Virtual Reality manuscript No.
(will be inserted by the editor)

AR image generation using view-dependent geometry modification and texture mapping

Yuta Nakashima · Yusuke Uno · Norihiko Kawai · Tomokazu Sato · Naokazu Yokoya

Received: date / Accepted: date

Abstract Augmented reality (AR) applications often require virtualized real objects, i.e., virtual objects built based on real objects and can be rendered from arbitrary viewpoint. In this paper, we propose a view-dependent geometry modification- and texture mapping-based method for real object virtualization and AR image generation. The proposed method is a hybrid of model- and image-based rendering techniques. It uses multiple input images of the real object as well as its three-dimensional (3D) model obtained from an automatic 3D reconstruction technique. Even with state-of-the-arts, the accuracy of reconstructed 3D model can be insufficient, resulting in visual artifacts due to, e.g., missing object boundaries. The proposed method thus generates a depth map from a 3D model of a virtualized real object and expands its region in the depth map to prevent missing object boundaries. Since such object region expansion in a depth map results in disclosure of background pixels in input images, especially undesirable for AR applications, we preliminary extracts object regions and use them for texture mapping. With our GPU implementation for real-time AR image generation, we have experimentally demonstrated that use of expanded geometry reduces the number of required input images and maintains the visual quality.

Keywords View-dependent geometry modification · view-dependent texture mapping · augmented reality · free-viewpoint image generation

Y. Nakashima, Y. Uno, N. Kawai, T. Sato, and N. Yokoya
Nara Institute of Science Technology (NAIST), 8916-5 Takayama-cho, Ikoma, Nara, 630-0192, Japan
Tel.: +81-743-72-5293
Fax: +81-743-72-5299
E-mail: {n-yuta, yusuke-u, norihi-k, tomoka-s, yokoya}@is.naist.jp

1 Introduction

With recent popularization of augmented reality (AR) technology, many applications have been proposed so far and some of them are even available to ordinary users. Such applications sometimes require to virtualize a real object (i.e., to build a virtual object from a real object) and to generate AR images with them. Especially for applications like virtual furniture arrangement and communication tools capable of showing real objects on the display at the other end, ordinary users of AR applications must build virtualized real objects by themselves. These applications strongly demand methods for real object virtualization and AR image generation.

Besides the availability to ordinary users, the requirements of real object virtualization and AR image generation can be summarized as follows:

(i) The visual quality of AR images based on virtualized real objects should be high.
(ii) The data size for virtualized real objects must be acceptably small for saving storage or lightweight transmission.

Most AR applications use a model-based technique for virtualizing real objects and generating AR images (Azuma, 1997). This technique uses three-dimensional (3D) models of real objects, which can be hand-crafted or automatically reconstructed using a 3D reconstruction technique. Figure 1(top) shows an example of such a technique with a 3D model that is automatically reconstructed by (Jancosek and Pajdla, 2011). On the other hand, research efforts have been dedicated to image-based techniques, which do not rely on three-dimensional (3D) models but on a number of images from which a

