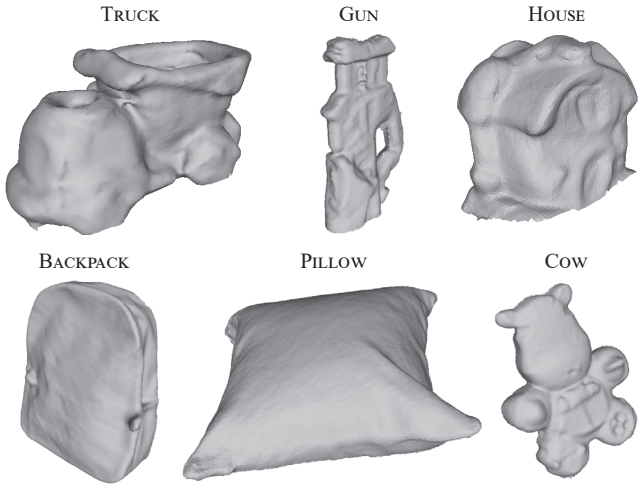

Fig. 10. Estimated geometry for the objects in Fig. 9.

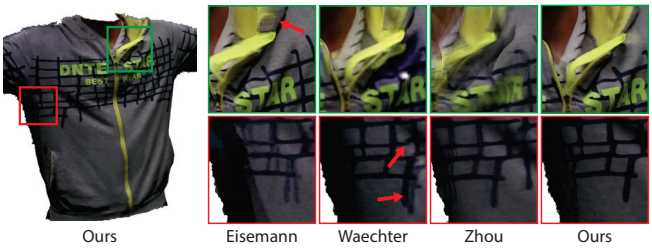

Fig. 11. Comparison against other approaches on a challenging scene.

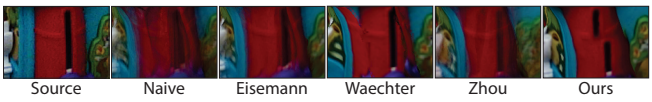

Fig. 12. We show a small inset on the side of the toy house in the House scene (see Fig. 9). The input images are significantly misaligned as can be seen by the blurriness of the Naïve approach. Our method is able to correct the misalignments and produce a plausible result. However, our patch-based approach is not able to always preserve the semantic information. For example, our method produces a result, where the single hole is broken down into two separate pieces. Other approaches are able to produce results with a single hole, but they suffer from tearing and blurring artifacts.

Limitation. The main limitation of our approach is that patch-based synthesis generally produces plausible results, but in some cases is not able to preserve the semantic information, as shown in Fig. 12. Here, although our approach corrects the significant misalignments and produces plausible results, it is unable to preserve the structure of the hole (see the source inset).

6 OTHER APPLICATIONS

In this section, we discuss several applications of our patch-based system including texture hole-filling and reshuffling as well as multiview camouflage. Note that, although patch-based synthesis has been previously used for *image* hole-filling and reshuffling [Barnes et al. 2009; Simakov et al. 2008], these methods are not suitable in our application because of lack of consistency.

6.1 Texture Hole-filling

In some cases, the texture of a real-world object may contain unwanted regions (holes) that we wish to fill in. One example of this

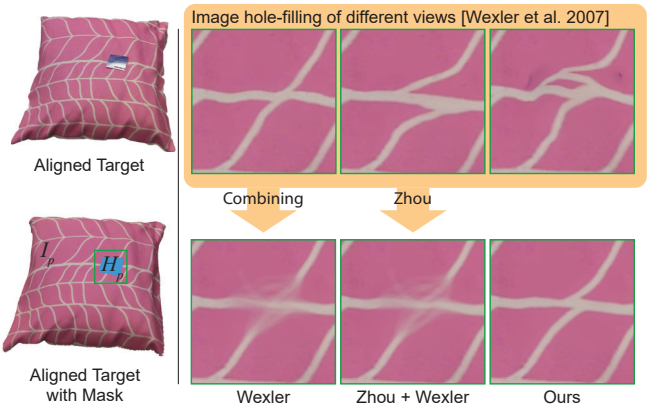

Fig. 13. We first use our system to synthesize aligned target images, one of which shown on the top left. We then mark the unwanted region (the sticker on the pillow) as the hole and project it to all the other views to obtain holes in other targets. The top row shows three views of the hole-filled results using the traditional patch-based synthesis [Wexler et al. 2007] to fill in the hole at each target image independently (we project the results to the same view for better comparison). Although the hole-filled results at each view are plausible, combining them produces texture with ghosting artifacts because of their inconsistencies. Aligning the hole-filled images using the method of Zhou and Koltun [2014] only slightly reduces the blurriness. Our method completes the holes in different targets in a photometrically consistent way, and thus, is able to produce artifact-free results.

