Joint Texture and Geometry Optimization for RGB-D Reconstruction

Yanping Fu[1] Qingan Yan[2] Jie Liao[1] Chunxia Xiao[1,3,4]
[1] School of Computer Science, Wuhan University, China
[2] Silicon Valley Research Center, JD.com, United States
[3] National Engineering Research Center for Multimedia Software, Wuhan University, China
[4] Institute of Artificial Intelligence, Wuhan University, China
{ypfu,liaojie,cxxiao}@whu.edu.cn, qingan.yan@jd.com

Abstract

Due to inevitable noises and quantization error, the reconstructed 3D models via RGB-D sensors always accompany geometric error and camera drifting, which consequently lead to blurring and unnatural texture mapping results. Most of the 3D reconstruction methods focus on either geometry refinement or texture improvement respectively, which subjectively decouples the inter-relationship between geometry and texture. In this paper, we propose a novel approach that can jointly optimize the camera poses, texture and geometry of the reconstructed model, and color consistency between the key-frames. Instead of computing Shape-From-Shading (SFS) expensively, our method directly optimizes the reconstructed mesh according to color and geometric consistency and high-boost normal cues, which can effectively overcome the texture-copy problem generated by SFS and achieve more detailed shape reconstruction. As the joint optimization involves multiple correlated terms, therefore, we further introduce an iterative framework to interleave the optimal state. The experiments demonstrate that our method can recover not only fine-scale geometry but also high-fidelity texture.

1. Introduction

With the emergence of RGB-D sensors, it has become more convenient to reconstruct the scenes and objects in our daily life. Moreover, after the vigorous development of the 3D reconstruction community in recent years, the 3D reconstruction technology using RGB-D sensor has achieved a qualitative leap. We can reconstruct more detailed 3D models [3, 8, 18, 19, 21, 31, 32, 36, 38] and texture [2, 15, 20, 22, 28, 30, 40] for indoor scenes. However, the reconstructed models are far from being directly applied to applications, as the geometric accuracy and texture quality of 3D reconstruction results do not meet the requirement of applications like VR/AR, games, CAD manufacturing, and 3D printing.

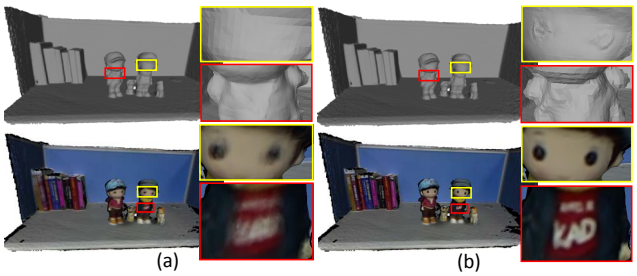

Figure 1. Joint texture and geometry optimization on RGB-D scanned geometry. (a) Without any optimization. (b) With the proposed joint optimization.

A high-quality 3D reconstructed model via the RGB-D sensor should reach two basic requirements, correct geometry and high-fidelity texture. They are mainly degraded by three factors: (1) the measuring error introduced by data acquirement equipment like noises, lens distortion and quantization error, (2) the accumulated errors during camera pose estimation and (3) the geometric error due to the sharp geometric feature over-smoothed by the moving weighted average of truncated signed distance field (TSDF) [7], which is commonly utilized as the implicit representation of depth data integration. Due to the geometric error and the camera drifting, the result of texture mapping in 3D reconstruction inevitably exhibits blurring and ghosting artifacts.

To achieve high-quality 3D reconstruction based on RGB-D sensor, various methods have been put forward. Wu *et al.* [33] and Zollhöfer *et al.* [41] refined reconstructed geometries with the guidance of shading information provided by color images, which have higher resolution than corresponding depth images, and hence are able to provide more visual cues. However, SFS-based geometry refinement easily suffers from the texture-copy problem [25]. Rather than modifying geometric shapes, Fu *et al.* [11], Bi *et al.* [2]

2575-7075/20/$31.00 ©2020 IEEE
DOI 10.1109/CVPR42600.2020.00599

and Zhou and Koltun [40] kept the geometry unchanged but modified the texture of the 3D model to compensate for geometric errors derived from reconstruction calculation. Different from previously mentioned works which only focus on either geometry refinement or texture optimization, Intrinsic3D [20] jointly optimized geometry, texture, camera poses, and scene lighting based on SFS and spatially-varying spherical harmonics (SVSH) from subvolumes. While this method is effective in certain scenes, it is super time-consuming and also suffers from the problem of texture-copy.

In this paper, we also jointly optimize camera poses, geometric detail and texture. However, different from [20], we build the proposed method directly on the reconstructed mesh model, which can achieve more detailed geometry than encoded in a TSDF representation. Therefore, we propose a novel method to enhance the geometry and utilizes the enhanced shape normals as a prior guidance to drive the adjustment of vertex positions; meanwhile, the texture and geometric consistency serve as complementary constraints to ensure the movement is reasonable. Accordingly, the updated geometry mesh will benefit the correction of textures and camera poses and not trigger texture-copy artifacts. The geometry and texture optimization results of the proposed method are shown in Figure 1. We demonstrate the effectiveness of the proposed method on various datasets. The results show that the proposed method can not only effectively enhance the geometry detail but also refine the texture of the reconstructed model.

2. Related Work

Geometry Refinement. Many methods [14, 16, 24, 26, 33, 37] focus on depth image refinement to improve the quality of 3D reconstruction. Although these methods work well in depth image refinement, their contributions to the final 3D model reconstruction are limited due to the inherent defect of the 3D reconstruction algorithm. Some other methods directly optimize the geometric model to refine the reconstruction. Xie *et al.* [34] proposed the angle profile as a new measurement to infer the hidden micro-structures from the existing surface and enhanced the geometric detail. Deng *et al.* [9] presented a variational method based on subdivision representation for both lighting and geometry, which was suitable for surface reconstruction in unknown illumination conditions. Romanoni *et al.* [27] refined a semantically annotated mesh through single-view variational energy minimization coupled with photo-consistency. Choe *et al.* [4] used shading cues captured from an IR camera instead of the RGB camera to refine the geometry of 3D mesh. Dzitsiuk *et al.* [10] used plane priors to filter out noise vertices during online reconstruction and got a clean and complete reconstructed model. Although the above methods can refine the geometry of the reconstruction result, they do not consider texture or do not perform any texture optimization.

