Dynamic Omnidirectional Texture Synthesis for Photorealistic Virtual Content Creation

Chih-Fan Chen*
Institute for Creative Technologies, University of Southern California

Evan Suma Rosenberg†
University of Minnesota

Abstract

We present a dynamic omnidirectional texture synthesis (DOTS) approach for generating real-time virtual reality content captured using a consumer-grade RGB-D camera. Compared to a single fixed-viewpoint color map, view-dependent texture mapping (VDTM) techniques can reproduce finer detail and replicate dynamic lighting effects that become especially noticeable with head tracking in virtual reality. However, VDTM is very sensitive to errors such as missing data or inaccurate camera pose estimation, both of which are commonplace for objects captured using consumer-grade RGB-D cameras. To overcome these limitations, our proposed optimization can synthesize a high resolution view-dependent texture map for any virtual camera location. Synthetic textures are generated by uniformly sampling a spherical virtual camera set surrounding the virtual object, thereby enabling efficient real-time rendering for all potential viewing directions.

Keywords: virtual reality, view-dependent texture mapping, virtual content creation.

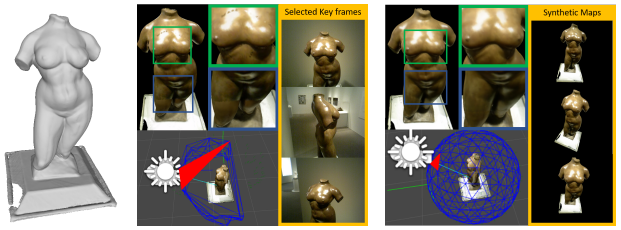
Index Terms: Computing methodologies—Computer Graphics—Graphics systems and interfaces—Virtual reality; Computing methodologies—Computing graphics—Image manipulation—Appearance and texture representations; Computing methodologies—Computer Graphics—Image manipulation—Image-based rendering

1 Introduction

Creating photorealistic virtual reality content has gained more importance with the recent proliferation of head-mounted displays (HMDs). However, manually modeling high-fidelity virtual objects is not only difficult but also time consuming. An alternative is to scan objects in the real world and render their digitized counterparts in the virtual world. Reconstructing 3D geometry using consumer-grade RGB-D cameras has been an extensive research topic and many techniques have been developed [10] with promising results. However, replicating the photorealistic appearance of reconstructed objects is still an open question. Existing methods (e.g., [1, 21]) compute the color of each vertex by averaging the colors from all captured images. Blending colors in this manner results in lower fidelity textures that appear blurry especially for objects with non-Lambertian surfaces (see Figures 8 and 9). Furthermore, this approach also yields textures with fixed lighting that is baked onto the model. These limitations become especially noticeable when viewed in head-tracked virtual reality displays, as the surface illumination (e.g. specular reflections) does not change appearance based on the user's physical movements.

To improve color fidelity, techniques such as View-Dependent Texture Mapping (VDTM) have been introduced [4, 8, 14]. In this approach, the texture is dynamically updated in real-time using a subset of images closest to the current virtual camera position. Although these methods typically result in improved visual quality, the dynamic transition between viewpoints is potentially problematic, especially for objects captured using consumer RGB-D cameras. This is due to the fact that the input sequences often cover only a limited range of viewing directions, and some frames may only partially capture the target object. To avoid the undesired rendering results, either controlled capturing conditions or carefully preselected frames from the input sequences are essential for previous methods. Neither solution is ideal for automating the end-to-end capture-to-rendering process. In this paper, we propose dynamic omnidirection texture synthesis (DOTS). The following outlines the major contributions of this paper:

- DOTS is a novel fully automatic virtual content creation pipeline from capture-to-rendering that supports omnidirectional real-time rendering.
- DOTS computes an optimized high-resolution synthetic texture from a set of low-resolution images with inaccurate camera poses for any particular target viewpoint.
- DOTS is robust to unconstrained capture conditions such as non-uniform camera trajectories, missing coverage, and partial object views.



(a) Geometry (b) Traditional VDTM (c) Our proposed method

Figure 1: (a) The reconstructed model used for different image-based rendering methods. (b) The key frames selected from the original video by traditional VDTM (e.g., [4]) are not uniformly distributed around the 3D model because they are dependent upon the camera trajectory during object capture. Moreover, some frames may only partially capture the target object. Thus, this leads to an irregular triangulation (red) and results undesirable visual artifacts in the synthetic novel view generated from those three images. (c) In contrast, the synthetic maps generated by DOTS cover all potential viewing directions and the triangulation is uniform, resulting in seamless view-dependent textures.

