Seamless Color Mapping for 3D Reconstruction with Consumer-Grade Scanning Devices

Bin Wang[1](✉), Pan Pan[1], Qinjie Xiao[2], Likang Luo[2], Xiaofeng Ren[1], Rong Jin[1], and Xiaogang Jin[2]

[1] Machine Intelligence Technology Lab, Alibaba Group, Hangzhou, China
{ganfu.wb,panpan.pp,x.ren,jinrong.jr}@alibaba-inc.com

[2] State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China
826464268@qq.com, style_luo@163.com, jin@cad.zju.edu.cn

Abstract. Virtual Reality provides an immersive and intuitive shopping experience for customers. This raises challenging problems of reconstructing real-life products realistically in a cheap way. We present a seamless texturing method for 3D reconstructed objects with inexpensive consumer-grade scanning devices. To this end, we develop a two-step global optimization method to seamlessly texture reconstructed models with color images. We first perform a seam generation optimization based on Markov random field to generate more reasonable seams located at low-frequency color areas. Then, we employ a seam correction optimization that uses local color information around seams to correct the misalignments of images used for texturing. In contrast to previous approaches, the proposed method is more computationally efficient in generating seamless texture maps. Experimental results show that our method can efficiently deliver a seamless and high-quality texture maps even for noisy data.

Keywords: Texture mapping · Markov random field · Seamless color optimization

1 Introduction

Texture mapping plays an important role in reconstructing virtual versions of real-life products for E-Commerce applications with inexpensive consumer-grade scanning devices. This raises challenging problems to reconstruct seamless, high-quality texture maps from noisy data, such as inaccurate geometry, imprecise camera poses, and optical distortions of consumer-grade cameras. Existing methods such as Waechter *et al.* [24] efficiently select suitable images to texture faces

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-11009-3_39) contains supplementary material, which is available to authorized users.

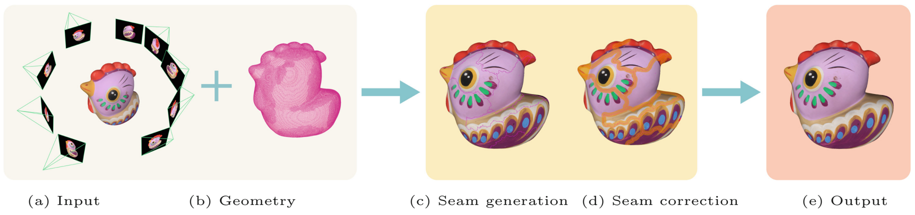

Fig. 1. The pipeline of our texture mapping process: (a) a set of images registered to a corresponding geometry (b) are taken as input, then the seam generation process (c) selects suitable images to texture faces on geometry, which creates most of the seams across low-frequency color areas. Finally, a seam correction optimization (d) corrects misalignments around seams (shown in translucent orange blocks) and generates a high-quality texture mapping output (e) (Color figure online)

on geometric models, but their method may generate visible seams, blurring and ghosting artifacts on the generated texture maps. Recently, Zhou and Koltun [26] use dense, global color information to correct the misalignments of images used for texturing, which produces impressive color maps. However, their approach suffers from large computational consumption. In this paper, we improve seamless texture maps by generating optimal seams with a bypass optimization and correcting the misaligned seams efficiently using local color information.

Our approach achieves both efficiency and seamless texture maps by a two-step global optimization. For the first step, we present a novel optimization based on Markov random field (MRF) that selects suitable images to texture geometric meshes and generates optimal seams located at low-frequency texture regions on texture maps. Our optimization incorporates color discrepancies between the textures of adjacent faces on meshes. As a result, low-frequency texture regions will be more appropriate to create seams, which results in lower energy for MRF-based optimization, and visible seams are thus diminished. As the seams cannot be completely eliminated by the first step, we perform a joint optimization in the second step in order to maximize the color consistency around the seams, which further eliminates the misaligned seams. We estimate camera poses and local warping of images used for texturing geometry. Specially, we only estimate the color consistency of vertices around seams and warp local image patches where seams exist for efficiency.

The contributions of our approach can be summarized as follows. Firstly, we present a seam generation optimization based on MRF to create optimal seams on low-frequency color areas. Then, we propose a seam correction optimization which can efficiently correct misaligned errors. Finally, we present a two-step optimization framework that can efficiently generate seamless texture maps, a main problem of texture mapping. Experimental results demonstrate that our approach can provide a better color representation with much lower computational cost compared to existing methods.

2 Related Works

3D Acquisition. As consumer-grade depth cameras make 3D acquisition more and more affordable and convenient, geometric acquisition using RGB-D is highly anticipated [19,23,25]. The pioneer work of KinectFusion proposed by Izadi *et al.* [11] reconstructs the scene's geometry with a volumetric representation. Nießner *et al.* [20] propose a real-time online 3D reconstruction system using an efficient geometric representation based on hashing. Zhou and Koltun [25] reconstruct dense scenes with points of interest using RGB-D cameras. Another popular method of geometric reconstruction is structure from motion. Ackermann *et al.* [1] propose a photometric stereo technique to reconstruct outdoor scenes. These methods based on structure from motion and RGB-D images are flexible enough to reconstruct geometry ranging from fine-scale objects to large-scale scenes. However, they generate 3D models with much noises [4,10,14]. Accurate geometric reconstruction can be obtained by structured light scanning systems [9,18]. Gupta *et al.* [8] present a structured light system to reconstruct high-quality geometry with global illumination. In this paper, we use data scanned from a low-cost structured light system, which consists of an ordinary projector and a RGB industry camera.

Vertex Texturing Methods. Vertex texturing methods encode color information as per-vertex color. Nießner *et al.* [20] and Shan *et al.* [22] integrate multi-view color samples, which lead to blurring and ghosting artifacts due to the misaligned errors. Zhou and Koltun [26] use dense, global color information to estimate the photometric consistency of all vertices on the object's mesh. Their method corrects misaligned errors and improves texture mapping fidelity. However, in order to describe the high-quality details of objects, their approach estimates the color for lots of vertices on the objects' meshes, which is time-consuming and may lose the advantage of texture mapping that represents high-quality details with low geometric representations.

