Efficient Use of Textured 3D Model for Pre-observation-based Diminished Reality

Shohei Mori*, Fumihisa Shibata, Asako Kimura, and Hideyuki Tamura
Ritsumeikan University

Figure 1: Typical results of the proposed pre-observation-based diminished reality system. After the dense coverage of view-dependent images of the scene, the object, i.e., a cat placed in front of the stuffed bear (left), is seamlessly deleted (middle). The proposed system allows the user to move in six degrees-of-freedom under dynamic lighting during the deletion (right).

ABSTRACT

Diminished reality (DR) deletes or diminishes undesirable objects from the perceived environments. We present a pre-observation-based DR (POB-DR) framework that uses a textured 3D model (T-3DM) of a scene for efficiently deleting undesirable objects. The proposed framework and T-3DM data structure enable geometric and photometric registration that allow the user to move in six degrees-of-freedom (6DoF) under dynamic lighting during the deletion process. To accomplish these tasks, we allow the user to pre-observe backgrounds to be occluded similar to existing POB-DR approaches and preserve hundreds of view-dependent images and triangle fans as a T-3DM. The proposed system effectively uses the T-3DM for all of processes to fill in the target region in the proposed deletion scheme. The results of our experiments demonstrate that the proposed system works in unknown 3D scenes and can handle rapid and drastic 6DoF camera motion and dynamic illumination changes.

Keywords: Diminished reality, mixed/augmented reality, image-based rendering, tracking, color correction.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information System—Artificial, augmented, and virtual realities

1 INTRODUCTION

Augmented reality and mixed reality (AR & MR) seamlessly merge reality and virtuality to enhance the user perception of reality. Diminished reality (DR) visually deletes or diminishes undesirable objects from the perceived environments (**Figure 1**). DR is considered a concept contrary to AR/MR; thus, a combination of these concepts is expected to lead to an unconstrained reality [1][2]. In most DR studies, the objects to be diminished are already determined as targets of interest (e.g., pedestrians [13], devices [17], walls [9], and buildings [3]); therefore, such studies implicitly demonstrate how to diminish the target objects in specific situations. If we view problems to be solved by DR from another perspective, we can consider that such visual object removal scheme corresponds to a type of "undo" function because it allows the user to partially regain past views that are considered better states than the current state.

In other words, undoing an action to place something in the real world corresponds to DR. This interpretation and its implementation are especially effective when the user "virtually undoes" an action "done" by a third party, e.g., the ideal scenery for filming is lost because of newly placed stage sets, a new building occludes a landscape, a working signboard is placed in front of a direction board, and an old building is reconstructed.

To regain past views in the current one, it is necessary to describe views beforehand in a format that can be recovered in the current view. This can be performed using DR that considers pre-observation. According to existing attempts [11][12][13][17], we consider that the performance of pre-observation-based DR depends on the complexity of backgrounds and camera movement. Thus, allowing multiple degrees of freedom makes DR challenging. In addition, pre-observation-based DR systems must handle lighting changes because views observed in the past must be recovered in the current view. Each of these problems has been actively researched, and several methods have been proposed to address these problems, such as six degrees-of-freedom (6DoF) camera pose estimation, arbitrary viewpoint image generation, and optical characteristic and lighting estimation. To date, there have been few implementations of DR systems compared to AR/MR systems; however, both technologies are considered important.

In this study, we present a pre-observation-based DR (POB-DR) framework (**Figure 2**). Owing to the framework that effectively and efficiently mediates a pre-fetched textured 3D model (T-3DM), the proposed system allows comparative viewpoint and dynamic lighting changes to existing approaches.

* mori@rm.is.ritsumei.ac.jp

LEAVE 0.5 INCH SPACE AT BOTTOM OF LEFT COLUMN ON FIRST PAGE FOR COPYRIGHT BLOCK

978-1-4673-8471-1/15 $31.00 © 2015 IEEE
DOI 10.1109/ISMARW.2015.16

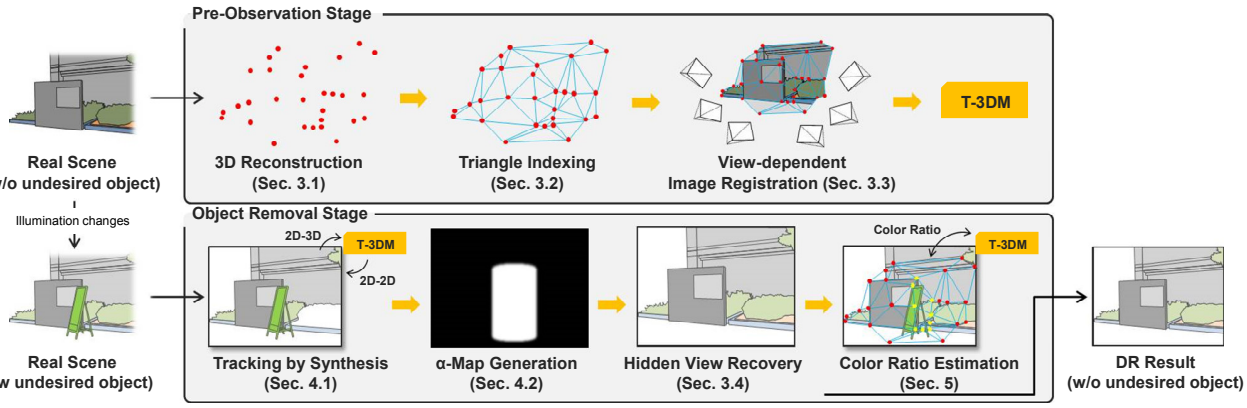

Figure 2: Workflow of the proposed method

- An efficient framework for POB-DR
  Geometric and photometric registration using prefetched T-3DM: Recyclic use of a T-3DM for camera trackcing, arbitrary viewpoint hidden area recovery, and its color correction
- An efficient T-3DM data structure for the tasks

The remainder of this paper is organized as follows. Section 2 describes related work and compares previous studies to the proposed method. Section 3 describes the components of a T-3DM and the steps required to construct a T-3DM. Given a T-3DM, we describe the proposed geometric registration method in Section 4, and we introduce the proposed photometric registration method in Section 5. In Section 6, we demonstrate the proposed system and discuss results. Finally, we summarize and discuss future work in Section 7.