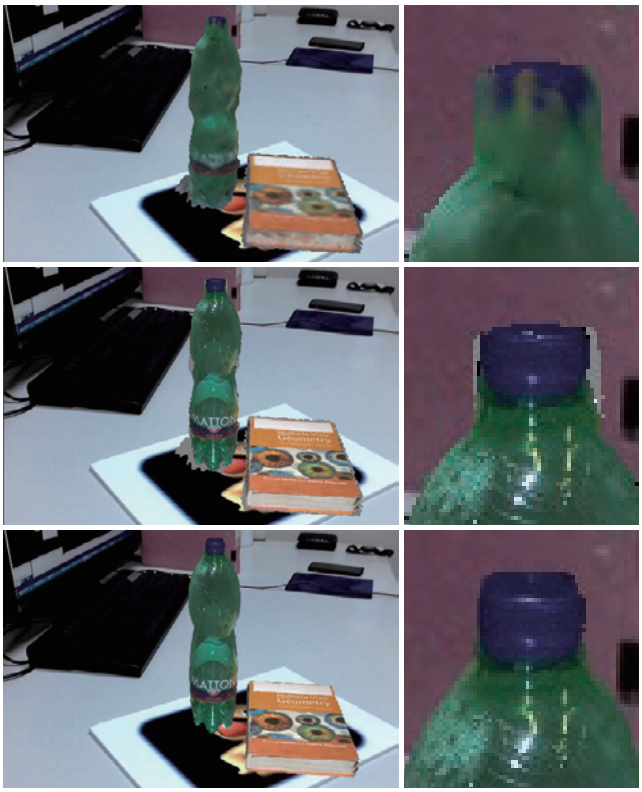


Fig. 1 Examples of AR images by model-based technique (Jancosek and Pajdla, 2011) (top), model-based technique with VDTM (middle), and proposed method (bottom). Images on right column are their closeups around bottle cap.

novel view image is synthesized, e.g., Levoy and Hanrahan (1996); Gortler et al (1996).

Unfortunately, these techniques suffer from some drawbacks. Since hand-crafting 3D models requires special skills and is practically impossible for ordinary users, model-based techniques need to use automatically reconstructed 3D models. However, the accuracy of such 3D models is usually insufficient, which leads to loss of the details and undesired rough boundaries (false boundaries that are not faithful to the original shape) as shown in Fig. 1(top). For the image-based techniques, the number of images is usually enormous for high quality image synthesis; the burden of image acquisition may keep ordinary users from using such image-based techniques.

To overcome the drawbacks, image-based techniques have been extended to leverage the geometry of real objects. One of the most well-known techniques is view-dependent texture-mapping (VDTM) in Debevec et al (1998). This technique applies to each face of a 3D mesh model the texture that is an image captured from the viewpoint closest, among multiple input images, to that of the image to be synthesized. Using VDTM, the detailed shapes inside the object region can be regained from a simplified or inaccurate 3D models. However, VDTM-based AR images still suffer from the rough boundary due to missing or extra volumes in the 3D models: Since VDTM applies textures directly to the 3D model, its boundary appears as is in AR images, leading missing object regions and exposure of the background pixels (the gray pixels around the bottle cap shown in Fig. 1(middle)).

In this paper, we propose a novel method for generating AR images with a virtualized real object. The proposed method can be deemed as an image-based technique leveraging geometry, which takes a 3D model and multiple images of a real object as its input. To remedy the problem of rough boundaries, we introduce view-dependent geometry modification (VDGM): After the depth map of the virtualized real object is generated in the rendering pipeline, VDGM expands the object region in it. Since the expanded geometry is likely to expose excessive background regions, which is not desirable in AR image generation, we preliminarily extract the foreground regions (i.e., image regions containing the real object to be virtualized) in the input images and take advantage of their smooth boundaries as shown in Fig. 1(bottom). The main contributions of this paper can be summarized as follows.

- We propose a novel method for virtualizing real objects and generating AR images using an image-based technique with 3D models. Being different from existing techniques, the proposed method adaptively modifies the geometry of objects (VDGM) to alleviate artifacts around object boundaries that is significant in VDTM-based AR images.
- We propose to apply the discrete Poisson equation to geometry modification for retaining the discontinuity in depth maps due to partial self-occlusion. We illustrate that this prevents undesired visual artifacts.
- We experimentally demonstrate the visual quality of AR images obtained by the proposed method. We also show that the proposed method requires less input images than existing image-based rendering techniques thanks to an automatically constructed 3D model.

The rest of the paper is organized as follows. In the next section, we review related work. Section 3 introduces an overview of the proposed method and the details are given in Sections 4 and 5. After describing implementation details in Section 6, the performance of the proposed method is validated in Section 7. Section 8 concludes the paper.