Face Texturing Methods. Face texturing methods such as [7,15,24] are based on Markov random field, and they select one single image to texture each face on the objects' meshes. These methods can generate texture maps with lots of details. However, these methods cannot perfectly address the misalignments of images resulting in blurring and ghosting artifacts. Lempitsky and Ivanov [15] diminish visible seams by performing a global color adjustment following with Poisson editing [21]. However, blurring and ghosting artifacts around seams may still occur due to noisy input data. Other approaches are proposed to generate seamless texture maps using geometry information. For example, Barnes *et al.* [3] provide an interactive method to manually correct misaligned errors between the geometry model and images, which is not suitable for E-Commerce applications. Bi *et al.* [5] propose a patch-based optimization that incorporates geometry information. Their method estimates the bidirectional similarity of different images, which suffers from high computational cost. Recently, some deep learning-based approaches are developed for real-world texture reconstruction using texture synthesis [16,17].

3 Overview

As shown in Fig. 1, our pipeline takes an object's mesh and a set of images registered to the mesh as input, and generates a high-quality and seamless texture map for the object. Our approach starts with the seam generation process that takes advantage of a novel MRF formulation to select the "best view" texturing per face on the mesh. Existing methods consider "best view" selection as a Graph Cuts optimization [6]. The main idea of our seam generation optimization formulation consists of two energy terms. The first term (data term) selects high-resolution images to texture each face, and the second term (smooth term) provides a smooth representation. The energy function can be solved by a MRF solution [13]. However, traditional methods are not robust for noisy data, resulting in blurring and ghosting artifacts [24]. We redefine the energy function of MRF by employing an easy-to-compute data term and a smooth term by taking advantage of the color differences between adjacent faces, which generates more reliable invisible seams. Details of our seam generation are presented in Sect. 4.1.

To reduce the remaining visible seams that cannot be fully diminished by the seam generation step, we develop a seam correction optimization to deal with misalignments in Sect. 4.2. Inspired by Zhou and Koltun [26] and Bi *et al.* [5], we correct the texture regions around the seams to generate a consistent color 3D representation. Compared to the existing method [24], we design a close-form solution (Fig. 1(d)) to obtain plausible results with a low computational cost. Finally, we use the color adjustment method of Waechter *et al.* [24] to deal with luminance inconsistency caused by variance of lighting on the textured results.

4 Approach

Our approach can texture a 3D object with less perceptible texture seams and higher fidelity. This section details the two key steps. Section 4.1 describes the seam generation optimization step, and Sect. 4.2 describes the seam correction step.

4.1 Seam Generation Optimization

Our seam generation optimization divides the mesh into blocks of faces (as shown in Fig. 2), and the faces in a block corresponds to the same image. The boundaries between blocks are perceived as texture seams in the textured mesh. The input of our method includes an object's triangle mesh and a set of object's images registered to the object mesh. We represent the triangle faces on the mesh as $\boldsymbol{F} = \{F_1, F_2, \cdots, F_m\}$, and the corresponding texture images as $\boldsymbol{I} = \{I_1, I_2, \cdots, I_n\}$, where I_i is the texture image for view i. The projection between images and faces are calculated according to camera intrinsic and extrinsic parameters.

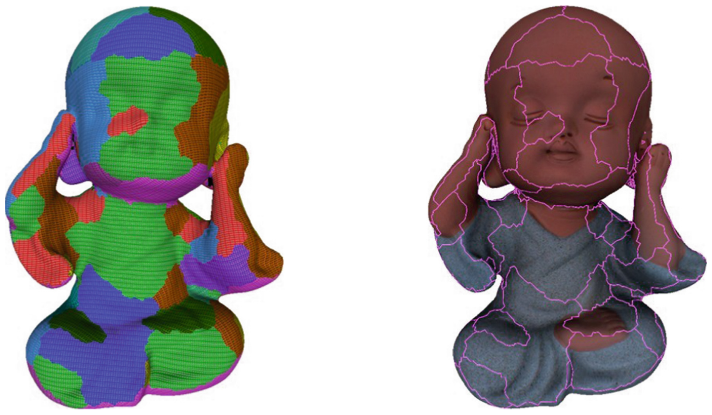
(a) Generated mesh blocks (b) Generated seams

Fig. 2. A mesh labeling example. Each color block in (a) represents a rendering texture to a face, and (b) shows the texturing result where pink contours indicate seams (Color figure online)

The generation of optimized blocks of faces or texture seams is formulated as a labeling problem, and we label each face F_j with a suitable image I_{l_i}. We assume a vector to represent the label relationship $\boldsymbol{L} = \{L_1, L_2, \cdots, L_j, \cdots, L_m\} \in \{1, 2, \cdots, i, \cdots, n\}^m$, and $L_j = i$ indicates that image I_i is used for texturing face F_j. As multiple images may correspond to one face, we should select the best candidate image view for each face. Here, we adopt MRF to solve this problem, and a common MRF energy function can be defined as:

$$E(\boldsymbol{L}) = E_d(\boldsymbol{L}) + \alpha \cdot E_s(\boldsymbol{L}). \quad (1)$$

The first energy term, namely the data term $E_d(\boldsymbol{L})$ represents the cost of texturing faces with texture images from a certain selection of views. The second term $E_s(\boldsymbol{L})$ represents the energy measuring the smooth level of the generated texture. α is a parameter to adjust the weight. In this paper, we propose a novel data term and a smooth term to effectively generate more accurate seams at low-frequency color areas, which cannot be solved by existing MRF techniques.

