VIEW-DEPENDENT VIRTUAL REALITY CONTENT FROM RGB-D IMAGES

Chih-Fan Chen, Mark Bolas, and Evan Suma Rosenberg [*]

Mixed Reality Lab, Institute for Creative Technologies, University of Southern California

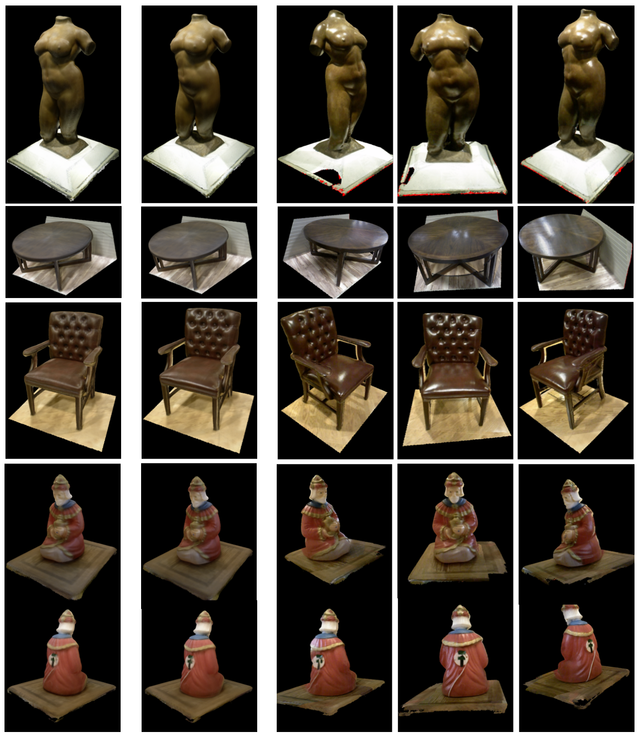

Fig. 1. (a) Fixed-texture models generated using the traditional approach of blending color images without texture optimization [1] and (b) with texture optimization [2]. The textures appear the same from all viewpoints. (c) Virtual objects generated using our real-time view-dependent approach, as viewed from three different perspectives.

ABSTRACT

High-fidelity virtual content is essential for the creation of compelling and effective virtual reality (VR) experiences. However, creating photorealistic content is not easy, and handcrafting detailed 3D models can be time and labor intensive. Structured camera arrays, such as light-stages, can scan and reconstruct high-fidelity virtual models, but the expense makes this technology impractical for most users. In this paper, we present a complete end-to-end pipeline for the capture, processing, and rendering of view-dependent 3D models in virtual reality from a single consumer-grade depth camera. The geometry model and the camera trajectories are automatically reconstructed and optimized from a RGB-D image sequence captured offline. Based on the head-mounted display (HMD) position, the three closest images are selected for real-time rendering and fused together to smooth the transition between viewpoints. The specular reflections and light-burst effects can also be preserved and reproduced. We confirmed that our method does not require technical background knowledge by testing our system with data captured by non-expert operators.

***Index Terms*—** Image-Based Rendering, Virtual Reality, Appearance Representation

1. INTRODUCTION

With the recent proliferation of consumer head-mounted displays (HMDs), there is increasing demand for realistic 3D content that can be integrated into virtual reality environments. However, creating photorealistic models is not only difficult but also time consuming and expensive. A simpler alternative involves scanning objects in the real world and rendering their digitized counterpart in the virtual world. The geometry of objects can be achieved by performing a 3D scan using widely available consumer-grade RGB-D cameras. The texture is determined by fusing data from multiple color images captured during the scan. Existing methods compute the color of each vertex by averaging the colors from all or these images. However, this technique of blending colors creates blurry, low-fidelity models with fixed lighting textures that are baked onto the model (see Figure 1 (a) and (b)). This limitation becomes more apparent when viewed in head-tracked virtual reality, as the illumination (e.g. specular reflections) does not change appropriately based on the user's viewpoint.