2 Related Work

A wide variety of techniques [15, 18, 19] have been proposed to incorporate 3D representations of physical objects in virtual environments. These methods can be categorized as *image-based*, *model-based*, or a *hybrid* of the two based on the way unobserved viewpoints are represented and visually reproduced.

*e-mail: cfchen@ict.usc.edu
†e-mail: suma@umn.edu

Image-Based Rendering Image-Based rendering (IBR) is a rendering method using photographs without utilizing the geometry of an object. Light field rendering (LFR) [7, 12] can synthesize unseen views if the number of images is sufficiently large to capture all the reflected lights coming from an object. To obtain such an input dataset, LFR requires a well-designed camera array or a programmable turntable. These specialized devices are very expensive and not practical for most users. Furthermore, without a 3D model, it is complicated to edit or interact with the content after capturing.

Model-Based Rendering On the other hand, model-based methods use geometric models with materials such as albedo, specular map, normal map, etc., to represent an object. Color mapping optimization [1,11,21] techniques have been developed to maximize the color (i.e., albedo) agreement between multiple input images. These methods produce higher visual quality for models with Lambertian surfaces. However, averaging all the observed colors of a non-Lambertian object can result in lower fidelity textures that appear blurry (see Figure 8). Moreover, fixed textures are not ideal to represent the dynamic illumination effects of non-Lambertian surfaces.

View-Dependent Texture Mapping View-dependent texture mapping (VDTM) [2, 4, 8, 9, 14] is a hybrid method that combines aspects of model-based and image-based rendering. Given a small set of selected images, it dynamically blends the color maps from different viewing directions to render the model's texture at run-time. Similar to image-based methods, VDTM can achieve more realistic surface color and illumination effects, while simultaneously maintaining the flexibility and interactivity of model-based approaches. However, VDTM is very sensitive to errors such as missing data or inaccurate camera pose estimation. Furthermore, earlier method required manual overhead for adjusting camera poses relative to the model [8]. In later work, structure-from-motion (SfM) and multi-view stereo were utilized to automate this process [2, 14, 17]. SfM estimates the camera poses and reconstructs the point clouds from visual odometry. This photometric reconstruction highly depends on matching the visual appearance between images, and as a result it tends to work poorly on objects with high specularity or repeated patterns. In contrast, RGB-D sensors (e.g., Kinect) can be used to integrate the depth data into a voxel volume which does not require color information [10]. However, the camera trajectories computed while scanning objects are typically prone to error that accumulates over time, resulting in a texture that appears blurry when mapped onto the reconstructed 3D mesh. Several approaches have been proposed to overcome this problem. For example, Hedman et al. [9] used additional high-dynamic range images for rendering and the corresponding depth information was computed from RGB-D camera instead of directly used the RGB-D sequence. Chen et al. [4] used a color mapping optimization method to further refine the computed trajectories for better rendering quality in virtual reality. However, previous VDTM work assumed the original images are well selected to cover all viewing directions and the whole reconstructed object is always in view, which is nonviable in casual scanning performed by non-experts. In other words, the following open problems have not been sufficiently addressed in previous work :

- Covering all potential view directions during object capture using a hand-held RGB-D camera is non-trivial. Furthermore, the spatial and temporal distributions of camera trajectories are non-uniform and often inconsistent (Figure 2).
- The unconstrained camera trajectory typically results in sub-optimal triangulation, which in turn can lead to artifacts such as sharp color discontinuities or erroneous surface illumination (Figures 1(green) and 7).
- The texture generated for a particular viewpoint can have large regions of missing data when the closest images in the input sequence contain only partial views of the captured object. (Figures 1(blue) and 9).