case is shown in Fig. 13, where the sticker on the pillow is not desired and should be removed from the final texture map. To do so, we begin by synthesizing aligned target images using our system. We then mark the hole region (shown in blue) in one of the aligned target images. This region can be simply projected to the other views to generate the hole in all the targets. These marked regions basically divide each target image into hole H_i and input I_i (the region outside the hole).

Here, the goal is to fill in the holes, H_i, by drawing information from each input, I_i, while preserving the photometric consistency of the filled holes. This is very similar to the main properties of our energy function in Eq. 5, and thus, our system can be used to perform the hole-filling process. Note that, this problem is related to multi-view hole-filling which has been proposed in a few recent techniques [Baek et al. 2016; Thonat et al. 2016], but we present a way to perform this task using our texture mapping framework.

We do this by setting the sources to the inputs, $S_i = I_i$, and the targets to the holes, $T_i = H_i$ in Eq. 5. In this case, our optimization draws information from the sources (regions outside the holes) to fill in the targets (holes) in a consistent way. This is done by performing the patch search from the regions outside the hole to the holes and voting these patches to reconstruct only the hole regions. For initialization, instead of using the sources, we smoothly fill in the holes from the boundary pixels using MATLAB's roifill function. We also omit the completeness term in the BDS energy term (see Eq. 1) which is responsible for bringing *most* of the information from the source to the target images. Note that, while this is a requirement for alignment, it is not necessary for hole-filling since we only need partial information from the inputs to fill in the holes.

We compare our approach to patch-based image hole-filling [Wexler et al. 2007] in Fig. 13. Although performing the hole-filing separately can produce plausible results at each view

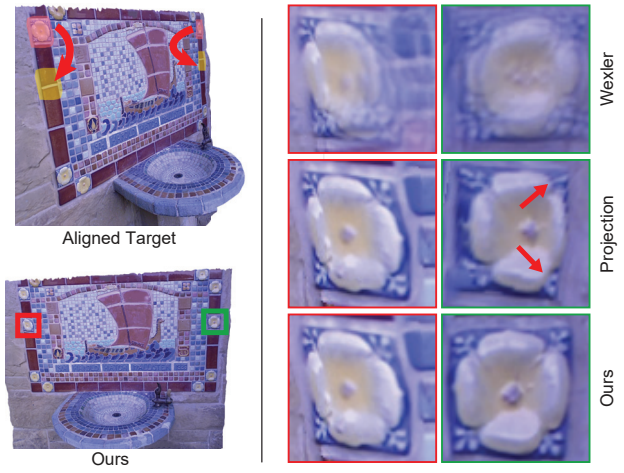

Fig. 14. We show one of the aligned target images on the left. Here, the goal is to plausibly copy the regions in red to the desired locations which are marked with yellow. We first use Simakov et al. [2008] to perform the reshuffling process for this target image. We then project the yellow masks to the other targets and use our hole-filling system to fill in the projected yellow masks. Performing the hole-filling independently for each target image produces inconsistent hole-filled results, which consequently produces textures with ghosting artifacts. Simply projecting the reshuffled result from one target to the other targets has problems at grazing angles. Our method is able to produce consistent results across different views and generate high-quality textures.

(top row), combining them generates a texture with ghosting artifacts (bottom row - left) because of their inconsistency. The method of Zhou and Koltun [2014] can be used to align the hole-filled images at different views (bottom row - middle). However, the final texture still contains ghosting artifacts since the inconsistencies cannot be corrected with warping. Our method is able to produce consistent hole-filled results in different views, and consequently, generate high-quality hole-filled texture.