To improve color fidelity, techniques such as View-Dependent Texture Mapping (VDTM) have been introduced [3, 4, 5]. In this approach, the system finds observed camera poses closest to the view point and uses the corresponding color images to texture the model. Previous work has used Structure-from-Motion and Stereo Matching to automatically generate the model and the camera trajectory. Although these methods typically result in higher color fidelity, the reconstructed geometric model is often less detailed and more

[*]e-mail:{cfchen, bolas, suma}@ict.usc.edu

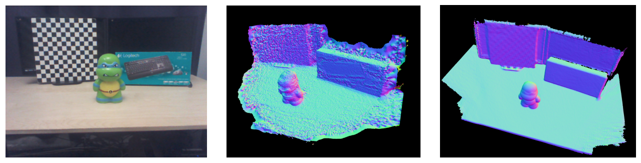

Fig. 2. Comparison of the 3D geometry reconstructed using structure from motion (SfM) with color images vs. KinectFusion with depth images.

prone to error than depth-based approaches. In this work, we leverage the strengths of both methods to create a novel view-dependent rendering pipeline (Figure 3). In our method, the 3D model is reconstructed from the depth stream using KinectFusion. The camera trajectory computed during reconstruction is then refined using the images from the color camera to improve photometric coherence. The color of 3D model is then determined at runtime using a subset of color images that best match the viewpoint of the observing user (Figure 1 (c)).

2. RELATED WORK

To visualize a real object in the virtual environment, image-based modeling and rendering (IBMR) [6] requires a set of color images captured from different viewpoints to generate a new, uncaptured, view of the object or scene. The light field rendering [7, 8, 9] used a ray tracing method to color each pixel of the novel view with the corresponding color from images in the database. Although these methods do not need the geometry information, they usually require a specially designed camera array and well-controlled environment for capturing data. On the other hand, generating the uncaptured view from VDTM requires a geometry model and the related camera position of each image. Each pixel color of the novel view is interpolated from the mesh of the 3D model and several appropriate images. As mentioned in [3], the visual affect is reduced due to the geometry information. However, manually reconstructing the model and setting the camera trajectory involves significant human effort. Several visual odemetry techniques such as SfM [10, 5] or v-SLAM [11] were proposed to automatically generate both the model and camera trajectory from color images. However, these methods depend solely on color information. Thus, they fail to build the geometry information if the visual features are not distinct or the appearance changes dramatically between frames due to illumination differences (Figure 2 (b)).

The advent of consumer-grade depth sensors such as the Microsoft Kinect has led to widespread interest in 3D scanning. Variants on the KinectFusion algorithm [1, 12] have gained popularity, due to its real-time reconstruction ability. To color geometric models, existing systems use a volumetric blending approach that integrates color samples over a voxel grid [1, 12, 13]. This produces color maps that convey only the objects general appearance. They suffer from blurring, ghosting, and other visual artifacts that are readily apparent at close range. Recent works [2, 14] had proposed color mapping optimization to produce higher quality models. They aim to find the best representative color of each pixel by several observations from a RGB sequence. However, the models with fixed textures lack material property and are not ideal to represent the true appearance from different viewpoints.

We propose a method that combines the advantage of the two different approaches. Our system can generate a 3D model and the corresponding camera trajectory that maximizes the color consistency from different viewpoints. Due to the accurate and dense geometry information, our system does not need to interpolate the pixel color in a novel view. Instead, we update the color of each vertex based on the user's viewpoint. Because of the independence of each vertex, the system can render the unseen view in real-time performance. This technique provides a better color appearance and a smoother transition when compared to a fixed-texture model.

3. PROPOSED FRAMEWORK

3.1. System Overview

We propose a system pipeline (Figure 3) to generate a view-dependent 3D model from a RGB-D camera that involves both offline and online stages. In the offline stage (Sec.3.2), our method uses the depth data to reconstruct a geometric model (Figure 2 (c)) and generate a corresponding camera trajectory. To maximize the texture quality and the cover area, we select color images based on the blurriness and the distribution in physical space. Color trajectory optimization is used to eliminate the noise from color-depth transformation and leads to better color consistency. In the online stage (Sec.3.3), the 3D model is projected to each color image for the visibility information, and the color of each vertex is precomputed by averaging all of the images. Note that the procedure above is performed only once. Based on the user's viewpoint, the closest images are fused to generate the texture at run-time.

3.2. Offline Stage

KinectFusion [1] is used to reconstruct a 3D model by integrating the depth data into a voxel volume over time. The camera pose of the latest frame is computed by comparing its depth image and the reconstructed model from previous frames. The 3D model $\boldsymbol{M}$ and the camera trajectory $\boldsymbol{T} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, \cdots \boldsymbol{T}_n\}$, where $\boldsymbol{T}_n$ is the extrinsic matrix of the n-th tracked camera pose, are generated after capturing. Compared to pure color images (Figure 2 (b)), the model created by the depth sensor is higher quality (Figure 2 (c)).