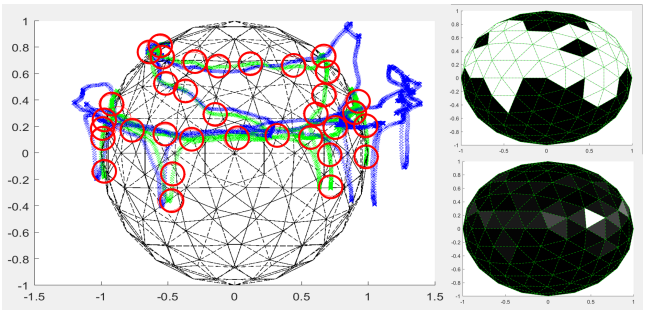


Figure 2: Visualization of a typical hand-held RGB-D capture sequence showing non-uniform spatial and temporal distribution. The blue line shows the estimated trajectory from the input depth stream. The green line shows the trajectory projected from the blue line onto a sphere surrounding the object. Red circles indicate the position of the selected key frames for VDTM. The spatial regions covered by the camera's trajectory are shown in the upper right image. Time spent in each region is displayed in the lower left image with brighter shades indicating longer duration.

We developed DOTS in order to overcome all three of the above limitations, which are crucial but were not considered in the previous VDTM methods [4, 9, 14]. The proposed approach can generate a set of complete synthetic texture maps for omnidirectional viewing, and is reasonably robust to inconsistent camera trajectories, partial views, and missing coverage during object capture.

3 Overview and Preprocessing

Overall Process The system pipeline is shown in Figure 3. Given an RGB-D video sequence, the geometry is first reconstructed from the original depth stream. A set of key frames is selected from the entire color stream, and the camera poses are optimized using the color information. Next, a virtual sphere is defined to cover the entire 3D model and the virtual camera poses are uniformly sampled and triangulated on the sphere's surface (Sec. 4.1). For each virtual camera pose, the corresponding texture is synthesized from several local frames and the pre-generated global texture (Sec. 4.5). At run-time, the user viewpoint provided by a head-tracked virtual reality display is used for selecting the synthetic maps to render the model in real-time (Sec. 4.6).

Geometry Reconstruction Any 3D reconstruction method can be used to obtain the geometry model with a triangular mesh representation. Similar to other papers [1, 4, 11, 21] that focus on the texture mapping for objects captured using handheld RGB-D cameras, we use Kinect Fusion [10] to construct the 3D model M from the depth sequences. The camera trajectory of the sequence is roughly estimated (the blue line in Figure 2) and can be used for the texture generation.

Spatial Key Frame Selection Using all frames I of the input video for generating the global texture is inefficient. We chose spatial key frame selection [4, 11, 16] to maximize the variation of viewing angles of the 3D model (red circles in Figure 2). The selected N key frames, $G = \{g_1, g_2, \cdots, g_N\} \in I$, with initial estimated camera poses, $T_G = \{t_{g_1}, t_{g_2}, \cdots, t_{g_N}\}$, are used for camera pose optimization.

Camera Pose Optimization The raw camera poses from the trajectory obtained by Kinect Fusion [10] are not accurate enough for mapping the texture onto the model correctly. Therefore, we use color mapping optimization [21] to obtain the images G and

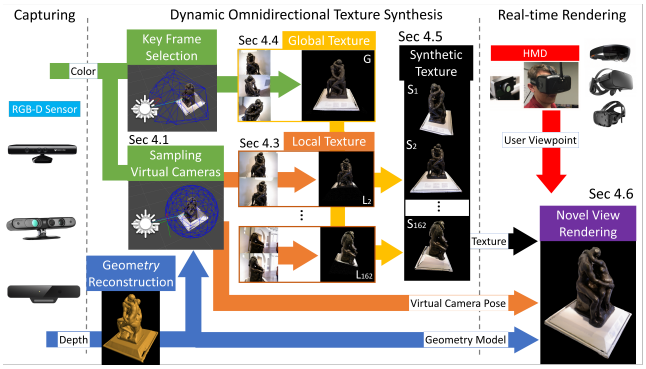


Figure 3: Overview of the DOTS content creation pipeline. Color and depth image streams are captured from a single RGB-D camera. The geometry is reconstructed from depth information and is then used to uniformly sample a set of virtual camera poses surrounding the virtual object. For each camera pose, a synthetic texture is blended from the global texture and local color images captured near the current location. The synthetic textures are then used to dynamically render the object in real-time based on the user's current viewpoint in virtual reality.

their optimized camera poses T_G as anchor images for our synthetic image generation.

4 Synthetic Texture Generation and Rendering

Our objective is to replicate a high-fidelity view-dependent model from a given low-resolution RGBD video. To achieve this goal, we need accurate camera poses and high-quality textures for real-time texture mapping. Instead of selecting VGA resolution images directly from the original video, we set virtual cameras surrounding the reconstructed geometry in 3D and generate the synthetic texture for each virtual camera.