We aim to reconstruct textures of objects with an ordinary size, and there are no scaling issues in our image data. Thus, it's not necessary to consider scaling in the data term as in Waechter *et al.* [24] which is time-consuming. Besides, Allene *et al.* [2] utilize a projected size as the data term, which is easy to calculate, but they cannot deal with blurring artifacts. Inspired by the method in [15], the metric employed by our data term is to measure the angle between the normal of a face and the camera view direction of an image. This metric accelerates our algorithm since it is computationally efficient. For face $F_j \in \boldsymbol{F}$, we use $f(L_j)$ to evaluate the texturing quality of F_j with view image $I_{L_j} \in \boldsymbol{I}$ $(L_j = i)$. If we can observe F_j from image I_{L_j}, we have $f(L_j) = 1 - (\boldsymbol{n}_j \cdot \boldsymbol{n}_{L_j})^2$; otherwise,

$f(L_j) = \infty$, where $\boldsymbol{n}_j$ is the normal vector of face F_j and $\boldsymbol{n}_{L_j}$ is the unit camera view vector of image I_{L_j}. We then have our data term $E_d(\boldsymbol{L})$ as follows:

$$E_d(\boldsymbol{L}) = \sum_{j=1}^{j=m} f(L_j). \tag{2}$$

For the smooth term, we minimize the average color difference between co-edged faces along the texture seams. We suppose that $e_{jk} \in \boldsymbol{E_{dge}}$ is the edge shared by face F_j and face F_k, $\boldsymbol{E_{dge}}$ is the set of edges shared by adjacent faces of the mesh. Since faces have greater color differences than edges, the average color of faces can better express discrepancies than method [15], which utilizes color discrepancies on the edges. Let $\boldsymbol{C}_{L_j}$ be the average color of pixels on the area where F_j is projected onto an image I_{L_j}, we use the following function to measure the cost of edge e_{jk}:

$$D(L_j, L_k) = \begin{cases} 0, & \text{if } L_j = L_k \\ d(\boldsymbol{C}_{L_j}, \boldsymbol{C}_{L_k})^2, & \text{otherwise,} \end{cases} \tag{3}$$

where $d(\cdot, \cdot)$ is the Euclidean distance on RGB color space. Thus, we have the smooth term $E_s(\boldsymbol{L})$:

$$E_s(\boldsymbol{L}) = \alpha \cdot \sum_{e_{jk} \in \boldsymbol{E_{dge}}} D(L_j, L_k). \tag{4}$$

The overall seam generation energy in Eq. (1) can be re-written as:

$$E(\boldsymbol{L}) = \sum_{j=1}^{j=m} f(L_j) + \alpha \cdot \sum_{e_{jk} \in \boldsymbol{E_{dges}}} D(L_j, L_k). \tag{5}$$

The energy function in Eq. (5) can be formulated as a probability distribution problem with Markov random field, which can be efficiently solved by the α-expansion Graph Cuts [6].

4.2 Seam Correction Optimization

Seam generation optimization produces reasonable seams that are less perceptible. There are still some noticeable seams as shown in Fig. 4 in the reconstructed model due to large misalignments of images. Seam correction optimization is designed to correct such seams by adjusting the content of selected images used for textures. Different from other global optimization methods such as Zhou and Koltun [26] that estimate colors for all vertices, our approach is more efficient since our optimization is only performed for a small set of vertices around seams.

When performing seam correction optimization, we take the following two factors into consideration. First, the color along seams should be consistent. Second, the textured appearance of the area around seams should be similar to the corresponding area in the selected images. To this end, we estimate the colors

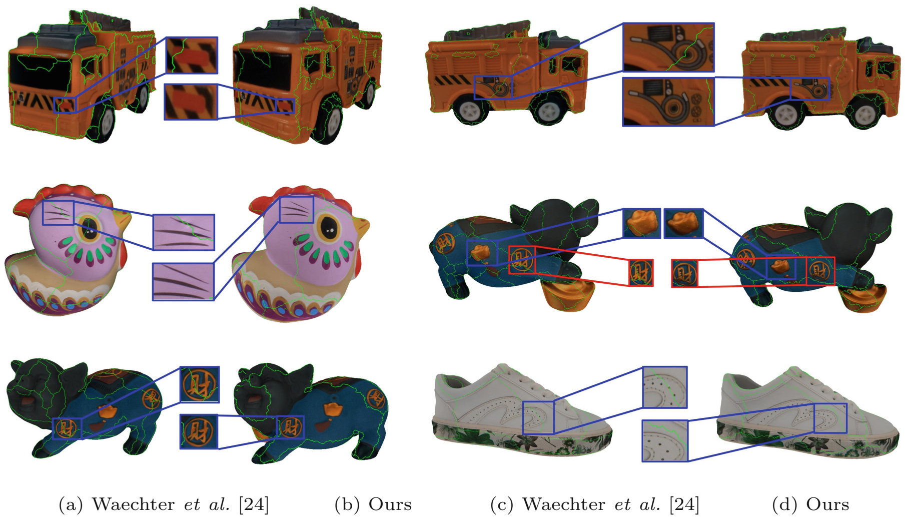

Fig. 3. The seam generation results of our method are tested on several datasets. We compare our results with Waechter *et al.* [24]. Our method outperforms the state-of-the-art view selection methods

for all vertices within the ranges on geometry where seams exist, as well as the camera poses for all images used for texturing geometry and the local warping of images patches that contain the seams to maximize the color similarity of the mapping around seams.