Texture Optimization. Waechter *et al.* [39] used the Markov Random Field to select an optimal image for each triangle face as texture. Then they proposed a global color adjustment strategy to alleviate the visible seams between textures. Zhou and Koltun [40] used a color consistency method to optimize the camera poses and a local image warping method to compensate for the geometric error, which can obtain sharp texture results. Fu *et al.* [11] proposed a global-to-local non-rigid optimization method to correct camera poses and compensate for the geometric error by texture coordinate warping. This method can effectively alleviate the seams between textures and obtain satisfactory texture results. Bi *et al.* [2] used a patch-based image synthesis strategy to generate a target color image for texture mapping to avoid texture misalignment caused by geometric error and camera drifting. Kim *et al.* [17] and Alldieck *et al.* [1] focused on texture mapping for dynamic objects and humans. 3DLite [15] applied a simple planar abstraction to represent the reconstructed model. Then they used the planar-based constraint to optimize the camera pose and the texture on the abstraction planes. Although these methods can generate visually plausible texture, they do not perform optimization for the geometry.

Joint Optimization. Wang *et al.* [30] used planar primitives to partition the model to generate a lightweight and low-polygonal mesh, then jointly optimized plane parameters, camera poses, texture and geometry using photometric consistency and planar constraints. However, this method relies on plane priors and is not suitable for complex non-planar scenes. Intrinsic3D [20] presented a novel method to optimize geometry, texture, camera poses, and scene lighting simultaneously, which was based on SFS and SVSH using subvolumes. They can obtain high-quality 3D reconstruction with consistent texture. However, this method relies on SFS and needs to decompose the illumination of the scene, which easily causes the problem of texture-copy.

3. Method

The proposed method aims at getting the 3D reconstructed model with fine-scale geometric detail and high-fidelity texture via a commodity RGB-D camera. We propose a joint optimization framework to this end. Figure 2 shows the overview of the proposed method.

In this section, we will elaborate on each step of the proposed method. Let M_0 represent the initial reconstructed mesh model, $\{\mathbf{v}_i\}$ be the vertex set of M_0. We use $\mathbf{D}$ to represent the depth image, $\mathbf{C}$ denotes the color image, and I represents the corresponding intensity image. M^C represents the reconstructed model M with texture color. $\mathbf{T}$ is the camera pose, which can transform a vertex $\mathbf{v}$ from the local camera coordinate system to the world coordinate

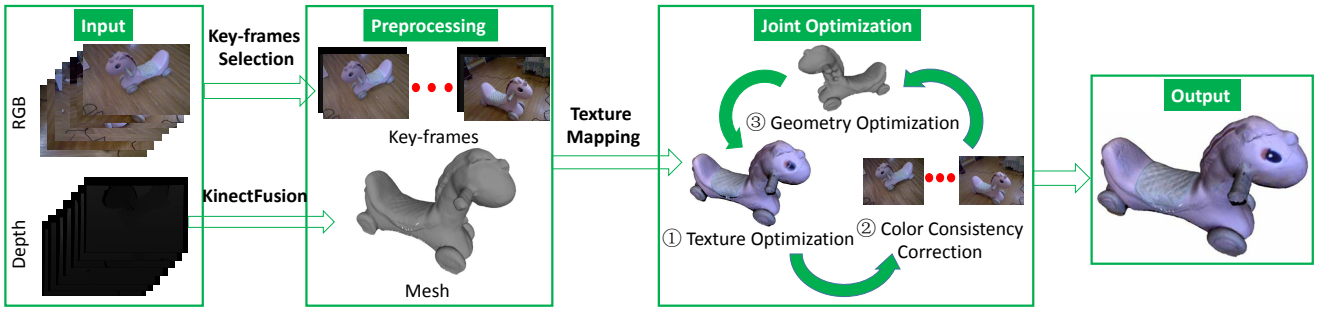

Figure 2. The overview of the proposed method. The input of the proposed method is an RGB-D sequence or stream. We utilize the depth images to reconstruct the initial 3D model and extract key-frames from the color images according to image quality. Subsequently, camera poses, geometry, texture, and color consistency between key-frames are jointly optimized in an iterative manner. The output is a 3D model with detailed geometry and high-fidelity texture.

system. We define $\mathbf{T}$ as:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{1}$$

where $\mathbf{R}$ is a rotation matrix and $\mathbf{t}$ is a translation vector.

Let Π represent the perspective projection including dehomogenization, which projects a vertex $\mathbf{v} = [x, y, z]^T$ from model M to the pixel $\mathbf{u}(u, v)$ of the image plane.

3.1. Mesh Reconstruction and Key-frames Selection

The input of the proposed method is an RGB-D sequence or stream acquired by a consumer RGB-D camera. We use the Microsoft Kinect to capture the RGB-D image sequence. Each RGB-D frame in the sequence includes a general RGB image and an aligned depth image. Then we use KinectFusion [23] to generate the 3D mesh model M_0 of the scene from the depth image sequence and record the evaluated camera pose $\mathbf{T}_i$ of each frame i. We further subdivide the reconstructed model according to the method [40]. KinectFusion can also be replaced with other state-of-the-art methods like BundleFusion [8] and the method of [5], which can obtain more accurate initial reconstructed model and camera poses.

To eliminate redundant color images and ensure the coverage of the reconstructed model, we extract the key-frames as the input texture images. The extraction procedure is performed in a recursive manner, where the new key-frame is selected if its corresponding camera pose satisfies the following formula:

$$C_i = \{C_i \in \Phi_{\text{KF}} : \angle(\mathbf{R}_k, \mathbf{R}_i) > 30^\circ \,||\, Dist(\mathbf{t}_k, \mathbf{t}_i) > 0.2\}, \tag{2}$$

where Φ_{KF} represents the key-frame set, k is the index of the last key-frame before the current frame i. $\angle(\mathbf{R}_k, \mathbf{R}_i)$ represents the angle between the two rotation matrices $\mathbf{R}_k$ and $\mathbf{R}_i$, and $Dist(\mathbf{t}_k, \mathbf{t}_i)$ represents the Euler distance between the two translation vectors $\mathbf{t}_k$ and $\mathbf{t}_i$. The field of view for Kinect is about 57° in horizontal and 43° in vertical, and the practical depth range is $0.5 \sim 3.5m$ [13], so we set the thresholds 30° and $0.2m$ respectively.