4.1 Sampling Virtual Camera

We uniformly sample virtual cameras surrounding the reconstructed geometry in 3D and set the size of the sphere, formed by the virtual cameras, large enough to cover the entire object in each synthesized virtual view. The sphere radius used in each virtual object is reported in Table 1. Note that unlike spatial and temporal key frame selection, the number of synthesized textures is independent of the length of input video and the camera trajectory. As shown in Figure 2, we sampled 162 virtual cameras $V = \{v_1, v_2, \cdots, v_{162}\}$ with known virtual camera poses $T_v = \{t_{v_1}, t_{v_2}, \cdots, t_{v_{162}}\}$ forming a total of 320 triangles to guarantee the angle between any viewing direction and its closest virtual camera is always smaller than 15°. Although the number of synthesized textures is 1.5 times larger than the number used in VDTM (see Table 1), DOTS covers all viewing directions, while VDTM covers only 18 % - 20 % of the entire sphere (e.g, the vertices on the sphere versus the red circles in Figure 2).

4.2 Weighting Function

For each virtual camera v_j with known camera pose t_{v_j}, we aim to generate one higher resolution texture s_j from several low-resolution frames. Unlike the method in [13], which assumes the camera poses of each frame is fixed, our objective function also optimizes the camera poses. To generate s_j, all images I from the original sequence are weighted and sorted based on their uniqueness with respect to v_j. The uniqueness is determined by the texture quality of each image from the original video and the similarity of the camera pose of each image with the virtual camera pose t_{v_j}. The blurriness metric from Crete et al [6] is used to evaluate the image quality. The distance and the angle between the camera of each frame with v_j are used for similarity evaluation. The overall weighting function of each image $i \in I$ with respect to the virtual camera v_j is defined as follows:

$$w_j(i) = \max\left(\frac{\cos\theta_i * b_i}{d_i^2}, \delta\right) \qquad \forall i \in I \tag{1}$$

where θ_i and d_i are the angle and the euclidean distance between the camera pose of image i and the virtual camera pose t_{v_j} and b_i represents the blurriness of image i. δ is a small number to prevent a non-positive value if θ_i is larger than 90°. In our case, we set $\delta = 10^{-3}$. Images with higher weights are chosen for optimization.

4.3 Local Texture Generation

For each virtual camera $v_j \in V$, we sorted the weights and selected a set of images L_j with highest weights from the original sequence, where $L_j = \{l_1, l_2, \cdots, l_K\} \in I$ with initial estimated camera poses $T_{L_j} = \{t_{l_1}, t_{l_2}, \cdots, t_{l_K}\}$. We set $K = 20$ in our work.

Because generating each synthetic texture is independent, we simplify the notation by re-annotating $\{v_j, L_j, T_j, w_j\}$ to $\{v, L, T, w\}$ respectively. We aim to simultaneously optimize the camera poses T_L and the synthetic texture s by minimizing the error between s and every local texture generated from image $l \in L$ using texture synthesis function Ψ (e.g., the two green rectangles in Figure 4 (a)). Thus, the error function is defined as follows:

$$E(T_L, s) = \sum_{l \in L} w'_l * \sum_{x \in X_l} (s(x) - \Psi(i_l, t_l, M, t_s, x))^2 \tag{2}$$

where Ψ renders the model M by image i and projects the rendered model to a known camera pose t_j. X_l are the set of vertices that are visible in both virtual view and the camera view of l, and $w'_l = w(l)/\sum_{l \in L} w(l)$ are the normalized weights of each image in L.

4.4 Global Texture Generation

As shown in Figure 4 (b), using only the local texture is insufficient for the entire model because the closest images may have only a partial view of the object. Thus, the selected key frames G in Section 3 is used to fill the missing information. The corresponding global texture of each key frame can be generated using Ψ. We then introduce a weighted global texture to our objective function, but keep T_G unchanged since it is already optimized. By fixing the camera poses T_G, Ψ can be further reduced to a global texture h_g with respect to the virtual camera pose t_s. The error function of global texture can be written as follows:

$$E_G(s) = \sum_{g \in G} w'_g * \sum_{x \in X_g} (s(x) - h_g(x))^2 \tag{3}$$

Where $w'_g = w(g)/\sum_{g \in G} w(g)$ and h_g is the global texture rendered from image $g \in G$. Two examples of global texture are shown in the red rectangles of Figure 4 (a).