After seam generation, we divide the mesh into different blocks textured with some selected images. We represent the image patch corresponding to each mesh block respectively as $\boldsymbol{P} = \{\boldsymbol{P}_1, \boldsymbol{P}_2, \cdots, \boldsymbol{P}_k\}$. One image can be used to texture multiple blocks (e.g. $\{\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3\} \in I_i$). We define the mesh blocks as $\boldsymbol{B} = \{\boldsymbol{B}_1, \boldsymbol{B}_2, \cdots, \boldsymbol{B}_k\}$. The relationship between an image patch and its corresponding mesh block is a perspective transformation calculated as $\boldsymbol{P}_k = \boldsymbol{\mathcal{K}} \boldsymbol{T}_k \boldsymbol{B}_k$ ($\boldsymbol{T} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, \cdots, \boldsymbol{T}_k\}$ denotes the external camera parameters, and $\boldsymbol{\mathcal{K}}$ denotes the internal camera parameters). $\boldsymbol{E} = \{E_1, E_2, \cdots, E_k\}$ denotes the edges set for each mesh block B_k. $\boldsymbol{F} = \{\boldsymbol{F}_1, \boldsymbol{F}_2, \cdots, \boldsymbol{F}_k\}$ denotes the control lattices for each image patch $\boldsymbol{P}_k$, which is used to warp image patch $\boldsymbol{P}_k$. We choose a vertex $\boldsymbol{v} \in \boldsymbol{B}_k$ as a candidate vertex for optimization, and the shortest geodetic distance from vertex $\boldsymbol{v}$ to edge set E_k is defined as $g(\boldsymbol{v}, E_k)$. We define a proper control range in each image patch for correction, in which only the vertex $\boldsymbol{v}$ with $g(\boldsymbol{v}, E_k)$ less than γ is used. The objective function can then be described as:

$$E_{correction}(\boldsymbol{C}, \boldsymbol{F}, \boldsymbol{T}) = \sum_k \Big(\sum_{\substack{\boldsymbol{v} \in \boldsymbol{B}_k, \\ g(\boldsymbol{v}, E_k) < \gamma}} w(\boldsymbol{v}) \cdot e^2 + \beta \cdot \boldsymbol{F}_k^\top \boldsymbol{F}_k \Big), \tag{6}$$

where $\boldsymbol{C} = \{C_{\boldsymbol{v}} | g(\boldsymbol{v}, E_k) < \gamma\}$ denotes the set of gray-scale color estimated for vertices around seams on the mesh, $\boldsymbol{F}$ denotes the pixel color corrections and $\boldsymbol{T}$

denotes the camera pose transformations. During optimization, the variables are optimized to correct visible seams. The second term is a regular term penalizing image patches from excessive deformation $\boldsymbol{F}$. $w(\boldsymbol{v})$ is the weight of vertex $\boldsymbol{v}$ representing the color discrepancies of the faces around vertex $\boldsymbol{v}$:

$$w(\boldsymbol{v}) = \frac{1}{|\boldsymbol{\mathcal{E}}(\boldsymbol{v})|} \sum_{e_{jk} \in \boldsymbol{\mathcal{E}}(\boldsymbol{v})} d(C_{L_j}, C_{L_k})^2, \tag{7}$$

where $\boldsymbol{\mathcal{E}}(\boldsymbol{v})$ is the set of edges that connect with vertex $\boldsymbol{v}$. Since we texture each face on geometry in the seam generation step, $w(\boldsymbol{v})$ can be precomputed and is constant in the seam correction step, which gives more weight to the vertex located at high-frequency color areas around seams.

The first term of the objective function estimates the color similarity for each vertex in $\{\boldsymbol{v} | g(\boldsymbol{v}, E_k) < \gamma\}$, e^2 is the residual value, and is described as:

$$e = C_{\boldsymbol{v}} - \mathcal{L} \circ \boldsymbol{\mathcal{F}}_k(\boldsymbol{\mathcal{K}} \cdot \boldsymbol{T}_k \cdot \boldsymbol{v}). \tag{8}$$

For each vertex $\boldsymbol{v}$, we project $\boldsymbol{v}$ into the image plane of $\boldsymbol{P}_k$ according to camera intrinsic parameter $\boldsymbol{\mathcal{K}}$ and camera poses $\boldsymbol{T}_k$ (we assume that $\boldsymbol{u} = \boldsymbol{\mathcal{K}} \cdot \boldsymbol{T}_k \cdot \boldsymbol{v}$), and then warp image patch $\boldsymbol{P}_k$ according to the following color correction function:

$$\boldsymbol{\mathcal{F}}_k(\boldsymbol{u}) = \boldsymbol{u} + \boldsymbol{B}(\boldsymbol{u}) \cdot \boldsymbol{F}_k, \tag{9}$$

where $\boldsymbol{B}(\boldsymbol{u})$ is a vector of B-spline base function which controls the vector of control lattices. To this end, we use $\mathcal{L}(\boldsymbol{u})$ to evaluate the gray-scale intensity of the projective point of vertex $\boldsymbol{v}$.

We solve Eq. (6) using an alternating optimization approach to optimize the image correction inspired by Zhou *et al.* [27]. We first fix $\boldsymbol{F}$ and $\boldsymbol{T}$ to optimize $\boldsymbol{C}$, then fix $\boldsymbol{C}$ to optimize $\boldsymbol{F}$ and $\boldsymbol{T}$, and vice versa. When $\boldsymbol{F}$ and $\boldsymbol{T}$ are fixed, Eq. (6) degenerates to a least-square optimization problem, and we use the following equation to calculate the average gray-scale value of vertex $\boldsymbol{v}$:

$$C_{\boldsymbol{v}} = \frac{1}{|\boldsymbol{\mathcal{I}}_{\boldsymbol{v}}|} \sum_{\substack{k, \\ \boldsymbol{P}_k \in \boldsymbol{\mathcal{I}}_{\boldsymbol{v}}}} \mathcal{L} \circ \boldsymbol{\mathcal{F}}_k(\boldsymbol{\mathcal{K}} \cdot \boldsymbol{T}_k \cdot \boldsymbol{v}), \tag{10}$$

where $\boldsymbol{\mathcal{I}}_{\boldsymbol{v}}$ is the set of image patches that can observe $\boldsymbol{v}$.