The hand-held depth camera easily introduces motion blurring and jitters. To avoid the influence of the blurring images in key-frames Φ_{KF} on texture and geometry optimization, we replace the initial key-frames with the clearest frames beside them. We use the image blurring measurement of [6] to get the score of each frame within the window around the key-frame. Then we replace the key-frame using the frame which obtains the highest score within the window. We set the window size to $[-5, 15]$.

3.2. Joint Geometry and Texture Optimization

After preprocessing, we can get an initial 3D mesh model M_0, a key-frame sequence $\{C_k \in \Phi_{\text{KF}}\}$ with corresponding camera poses $\{\mathbf{T}_k\}$. Due to geometric errors and camera drifting, it is difficult to obtain consistent texturing result for the reconstructed model through the acquired key-frames. In order to restore high-fidelity geometries and textures, we jointly refine the reconstructed 3D model and texture to recover high-frequency geometry details and high-quality textures. We implement our joint optimization strategy in three steps: (1) Optimizing the camera pose of each key-frame by minimizing the color inconsistency between the vertex on the model and its projection on each visible key-frame. (2) Correcting color inconsistency caused by illumination changes between key-frames. (3) Optimizing the position of the vertex on the model to make it not only physical correct but also color-consistent with its projection on visible key-frames.

3.2.1 Camera Poses and Texture Optimization

We first compute the initial texture of M_0 and get the texture model M_0^C. To get the texture of the mesh model M, we project each vertex $\mathbf{v}$ on M onto all the visible key-frames

to get all the color values of this vertex on the key-frames, then calculate the color of the vertex $\mathbf{v}$ by weighted averaging. We use $\kappa_{ij} = cos(\theta)^2/d^2$ as the weight of each vertex $\mathbf{v}_i$ corresponding to the j-th key-frame, where θ is the angle between the normal of vertex $\mathbf{v}_i$ and its view direction at the j-th key-frame, and d is the distance from vertex $\mathbf{v}_i$ to the camera center of the key-frame j. Due to the camera drifting and geometric error, it is easy to get inaccurate texture colors via projection, therefore, resulting in blurring and ghosting artifacts in the texture model M_0^C as shown in Figure 1(a).

During each iteration of the joint optimization, we first optimize the camera pose $\mathbf{T}$ of each key-frame. We adopt a similar idea of [40] to optimize the camera pose of each key-frame to ensure that the texture of the model M^C is as consistent as possible with the texture obtained by projecting it onto all the visible key-frames. The difference is that we consider not only color consistency but also geometric consistency, which is more robust to the texture-less scene. The objective E_{tex} is defined as:

$$E_{\text{tex}} = \lambda_c E_c + \lambda_g E_g, \tag{3}$$

where E_c is the photometric consistency term and E_g is the geometric consistency term. We experimentally set $\lambda_c = 1$ and $\lambda_g = 100$ to balance the metric difference between terms, and put more weight on geometric consistency for alleviating camera drifting in texture-less.

E_c is defined the same as the method of [40], which ensures the photometric error between the vertex on M^C and its corresponding projected texture on each key-frame is minimum.

$$E_c = \sum_{i}^{\#\text{KF}} \sum_{j}^{\#\text{vert}} \left(C(\mathbf{v}_j) - I_i(\Pi(\mathbf{T}_i^{-1}\mathbf{v}_j))\right)^2, \tag{4}$$

where $C(\mathbf{v}_j)$ is the intensity value of vertex $\mathbf{v}_j$ on the texture model M^C, i is the index of the key-frames, j is the index of vertex on M^C, #KF represents the number of the key-frames and #vert represents the number of vertices.

E_g ensures that the depth of each vertex on M^C should be as consistent as possible with the corresponding depth on the depth image of each visible key-frame.

$$E_g = \sum_{i}^{\#\text{KF}} \sum_{j}^{\#\text{vert}} \left(\varphi(\mathbf{T}_i^{-1}\mathbf{v}_j) - D_i(\Pi(\mathbf{T}_i^{-1}\mathbf{v}_j))\right)^2, \tag{5}$$

where $\varphi(\mathbf{v})$ is a function which fetches the third element of the vector $\mathbf{v}$.

After the camera pose optimization, we calculate the texture color of each vertex $\mathbf{v}_i$ on the model M to generate a new color texture model M^C as described above.

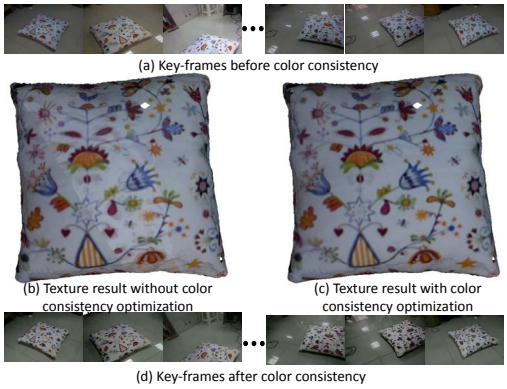

Figure 3. Qualitative comparisons between key-frames and texture results with and without color consistency correction.

3.2.2 Key-frames Color Consistency

The color images captured by the RGB-D sensor are easy to be affected by the factors such as the auto-white balancing, auto-exposure, and the illumination changes. These factors will cause color inconsistency between the color images captured from different views. To reduce the negative influences of the color inconsistency on our joint optimization, we optimize the color consistency across different key-frames after camera poses optimization in each iteration.