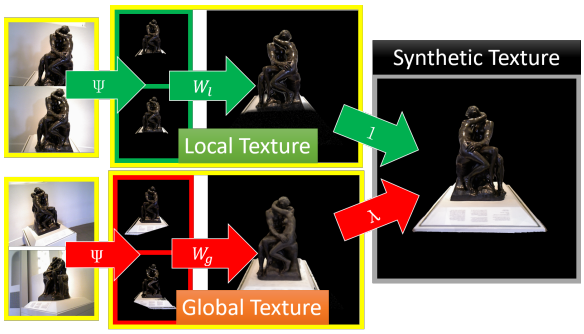
4.5 Synthetic Texture Generation

To combine the global and local error function, λ defines the weights of the global texture term. Eq. 2 can be rewritten as follows:

$$\begin{aligned} E(T_L, s) = \;& E_L + \lambda * E_G \\ =\; & \sum_{l \in L} w''_l * \sum_{x \in X_l} (s(x) - \Psi(i_l, t_l, M, t_s, x))^2 \\ & + \lambda \sum_{g \in G} w''_g * \sum_{x \in X_g} (s(x) - h_g(x))^2 \end{aligned} \tag{4}$$

where $w''_l = w(l)/(\sum_{l \in L} w(l) + \lambda \sum_{g \in G} w(g))$, $w''_g = w(g)/(\sum_{l \in L} w(l) + \lambda \sum_{g \in G} w(g))$.

Note that to avoid re-estimating the camera poses of the global texture, they are excluded from the selection of local textures. The objective function is not a linear least square function because of the texture synthesis function Ψ, thus the optimization process iteratively minimized the error function with respect to each $t_l \in T_L$ and s. Note that the weights of the global textures are comparatively smaller than the weights of local textures. We further decrease it by setting λ to 0.1. The regions covered by local texture will not be affected much, but the uncovered area with the missing texture can be filled in with the weighted global texture. As shown in Figure 4 (c), the optimized textures S not only maximize the color agreement of local textures but also seamlessly blends the global textures. Only the derived high-resolution synthetic textures S and the corresponding virtual camera poses T_S are used for real-time image-based rendering.



(a) overview of texture blending

(b) local texture only

(c) local and global textures

Figure 4: (a) The synthetic texture is blended from the local texture (green rectangle) and the global texture (red rectangle). (b) It was not possible to completely cover the entire 3D model using only the local texture information. (c) By blending with the global texture, complete coverage is obtained.

Choice of Texture Resolution In DOTS, there is no constrain on the resolution of synthetic texture and can be set to any arbitrary positive number. We set the resolution as 2048×2048, which has two advantages. First, by blending several VGA resolution (640×480) images with VGA, our method achieves higher resolution. A comparison of VGA resolution versus high resolution is shown in Figure 5. The resolution of the blue rectangle of the synthetic texture is approximately 2 times that of the red rectangle of the original image. Second, to achieve better performance, most game engines and graphics APIs require the texture resolution to be power-of-two. Thus, when importing the texture, the selected frames from original video are either being stretched or padded to fulfill the requirement, while the generated synthetic textures remain the same.

4.6 Real-time Image-based Rendering

At run-time, the HMD pose is provided by the Oculus Rift CV1 and two external Oculus Sensors. The traditional VDTM method computes the euclidean distance between the user's head position and all camera poses and then selects the closest images for rendering. In contrast, DOTS systematically samples all spherical virtual camera poses surrounding the virtual object. Thus, the vector from the center of the model to the HMD position only intersects with one

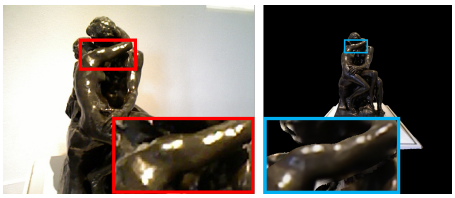


Figure 5: Comparison of an input image captured from the RGB-D camera (left) with a high-resolution synthetic texture generated using the proposed method (right).

Figure 6: Three virtual objects used for comparison.

triangle mesh of the virtual sphere surrounding the model (e.g., Figure 1(c) bottom-left). Barycentric coordinates are used to compute the weight and blend the color from the three synthesized textures $\{s_1, s_2, s_3\}$ of the intersected triangle to generate the novel view.