When $\boldsymbol{C}$ is fixed, we perform an inner iterative strategy to solve $\boldsymbol{F}$ and $\boldsymbol{T}$. We fix $\boldsymbol{F}$ and assume that we perform little rotation for each image. With this assumption, we can solve the camera pose as a linear system. By approximating the external camera matrix as a 6-vector $\boldsymbol{T}_k = \{\alpha_k, \beta_k, \lambda_k, a_k, b_k, c_k\}$, we can independently solve a linear system with 6 parameters for each image patch $\boldsymbol{P}_k$.

After that, we fix $\boldsymbol{C}$ and $\boldsymbol{T}$ to optimize $\boldsymbol{F}$. Then $\boldsymbol{u} = \boldsymbol{\mathcal{K}} \cdot \boldsymbol{T}_k \cdot \boldsymbol{v}$ is a constant, and $\boldsymbol{\mathcal{F}}_k(\boldsymbol{u})$ is a linear combination of $\boldsymbol{F}_k$, and we have:

$$E_{correction}(\boldsymbol{F}) = \sum_k \Big(\sum_{\substack{\boldsymbol{v} \in \boldsymbol{B}_k, \\ g(\boldsymbol{v}, E_k) < \gamma}} w(\boldsymbol{v}) \cdot (C_{\boldsymbol{v}} - \mathcal{L} \circ \boldsymbol{\mathcal{F}}_k(\boldsymbol{F}_k))^2 + \beta \cdot \boldsymbol{F}_k^{\top} \boldsymbol{F}_k \Big). \tag{11}$$

Equation (11) is a least-square system and can be efficiently solved. We continue the alternating optimization iteratively until it converges.

5 Results

We evaluated the performance of our proposed method using our test datasets. All experiments were performed on a commodity workstation with an Intel i5 3.2 GHz CPU and 8 GB of RAM. We first presented details about the test data, and then evaluated the seam generation process and seam correction process. Finally, we compare our method to the state-of-the-art approaches.

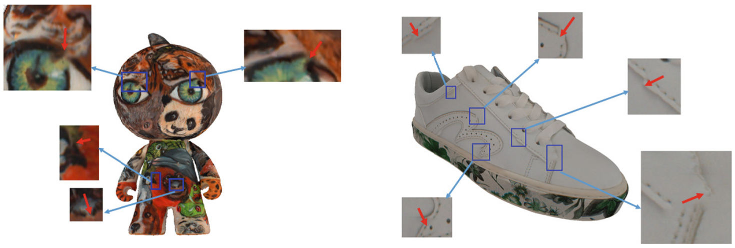

Fig. 4. Some noticeable seams still remain in the textured object due to the misalignment of texture images that cannot be completely avoided by seam generation optimizations

Test Datasets. Our datasets were captured from real-life products (shoes, arts, crafts, etc.) and were reconstructed by the following steps. For each object, we first used a consumer-level structured light 3D scanner to generate a registered point cloud. The scanner contained an RGB industry camera and a normal projector, which was inexpensive. Calibration was performed before the scanning procedure. Then we meshed the point cloud by surface reconstruction [12]. Since the calibrated parameters changed slightly due to the heat transfer in the environment (especially for the parameters of projector), the reconstructed 3D geometry, camera poses and images suffered from noises. Besides, geometric errors would also be introduced by the point cloud registration step. In general, the error of the geometry model was about 3 mm to 5 mm, and the texture reprojection error was about 5 pixels to 15 pixels (the resolution of the captured images was 3456×2304).

Seam Generation Evaluation. We first evaluated the contribution of the weight α in Eq. (1). As discussed in Sect. 4.1, the weight α kept a balance between the data term and the smooth term. As shown in Fig. 6, we found that seams will bypass most of the high-frequency color areas on texture images if we set a larger α. Since a larger weight of $E_s(\boldsymbol{L})$ might result in a larger value of $E_d(\boldsymbol{L})$, the resolution of texture might be reduced. Hence, we need to find an appropriate α value which will not increase $E_d(\boldsymbol{L})$ significantly. We estimated the incremental percentage of $E_d(\boldsymbol{L})$ for different values of α:

$$\Delta E_{d,\alpha_i}(\boldsymbol{L}) = \frac{E_{d,\alpha_i}(\boldsymbol{L}) - E_{d,\alpha_{i-1}}(\boldsymbol{L})}{E_{d,\alpha_{i-1}}(\boldsymbol{L})}(i \geqslant 2). \tag{12}$$

We set different α values (from 50 to 300) marked as α_i, and tested its influence on $E_d(\boldsymbol{L})$. Estimated data were shown in Table 1 and Fig. 5, when $\alpha \leqslant 200$, the average incremental percentage of $E_d(\boldsymbol{L})$ was about 5% and increased to about 15% when $\alpha > 200$. It meant that if we set α too large for the seam generation optimization, the optimization will select images with larger intersection angles between the view direction and the face normal to texture geometric meshes, which decreased the resolution of the texture. Thus, we set $\alpha = 200$ as a trade-off to balance the seam visibility and the resolution of texture.

Since [7,15,24] used similar ideas dealing with seams considering color differences, labels of vertices or edges for faces along seams, we compared our seam generation strategy to the method of Waechter *et al.* [24]. The method of Waechter *et al.* [24] integrated the colors of image patches projected to a face as the data term energy, which favored close-up views. This approach is suitable for large-scale models. Different from their method, we used the angle between the face normal and the camera view direction for the data term. In addition, the smooth term in [24] is based on the Potts model, while our smooth term was based on the color difference between faces adjacent to seams. As shown in Figs. 3 and 8, our method can generate more reasonable seams.