2 Related work

Various techniques have been proposed for virtualizing real objects and generating novel view images based on virtualized real objects. One extreme of such techniques is the fully model-based one, in which a 3D model of a real object is hand-crafted or automatically reconstructed using such a 3D reconstruction technique as (Furukawa et al, 2010; Jancosek and Pajdla, 2011; Kolev et al, 2012). Good comparison of some existing algorithms can be found in (Seitz et al, 2006) for automatic 3D reconstruction. Novel view images are synthesized by an ordinary rendering pipeline applied to the 3D model. As mentioned in Section 1, the problems of these techniques are that hand-crafting 3D models is not easy for ordinary users and the accuracy of automatic 3D reconstruction may be insufficient even with the state-of-the-art algorithms.

Techniques at the other extreme are fully image-based ones (Levoy and Hanrahan, 1996; Gortler et al, 1996). They model rays from a scene based on input images and synthesize a novel view image using the modeled rays. Their visual quality is sufficient if the number of input images is large; however, capturing an enormous number of images with accurate camera poses can be laborious for ordinary users.

Between the extremes of the model- and image-based techniques, a spectrum of various techniques have been proposed so far, which can be basically deemed as image-based techniques but leverage the geometry of real objects to improve the visual quality and reduce the number of input images. VDTM originally proposed by Debevec et al (1998) hand-crafts simplified 3D models of a real scene and applies a texture to each face of the models. The texture is selected from input images that is captured from the viewpoint closest to the novel view image to be synthesized. Bastian et al (2010) interactively build a relatively accurate 3D model of a real object to skip hand-crafting 3D models from scratch and use the VDTM technique to color it. Irani et al (2002), on the other hand, have proposed to reduce the number of images without explicitly gaining the geometry of a scene; however, they implicitly use geometric information through color consistency test.

More recent techniques leverage a roughly estimated geometry as a proxy of 3D models. For example, the unstructured light field technique by Davis et al (2012) uses several types of proxies, i.e., planes and triangle meshes based on feature points used in simultaneous localization and mapping (SLAM) techniques, e.g., Klein and Murray (2007). Compared to the original light field technique (Levoy and Hanrahan, 1996), this technique drastically reduces the number of input images required for sufficient visual quality. Chaurasia et al (2013) also exploit feature points obtained during structure-from-motion such as (Wu, 2011). Instead of explicitly representing the geometry, they assign feature points to superpixels and use it as a proxy of the scene geometry.

Our proposed method also lies between the model- and image-based techniques, relying much on 3D models of real objects that are hand-crafted or automatically reconstructed using, e.g., a technique by Jancosek and Pajdla (2011). Our position, however, is that the 3D models are not sufficiently accurate to synthesize novel view images, and we further modify the 3D models depending on the viewpoint of the novel view image, which we referred to as view-dependent geometry modification (VDGM), to remedy the problem of rough boundaries shown in Fig. 1(middle). Similar ideas are employed by Buehler et al (2001) and Davis et al (2012) as a proxy. They build triangle meshes given a viewpoint in novel view image synthesis. However, as Davis et al (2012) noted, this approach may suffer from temporal discontinuity in a sequence of synthesized images due to topological change in the triangle meshes. They also give visual artifacts around jump edges because their triangle mesh generation does not take into account the occluding relationship among objects. In contrast, our proposed method modifies the geometry in the pixel-wise depth domain, in which topological change never involves. Our VDGM also considers the occluding relationship based on the original 3D models to avoid the visual artifacts around jump edges.

In addition, being different from the techniques in Davis et al (2012) and Chaurasia et al (2013), our proposed method is tailored for AR image generation, which usually deals with small objects but not an entire scene. This enables us to take advantage of foreground extraction to smooth the object boundaries.

3 Overview of the proposed method

An overview of the proposed method is shown in Fig. 2. Given set $S \in \{I_n | n = 1, \ldots, N\}$ of multiple input images of the real object to be virtualized (called the target, hereinafter), the proposed method generates AR images through the offline and online stages.