It is worth noting that we do not hole-fill the geometry. Therefore, our method can only fill in texture holes, if their underlying geometry is not complex, like the one in Fig. 13. Extending our system to also fill in geometries is an interesting topic for future research.

6.2 Texture Reshuffling

As shown in Fig. 14, our method can also be used to copy parts of a texture (marked with red masks) to other regions within the texture (marked with yellow). Again before starting the reshuffling process we synthesized aligned targets using our system. We then mark some regions in one of the target images (reshuffling target) and the goal is to replicate them in a plausible way in the desired locations (yellow masks in Fig. 14). Moreover, the synthesized contents at the new locations of the reshuffling target need to be consistent with all the other target images.

To do this, we first perform the single image reshuffling [Simakov et al. 2008] and synthesize a replica of the regions of interest in the new locations. Note that, this process is performed exactly like Simakov et al. [2008] and only on the reshuffling target. At this point, the other targets are not consistent with this target image in the areas where the reshuffling is performed (yellow regions).

We address this issue, by first projecting the yellow masks to the other targets. We then use our described hole-filling system to fill in the projected yellow masks in other targets. Note that here we do not modify the reshuffling target and it is only used to force the other targets to produce consistent content in the regions defined with the yellow mask. Formally speaking, this means that we remove the E_{BDS} term corresponding to the reshuffling target in Eq. 5.

This process produces targets that are consistent with the reshuffling target, as shown in Fig. 14. Again, the textures produced by hole-filling each target separately using Wexler et al.'s approach [2007] contain ghosting artifacts. Moreover, projecting the content of the yellow mask from the reshuffling target to the other targets produces blurriness. Our method produces high-quality results.

Fig. 15. We show three views of a camouflaged box generated by our approach and Owens et al.'s method[2014]. Comparing to Owens et al.'s technique, we are able to produce a reasonable texture map.

6.3 Multiview Camouflage

Our method could also be used to camouflage a 3D object from multiple viewpoints. Here, the input is a set of images of a scene and the geometry of a 3D object that needs to be artificially inserted into the scene and camouflaged. This is done by producing a consistent texture map for the geometry to make it invisible from different viewpoints. This problem can be viewed as image-based texture mapping for a highly inaccurate geometry, where the geometry of the scene is modeled with the 3D object. We compare the result of our technique for camouflaging a box against Owens et al.'s method [2014] in Fig. 15. Note that, their approach is specifically designed for this application and is limited to camouflaging boxes. Therefore, their approach produces high-quality results in this case. In comparison, our framework is able to handle this additional application and produce reasonable results. Moreover, our method is not limited to boxes and is able to handle any other objects (see supplementary video).

7 CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a novel global patch-based optimization system for image-based texture mapping. We correct the misalignments caused by the inaccuracies in the geometry, camera poses, and optical distortions of the input images, by synthesizing an aligned image for each source image. We propose to do this using a novel patch-based energy function that reconstructs photometrically-consistent aligned images from the source images. To solve our energy function efficiently, we propose a two step approach involving a modified patch search and vote followed by a reconstruction stage. We show that our patch-based approach is

effective in handling large inaccuracies and outperforms state-of-the-art approaches. Moreover, we demonstrate other applications of our system such as texture editing and multiview camouflage.

In the future, it would be interesting to extend our system to directly correct the inaccuracies of the geometry and camera poses, in addition to producing the aligned images. Moreover, we would like to investigate the possibility of using our system for wide baseline view interpolation, where information from a set of images need to be combined to produce consistent novel view images.

APPENDIX

Here, we discuss the derivation of Eq. 8, which computes the targets that minimize the second term of Eq. 5. To start, we rewrite E_2 as:

$$E_2 = \frac{1}{N}\sum_{k=1}^{N}\sum_{\mathbf{x}_k}\sum_{j=1}^{N} w_j(\mathbf{x}_{k\to j})\left(T_j(\mathbf{x}_{k\to j}) - M_k(\mathbf{x}_k)\right)^2. \quad (11)$$