As the data is captured from a handheld camera, some of the color images are not ideal for rendering due to motion

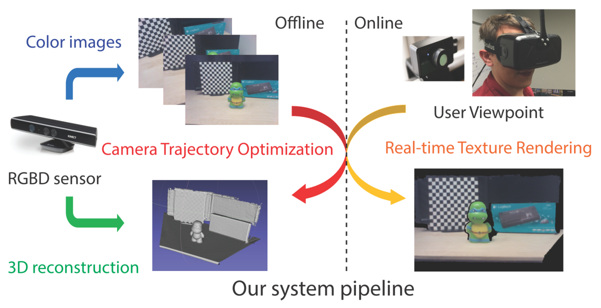

Fig. 3. An overview of the complete pipeline, consisting of two phases: (1) offline processing/optimization, and (2) online view-dependent texture rendering.

blur. Using blurry images does not improve the quality of texture rendering and also increases memory usage at run-time. Instead of using every image from the trajectory, we select a set of color images I_i with good image quality. In contrast to Zhou et al. [2], who select frames from different time intervals, we aim to maximize the covered area of viewpoints in our online stage view-dependent rendering. To achieve this purpose, the color images are first ranked by the blurriness [15]. We add the top ranked images such that the distance between each selected image and the images already in I_i must be larger than d cm, until we select K frame (see Algorithm 1). In our experiments, K was 100-150 and d was 5-10, based on the total number of images and the covered area of the object while capturing the data.

```
Data: Color sequence $I$
Result: Selected color image sequence $I_{sel}$
Ranked color sequence $I_R$= sort(blurriness($I$));
$I_{sel} = \emptyset$ ;
while $i \in I_R$ do
    if $(P_i - P'_i > d, \forall i' \in I_{sel})$ then
        $I_{sel} = I_{sel} \cup \{i\}$
    end
    if (length($I_{sel}$ == $K$)) then
        break;
    end
end
```

Algorithm 1: Key Frame Selection

The initial camera trajectories from KinectFusion are not sufficiently accurate for texture mapping because they are purely based on geometry. To maximize the color and geometric agreement, first we calibrate and align the color and IR camera. Then, we apply Color Mapping Optimization [2] to yield even more accurate camera poses. The objective function minimizes the difference between the color of the vertices and their corresponding color in each frame. Note that we only use the alternating optimization method; as noted by Narayan et al. [14] the deformation grid optimization method sometimes leads to divergence. This approach iteratively solves the problem from the initial state $\boldsymbol{T}_{sel}$ and converges to $\boldsymbol{T}_{opt}$ by Gauss-Newton method.

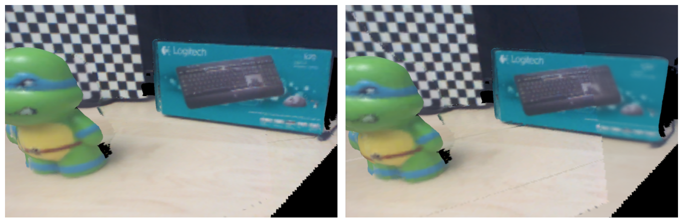

Fig. 4. A view-dependent texture rendered using three source images (left) compared with a texture generated using a single image (right).

3.3. Online Stage

To update the vertex color, the visibility is used to avoid incorrect texture mapping. We project the 3D model to each image space and generate the associated depth images for a visibility check. If the distance from the vertex to the image is larger than the value in the depth image, the vertex is considered invisible. By using visibility information, the vertex color can be updated in a parallel fashion, even without geometric information, to achieve real-time performance. We also pre-compute the basic color for each vertex by averaging the RGB value from all images that pass the visibility check. The vertex color remains the same if not rendered by any additional images. These two procedures only need to be applied once unless images are added or removed from the database, so it does not effect the process time at run-time.

At the online stage, we sample images based on the euclidean distance between the user's HMD position and all the camera positions in our database. The HMD position $\boldsymbol{p_u}$ is provided by the Oculus Rift DK2's position tracking camera. The camera position of each image can be computed from the transformation matrix T_{opt} in Sec. 3.2.3. Each transformation matrix $\boldsymbol{T}_i \in T_{opt}$ can be decomposed to a rotation matrix $\boldsymbol{R_i} \in \mathbb{R}^{3X3}$ and a translation matrix $\boldsymbol{t_i} \in \mathbb{R}^{1x3}$. The camera position is obtained by $\boldsymbol{p_i} = -\boldsymbol{R_i}^T \boldsymbol{t_i}$.

$$\boldsymbol{I_r} = \arg\min_i ||\boldsymbol{p_i} - \boldsymbol{p_u}||^2, 0 \le i \le K \qquad (1)$$

Utilizing the closest image to render the model creates a sudden transition from one image to another. It also produces a sharp edge between updated vertices and the others (Figure. 4) (left). Selecting more images can achieve smooth transitions with head movement but at a cost; details, such as specularities and light-bursts will be lost (e.g., in the fixed-texture model colored by all images). In our experiment, we select three images to not only preserve the detail but also smoothly switch from different viewpoints (Figure. 4).