5 Results

5.1 Data Sets

The dataset [5] contains thousands of RGB-D sequences captured by Primesense. The raw color and depth are not synchronized, so we assigned the color images to the depth images with the smallest time-stamp difference. Since the streams are both 30 fps, the shifting error is small and can also be handled by the camera pose optimization. We tested our system with three different models: Torso of Elevation, the Kiss by Rodin, and an antique leather chair (corresponding to IDs 3887, 4252 and 5989 in the database respectively). The detail of each object is shown at Table 1 and some example images are shown in Figure 6. The vertex and surface is reconstructed using KinectFusion [10], and the key frames are selected from the color/depth stream. The decimated model of the sculpture is used in the second row of Figure 8.

5.2 Visual Analysis

Although there are many proposed VDTM methods such as [2, 9, 14] or other IBR methods [3, 20] can achieve real-time in AR, the frame rate is still much slower than the required frame rate (i.e., 90 fps) for real-time VR applications. To ensure a realistic and comfortable experience in VR, we compare our DOTS with a previous VDTM [4] approach that can both render the model in under 8 milliseconds (i.e., more than 125 fps). In Figure 1, DOTS systematically sampled spherical virtual camera set surrounding the target object, while the selected key frames in VDTM are unstructured. For both methods, three images from the highlighted triangles are used to render the model. Because the selected virtual cameras for

Object	vertex	surface	key frames	color/depth
Torso of elevation (decimated)	208K 1.3K	406K 2.5K	101	3210 / 3225
The Kiss	280K	544K	116	3989 / 4007
Chair	255K	495K	98	3299 / 3313

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

DOTS have similar viewing directions, the viewpoint generated by blending the three corresponding synthetic textures exhibits fewer artifacts compared to VDTM, which uses the three closest images from original capture sequence. Moreover, the synthetic textures provide omnidirectional coverage of the virtual object, while the selected key frames in VDTM sometimes only partially capture the target object. Because of this missing texture information, VDTM can only synthesize a partial unobserved view, thereby resulting in sharp texture discontinuities (i.e. seams), while DOTS can render the model with such artifacts. Similar observations can be made among all four virtual objects. As shown in Figure 9, the fixed texture method (red rectangle) results in a lower fidelity (blurry) appearance. Although the model rendered using VDTM can achieve a more photorealistic visual appearance (leftmost in blue rectangle) than fixed texture method, the region of high-fidelity viewpoints is limited. Due to the unconstrained capture process, this region varies between different objects and is highly dependent on the specific motions of the hand-held RGB-D camera. Texture defects and unnatural changes between viewpoints are not pleasant during virtual reality experiences where user freedom is encouraged and the view direction cannot be predicted in advance. However, in contrast to VDTM, DOTS generates omnidirectional synthetic textures that produce visually reasonable results even under such conditions.

It is worth noting that DOTS can generate good appearance even if the HMD position is significant deviates from the captured trajectory, while VDTM results unnatural specularity highlights and noticeable artifacts. For example, the top views of three different objects rendered by VDTM (Figure 7 left) have severe artifacts because there is a lack of data around the HMD position, the closest images of VDTM are selected from far away in both distance and the viewing angle. The texture used to render the female sculpture comes from both front views and side views, resulting in inconsistent highlights on the shoulder and the chest. Several texture edges on the base of The Kiss and the back of the chair is observed because of the unstructured triangulation.

(a) Traditional VDTM [4] (b) DOTS

Figure 7: Rendering results from viewpoints that were not covered during object capture (e.g. a top-down view of the virtual objects). (a) The model is rendered incorrectly by VDTM. For example, the specularity highlights on the female sculpture's shoulder and chest are conflicting with each other. Blending textures from distant camera poses results in noticeable color discontinuity on the base of The Kiss and the back of the chair. (b) The model rendered by DOTS presents reasonable visual quality even though the viewpoint was never observed in the capture sequence.

In Figure 8, we decimated the reconstructed female sculpture from 208K vertices to 1.3K vertices (i.e., 0.6% of the original). The visual fidelity of the model decreased dramatically when rendered using the fixed texture. The texture mapping failed for VDTM, resulting in extremely undesirable artifacts. However, when rendered using DOTS, the reduced polygon mesh retained a high-fidelity appearance that was nearly indistinguishable from the original model.