Inspired by [15], which used the method of NRDC [12] to color inconsistency between images, we also adopt a three spline curves $B_j(C(\mathbf{v}_i)) = (B_j^{\{r,g,b\}}(C(\mathbf{v}_i)))$ as a color transfer function to transfer the color of the vertex $\mathbf{v}_i$ on the model M^C to the color of its projection on the j-th key-frame. With the texture color of the model M^C as a reference, the color of all key-frames is corrected to achieve color consistency between all key-frames. We use the following formula to find the optimal spline curves B as color transfer function for each key-frame:

$$E_{color} = \sum_{i}^{\#\text{KF}} \sum_{j}^{\#\text{vert}} ||C(\mathbf{v}_j) - B_i(q_{ij})||^2 + \lambda_b \sum_{i}^{\#\text{KF}} \sum_{j}^{\#\text{vert}} (B_i^{'}(x_j) - 1)^2, \tag{6}$$

where q_{ij} is the color value of the pixel on the i-th key-frame corresponding to the projection of the vertex $\mathbf{v}_j$. According to the suggestion of [15], we regularize the derivatives of the transfer functions $B_i^{'}(x_j) \approx 1$. We also set $\lambda_b = 0.1$ and x_j is from 0 to 250 with interval 25.

After receiving the optimal color transfer function for each key-frame according to Eq. 6, we adjust the color of each key-frame using the corresponding color transfer function B to achieve color consistency across key-frames, as shown in Figure 3.

3.2.3 Geometry Optimization

In this step, we recompute the color value of each vertex on the model M^C. If the color of each vertex is not consistent

with the color value that it obtains from the key-frame by perspective projection, we consider that it is due to geometric reconstruction error. Therefore, we need to further refine the position of the vertex on the model M^C to correct the reconstructed geometric error.

To optimize the color consistency between the reconstructed model M^C and the projected texture on the key-frames, we compute an offset vector for each vertex on the model to correct the geometric error. We define the energy function as:

$$E_{\text{geo}} = E_{\text{tex}} + \lambda_H E_H + \lambda_L E_L + \lambda_R E_R, \tag{7}$$

where E_{tex} is the photometric and geometric consistency term defined as Eq. 3, E_H is the high-boost enhancement term, E_L is the Laplacian data term, and E_R is the regularization term. λ_H, λ_L and λ_R are coefficients balancing between terms.

E_H is used to boost the high-frequency geometric detail of the reconstructed model M^C. First, we separate the high-frequency detail of the model M_0 by subtracting the smoothed version of M_0 from the original model M_0. Then we amplify the high-frequency of the model using the high-boost strategy [35]. Finally, we take the high-boost normal as a normal consistency constraint to refine the geometry of the reconstructed model according with the photometric consistency and geometric consistency as guidances. The high-boost enhancement adjusts the position of the vertex to fit the high-boost normal of the triangle adjacent to the vertex. The normal-based error at vertex $\mathbf{v}$ is defined as the area-weighted sum of squared difference between smooth normal $\mathbf{n}_s(R)$ of triangle R and high-boost normal $\mathbf{n}_h(R)$ of triangle R:

$$E_n = \sum_{i \in \Upsilon(\mathbf{v})} A(R_i)(\mathbf{n}_s(R_i) - \mathbf{n}_h(R_i))^2, \tag{8}$$

where $\Upsilon(\mathbf{v})$ is the index set for 1-ring neighborhood triangles around $\mathbf{v}$, R_i represents the i-th adjacent triangle of vertex $\mathbf{v}$. $A(R)$ is the area of the triangle R. $\mathbf{n}_s(R)$ and $\mathbf{n}_h(R)$ are the smoothed normal and the boosted normal of triangle R, they can be calculated according to work [35].

We can minimize Eq. 8 to enhance the position of vertex $\mathbf{v}$ to fit the high-frequency detail of the reconstructed model. The partial derivative of Eq. 8 with respect to $\mathbf{v}$ is given as:

$$\frac{\partial E_n}{\partial \mathbf{v}} = 2 \sum_i \left(\frac{\partial A(R_i)}{\partial \mathbf{v}} - \frac{\partial A(S_i)}{\partial \mathbf{v}} \right), \tag{9}$$

where S_i is a triangle generated by projecting triangle R_i onto the plane defined by the boosted normal $\mathbf{n}_h(R)$. Then we can update the position of vertex $\mathbf{v}$ as:

$$\mathbf{v}^h = \mathbf{v} - \lambda_{hb} \sum_{i \in \Upsilon(\mathbf{v})} \left(\frac{\partial A(R_i)}{\partial \mathbf{v}} - \frac{\partial A(S_i)}{\partial \mathbf{v}} \right), \tag{10}$$

where $\mathbf{v}^h$ is the refined position by high-boost enhancement. We set $\lambda_{hb} = 0.2$ in all the experiments.

To recover high-frequency geometric details, the optimized position of each vertex on the reconstructed model should restore the high-frequency detail as much as possible. We define the high-boost term E_H as:

$$E_H = \sum_i^{\#\text{vert}} ||\mathbf{v}_i - \mathbf{v}_i^h||^2. \tag{11}$$

E_L uses the Laplacian operator [29] to preserve local geometric features and suppress irregular vertex movement during geometric optimization.

$$E_L = \sum_i^{\#\text{vert}} ||\mathbf{v}_i - \frac{1}{\sum_j \omega_{ij}} \sum_{j \in \Omega_i} \omega_{ij} \mathbf{v}_j||^2, \tag{12}$$

where Ω_i is the index set for vertices in 1-ring neighborhood of vertex $\mathbf{v}_i$ and j belongs to Ω_i, ω_{ij} is the weight between vertex $\mathbf{v}_i$ and $\mathbf{v}_j$. In our implementation, we set $\omega_{ij} = 1$.

E_R ensures that each vertex $\mathbf{v}$ will not deviate too far away during the optimization procedure and is defined as:

$$E_R = \sum_i^{\#\text{vert}} ||\mathbf{v}_i - \widetilde{\mathbf{v}}_i||^2, \tag{13}$$

where $\widetilde{\mathbf{v}}_i$ is the adjusted position of vertex $\mathbf{v}_i$ from last iteration.