2 Related Work

DR is accomplished by filling in a region of interest (ROI) with a recovered image of backgrounds occluded by an undesirable object. Existing DR approaches are categorized into image inpainting- and observation-based approaches.

2.1 Inpainting-based DR

When hidden backgrounds cannot be observed (e.g., undesirable objects are fixed onto the ground or walls), hidden regions are estimated by the surrounding areas and filled with plausible results. Inpainting-based DR is an image inpainting or video inpainting method designed for real-time operation [4][5][6][7]. Herling and Broll proposed a real-time image inpainting algorithm based on appearance and spatial cost functions, heuristic optimization of the cost functions, and multi-resolution optimization [5]. Kawai *et al*. improved real-time performance using a multi-threading function, i.e., two threads are assigned independently to computationally expensive image inpainting and other implementations [6]. They also presented a model for global and local luminance changes to modify their image inpainting results. Inpainting-based DR is essentially designed for on-screen object removal; therefore, it is difficult to apply such methods to 3D scenes. At present, they can be applied to several planes [7].

2.2 Observation-based DR

The objective of observation-based DR is semantic recovery of backgrounds hidden by undesirable objects by observing the backgrounds in real time using multiple cameras or before the undesirable objects are placed in the environment. Observation-based DR can be separated into real-time observation-based DR (ROB-DR) and pre-observation-based DR (POB-DR). ROB-DR is an approach that can handle dynamic backgrounds. In ROB-DR, multiple cameras are placed in an environment for real-time observation of the occluded backgrounds, and the results are warped to the user's viewpoint to visualize the occluded region. Assuming they are appropriately synchronized or calibrated beforehand, camera synchronization problems and the use of different optical systems are generally ignored. In addition, backgrounds are assumed to be simple because there are practical limits to the density of camera assignment in an environment.

Enomoto and Saito assumed multiple users with handheld cameras tracked with AR markers [8]. In their system, each user partially observes hidden regions at different viewpoints and shares their results assuming that a region hidden from a user is observable by the other users. Barnum *et al*. presented a method for see-through walls. To accomplish this task, they use calibrated surveillance cameras to observe an occluded area, and a hidden area can be separated into a planar foreground and planar background [9]. Jarusirisawad *et al*. accomplish ROB-DR in a 3D scene using multiple cameras and a plane sweep algorithm by excluding occluding objects from the predefined projective grid space [10]. Zokai *et al*. proposed a paraperspective projection model to handle 3D backgrounds using multiple cameras [11]. While this method can handle 3D scenes, occluded surfaces are assumed to be Lambertian and visible from most of the cameras to find correct matches. Although these methods can handle dynamic backgrounds, practical systems tend to be quite complicated, and it is difficult to accomplish high quality object removal.

On the other hand, POB-DR methods can achieve high-quality object removal under the assumption that the areas to be occluded can be observed in advance. In this case, the hidden backgrounds can be modeled precisely, and the necessary data is organized beforehand. Although POB-DR loses the capability to handle dynamic backgrounds, such methods tend to be lightweight and offer high-quality results. Even if the backgrounds are geometrically static, lighting conditions will change according to time. Consequently, a photometric registration scheme is necessary for POB-DR. Even though these are universal problems for POB-DR, many POB-DR systems handle only static scenes under static lighting conditions because they assume indoor scenarios or the object removal is performed immediately after pre-observation.

Lepetit *et al*. proposed a method to remove an undesirable object from a video recorded using a moving camera, assuming that the occluded area is visible in different frames [12]. Their system tracks and segments the undesirable object using a semi-interactive method and fills the occluded region with an image

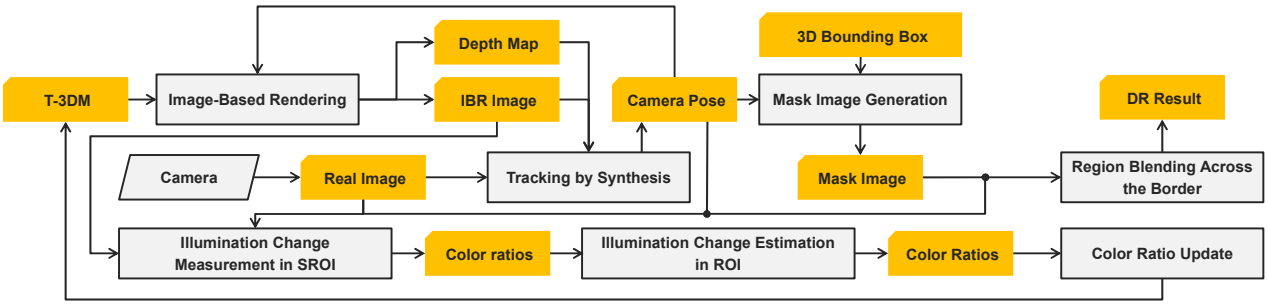

Figure 3: Data flow of the proposed method. Note that it recycles IBR image and corresponding depth map (z-buffer) both for a geometric and photometric registration.

patch warped from a different frame in which the occluded region is fully visible. Li *et al*. proposed a framework to use large photo collections on the Internet to eliminate undesirable objects in outdoor scenes [13]. Their framework includes the selection of appropriate images from a collection using normalized cross correlation criteria and mean-value coordinate (MVC) blending to synthesize an image. However, such approaches essentially assume Lambertian scene, such as stone buildings or statues, distant backgrounds that do not change in perspective, or limited camera motion.