Figure 8: (top) A 3D model rendered using per-vertex color. (bottom) A reduced polygon mesh rendered using a UV map. From left to right: the untextured geometric model, fixed texture mapping [21], VDTM [4], and DOTS.

5.3 Time and Space Complexity Comparison

The time complexity of VDTM (i.e., the camera pose optimization) is $O(vN)$, where v is the vertex number and N is the number of key frames. For each synthetic view generation, it takes additional computation $O(vK)$, where K is the selected local images. In our experiments, it takes 2 to 3 hours for camera pose optimization and another 2 to 3 hours to generate all 162 synthetic textures on a Macbook Pro with an Intel i7-4850HQ CPU, Nvidia GeForce GT750M GPU and 16 GB of RAM. The space complexity for VDTM is based on the number of key frames, while the complexity for DOTS is based on the desired resolution of the synthetic texture and the number of synthetic views.

6 Conclusion and Future Work

We present dynamic omnidirectional texture synthesis (DOTS), a novel approach for rendering virtual reality content captured using a consumer-grade RGB-D camera. The proposed objective function is used to generate optimized synthetic textures for real-time free-viewpoint rendering. The capture-to-rendering process is fully automated and does not require any expert knowledge or human effort. Visual comparison confirmed that DOTS can handle more extreme cases such as viewing perspectives that were not directly covered during object capture.

Limitations and Future Work Due to the nature of image-based rendering techniques, real world illumination is baked into the texture. During object capture, a controlled lighting environment and fixed white balance and exposure of the camera is expected. Another limitation is synthesizing accurate specular reflections. Since specular reflections have a strong non-local effect and are strongly related to the geometry and lighting conditions, the synthetic views created from interpolating between color images may have visual artifacts. However, this limitation exists in all image-based rendering methods and is beyond the scope of this paper. In order to estimate the surface illumination properties of the reconstructed virtual object, we plan to investigate methods for separating the texture into diffuse and specular maps. Furthermore, replacing the uniform sampling with an adaptive sampling algorithm based on either the geometry of the object or the area of interest (e.g., sampling more virtual cameras in front of the reconstructed object) could further improve the transition smoothness.

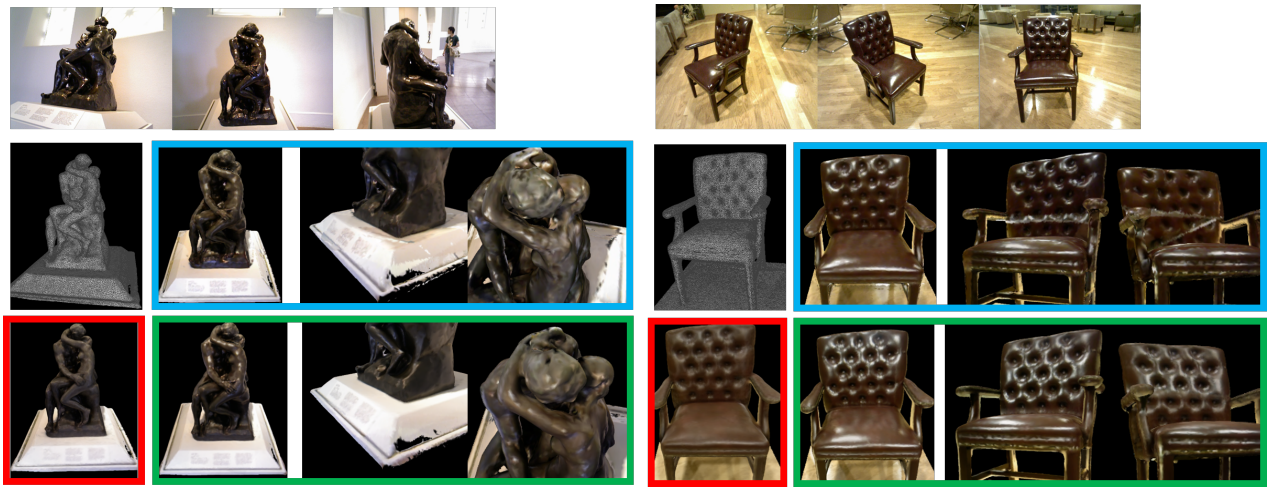


Figure 9: Appearance comparison results for each virtual object used in our paper. The top row shows three example images from original video. The second row are the geometry model and the results from VDTM [4] (in blue rectangle). Note that although VDTM can achieve good visual quality (lesfmost), the region of high-fidelity viewpoints is limited. The third row is the fixed texture [21] (in red rectangle) and the results of DOTS (in green rectangle). In contrast to VDTM, DOTS generates omnidirectional synthetic texture maps that produce visually reasonable results.