3.3. Minimization

We optimize the parameters $(\mathbf{T}, B, \mathbf{V})$ in an iterative manner, where we apply external iterations to perform joint optimization. In each external iteration, we preform three internal optimization in turn. First, we fix B and $\mathbf{V}$, and minimize $E(\mathbf{T}) = E_{\mathbf{tex}}$ to optimize the camera pose $\mathbf{T}$ of each key-frame. Second, we optimize $E(B) = E_{\text{color}}$ to recompute the color transfer function B of each key-frame according to the optimized camera pose $\mathbf{T}$ and M^C. Finally, we fix $\mathbf{T}$ and B to minimize $E(\mathbf{V}) = E_{\text{geo}}$ to refine each vertex $\mathbf{v}$ on the reconstructed model M^C.

To achieve a balance between convergence and performance, we find that setting the external iteration number to 5, and the internal iteration number respectively in each step to 50, 10, and 20 can achieve good performance. The trend of convergence rate and number of external iterations is illustrated in Figure 4.

4. Results

In this section, we first evaluate our method against the state-of-the-art methods [2, 11, 20, 40] on public RGB-D datasets and the datasets acquired by ourselves using Kinect. Then we perform the ablation studies to validate the

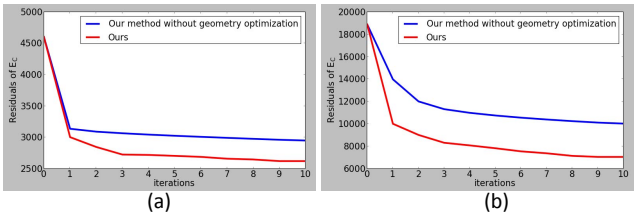

Figure 4. The changes of E_c with and without geometry optimization according to different external iteration numbers on dataset (a) *Tomb-statuary* and (b) *Bricks*.

effectiveness of each component in our method. Finally, we discuss the limitations of our method. All the experiments are conducted on a computer with an Intel i7 3.6GHz CPU and 8GB RAM, and GeForce GTX1060 6GB. We use the codes publicly released by the authors [20], [40] and [11], and a implementation version of [2] by ourselves. We empirically set $\lambda_H = 1 \times 10^4$, $\lambda_L = 1000$ and $\lambda_R = 1000$.

4.1. Evaluation on Public Datasets

We first compare the texture and geometry optimization results between our method and its most related method Intrisic3D [20] on datasets provided by Intrisic3D. From the close-up of the comparison results in Figure 5, it is notable that our method achieves more striking results in both texture optimization and geometry optimization. Since Intrinsic3D [20] is based on SFS, it works better in some tiny geometric details, for example, the eyes and eyebrows in the dataset *tomb-statuary*. However, due to the inherent defect of the SFS-based method, Intrinsic3D is easy to suffer from texture-copy, as shown in the datasets *gate* and *bricks*. In addition, our method introduces a high-boost term to fit the normal of high-frequency geometric detail, which can restore better medium scale geometric details than Intrinsic3D, as shown in Figure 5.

We also compared our method with the state-of-the-art methods of Zhou *et al.* [40], Fu *et al.* [11] and Bi *et al.* [2] on texture optimization, and Intrinsic3D [20] on texture and geometry optimization on the dataset *fountain* provided by [40], as shown in Figure 7. Notable that the texture optimization results generated by our method are comparable to [2, 40, 11], which are specially designed for texture optimization. While they perform well on textures, the geometries are still remain unchanged. Instead, our method optimizes not only the texture mapping result but also the geometry, and produces clear and detailed results as compared to Intrinsic3D [20], which also conducts a joint optimization scheme.

4.2. Evaluation on Collected Datasets

To demonstrate that the proposed method is effective to any consumer RGB-D cameras, we use the Microsoft

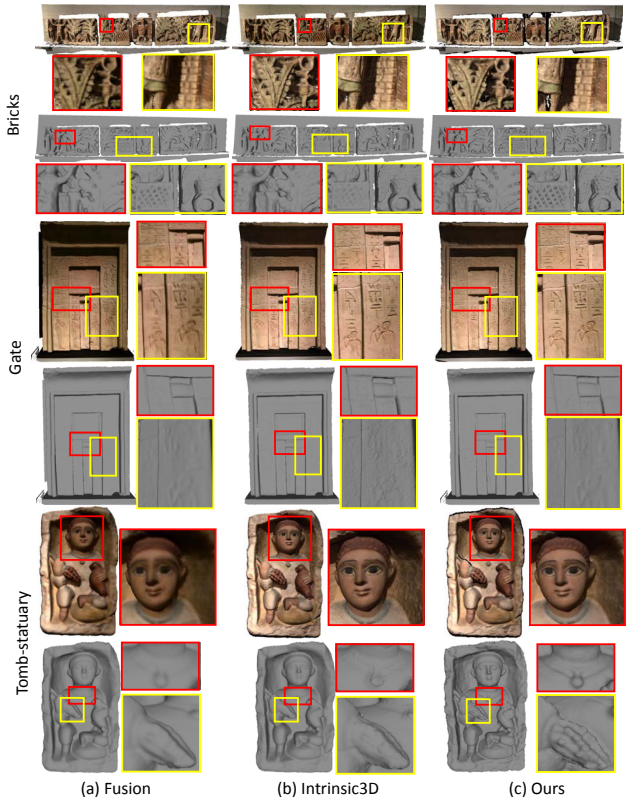

Figure 5. The comparison results with Intrinsic3D [20] on the datasets provided by Intrinsic3D [20]. (a) The reconstructed model and texture results. (b) The texture and geometry optimization results provided by Intrinsic3D [20]. (c) The texture and geometry optimization results of the proposed method.

Kinect v1 captured RGB-D image sequences as input.