At the offline stage, the proposed method uses a structure-from-motion technique, e.g., (Wu, 2011), to estimate the pose of the n-th camera C_n that shot the image $I_n \in S$. We denote the estimated rotation matrix and translation vector for I_n by $\mathbf{R}_n$ and $\mathbf{t}_n$ in an arbitrary coordinate system. Then, 3D model M is hand-crafted or is constructed, e.g., from S using a 3D reconstruction technique. We can adopt any technique such as (Jancosek and Pajdla, 2011) for automatic 3D

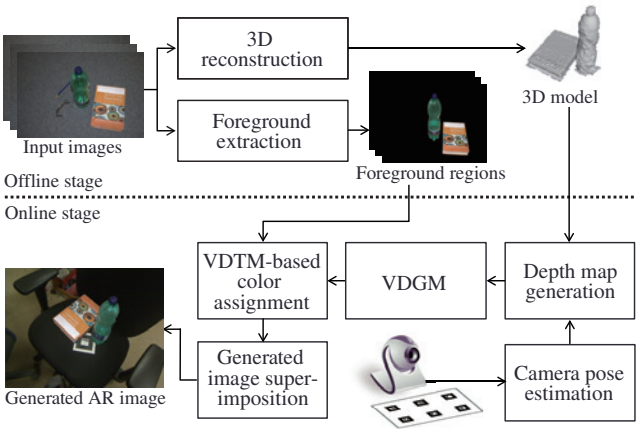


Fig. 2 Overview of proposed method.

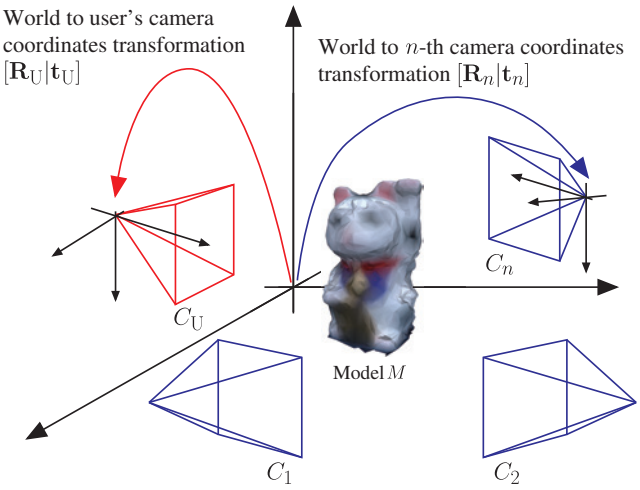


Fig. 3 Relationship among coordinate systems used in our proposed system.

reconstruction. The user may modify M so that only the target is rendered in the AR application. Also target region Ω_n in $I_n \in S$ is extracted, e.g., by using the GrabCut algorithm (Rother et al, 2004).

At the online stage, we generate AR images for the user's camera C_U. The image from C_U is denoted by I_U, capturing the actual environment onto which the rendered target is to be superimposed. The proposed method firstly estimates the pose of C_U using such a technique as ARToolkit (Kato and Billinghurst, 1999), PTAM (Klein and Murray, 2007), and Metaio SDK (Metaio GmbH, 2014), in real-time. The camera pose is denoted by $\mathbf{R}_U$ and $\mathbf{t}_U$. For notation simplicity, we assume that the coordinate system for $\mathbf{R}_U$ and $\mathbf{t}_U$ is the same as one for $\mathbf{R}_n$ and $\mathbf{t}_n$, called the world coordinate system, but if necessary, we can apply any constant transformation to bring coordinates in one system to the other. The relationship among the coordinate systems used in the proposed method is summarized in Fig. 3. From the estimated camera pose and the 3D