ACKNOWLEDGMENTS

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.

REFERENCES

[1] S. Bi, N. K. Kalantari, and R. Ramamoorthi. Patch-based optimization for image-based texture mapping. *ACM Trans. Graph.*, 36(4):106:1–106:11, jul 2017. doi: 10.1145/3072959.3073610

[2] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pp. 425–432. ACM, New York, NY, USA, 2001. doi: 10.1145/383259.383309

[3] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph.*, 32(3):30:1–30:12, jul 2013. doi: 10.1145/2487228.2487238

[4] C. F. Chen, M. Bolas, and E. S. Rosenberg. View-dependent virtual reality content from rgb-d images. In *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 2931–2935, Sept 2017. doi: 10.1109/ICIP.2017.8296819

[5] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun. A large dataset of object scans. *arXiv:1602.02481*, 2016.

[6] F. Crete, T. Dolmiere, P. Ladret, and M. Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. *Proc. SPIE*, 6492:64920I–64920I–11, 2007. doi: 10.1117/12.702790

[7] A. Davis, M. Levoy, and F. Durand. Unstructured light fields. *Comput. Graph. Forum*, 31(2pt1):305–314, may 2012. doi: 10.1111/j.1467-8659.2012.03009.x

[8] P. Debevec, Y. Yu, and G. Boshokov. Efficient view-dependent image-based rendering with projective texture-mapping. Technical report, University of California at Berkeley, Berkeley, CA, USA, 1998.

[9] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow. Scalable inside-out image-based rendering. *ACM Trans. Graph.*, 35(6):231:1–231:11, nov 2016. doi: 10.1145/2980179.2982420

[10] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. 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*, UIST '11, pp. 559–568. ACM, New York, NY, USA, 2011. doi: 10.1145/2047196.2047270

[11] J. Jeon, Y. Jung, H. Kim, and S. Lee. Texture map generation for 3d reconstructed scenes. *The Visual Computer*, 32(6):955–965, Jun 2016. doi: 10.1007/s00371-016-1249-5

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

[13] R. Maier, J. Stckler, and D. Cremers. Super-resolution keyframe fusion for 3d modeling with high-quality textures. In *International Conference on 3D Vision*, pp. 536–544, Oct 2015. doi: 10.1109/3DV.2015.66

[14] Y. Nakashima, F. Okura, N. Kawai, H. Kawasaki, A. Blanco, and K. Ikeuchi. Realtime novel view synthesis with eigen-texture regression. *Proceedings of British Machine Vision Conference)*, 2017.

[15] F. Remondino and S. El-Hakim. Image-based 3d modelling: A review. *The Photogrammetric Record*, 21(115):269–291, 2006. doi: 10.1111/j.1477-9730.2006.00383.x

[16] T. Richter-Trummer, D. Kalkofen, J. Park, and D. Schmalstieg. Instant mixed reality lighting from casual scanning. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 27–36, Sept 2016. doi: 10.1109/ISMAR.2016.18

[17] T. Rongsirigul, Y. Nakashima, T. Sato, and N. Yokoya. Novel view synthesis with light-weight view-dependent texture mapping for a stereoscopic hmd. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 703–708, July 2017. doi: 10.1109/ICME.2017.8019417

[18] H. Shum and S. B. Kang. A review of image-based rendering techniques. *Proceedings of IEEE/SPIE Visual Communications and Image Processing (VCIP)*, 4067:2–13, 2000. doi: 10.1117/12.386541

[19] M. Waechter, M. Beljan, S. Fuhrmann, N. Moehrle, J. Kopf, and M. Goesele. Virtual rephotography: Novel view prediction error for 3d reconstruction. *ACM Trans. Graph.*, 36(1):8:1–8:11, Jan 2017. doi: 10.1145/2999533

[20] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle. Surface light fields for 3d photography. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pp. 287–296. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000. doi: 10.1145/344779.344925

[21] Q.-Y. Zhou and V. Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Trans. Graph.*, 33(4):155:1–155:10, July 2014. doi: 10.1145/2601097.2601134