Figure 8 shows the qualitative geometry comparisons between our method and Intrinsic3D [20] on our newly collected datasets. From the close-up images, we can see that our method can restore geometry with more high-frequency details. Due to the limited resolution of color and depth images captured by Kinect $v1$, it is challenging to directly capture very high-frequency geometric and textural details. Yet, our method is able to take into account not only the geometric and color consistency but also the normal enhancement of high-frequency geometry. Therefore, our method can restore more high-frequency geometric details. Besides, due to the ambiguity of the SFS-based method on processing textural and geometric details, Intrinsic3D is easy to introduce texture-copy artifacts, as shown in Figure 9. To restore high-quality geometric details, Intrinsic3D [20] requires to divide the TSDF many times, which is both time and memory consuming. However, our method directly optimizes the mesh of the reconstructed model, which requires less memory and time to reach the geometry and texture optimization goals. The running time of the proposed method

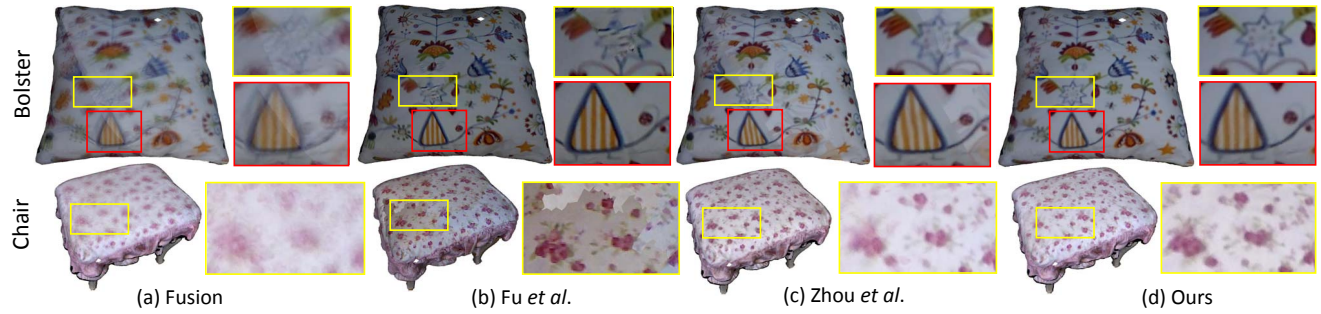

Figure 6. The texture optimization comparison results. (a) The texture results by KinectFusion [23]. (b) The texture results by Fu *et al.* [11]. (c) The texture results by Zhou *et al.* [40]. (d) The texture results by the proposed method.

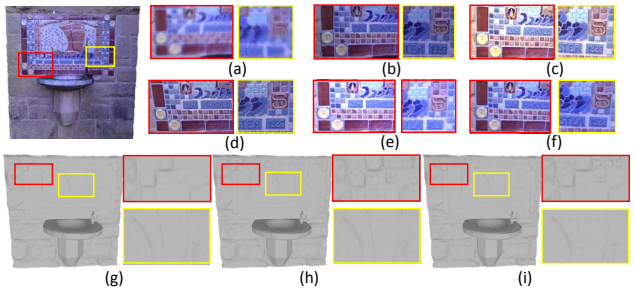

Figure 7. The qualitative comparisons with the state-of-the-art methods on public dataset *fountain* provided by [40]. (a) and (g) are the texture and geometry result by KinectFusion [23]. (e) and (h) are the texture and geometry optimization results by Intrinsic3D [20]. (f) and (i) are the texture and geometry optimization results by the proposed method. (b), (c) and (d) are the texture results by [11], [40] and [2] respectively.

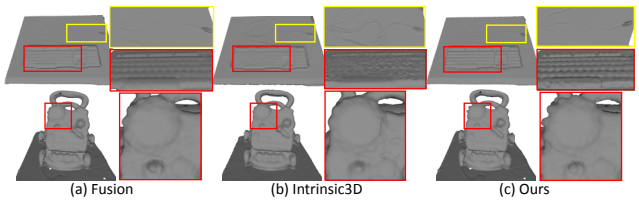

Figure 8. Qualitative geometry comparisons between KinectFusion [23], Intrinsic3D [20] and our method on datasets *keyboard* and *walker* captured by ourselves using Kinect.

is reported in Table 1. In contrast, Intrinsic3D [20] has to spend days on processing these datasets in the same computing environment.

Figure 6 shows the texture comparisons between our method and the methods of [40] and [11]. Due to the auto-exposure and illumination changes, color images captured from different views easily suffer from color inconsistency. From Figure 6, it is can be seen that the color inconsistency between color frames have a significant impact on the texture result of [40] and [11]. Since the method [40] does not use any color consistency processing, when there are drastic color inconsistencies between key-frames, the final results demonstrate obvious brightness inconsistency, as shown in the yellow box in Figure 6(c). While the method [11] uses a global to local non-rigid correction to stitch the textures, the seams between textures cannot be eliminated completely, as shown in Figure 6(b). In contrast, the proposed method uses an effective color consistency processing, so the generated texture color tends to be consistent. Additionally, the method of [40] uses a local warp method to compensate for the reconstructed geometric error to correct texture misalignment. Therefore, there will be some significant distortion in the texture result, as shown in the red box of Figure 6(c). However, the proposed method directly performs color consistency between key-frames and refines the geometry, which can get better texture and geometric details than these methods.

Dataset	#face	#vert	#KF	Time
Bricks	1,248,351	666,927	31	75.26
Gate	1,270,574	648,874	39	117.60
Tomb-statuary	475,024	243,443	11	31.32
Keyboard	537,900	270,767	12	41.29
Walker	1,616,112	817,145	18	52.05

Table 1. Running time (minute) of the proposed method on different datasets.

4.3. Ablation Studies

In this section, we investigate the effectiveness of each component of our method. To assess the effectiveness of the color consistency optimization, we compare the texture mapping results of our method with and without color consistency optimization, as demonstrated in Figure 3. It can be observed that color consistency optimization can correct color consistency across key-frames and effectively suppress the influence of illumination changes on the texture-mapping result, as shown in Figure 3(c).