These practical problems in POB-DR will be mitigated by image-based rendering (IBR) approaches, e.g., plenoptic modeling known as light field rendering [14], lumigraph rendering [15], and view-dependent texture mapping [16]. The proposed approach is closely related to that of Cosco *et al*., which uses an IBR approach to recover hidden views to delete a haptic device placed on a specular tabletop [17]. However, the proposed method differs in that we do not use AR markers to estimate camera pose. Thus, such markers do not remain in the final results. In addition, the proposed system considers two additional factors for calculating IBR blending weight to obtain more natural results. The proposed system does not require users to input geometric information of the scene manually and allows illumination changes after pre-observation.

We claim that existing POB-DR approaches tend to partially ignore the abovementioned universal problems in POB-DR because they are intended to perform well for very specific scenarios. In contrast, we do not assume specific scenarios and attempt to build a framework to solve such problems. The proposed system provides high-quality hidden view recovery based on IBR and is designed to use the IBR results to track a scene and measure color tone changes (**Figure 3**). Since the IBR results are used for the multiple purposes, the proposed framework is efficient in terms of data redundancy when compared to cases that employ a camera tracking and a hidden view recovery scheme independently.

3 T-3DM Construction and Rendering

First, a scene is described in the form of a T-3DM in the pre-observation stage to render it through our IBR pipeline. A T-3DM is constructed from multiviewpoint images of a target scene. A T-3DM has the following components.

- M view-dependent images
- N triangle fans and K triangles
- N RGB color ratio vectors

A triangle fan is a set of triangles having a shared vertex that is connected to neighboring vertices. The IBR pipeline treats each view-dependent image as a bundle of light rays and treats triangle fans as a geometric proxy. This section describes the construction of a T-3DM and the IBR pipeline to obtain an arbitrary viewpoint image (**Figure 4**). Note that RGB color ratio vectors are described in Section 5.

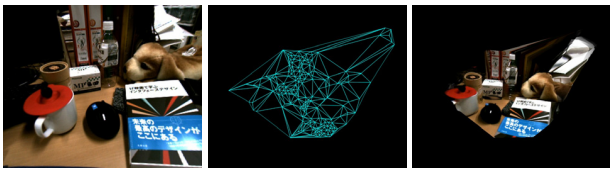

Figure 4: Examples of T-3DM. From left to right, real scene: geometry proxy (3D polygon meshes), and IBR rendering result.

3.1 3D Reconstruction

First, we abstract the structure of an unknown 3D scene as a 3D point cloud including N 3D points. Note that we consider that any 3D reconstruction approaches will work for building a 3D point set; however, the number of points will affect the subsequent processing time and the quality of IBR because the proposed IBR pipeline renders a scene using polygon meshes, i.e., triangle fans.

In our implementation, we use a simple stereo technique to simplify the following triangle indexing. When the system is started, the user waves a calibrated handheld camera. The system then fetches continuous frames during the motion and Harris corners [18] detected in the initial frame are tracked to the last frame with a Lucas-Kanade (LK) tracker [19] throughout the frames. Tracked points with large photometric error are discarded during corner tracking. Given the 2D–2D correspondences in the initial and the last frame, an eight-point algorithm [20] with random sample consensus (RANSAC) [21] outlier removal can estimate an essential matrix and triangulate N Harris corners. Consequently, in this implementation, the T-3DM initially contains two view-dependent images.

3.2 Triangle Indexing

To fill in spaces between 3D points, they are connected to its surrounding vertices to construct 3D polygon meshes. We construct 3D polygons as triangle fans both for efficient IBR on graphics hardware and photometric registration as described in Sections 3.4 and 5 respectively. We also preserve the indices of triangles to generate a screen-space z-buffer to handle occlusion checks (Section 3.4).

In our implementation, the system first connects N Harris corners in the initial frame of the 3D reconstruction step using 2D Delaunay triangulation. Then, each triangle is searched to find triangles that share a vertex and combined as N triangle fans. Since each Harris corner has a 2D–3D correspondence, the 2D triangle fans correspond to the 3D triangle fans.

3.3 View-dependent Image Registration

After triangle indexing, the system begins to track the scene using feature point-based camera pose estimation, which is described in Section 4.1. The system is then ready for insertion of view-dependent images. To avoid troublesome key binding recording, the system inserts images automatically based on criteria similar to those of key frame insertion [22]. Note that tracking quality must be good, and the camera must be a minimum distance from the nearest camera already in the T-3DM. The tracking quality is considered high when the standard deviation and the mean value of projection errors are sufficiently low. In addition, the camera movement must be slow to avoid inserting unclear images due to motion blur.

3.4 T-3DM Rendering as Hidden View Recovery

Blending weight update: The IBR pipeline uses a GPU rendering pipeline to draw the T-3DM in real time. A triangle fan is texture mapped with the top k weighted cameras. Based on the literature [23], our IBR pipeline uses three types of weight to calculate the final weight to draw a triangle fan. Each view-dependent image is projected onto the triangle fan using projective texture mapping [24] for efficiency. Finally, every triangle fan in the current view is compounded using alpha blending on a screen. Equations (1) to (3) represent angular weight $w_{\mathrm{ang}}(i)$, resolution weight $w_{\mathrm{ang}}(i)$, and the field of view weight $w_{\mathrm{ang}}(i)$ of the i^{th} camera c_i respectively. Equation (4) gives the final weight $w(i)$ of the c_i.