Each vertex is mapped to image planes to retrieve their corresponding RGB values. We compute the vector from the

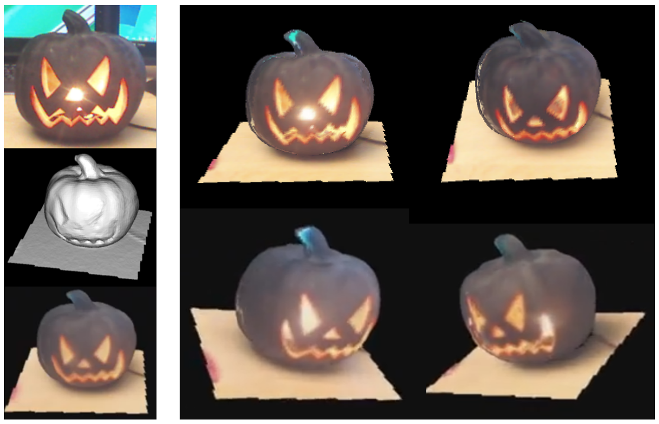

Fig. 5. (top left) An image of a real object captured using a Kinect v1. (middle left) The untextured 3D model. (bottom left) The model with a fixed texture generated from blending the source images (right). The model with view-dependent texture rendered from several viewpoints. Note the flame from the candle within the object.

model center to the HMD position $\boldsymbol{p_u}$. If it intersects the triangle formed by the three selected camera poses $\{\boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p_3}\}$ at $\boldsymbol{p}$, we use the barycentric coordinates to compute the weight.

$$\boldsymbol{p} = w_1\boldsymbol{p_1} + w_2\boldsymbol{p_2} + w_3\boldsymbol{p_3} \tag{2}$$

If the direction ray from the model center does not intersect with the triangle (e.g., the user position is outside the cover area), we use the inverse of euclidean distance as the weight. (i.e., $w_i = \|\boldsymbol{p} - \boldsymbol{p_i}\|_2^{-1}, i \in \{1, 2, 3\}$)

Before combining these values, we must perform a visibility check to detect occlusions. RGB values that fail the visibility check are then discarded (i.e., set the weight to zero). The remaining weights are normalized and the vertex color is updated by the new RGB value.

$$C(v) = w'_1\boldsymbol{c}_1 + w'_2\boldsymbol{c}_2 + w'_3\boldsymbol{c}_3 \tag{3}$$

where $C(v)$ represents the color of vertex v, $w'_i = w_i/(w_1 + w_2 + w_3), i \in \{1, 2, 3\}$, $\boldsymbol{c}_1$, $\boldsymbol{c}_2$, and $\boldsymbol{c}_3$ are the pixel colors retrieved from projecting the vertex onto the chosen images.

4. EXPERIMENTAL RESULTS

We used a Microsoft Kinect v1, which streams VGA resolution (640X480) depth and color images at 30 fps. The color images are captured with fixed exposure and white balancing. The position of specular reflection varies by viewpoint. The color images captured the light-burst effect and accurately reproduced it at run-time (see Figure 5).

Object	vertex	surface	images	color/depth images
Sculpture	208K	406K	108	3210 / 3225
Table	390K	763K	98	2906 / 2918
Chair	255K	495K	111	3299 / 3313
Figurine	99K	197K	96	2843 / 2856

Table 1. Information about the 3D models and images used in different examples.

We used the data from Choi et al. [16] to test our system, as it contains thousands of RGB-D sequences. We selected disparate models, including a female nude sculpture (3887), round table (5648), antique leather chair (5989), and an elderly male figurine (9933), where the number in parentheses is the index of the sequence in [16]. Figure 1 first shows the 3D model generated from KinectFusion without texture optimization (Figure 1 (a)) and with texture optimization [1] (Figure 1 (b)). The textures are identical from all viewing angles. In contrast, virtual objects generated using our real-time view-dependent approach preserves the specularity of the object from different viewpoint (Figure 1 c). It is worth noting that these sequences are captured by individuals who are not experts in computer vision or engineering. The detail of each model is shown in Table. 1.