Figure 4 shows the convergence rate of E_c of our method

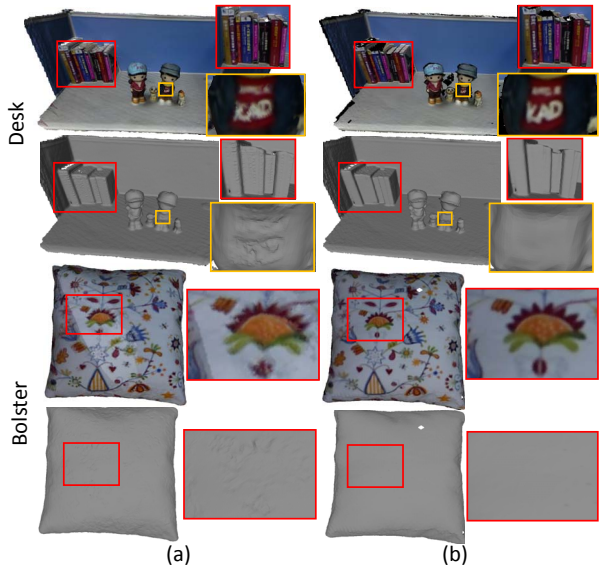

Figure 9. The texture-copy artifact comparison results. (a) The texture-copy artifact on the texture and geometry optimization of Intrinsic3D [20]. (b) Our method is not affected by texture-copy.

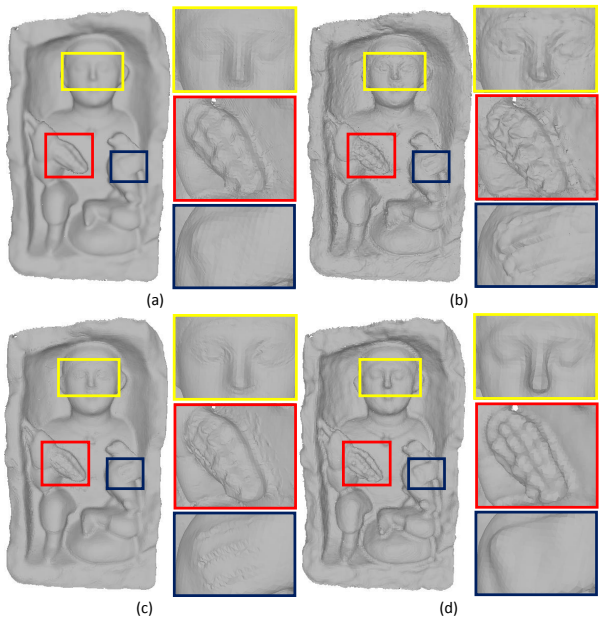

Figure 10. The comparison results of geometry optimization with different weights. (a) $\lambda_L = 1 \times 10^5$ and $\lambda_R = 1 \times 10^5$. (b) $\lambda_c = 100$ and $\lambda_H = 1 \times 10^6$. (c) $\lambda_c = 100$. (d) $\lambda_H = 1 \times 10^6$.

respectively without geometry optimization and with geometry optimization in external iterations. The value of E_c (Eq. 4) directly reflects the quality of the texture. From Figure 4, it is notable that our geometric optimization method make E_c converge faster and converge to a better value than the method without geometry optimization.

Figure 10 shows the mesh model comparisons between our method with different coefficients. To control variables, we only change one or two coefficients each time and keep the others as default. To demonstrate the effectiveness of Laplacian and regularization terms, we increase λ_L and λ_R from 1000 to 1×10^5. From Figure 10(a), we can see that the noises will significantly decrease while the detail is smoothed. To demonstrate the joint effectiveness of color consistency term and high-boost enhancement, we increase λ_C from 1 to 100 and λ_H from 1×10^4 to 1×10^6. It can be observed from Figure 10(b) that the geometric detail is enhanced while the vertices drifting. To demonstrate the effectiveness of the color consistency term, we increase λ_C from 1 to 100. It can be see from Figure 10(c) that the tiny geometric detail is enhanced while the high-frequency detail is not restored. To demonstrate the effectiveness of high-boost enhancement, we increase λ_H from 1×10^4 to 1×10^6. It can be observed from Figure 10(d) that the high-frequency detail is enhanced, but the tiny geometric detail is missing and some vertices drifting without color consistency and depth consistency constraints.

Limitations. While our method can effectively enhance the geometry and texture of the reconstructed model, it cannot get rid of the case that the geometry of the reconstruction is significantly missing. For example, due to the limited resolution and noises within depth image, some tiny objects may fail to be reconstructed in the initial session. In this case, even with the enhancement of our optimization framework, the missing geometry is still unable to be recovered.

5. Conclusion

In this paper, we have presented a joint optimization method to refine the texture and enhance the geometry of the 3D reconstruction by an RGB-D camera, which optimizes the camera poses, geometry and texture of the reconstructed model, and color consistency between key-frames simultaneously. First, we use the photometric consistency and the geometric consistency as cues to optimize the camera poses and texture of the reconstructed model. Then, we use the texture optimization result to correct the color inconsistency between key-frames. Finally, we refine the geometry of the reconstructed model by photometric and geometric consistency, as well as the normal of the high-frequency geometry cues. After such joint optimization, we can achieve not only high-fidelity textures but also high-quality geometries.

Acknowledgments. This work was partly supported by the National Key Research and Development Program of China (2017YFB1002600), the NSFC (No. 61672390, 61972298), the Key Technological Innovation Projects of Hubei Province (2018AAA062), Wuhan Science and Technology Plan Project (No. 2017010201010109). The corresponding author is Chunxia Xiao.

References

[1] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Detailed human avatars from monocular video. In *3DV*, 2018.

[2] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Patch-based optimization for image-based texture mapping. *ACM Transactions on Graphics*, 36(4):106–1, 2017.

[3] Yanpei Cao, Leif Kobbelt, and Shimin Hu. Real-time high-accuracy three-dimensional reconstruction with consumer RGB-D cameras. *ACM Transactions on Graphics*, 37(5):171, 2018.

[4] Gyeongmin Choe, Jaesik Park, Yu-Wing Tai, and In So Kweon. Refining geometry from depth sensors using IR shading images. *International Journal of Computer Vision*, 122(1):1–16, 2017.

[5] Sungjoon Choi, Qianyi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015.

[6] Frederique Crete, Thierry Dolmiere, Patricia Ladret, and Marina Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. In *SPIE*, 2007.