$$w_{\mathrm{ang}}(i) = \mathrm{norm}(\mathbf{t}_i - \mathbf{p}) \cdot \mathrm{norm}(\mathbf{t}_{\mathrm{cur}} - \mathbf{p}) \tag{1}$$

$$w_{\mathrm{res}}(i) = \max(0,\ 1 - (\|\mathbf{p} - \mathbf{t}_i\| - \|\mathbf{p} - \mathbf{t}_{\mathrm{cur}}\|) / \|\mathbf{p} - \mathbf{t}_i\|) \tag{2}$$

$$w_{\mathrm{fov}}(i) = \max(\mathrm{v}_{\mathrm{min}},\ \min(1,\ \mathbf{r}_i \cdot \mathbf{r}_{\mathrm{cur}})) \tag{3}$$

$$w(i) = (\alpha\, w_{\mathrm{ang}}(i) + \beta\, w_{\mathrm{res}}(i))\, \gamma\, w_{\mathrm{fov}}(i) \tag{4}$$

Here, $\mathbf{p}$ is a shared vertex position of a triangle fan, $\boldsymbol{t}_i$ and $\mathbf{r}_i$ are the i^{th} ($i \leqq M$) camera c_i translation vector and rotation vector, respectively. $\boldsymbol{t}_{\mathrm{cur}}$ and $\mathbf{r}_{\mathrm{cur}}$ are the current camera $\mathrm{c}_{\mathrm{cur}}$ translation vector and rotation vector, respectively, $\mathrm{v}_{\mathrm{min}}$ is minimum FoV, and α, β, and γ are user adjustable parameters. In our implementation, weights are updated according to equations (1) to (4), and camera indices are sorted by weight by the CPU. The weights are transferred to GPU memory as a k by N weight lookup table, and images are blended on screen space using the GPU by referencing the weights in the GPU memory. Thus, the lookup table is first transferred to the vertex shader and is referred to as the vertex lookup table. Then, the values are compensated in fragments to calculate each pixel value in a fragment shader

Whereas Cosco *et al*. use $k = 1$ camera, the proposed system uses $k \geq 1$ cameras. In addition, Cosco *et al*. use angular weight similar to Debevec's method [16], whereas we use the abovementioned three weights. Blending multiple cameras yields smoother appearance in DR results and a smooth appearance improves the subsequent camera tracking (Section 4.1).

Depth map generation and occlusion checks: In the rendering step, an IBR image and corresponding depth map of the current view are generated. To obtain a reasonable appearance of the T-3DM using the current hardware implementation of projective texture mapping in OpenGL, it is necessary to implement visibility checks to avoid projective textures passing through the occluded geometry [25]. However, we skip this process assuming that we preserve view-dependent images densely enough in the 3D space and distant cameras do not affect the current view.

Furthermore, we use an alpha blending scheme on the GPU to combine rendered triangle fans efficiently. Note that polygon sorting is required for accuracy. To avoid this expensive process, we have implemented occlusion checks using a depth map. First, the system generates a z-buffer as a depth map of the current view without drawing colors. Note that we preserve triangles to generate the depth map in the triangle indexing step (Section 3.2). Next, each pixel is colored with the IBR results if the depth of the triangle fans is close to the depth map.

For efficiency, the depth map and the IBR image are reused for the camera tracking of the next frame. The IBR image and the vertices of the T-3DM are used for color correction (Section 5).

4 Geometric Registration Using T-3DM

Supposing that we have a high quality T-3DM in the pre-observation stage, we can track the T-3DM and obtain the camera pose at the current frame. We employ a feature point-based approach to estimate the camera pose at each frame by minimizing projection error of the feature points detected on the T-3DM. To improve the quality of the final results, we implement a process to reduce gaps that appear around the ROI due to geometric and photometric registration residuals. This section describes the camera tracking and residual reduction process, which we refer to as region blending across the border (RBAB).

4.1 Tracking by Synthesis

In the object removal stage, the initial pose is given by matching one of the view-dependent images in the T-3DM. After the initial pose estimation, the camera pose in the following frames must be estimated. The system estimates the camera pose of the current frame using a tracking by synthesis [26] based approach. Tracking by synthesis uses textured 3D model data of a scene for tracking and provides benefits for DR camera tracking in terms of accuracy, robustness, and real-time performance, e.g., drift free model-frame registration, features matching without scale or affine invariant feature descriptors, and automatic culling and occlusion handling. In addition, tracking by synthesis can take advantage of current graphics hardware. Since we employ a T-3DM as textured 3D model data and the IBR pipeline to generate depth map and color buffer, the system is ready to track a scene using the tracking by synthesis framework as follows.

Model projection: An image of prebuilt 3D model data is rendered. In the proposed framework, an IBR image and a depth map are obtained in the previous object removal stage.

Feature extraction and matching: The rendering image and the current frame are matched to obtain 2D–2D correspondences. Since only points visible in the IBR image are useful, features are extracted in the IBR image and matched to the current frame (i.e. using Harris operator and LK tracker).

Feature back-projection: The corresponding depth map is referenced to obtain the 3D positions of the features. The system refers to the depth map to fetch the 3D position of the features in the IBR image. Since the features are tracked to the current frame, we obtain 3D–2D correspondences.

3D–2D registration: The perspective-n-point (PnP) problem is solved to estimate the 6DoF pose. We used the EPnP [27] with the RANSAC algorithm to estimate the pose of the tracked T-3DM.