We also compared the computational cost of the MRF-based optimization between Waechter *et al.*' method and our approach. As the authors described in [24], their main computational bottleneck relied on the data term, while the main computational cost of our MRF-based optimization relied on the smooth term, which calculates the average color of each face. Theoretically, the computation of average color was cheaper to calculate than the computation of Waechter *et al.*'

Table 1. Some experimental examples of α in Eq. (1). For our datasets, we set $\alpha = 200$ as a trade-off to balance the seam effect and the resolution of texture

α_i value	$E_d(\boldsymbol{L})$ for different test data					
	car	chicken	pig	doll1	shoes1	doll2
50	47,612	29,041	55,178	29,194	100,692	68,859
100	47,633	30,582	56,603	29,512	101,789	69,503
150	48,310	32,477	57,420	30,368	105,036	71,893
200	**48,826**	**34,698**	**59,387**	**32,152**	**108,605**	**74,337**
250	55,781	38,424	69,164	34,896	157,398	80,641
300	65,812	52,109	82,636	48,177	212,607	89,603

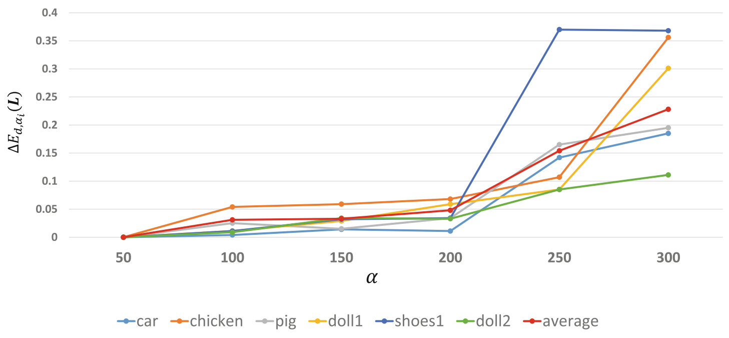

Fig. 5. The incremental percentages of $E_d(\boldsymbol{L})$

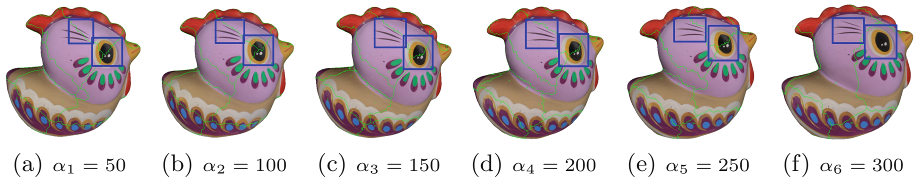

Fig. 6. The seam generation results. The seam generation scheme can bypass more high-frequency color areas as α increases

data term, which needs to compute the projected size and integrate pixel colors. In Table 2, we compared the MRF computational cost between our method and Waechter *et al.*' method quantitatively. The listed statistics conform to our theoretical analysis. Our computational advantage is more obvious when the data size grows large.

Table 2. The comparison of computational cost between our seam generation and Waechter *et al.* [24] for datasets with different sizes (Ours/Waechter *et al.*' method). Our approach is more computationally efficient than Waechter *et al.* [24] ("k", "m" represent thousand and million, respectively)

View number	Computational cost for different numbers of views and faces					
	50k	0.1m	0.2m	0.5m	1m	2m
8	6.2s/6.8s	7.2s/9.6s	8.7s/12.2s	12.5s/15.4s	14.5s/19.2s	23.9s/35.2s
12	8.9s/10.7s	10.5s/13.9s	13.8s/16.1s	18.5s/23.9s	21.7s/28.4s	37.8s/53.1s
24	19.6s/21.3s	22.4s/29.7s	27.2s/37.4s	38.1s/49.5s	44.6s/58.5s	75.2s/108.2s
32	25.1s/27.6s	29.1s/38.7s	35.2s/50.3s	51.4s/63.6s	58.3s/78.3s	96.7s/143.8s
48	41.4s/46.8s	48.2s/61.1s	51.6s/70.4s	73.6s/98.5s	87.3s/114.4s	152.8s/225.1s

Seam Correction Evaluation. We first compared our results to the approach of Zhou and Koltun [26]. Their approach used images of all views to optimize color for each vertex. However, if some images were blurred, it generated blurring artifacts. Different from them, our approach selected the best images to texture faces on meshes. As a result, our method can avoid blurring effects effectively. Moreover, since we used high-resolution images to texture faces, we can generate high-quality texture maps even for low-resolution meshes. Results shown in Fig. 7 indicate that our approach can generate better results for blurring cases.

To evaluate the optimization performance quantitatively, we defined a normalized residual error by dividing it with the number of vertices used for optimization, and was described as:

$$RE_{normalized} = \frac{\sum_{i=1}^{k} \sum e^2}{|\boldsymbol{v}|}. \tag{13}$$

In this way, the residual errors of ours and Zhou and Koltun' method [26] were comparable. We have shown the $RE_{normalized}$ results in Table 3.

From Table 3, we can find that our method converges faster than the method by Zhou and Koltun [26] (see the column named "Time per iter." and "# of iter."). Moreover, the computational cost of our approach outperforms Zhou and Koltun' method [26], especially for high-resolution meshes (see the column named "Total time" in Table 3). This can be explained as follows. Zhou and Koltun [26] utilized all vertices for optimization. Different from them, we only utilized related pixels around the seams for color optimization, and our computational cost was related to the number of edges of seams instead of the number of vertices.