[7] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996.

[8] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics*, 36(4):76a, 2017.

[9] Teng Deng, Jianmin Zheng, Jianfei Cai, and Tat-Jen Cham. Shading-based surface recovery using subdivision-based representation. *Computer Graphics Forum*, 38(1):417–428, 2019.

[10] Maksym Dzitsiuk, Jürgen Sturm, Robert Maier, Lingni Ma, and Daniel Cremers. De-noising, stabilizing and completing 3D reconstructions on-the-go using plane priors. In *ICRA*, 2017.

[11] Yanping Fu, Qingan Yan, Long Yang, Jie Liao, and Chunxia Xiao. Texture mapping for 3D reconstruction with RGB-D sensor. In *CVPR*, 2018.

[12] Yoav HaCohen, Eli Shechtman, Dan B Goldman, and Dani Lischinski. Non-rigid dense correspondence with applications for image enhancement. *ACM Transactions on Graphics*, 30(4):70, 2011.

[13] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE transactions on cybernetics*, 43(5):1318–1334, 2013.

[14] Yudeog Han, Joon Young Lee, and In So Kweon. High quality shape from a single RGB-D image under uncalibrated natural illumination. In *ICCV*, 2013.

[15] Jingwei Huang, Angela Dai, Leonidas J Guibas, and Matthias Nießner. 3DLite: towards commodity 3D scanning for content creation. *ACM Transactions on Graphics*, 36(6):203–1, 2017.

[16] Achuta Kadambi, Vage Taamazyan, Boxin Shi, and Ramesh Raskar. Polarized 3D: High-quality depth sensing with polarization cues. In *ICCV*, 2015.

[17] Jungeon Kim, Hyomin Kim, Jaesik Park, and Seungyong Lee. Global Texture Mapping for Dynamic Objects. *Computer Graphics Forum*, 2019.

[18] Jie Liao, Yanping Fu, Qingan Yan, and Chunxia Xiao. Pyramid multi-view stereo with local consistency. In *Computer Graphics Forum*, volume 38, pages 335–346, 2019.

[19] Lingjie Liu, Nenglun Chen, Duygu Ceylan, Christian Theobalt, Wenping Wang, and Niloy J Mitra. Curve Fusion: reconstructing thin structures from RGBD sequences. In *SIGGRAPH Asia*, page 218, 2018.

[20] Robert Maier, Kihwan Kim, Daniel Cremers, Jan Kautz, and Matthias Nießner. Intrinsic3D: High-quality 3D reconstruction by joint appearance and geometry optimization with spatially-varying lighting. In *ICCV*, volume 3, 2017.

[21] R. Maier, R. Schaller, and D. Cremers. Efficient online surface correction for real-time large-scale 3D reconstruction. In *BMVC*, 2017.

[22] Robert Maier, Jörg Stückler, and Daniel Cremers. Super-resolution keyframe fusion for 3D modeling with high-quality textures. In *3DV*, 2015.

[23] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[24] Roy Or-El, Rom Hershkovitz, Aaron Wetzler, Guy Rosman, Alfred M Bruckstein, and Ron Kimmel. Real-time depth refinement for specular objects. In *CVPR*, 2016.

[25] Roy Or El, Guy Rosman, Aaron Wetzler, Ron Kimmel, and Alfred M. Bruckstein. RGBD-Fusion: Real-time high precision depth recovery. In *CVPR*, 2015.

[26] Jaesik Park, Hyeongwoo Kim, Yu-Wing Tai, Michael S Brown, and Inso Kweon. High quality depth map upsampling for 3D-TOF cameras. In *ICCV*, 2011.

[27] Andrea Romanoni, Marco Ciccone, Francesco Visin, and Matteo Matteucci. Multi-view stereo with single-view semantic mesh refinement. In *ICCV*, 2017.

[28] Christiane Sommer and Daniel Cremers. Joint representation of primitive and non-primitive objects for 3D vision. In *3DV*, 2018.

[29] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. Laplacian surface editing. In *SGP*, 2004.

[30] Chao Wang and Xiaohu Guo. Plane-based optimization of geometry and texture for RGB-D reconstruction of indoor scenes. In *3DV*, 2018.

[31] Oliver Wasenmüller, Marcel Meyer, and Didier Stricker. CoRBS: Comprehensive RGB-D benchmark for SLAM using Kinect v2. In *WACV*, 2016.

[32] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016.

[33] Chenglei Wu, Michael Zollhöfer, Matthias Nießner, Marc Stamminger, Shahram Izadi, and Christian Theobalt. Real-time shading-based refinement for consumer depth cameras. *ACM Transactions on Graphics*, 33(6):200, 2014.

[34] Wuyuan Xie, Miaohui Wang, Xianbiao Qi, and Lei Zhang. 3D surface detail enhancement from a single normal map. In *ICCV*, 2017.

[35] Hirokazu Yagou, A Belyaevy, and D Weiz. High-boost mesh filtering for 3-D shape enhancement. *Journal of Three Dimensional Images*, 17(1):170–175, 2003.

[36] Qingan Yan, Long Yang, Ling Zhang, and Chunxia Xiao. Distinguishing the indistinguishable: Exploring structural ambiguities via geodesic context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3836–3844, 2017.

[37] Shi Yan, Chenglei Wu, Lizhen Wang, Feng Xu, Liang An, Kaiwen Guo, and Yebin Liu. DDRNet: Depth map denoising and refinement for consumer depth cameras using cascaded CNNs. In *ECCV*, 2018.

[38] Long Yang, Qingan Yan, Yanping Fu, and Chunxia Xiao. Surface reconstruction via fusing sparse-sequence of depth images. *IEEE Transactions on Visualization and Computer Graphics*, 24(2):1190–1203, 2018.

[39] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape-from-shading: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.

[40] Qianyi Zhou and Vladlen Koltun. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Transactions on Graphics*, 33(4):155, 2014.

[41] Michael Zollhöfer, Angela Dai, Matthias Innmann, Chenglei Wu, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Shading-based refinement on volumetric signed distance functions. *ACM Transactions on Graphics*, 34(4):96, 2015.