Our tracking system uses view-dependent textures; therefore, it is expected to find a visually appropriate position and orientation. Thus, the IBR pipeline explicitly affects the tracking results. As a typical example, we found that the estimated camera pose rattled because of texture switching induced by k-nearest-neighbor interpolation when k is a small value, such as 1 or 2 (as claimed in

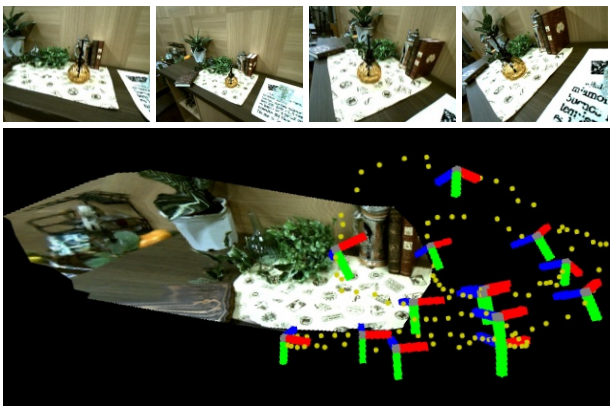

Figure 5: Visualization of camera poses estimated using tracking by the synthesis-based [26] approach. The top row shows several frames used for this tracking. The bottom figure shows plots of camera pose and a tracked T-3DM. Note that the undesirable object is not included in the T-3DM.

[28]). From a tracking perspective, setting k to a larger value may be a solution in principle; however, relative to quality and real-time requirements, this may be a shortsighted solution. Consequently, we empirically set k to 3 as the default.

The undesirable object should not exist at the pre-observation stage but exist in the object deletion stage. Thus, it must not be considered during tracking (**Figure 5**). However, the target region cannot be determined before the pose of the current frame is estimated because the system tracks the target object using a 3D bounding box. Thus, we suppose that feature points tracked into the target region are eliminated because of the low similarity between the tracked frames or are eliminated as outliers of robust estimation.

If the camera faces away from the scene and the tracker becomes lost, the camera is relocalized in the second step by matching the current image and the view-dependent image poses.

4.2 Region Blending Across the Border

We assume to delete an arbitrary static object from a scene. Thus, the following requirements for object detection must be met to obtain high-quality DR results.

- Object detection by user interaction
- Tracking of a static target object
- Artifact reduction around the ROI due to imperfect registration

We use a 3D bounding box surrounding a target object to determine a ROI. The bounding box is placed by a user, tracked, and projected to the screen. As shown in the middle of **Figure 6**, edge-like artifacts generally appear around the ROI in POB-DR because of registration errors.

Note that some methods have been proposed to handle such effects. Cosco *et al*. [17] used a set of approximated 3D bounding boxes of a target to determine a ROI. They assumed that vertices of a geometric proxy are uniformly aligned and set the blending weight of vertices surrounding the target region to zero to blend the recovered hidden view and the current view smoothly. However, this approach is not sufficient for our purpose if we use a vision technique to reconstruct vertices that are not aligned uniformly. Therefore, the blending will be partially sparse and vertices will be appreciably switched according to the camera motion in the DR results. Li *et al*. [13] used a vision-based object tracker and MVC blending, which runs in real time by limiting a ROI to a rectangle shape. We assume more drastic camera motion

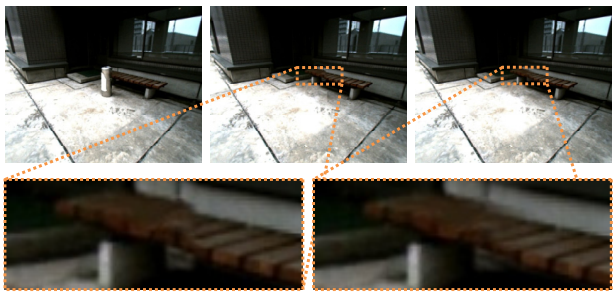

Figure 6: Comparison between DR results with and without RBAB. From left to right: current view, DR result without RBAB, and DR result with RBAB. Note that artifacts (e.g., edges) appear around the target region without RBAB processing.

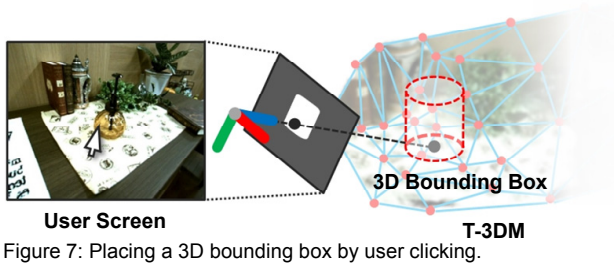

Figure 7: Placing a 3D bounding box by user clicking.

in a 3D scene; therefore, it is difficult to track a target using vision technology or approximate it using a simple rectangle. Thus, we use a 2D alpha blending style blending method, i.e., RBAB.

3D bounding box: Assuming a user camera moves more horizontally than vertically, we employ a cylindrical 3D bounding box because its appearance on a screen does not change drastically with motion. By projecting the bounding box, we can obtain a binary mask image I_{Mask}.

Set up with user interactions: The bounding box is first placed by the user clicking on the screen during the tracking (**Figure 7**). The clicked position is back-projected similar to feature back-projection (Sec. 4.1). Next, the bounding box is adjusted with respect to pose and expansion by keyboard operations.

Alpha map generation and blending: The mask image I_{Mask} is blurred using a box blur filter. The alpha map is applied to the alpha channel, representing the transparency of an IBR image. Finally, the IBR image is superimposed onto the current image.

Figure 6 shows a comparison between IBR image composition with and without RBAB. RBAB is quite simple but performs well in combination with the following photometric registration.

5 Photometric Registration Using T-3DM

In POB-DR, a hidden view is recovered using images captured before undesirable objects are placed in the environment. Therefore, photometric changes must be handled during the object deletion stage as described in Section 2. Many POB-DR studies only consider static illumination [12][17]. As mentioned above, Li *et al*. [13] used MVC blending to compensate for such differences in 2D image space. Kawai *et al*. modeled global and luminance changes in their semi-dynamic video inpainting approach [6].