Texture Maps Evaluation. Finally, we compared our final results to the approaches of Shan *et al.* [22], Waechter *et al.* [24] and Zhou and Koltun [26] qualitatively. For a fair comparison, all methods shared the same inputs. The results were shown in Fig. 8. Both [22] and [26] produced color for all vertices only. Shan *et al.* [22] blended color for vertices from views, resulting in blurry and ghosting artifacts shown in Fig. 8(a) and (e). Zhou and Koltun [26] generated

Table 3. Normalized residual error and average time cost per iteration. Our optimization converges faster and has a lower computational cost in each iteration

Model	Method/# of vertices	Normalized residual error					Time per iter.	# of iter.	Total time
		Initial	50 iter.	100 iter.	200 iter.	final			
Fig. 7(a)	Zhou *et al.*/50k	0.061	0.043	0.031	0.022	0.018	0.062s	302	18.8s
	Ours/50k	0.061	0.033	0.021	0.020	0.018	0.042s	276	11.6s
Fig. 7(b)	Zhou *et al.*/0.2m	0.065	0.047	0.038	0.028	0.019	0.180s	353	63.5s
	Ours/0.2m	0.065	0.039	0.029	0.020	0.019	0.046s	298	13.7s
Fig. 7(d)	Zhou *et al.*/1m	0.063	0.051	0.045	0.037	0.018	0.420s	868	364.5s
	Ours/1m	0.063	0.038	0.027	0.021	0.018	0.058s	325	18.85s

better results in Fig. 8(b) and (f), but their performance was limited by the number of vertices. Waechter *et al.* [24] performed texturing per face on mesh with a single image, their approach generated obvious seams because of noises (shown in Fig. 8(c) and (g)). With our two-step optimization, our approach was able to produce visually seamless texture maps (see Fig. 8(d) and (h)). The comparison results show that our approach can substantially improve texture mapping.

6 Conclusions

It is a challenging problem to reconstruct virtual versions of real-life products realistically with inexpensive consumer-grade scanning devices. We have presented a two-step optimization solution for seamless texture mapping with noisy data. The seams are generated from imperceptible texture regions, and the seam misalignments are corrected by the color consistency strategy. We evaluate our approach on a number of objects. Experimental results have shown that our method can efficiently generate visually seamless high-fidelity texture maps with realistic appearance at a low cost. More experimental results are shown in the supplementary video.

It is worth noting that our approach uses a small set of data around seams to correct misalignments, and thus may not be able to correct large noisy data. We mainly focus on indoor objects, and the occlusion problem is not yet addressed. We plan to extend our approach to data with even larger noises in our future work.

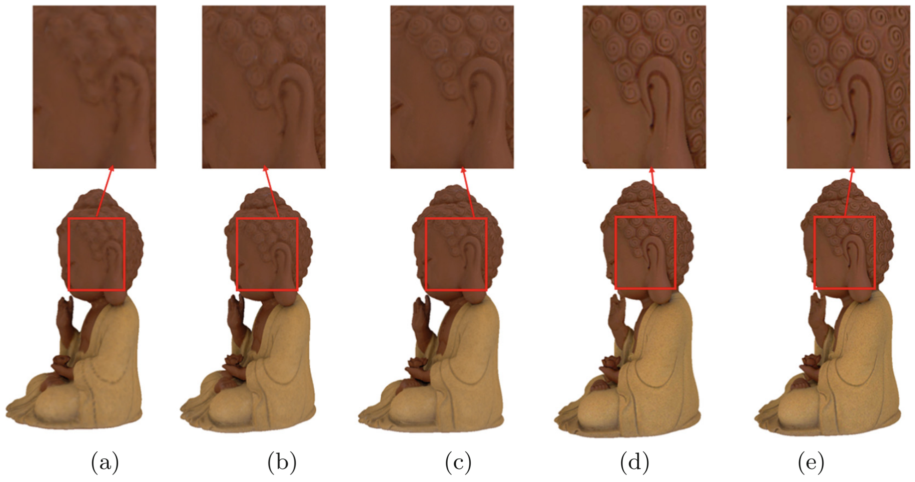

Fig. 7. (a)–(d) are the results of Zhou and Koltun' method by rendering meshes with vertex color. From left to right, the number of vertices of the models are 0.05 million, 0.2 million, 0.5 million, and 1 million, respectively. (e) is our result with 0.05 million vertices. We reconstruct high-quality texture maps with low geometric complexity