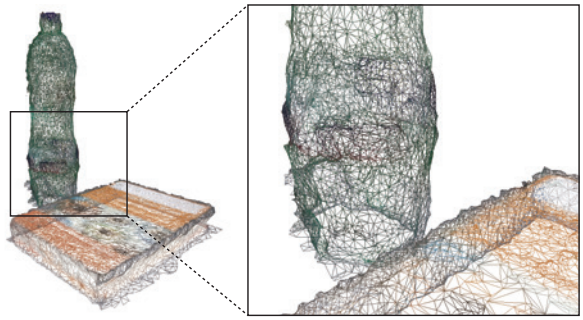


Fig. 4 Automatically reconstructed 3D mesh model and its close view, which illustrates missing and extra volumes. It can also imply that triangles around the object boundary yield false high-frequency components.

model, we generate a depth map using the same process as an ordinary rendering pipeline. Then, VDGM modifies the target region in the depth map. For synthesizing an image of the virtualized target, our VDTM, which is modified to work with depth maps, colors each pixel in the modified depth map using foreground pixels of images in S. Finally, the synthesized image is superimposed on I_U to generate an AR image.

As mentioned above, the proposed method does not use the shape of an automatically reconstructed or hand-crafted 3D model as is but expands and smooths the target region in the depth map by VDGM. This modification remedies the problem of rough boundaries: Generally, automatically reconstructed or hand-crafted 3D models are not completely consistent with the target silhouette in I_n. This is because hand-crafted 3D models are sometimes over-simplified and automatically reconstructed 3D models contain missing or extra volumes as shown in Fig. 1(middle), which also result in false high-frequency components in the object boundary (Fig. 4). VDGM reduces the visual artifacts due to these problems by (i) expanding and smoothing the target geometry in the depth map and by (ii) determining the color of each pixel based on the foreground pixels of $I_n \in S$ to avoid artifacts due to excessive expansion.

The following sections give the details of the online stage, i.e., VDGM and VDTM-based color assignment. Techniques used at the offline stage are borrowed from existing ones specified in Section 6.

4 View-dependent geometry modification

Given the estimated pose of camera C_U, the ordinary rendering pipeline generates the depth map with model M, whose j-th depth value is denoted by d_j. Since M is not sufficiently accurate, the boundary of the target region in the depth map is chipped as shown in Fig. 5(left), which directly leads to the problem of rough

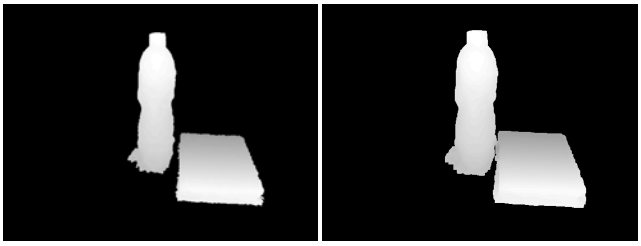


Fig. 5 Examples of original depth map (left) and expanded one (right). Non-black regions in left and right depth maps represent Ω_{E} and Ω'_{E}, respectively.

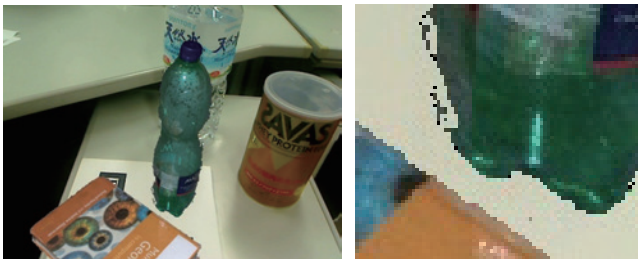


Fig. 6 Example AR image with discontinuity inside original target region (left) and its closeup (right).