We deal with RGB illumination changes at a color correction level using the vertices of a geometric proxy as the sampling points. As shown in **Figure 8**, our implementation to estimate color tone changes in the ROI involves two steps. First, we measure color tone changes in the surrounding ROI (SROI, I_{Mask} = 0) and then propagate the results to the target ROI (ROI, I_{Mask} = 1). The results affect the IBR of the next frame (Figure 3). Details of our photometric registration are described below.

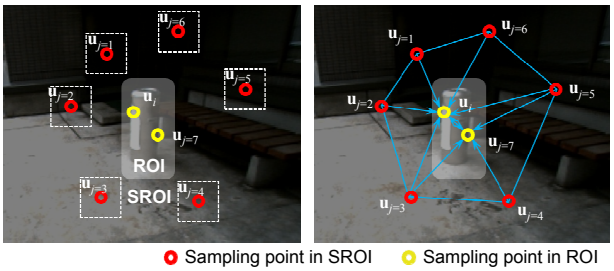

Figure 8: Color ratio measurement in SROI (Left) and color ratio estimation in ROI (Right). In this figure, $\mathbf{u}_i$ is estimated using $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$, $\mathbf{u}_7$, $\mathbf{u}_5$, and $\mathbf{u}_6$.

Figure 9: RGB color ratio measurement in SROI. Top row shows real scene with lighting and bottom row shows corresponding rendering results of color corrected T-3DM. Note that shadows are not casted like in the real scene since this method works in color correction level.

Illumination change measurement in SROI: First, all N vertices of the geometric proxy are projected to the screen as sampling points $\mathbf{u}_j$ ($j \leq N$). $w \times w$ (15×15 by default) patches are generated around the sampling points $\mathbf{u}_j$ belonging to the SROI in the current view $\mathrm{I}_{\mathrm{Cam}}$ and the corresponding IBR image $\mathrm{I}_{\mathrm{IBR}}$. Then, the j^{th} RGB color ratio vector $\mathbf{v}_{\mathrm{RGB},j}$ between the patch pairs is calculated (Eq. (5)). **Figures 9** shows the results.

$$\mathbf{v}_{\mathrm{RGB},j} = \frac{1}{w \times w}\left(\sum_{w \times w} \mathrm{I}_{\mathrm{Cam}}(\mathbf{u}_j) / \sum_{w \times w} \mathrm{I}_{\mathrm{IBR}}(\mathbf{u}_j)\right) \qquad (5)$$

Illumination change estimation in ROI: In the ROI, it is impossible to measure RGB color ratios because undesirable objects are present in the current view $\mathrm{I}_{\mathrm{Cam}}$ but do not occur in the corresponding IBR image $\mathrm{I}_{\mathrm{IBR}}$. Thus, the RGB color ratio vector $\mathbf{v}_{\mathrm{RGB},j}$ in ROI is calculated using a weighted average of the RGB color ratio vectors $\mathbf{v}_{\mathrm{RGB},k}$ of the j^{th} triangle fan and each distance d_k (Eq. (6)). Given the initial value $(1, 1, 1)^{\mathrm{T}}$, the vector $\mathbf{v}_{\mathrm{RGB},j}$ is updated frame by frame.

$$\mathbf{v}_{\mathrm{RGB},j} = \frac{1}{n}\left(\sum_{n} \mathbf{v}'_{\mathrm{RGB},k} / d_k\right) \qquad (6)$$

RGB color ratio vectors are set to the T-3DM attributes and are compensated over polygon meshes using a smooth shading mechanism on the GPU. In our IBR pipeline, the RGB color vectors are multiplied to the original T-3DM pixel colors in the screen space to obtain a color corrected pixel. **Figures 10** shows the results.

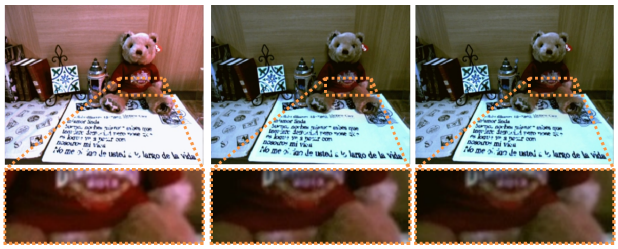

Figure 10: RGB color ratio propagation in the ROI using sampling points in SROI. As time progresses from the left figure to the right figure, RGB color ratios in the ROI decrease frame by frame because of illumination changes.

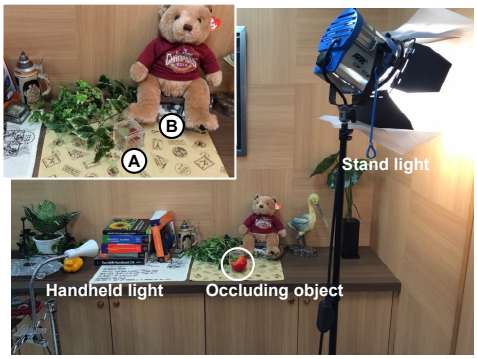

Figure 11: Indoor test scene. Lighting is changed using the stand light with a diffuser sheet and the handheld light.

6 Results

Here, we present experimental results obtained using real data. We demonstrate our POB-DR system in indoor (**Figure 11**) and outdoor scenes. This evaluation was performed during live operation using a hand-held camera and a mobile laptop PC with identical tunable parameters.

6.1 System Configuration

In the following evaluations, we used a PointGrey Flea3 camera, operating at 60 Hz, 640 × 512 resolution, 24-bit RGB color, and a lens with a 71.7° (H) FoV. Note that the camera has precalibrated intrinsic parameters. We used a Windows 8.1 laptop with an Intel Core i7 4500U 1.8 GHz CPU, Intel HD Graphics 4400 GPU, and 8.0 GB memory. The system was implemented using Microsoft Visual Studio 2013 with C++, the OpenGL graphics API, and OpenCV as the computer vision library primarily for the LK tracker and PnP solver. The proposed IBR pipeline works on a GPU using OpenGL Shading Language 3.3.