All the experiments were performed in Unity 5.3 on a Macbook Pro with an Intel i7-4850HQ CPU, Nvidia GeForce GT750M GPU and 16 GB of RAM. Our method can render the models in Figure 1 in 10-15 milliseconds (i.e., 70-90 fps), making it sufficient for real-time high-frequency rendering.

5. CONCLUSION

We propose a novel pipeline for rapidly creating photorealistic virtual reality content with only one consumer-grade RGB-D sensor. Our system automatically generates the geometric model and the camera trajectory of selected color images from a RGB-D sequence. It also generates a texture for the 3D model that changes based on the HMD position in real-time, and is robust enough to extend to viewpoints that were not originally captured. By fusing weighted vertex color from multiple images, we can smoothly transition the texture from one viewpoint to another. We demonstrated that our system will correctly reproduce the appearance of objects captured by individuals without expert knowledge, making it a useful application for real-world 3D scanning.

6. ACKNOWLEDGEMENT

This work is sponsored by the U.S. Army Research Laboratory (ARL) under contract number W911NF-14-D-0005. Statements and opinions expressed and content included do not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

7. REFERENCES

[1] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon, "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*, New York, NY, USA, 2011, UIST '11, pp. 559–568, ACM.

[2] Qian-Yi Zhou and Vladlen Koltun, "Color map optimization for 3d reconstruction with consumer depth cameras," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 155:1–155:10, July 2014.

[3] Paul Debevec, Yizhou Yu, and George Boshokov, "Efficient view-dependent image-based rendering with projective texture-mapping," Tech. Rep., University of California at Berkeley, Berkeley, CA, USA, 1998.

[4] John Bastian, Ben Ward, Rhys Hill, Anton Hengel, and Anthony Dick, "Interactive modelling for ar applications," in *In 2010 IEEE International Symposium on Mixed and Augmented Reality, IEEE*, 2010.

[5] Yuta Nakashima, Yusuke Uno, Norihiko Kawai, Tomokazu Sato, and Naokazu Yokoya, "Ar image generation using view-dependent geometry modification and texture mapping," *Virtual Reality*, vol. 19, no. 2, pp. 83–94, 2015.

[6] Harry Shum and Sing B. Kang, "Review of image-based rendering techniques," in *Proc. SPIE*, 2000, vol. 4067, pp. 2–13.

[7] Marc Levoy and Pat Hanrahan, "Light field rendering," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1996, SIGGRAPH '96, pp. 31–42, ACM.

[8] Mark Bolas, Ashok Kuruvilla, Shravani Chintalapudi, Fernando Rabelo, Vangelis Lympouridis, Christine Barron, Evan Suma, Catalina Matamoros, Cristina Brous, Alicja Jasina, Yawen Zheng, Andrew Jones, Paul Debevec, and David Krum, "Creating near-field vr using stop motion characters and a touch of light-field rendering," in *ACM SIGGRAPH 2015 Posters*, New York, NY, USA, 2015, SIGGRAPH '15, pp. 19:1–19:1, ACM.

[9] J. Thatte, J. B. Boin, H. Lakshman, G. Wetzstein, and B. Girod, "Depth augmented stereo panorama for cinematic virtual reality with focus cues," in *2016 IEEE International Conference on Image Processing (ICIP)*, Sept 2016, pp. 1569–1573.

[10] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen, "Unstructured lumigraph rendering," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 2001, SIGGRAPH '01, pp. 425–432, ACM.

[11] Abe Davis, Marc Levoy, and Fredo Durand, "Unstructured light fields," *Comput. Graph. Forum*, vol. 31, no. 2pt1, pp. 305–314, May 2012.

[12] T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard, and J.B. McDonald, "Kintinuous: Spatially extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, Jul 2012.

[13] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3D mapping with an RGB-D camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, Feb 2014.

[14] K. S. Narayan and P. Abbeel, "Optimized color models for high-quality 3d scanning," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 2503–2510.

[15] Frederique Crete, Thierry Dolmiere, Patricia Ladret, and Marina Nicolas, "The blur effect: perception and estimation with a new no-reference perceptual blur metric," in *Proc. SPIE*, 2007, vol. 6492, pp. 64920I–64920I–11.

[16] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun, "A large dataset of object scans," *arXiv:1602.02481*, 2016.