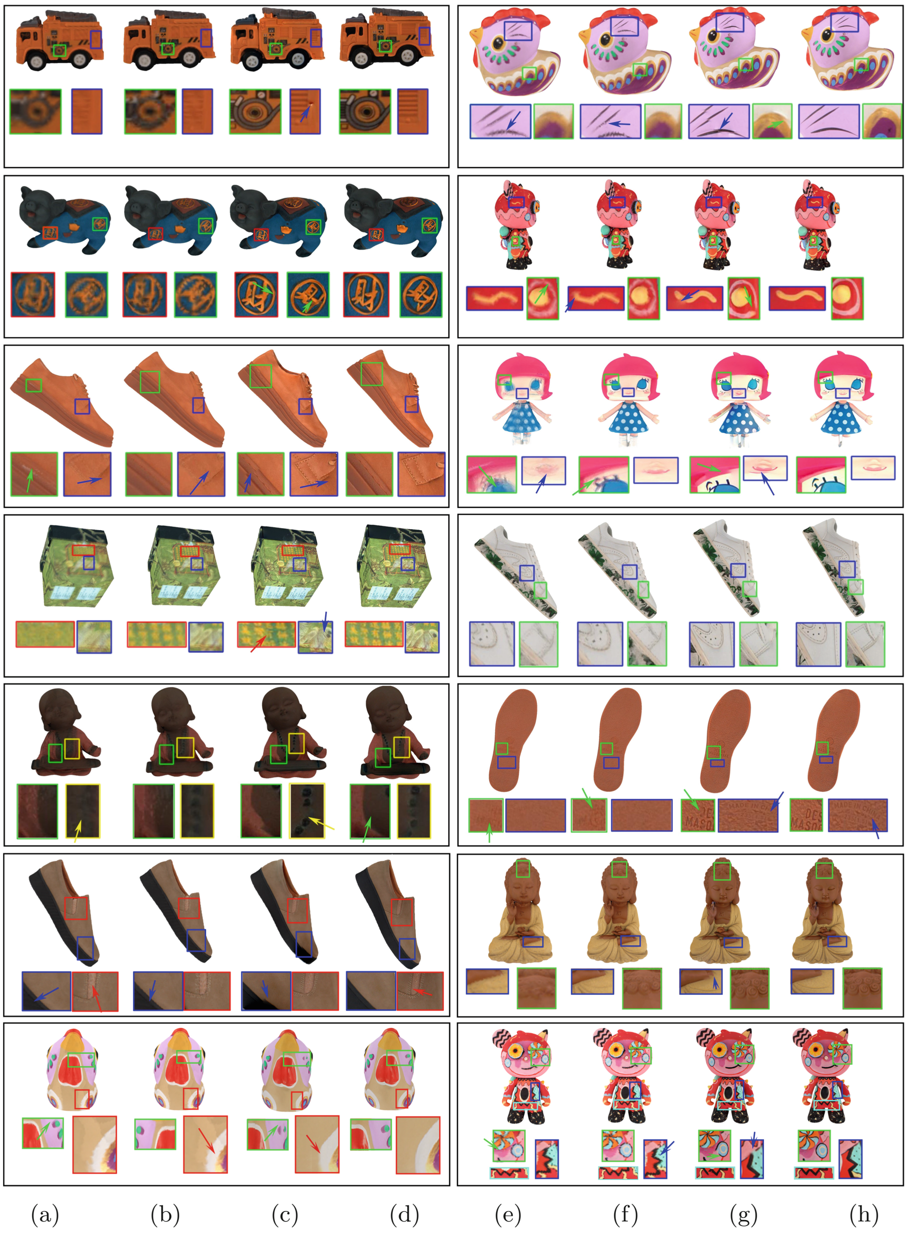

Fig. 8. We compare our method to the state-of-the-art methods with the same inputs. Shan *et al.* [22] (a, e) generate blurring and ghosting artifacts for noise data. Zhou and Koltun [26] perform better, but their results suffer from low-resolution geometry (b, f). Waechter *et al.* [24] select an image texturing per face by penalizing a global optimization, resulting in visible seams for noisy data (c, g). Our method can produce realistic and high-fidelity texture maps (d, h)

Acknowledgments. Xiaogang Jin is supported by the National Key R&D Program of China (No. 2017YFB1002600).

References

1. Ackermann, J., Langguth, F., Fuhrmann, S., Goesele, M.: Photometric stereo for outdoor webcams. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
2. Allene, C., Pons, J.P., Keriven, R.: Seamless image-based texture atlases using multi-band blending. In: 19th International Conference on Pattern Recognition (ICPR) (2008)
3. Barnes, C., Goldman, D.B., Shechtman, E., Finkelstein, A.: The patchmatch randomized matching algorithm for image manipulation. Commun. ACM **54**(11), 103–110 (2001)
4. Bernardini, F., Martin, I.M., Rushmeier, H.: High-quality texture reconstruction from multiple scans. IEEE Trans. Vis. Comput. Graph. **7**(4), 318–332 (2001)
5. Bi, S., Kalantari, N.K., Ramamoorthi, R.: Patch-based optimization for image-based texture mapping. ACM Trans. Graph. **36**(4), 106:1–106:11 (2017)
6. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
7. Gal, R., Wexler, Y., Ofek, E., Hoppe, H., Cohen-Or, D.: Seamless montage for texturing models. In: Computer Graphics Forum, pp. 479–486 (2010)
8. Gupta, M., Agrawal, A., Veeraraghavan, A., Narasimhan, S.G.: Structured light 3D scanning in the presence of global illumination. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2011)
9. Gupta, M., Nayar, S.K.: Micro phase shifting. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
10. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: using depth cameras for dense 3D modeling of indoor environments. In: Khatib, O., Kumar, V., Sukhatme, G. (eds.) Experimental Robotics, pp. 477–491. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-28572-1_33
11. Izadi, S., et al.: KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST) (2011)
12. Kazhdan, M., Hoppe, H.: Screened poisson surface reconstruction. ACM Trans. Graph. **32**(3), 29:1–29:13 (2013)
13. Kindermann, R., Snell, J.L.: Markov Random Fields and Their Applications, vol. 1. American Mathematical Society, Providence (1980)
14. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view RGB-D object dataset. In: IEEE International Conference on Robotics and Automation (ICRA) (2011)
15. Lempitsky, V., Ivanov, D.: Seamless mosaicing of image-based texture maps. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2007)
16. Li, C., Wand, M.: Combining Markov random fields and convolutional neural networks for image synthesis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
17. Li, C., Wand, M.: Precomputed real-time texture synthesis with markovian generative adversarial networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 702–716. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_43

18. Moreno, D., Son, K., Taubin, G.: Embedded phase shifting: robust phase shifting with embedded signals. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015
19. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. IEEE Trans. Robot. **33**(5), 1255–1262 (2017)
20. Nießner, M., Zollhöfer, M., Izadi, S., Stamminger, M.: Real-time 3D reconstruction at scale using voxel hashing. ACM Trans. Graph. **32**(6), 169:1–169:11 (2013)
21. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. In: ACM SIGGRAPH 2003 Papers (2003)
22. Shan, Q., Adams, R., Curless, B., Furukawa, Y., Seitz, S.M.: The visual turing test for scene reconstruction. In: International Conference on 3D Vision (3DV) (2013)
23. Song, S., Xiao, J.: Tracking revisited using RGBD camera: unified benchmark and baselines. In: Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV) (2013)
24. Waechter, M., Moehrle, N., Goesele, M.: Let there be color! Large-scale texturing of 3D reconstructions. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 836–850. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_54
25. Zhou, Q., Koltun, V.: Dense scene reconstruction with points of interest. ACM Trans. Graph. **32**(4), 112:1–112:8 (2013)
26. Zhou, Q., Koltun, V.: Color map optimization for 3D reconstruction with consumer depth cameras. ACM Trans. Graph. **33**(4), 155:1–155:10 (2014)
27. Zhou, Q.-Y., Park, J., Koltun, V.: Fast global registration. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 766–782. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_47