6.2 DR Results and Discussion

Here, we demonstrate the proposed system with an indoor test scene. To demonstrate the proposed system in a practically difficult scene for POB-DR, we placed fake leaves, a glass object, and a stuffed bear in the scene (Figure 11). We used a standing light and a handheld light to dynamically change lighting conditions.

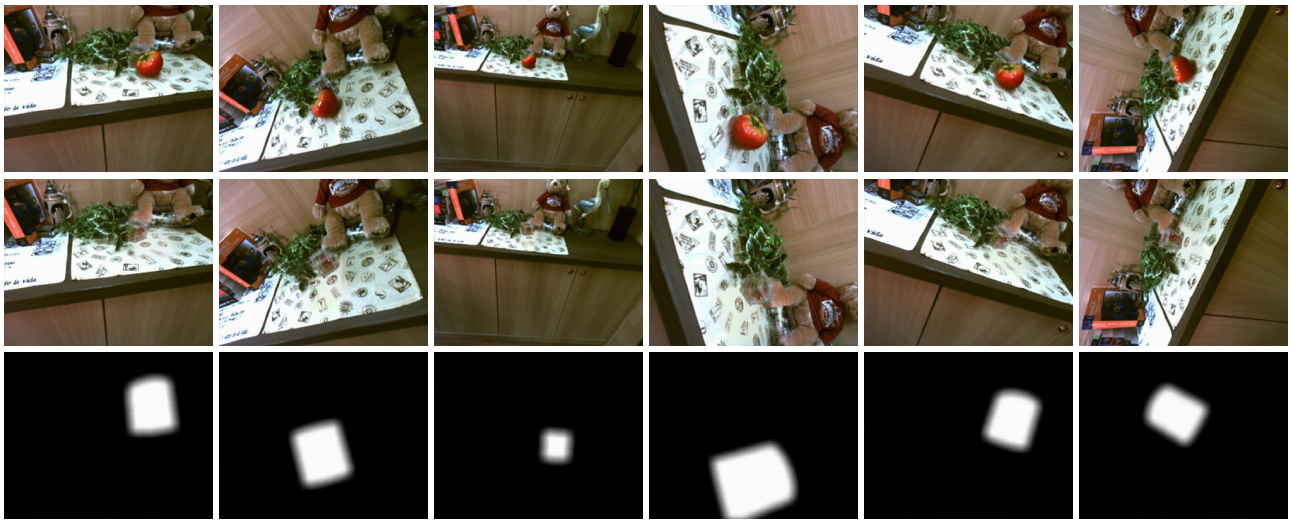

Figure 12: Pairs of original frames, corresponding DR results, and mask images for the indoor test scene. Illumination was varied (from the first row to the other rows) after pre-observation.

First, we observed the scene lit by the standing light before placing an occluding object. We then constructed a T-3DM. After the pre-observation stage, we placed the occluding object at position "A" in Figure 11. Then, we turned the standing light off during the object removal stage. The white balance of the camera was not changed between the pre-observation and object removal stages. **Figure 12** shows the DR results. As expected, the occluding object was visually deleted while the camera moved freely in the 3D space even though the illumination was changed. **Table 1** shows the processing time for each step as well as total processing time.

Table 1: Processing Time

Number of vertices/triangle fans	393
Number of view-dependent images	423
Tracking by Synthesis	45.7 ms
Color ratio estimation	2.3 ms
IBR and synthesis	22.2 ms
Data manipulation	16.5 ms
Total	86.7 ms

Processing time for each step was quite stable throughout 1500 frames. Compared to an MVC blending approach [13], our photometric registration is not significantly affected by the size of the ROI and ran quickly due to efficient use of the T-3DM. Although our implementation is simple color correction level photometric registration, the results are good and consistent.

We applied the proposed system to a scene under dynamic lighting conditions. In this evaluation, we placed the target object at "B" in Figure 11 and used the handheld light to dynamically illuminate the scene in the object removal stage. **Figure 13** shows the results. Our photometric registration requires several frames to complete color correction because the RGB color ratios of a sampling point are propagated from other sampling points connected in its triangle fan frame by frame. **Figure 14** shows results for some outdoor scenes.

While the proposed system performed well for most frames, some known issues led to inconsistencies. For example, IBR images were blurred and tracking was lost when the camera reached areas that were sparsely covered by view-dependent images. The most straightforward solution to this problem is to preserve a greater number of view-dependent images more densely in the pre-observation stage, even though this will result in increased data volume for the T-3DM. Note that we must also consider the hardware used in these evaluations. In our current implementation, we use IBR images on the GPU for both camera tracking and color correction on the CPU; thus, an IBR image must be transferred from the GPU memory to the main memory for each frame. This process is generally slow on current GPUs because it requires synchronization with the CPU. In addition, the BUS bandwidth may be limited on some hardware. We selected a CPU with an integrated GPU and transferred quarter-sized IBR and alpha map images, which can resolve the above problems.

7 Conclusion

We have presented a geometric and photometric registration method for POB-DR. The system begins with 3D reconstruction, in which hundreds of view-dependent images are automatically captured while a camera is waved around the scene. The results are then combined to construct a T-3DM. After an undesirable object is placed in the environment, the system visually eliminates the object effectively and efficiently using the prefetched T-3DM. In other words, the act of placing the object is virtually undone in an MR space. The T-3DM is used by a tracking by synthesis-based camera tracker and color correction level illumination adaptation for precise hidden view recovery. Finally, the generated IBR image is synthesized to the current image using a computationally inexpensive region blending technique. The results demonstrate that the proposed system can eliminate an undesirable object from a 3D structured scene under dynamic illumination with drastic viewpoint changes.