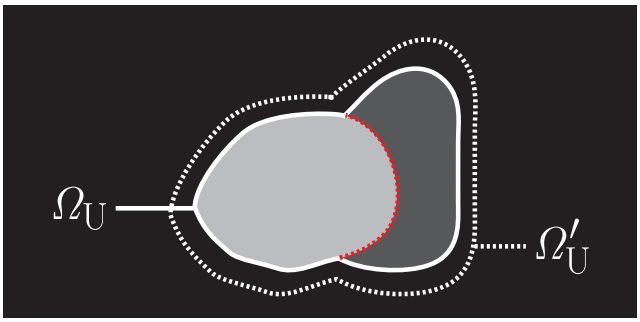


Fig. 7 Illustrative example of depth map with definitions of regions Ω_{U} (solid white line) and Ω'_{U} (dashed white line). Red dashed line indicates pairs of pixels excluded from A_{U}.

boundaries. VDGM reduces the false high-frequency components by smoothing the target region and by expanding the target region to ensure that the pixels at the target boundaries in images in S form the boundary of the generated image. A straightforward approach is to apply a low-pass filter (i.e., a box or Gaussian filter). However, we have found that it spoils actual jump edges between different surfaces, and that the inaccurate 3D model can cause discontinuity even in the inside of the original target region as shown in Fig. 6. In this work, we have developed novel formulation of VDGM using discrete Poisson equation, which can be solved efficiently on GPUs. Appropriate coefficients of the equation preserves the jump edges while smoothing and expanding the target regions.

Let Ω_{U} be the region onto which 3D mesh model M is projected given the rotation $\mathbf{R}_{\mathrm{U}}$ and translation $\mathbf{t}_{\mathrm{U}}$ for the user's camera C_{U}, i.e., d_j is defined over $j \in \Omega_{\mathrm{U}}$, as shown in Fig. 7. We also denote an expanded region by Ω'_{U}, obtained by applying the morphological erosion operator, which can be iterated several times. The erosion operator determines how much the target region is expanded. Discrete Poisson equation-based VDGM assigns actual depth values to the expanded region as well as smoothing the original depth values.

The requirements for VDGM are as follows:

(i) A new depth value must be close to its original value, if any, for correct color assignment.
(ii) A new depth value must also be close to its adjacent depth values for smoothness of the boundary of the generated novel viewpoint images to prevent visual artifacts, such as in Fig. 6.
(iii) Requirement (ii) can be ignored to preserve the jump edges.

Letting d'_j be the j-th depth value in expanded region Ω'_{U}, we can express the requirements above in the following minimization problem over d'_j for all $j \in \Omega'_{\mathrm{U}}$ based on discrete Poisson equation:

$$\min_{\mathbf{d}'} \sum_{j \in \Omega_{\mathrm{U}}} (d'_j - d_j)^2 + \sum_{(j,i) \in A_{\mathrm{U}}} (d'_j - d'_i)^2, \tag{1}$$

where $\mathbf{d}' = \{d'_j | j \in \Omega'_{\mathrm{U}}\}$. The first term is calculated over original target region Ω_{U} for (i), and the second term smooths the depth values for (ii). A_{U} is the set of all pairs of adjacent pixels in Ω'_{U} excluding (j, i) such that

$$|d_j - d_i| > \theta_{\mathrm{JE}} \tag{2}$$

so as to preserve the jump edges in the original depth map, where we assume that adjacent pixels whose depth values' difference is greater than a predetermined threshold θ_{JE} form a jump edge. Depth values d'_j for $j \in \Omega'_{\mathrm{U}} \backslash \Omega_{\mathrm{U}}$ are determined solely based on the second term for expanding the target region, where the operator "\" stands for the relative complement.

The minimization problem in Eq. (1) is equivalent to a symmetric and positive-definite linear system obtained by setting its partial derivatives with respect to d'_j to zero for all $j \in \Omega'_{\mathrm{U}}$. Since the coefficient matrix of the system is sparse, we can solve it by conjugate gradient (CG) for sparse systems, which works efficiently on GPUs. Figure 5(right) shows the depth map after VDGM.