To compute the optimum targets, we first need to differentiate the error with respect to each target as:

$$\frac{\partial E_2}{\partial T_i(\mathbf{x}_i)} = \frac{\partial \sum_{k=1}^{N}\sum_{\mathbf{x}_k} w_i(\mathbf{x}_{k\to i})\big(T_i(\mathbf{x}_{k\to i}) - M_k(\mathbf{x}_k)\big)^2}{\partial T_i(\mathbf{x}_i)}, \quad (12)$$

where we remove the normalization factor, since it does not affect the optimum result. Moreover, since we differentiate with respect to the i^{th} target, we set $j = i$. Here, for each k, the summation is over all pixels of image k. Since we take the derivative with respect to the i^{th} target, we should backproject each term from k to i. By ignoring the effect of interpolation in the projection, we have:

$$\frac{\partial E_2}{\partial T_i(\mathbf{x}_i)} = \frac{\partial \sum_{k=1}^{N}\sum_{\mathbf{x}_i} w_i(\mathbf{x}_i)\big(T_i(\mathbf{x}_i) - M_k(\mathbf{x}_{i\to k})\big)^2}{\partial T_i(\mathbf{x}_i)}, \quad (13)$$

where we used the fact that $\mathbf{x}_{i\to k\to i} = \mathbf{x}_i$. By taking the derivative in the above equation and setting it equal to zero, T_i's can be calculated as defined in Eq. 8. Note that, since the derivative is with respect to a single pixel of the target image $\mathbf{x}_i$, we remove the summation over all pixels before taking the derivative.

ACKNOWLEDGMENTS

We would like to thank Pradeep Sen for valuable discussions. This work was supported in part by ONR grant N000141512013, NSF grants 1451830 and 1617234, and the UC San Diego Center for Visual Computing. Preliminary experiments for this project were funded by NSF grants 1342931 and 1321168.

REFERENCES

Ehsan Aganj, Pascal Monasse, and Renaud Keriven. 2010. Multi-view Texturing of Imprecise Mesh. In *ACCV*. 468–476.

S. H. Baek, I. Choi, and M. H. Kim. 2016. Multiview Image Completion with Space Structure Propagation. In *CVPR*. 488–496.

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM TOG* 28, Article 24 (2009), 11 pages. Issue 3.

Pierre Bénard, Forrester Cole, Michael Kass, Igor Mordatch, James Hegarty, Martin Sebastian Senn, Kurt Fleischer, Davide Pesare, and Katherine Breeden. 2013. Stylizing Animation by Example. *ACM TOG* 32, 4, Article 119 (2013), 12 pages.

Fausto Bernardini, Ioana M. Martin, and Holly Rushmeier. 2001. High-Quality Texture Reconstruction from Multiple Scans. *IEEE TVCG* 7, 4 (2001), 318–332.

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured Lumigraph Rendering. In *SIGGRAPH*. 425–432.

M. Corsini, M. Dellepiane, F. Ganovelli, R. Gherardi, A. Fusiello, and R. Scopigno. 2013. Fully Automatic Registration of Image Sets on Approximate Geometry. *IJCV* 102, 1-3 (2013), 91–111.

Massimiliano Corsini, Matteo Dellepiane, Federico Ponchio, and Roberto Scopigno. 2009. Image-to-Geometry Registration: a Mutual Information Method exploiting Illumination-related Geometric Properties. *CGF* 28, 7 (2009), 1755–1764.

Frederique Crete-Roffet, Thierry Dolmiere, Patricia Ladret, and Marina Nicolas. 2007. The Blur Effect: Perception and Estimation with a New No-Reference Perceptual Blur Metric. In *HVIE*.

Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. 1996. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-based Approach. In *SIGGRAPH*. 11–20.

M. Dellepiane, R. Marroquim, M. Callieri, P. Cignoni, and R. Scopigno. 2012. Flow-Based Local Optimization for Image-to-Geometry Projection. *IEEE TVCG* 18, 3 (2012), 463–474.

M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. 2008. Floating Textures. *CGF* 27, 2 (2008), 409–418.

Thomas Franken, Matteo Dellepiane, Fabio Ganovelli, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. 2005. Minimizing user intervention in registering 2D images to 3D models. *The Visual Computer* 21, 8 (2005), 619–628.

Ran Gal, Yonatan Wexler, Eyal Ofek, Hugues Hoppe, and Daniel Cohen-Or. 2010. Seamless Montage for Texturing Models. *CGF* 29, 2 (2010), 479–486.

Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable Inside-Out Image-Based Rendering. *ACM TOG* 35, 6 (2016), 231:1–231:11.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. *UIST* (2011), 559–568.

Ondřej Jamriška, Jakub Fišer, Paul Asente, Jingwan Lu, Eli Shechtman, and Daniel Sýkora. 2015. LazyFluids: Appearance Transfer for Fluid Animations. *ACM TOG* 34, 4, Article 92 (2015), 10 pages.

Nima Khademi Kalantari, Eli Shechtman, Connelly Barnes, Soheil Darabi, Dan B Goldman, and Pradeep Sen. 2013. Patch-based High Dynamic Range Video. *ACM TOG* 32, 6 (2013).

V. Lempitsky and D. Ivanov. 2007. Seamless Mosaicing of Image-Based Texture Maps. In *CVPR*. 1–6.

Hendrik P.A. Lensch, Wolfgang Heidrich, and Hans-Peter Seidel. 2001. A Silhouette-Based Algorithm for Texture Registration and Stitching. *Graphical Models* 63, 4 (2001), 245 – 262.

Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time Dense Surface Mapping and Tracking. In *ISMAR*. 127–136.

Eyal Ofek, Erez Shilat, Ari Rappoport, and Michael Werman. 1997. Multiresolution Textures from Image Sequences. *IEEE Computer Graphics and Applications* 17, 2 (1997), 18–29.

A. Owens, C. Barnes, A. Flint, H. Singh, and W. Freeman. 2014. Camouflaging an Object from Many Viewpoints. In *CVPR*. 2782–2789.

Frédéric Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David H. Salesin. 1998. Synthesizing Realistic Facial Expressions from Photographs. In *SIGGRAPH*. 75–84.

Kari Pulli and Linda G. Shapiro. 2000. Surface Reconstruction and Display from Range and Color Data. *Graphical Models* 62, 3 (2000), 165 – 201.

Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. 2006. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *CVPR*. 519–528.

Pradeep Sen, Nima Khademi Kalantari, Maziar Yaesoubi, Soheil Darabi, Dan B Goldman, and Eli Shechtman. 2012. Robust Patch-Based HDR Reconstruction of Dynamic Scenes. *ACM TOG* 31, 6 (2012).

Eli Shechtman, Alex Rav-Acha, Michal Irani, and Steve Seitz. 2010. Regenerative Morphing. In *CVPR*. 615–622.

D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. 2008. Summarizing visual data using bidirectional similarity. In *CVPR*. 1–8.

Sudipta N. Sinha, Drew Steedly, Richard Szeliski, Maneesh Agrawala, and Marc Pollefeys. 2008. Interactive 3D Architectural Modeling from Unordered Photo Collections. *ACM TOG* 27, 5, Article 159 (2008), 10 pages.

Ioannis Stamos and Peter K. Allen. 2002. Geometry and Texture Recovery of Scenes of Large Scale. *Computer Vision and Image Understanding* 88, 2 (2002), 94–118.

T. Thonat, E. Shechtman, S. Paris, and G. Drettakis. 2016. Multi-View Inpainting for Image-Based Scene Editing and Rendering. In *IEEE 3DV*. 351–359.

Yochay Tzur and Ayellet Tal. 2009. FlexiStickers: Photogrammetric Texture Mapping Using Casual Images. *ACM TOG* 28, 3, Article 45 (2009), 10 pages.

Luiz Velho and Jonas Sossai Jr. 2007. Projective Texture Atlas Construction for 3D Photography. *The Visual Computer* 23, 9 (2007), 621–629.

Michael Waechter, Nils Moehrle, and Michael Goesele. 2014. Let there be color! Large-scale texturing of 3D reconstructions. In *ECCV*. Springer, 836–850.

Yonatan Wexler, Eli Shechtman, and Michal Irani. 2007. Space-Time Completion of Video. *IEEE PAMI* 29, 3 (2007), 463–476.

Qian-Yi Zhou and Vladlen Koltun. 2014. Color Map Optimization for 3D Reconstruction with Consumer Depth Cameras. *ACM TOG* 33, 4, Article 155 (2014), 10 pages.