Acknowledgments

This work was supported in part by a Grant-in-Aid for Scientific Research (S) Grant Number 24220004 and a Grant-in-Aid for the Japan Society for the Promotion of Science Fellows Grant Number 259193.

References

[1] Steve Mann, "Mediated reality," TR 260, M.I.T. Media Lab Perceptual Computing Section, Cambridge, Massachusetts, 1994.

Figure 13: Pairs of original frames and corresponding DR results for the indoor test scene under dynamic lighting.

Figure 14: Pairs of original frames and the corresponding DR results for outdoor scenes.

[2] Steve Mann and James Fung, "VideoOrbits on eye tap devices for deliberately diminished reality or altering the visual perception of rigid planar patches of a real world scene," Proc. ISMR 2001, pp. 48-55, 2001.

[3] Yuichiro Takeuchi and Ken Perlin, "ClayVision: The (elastic) image of the city," Proc. CHI 2012, pp. 2411-2420, 2012.

[4] Jan Herling and Wolfgang Broll, "Advanced self-contained object removal for realizing real-time diminished reality in unconstrained environments," Proc. ISMAR 2010, pp. 207-212, 2010.

[5] Jan Herling and Wolfgang Broll, "PixMix: A real-time approach to high-quality diminished reality," Proc. ISMAR 2012, pp. 141-150, 2012.

[6] Norihiko Kawai, Masayoshi Yamasaki, Tomokazu Sato, and Naokazu Yokoya, "Diminished reality for AR marker hiding based on image inpainting with reflection of luminance changes," Proc. ITE Transactions on Media Technology and Applications, Vol. 1, No. 4, 343-353, 2013.

[7] Norihiko Kawai, Tomokazu Sato, and Naokazu Yokoya, "Diminished reality considering background structures," Proc. ISMAR 2013, pp. 259-260, 2013.

[8] Akihito Enomoto and Hideo Saito, "Diminished reality using multiple handheld cameras," Proc. ACCV 2007, pp. 130-150, 2007.

[9] Peter Barnum, Yaser Sheikh, Ankur Datta, and Take Kanade, "Dynamic seethroughs: Synthesizing hidden views of moving objects," Proc. ISMAR 2009, pp. 111-114, 2009.

[10] Songkran Jarusirisawad, Takahide Hosokawa, and Hideo Saito, "Diminished reality using plane-sweep algorithm with weakly-calibrated cameras," *Progress in Informatics*, Vol. 7, pp. 11-20, 2010.

[11] Siavash Zokai, Nassir Navab, Yakup Genc, and Julien Esteve, "Multiview paraperspective projection model for diminished reality," Proc. ISMAR 2003, pp. 217-226, 2003.

[12] Vincent Lepetit and Marie-O. Berger, "An intuitive tool for outlining objects in video sequences: Applications to augmented and diminished reality," Proc. ISMR 2001, pp. 159-160, 2001.

[13] Zhuwen Li, Yuxi Wang, Jiaming Guo, Loong-F. Cheong, and Steven Z. Zhou, "Diminished reality using appearance and 3D geometry of internet photo collections," Proc. ISMAR 2013, pp. 11-19, 2013.

[14] Marc Levoy and Pat Hanrahan, "Light field rendering," Proc. SIGGRAPH 1996, pp. 31-42, 1996.

[15] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen, "The lumigraph," Proc. SIGGRAPH, pp. 43-54, 1996.

[16] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach," Proc. SIGGRAPH, pp. 11-20, 1996.

[17] Francesco I. Cosco, Carlos Garre, Fabio Bruno, Maurizio Muzzupappa, and Miguel A. Otaduy, "Augmented touch without visual obtrusion," Proc. ISMAR 2009, pp. 99-102, 2009.

[18] Chris Harris, and Mike Stephens, "A combined corner and edge detector," Proc. Alvey Vision Conf., pp. 147-151, 1988.

[19] Jean-Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm," Intel Corporation Microprocessor Research Labs., Retrieved from http://robots.stanford. edu/cs223b04/algo_tracking.pdf, 2000.

[20] Richard I. Hartley, "In defense of the eight-point algorithm," *IEEE Trans. Pattern Recognition and Machine Intelligence*, Vol. 19, No. 6, pp. 580-593, 1997.

[21] Martin A. Fischler, and Robert C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," Comm. of the ACM, Vol. 24, pp. 381-395, 1981.

[22] Georg Klein and David Murray, "Parallel tracking and mapping for small AR workspaces," Proc. ISMAR 2007, pp. 225-234, 2007.

[23] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen, "Unstructured lumigraph rendering," Proc. SIGGRAPH 2001, pp. 425-432, 2001.

[24] Mark Segal, Carl Korobkin, Rolf V. Widenfelt, Jim Foran, and Paul Haeberli, "Fast shadows and lighting effects using texture mapping," Proc. SIGGRAPH 1992, pp. 294-252, 1992.

[25] Paul Debevec, Yizhou Yu, and George Borshukov, "Efficient view-dependent image-based rendering with projective texture-mapping," Proc. Eurographics 1998 Rendering Workshop, 1998.

[26] Gilles Simon, "Tracking-by-Synthesis using point features and pyramidal blurring," Proc. ISMAR 2011, pp. 85-92, 2011.

[27] Vincent. Lepetit, Francesc M.-Noguer, and Pascal Fua, "EPnP: An accurate O(n) solution to the PnP problem," Int. Journal Computer Vision, Vol. 81, No. 2, pp. 155-166, 2009.

[28] Abe Davis, Marc Levoy, and Fredo Durand, "Unstructured light fields," Vol. 31, No. 2pt1, pp.305-314, Proc. Eurographics, 2012.