5 View-dependent texture mapping-based color assignment

In our method, VDGM expands object regions in depth maps and smooths the depth values. This modification

removes the actual details in the target's 3D model as well, which may result in significant visual artifacts in generated novel view images. As mentioned in Debevec et al (1998), color assignment based on the idea of VDTM reduces these visual artifacts by choosing appropriate images in S for coloring each pixel. Therefore, the proposed method employs VDTM-based color assignment, consisting of visibility check, suitable image selection, and coloring using mixture weights.

5.1 Visibility check

For coloring the pixels in expanded region Ω'_{U}, the proposed method uses images in S; however, each image only contains a portion of the target. Obviously, the opposite side of the target is invisible in a single image. In addition, the target can be partially occluded by itself, which is referred to as self-occlusion. In order not to use such images for coloring corresponding pixels in the novel view image to be generated, we employ visibility check based on the depth maps of 3D model M for C_n's. The i-th depth value for C_n is denoted by $e_{n,i}$. The depth map for C_n can be preliminarily generated at the offline stage.

Given depth value d'_j and the pose of C_{U}, we can regain the 3D position of the corresponding point on M in the world coordinate system, which is denoted by $\mathbf{p}_j$. This point is then transformed to C_n's coordinate system using $\mathbf{R}_n$ and $\mathbf{t}_n$, and is converted to depth value $d'_{n,j}$. If 3D point $\mathbf{p}_j$ regained from the depth is visible in I_n and is projected to the i-th pixel of I_n, $d'_{n,j}$ is the same value as $e_{n,i}$. Otherwise, since $\mathbf{p}_j$ is occluded by different surfaces, $d'_{n,j}$ is larger than $e_{n,i}$. Therefore, if

$$|d'_{n,j} - e_{n,i}| < \theta_{\mathrm{VC}} \tag{3}$$

is satisfied with a certain threshold θ_{VC}, we set the visibility label $v_{n,j}$ to 1, and 0 otherwise, where $v_{n,j} = 1$ means that $\mathbf{p}_j$ is visible in I_n.

5.2 Suitable image selection

As mentioned in Debevec et al (1998), the details of the target that are lost during 3D modeling and VDGM can be reproduced by using the VDTM technique. Devebec et al's original method applies a selected image as the texture to each face of the 3D model; however, the proposed method represents the modified geometry of the target as a depth map. We thus introduce per-pixel VDTM, which selects a suitable image for each pixel in expanded region Ω'_{U}. Thanks to GPU implementation, this process is sufficiently fast for real-time novel view image generation.

Fig. 8 Visual artifacts due to selected image transition, exposing significant discontinuity in color within an object.

According to Debevec et al (1998), we first calculate similarity $s_{j,n}$ for regained 3D point $\mathbf{p}_j$ involving C_{U} and C_n defined by

$$s_{n,j} = \frac{(\mathbf{t}_{\mathrm{U}} - \mathbf{p}_j)^\top (\mathbf{t}_n - \mathbf{p}_j)}{\|\mathbf{t}_{\mathrm{U}} - \mathbf{p}_j\| \, \|\mathbf{t}_n - \mathbf{p}_j\|} + 1, \tag{4}$$

where $\top$ represents the transposition and $\|\cdot\|$ the Euclidean norm. We add one to make the similarity positive. This similarity corresponds to the angle formed by $(\mathbf{t}_{\mathrm{U}} - \mathbf{p}_j)$ and $(\mathbf{t}_n - \mathbf{p}_j)$, and a large value means a small difference in the view directions of these cameras. Suitable image $\hat{I}_j$ for $\mathbf{p}_j$ is given by $I_{\hat{n}_j}$ where $\hat{n}_j = \mathrm{argmax}_n \, s_{n,j}$. Considering the visibility of $\mathbf{p}_j$ in I_n, we modify this process to

$$\hat{I}_j = I_{\hat{n}_j} \text{ where } \hat{n}_j = \operatorname*{argmax}_n v_{n,j} s_{n,j}. \tag{5}$$

5.3 Coloring using mixture weights

Suitable image selection finds the best image based on a similarity measure; however, coloring solely based on this image causes significant visual artifacts due to transition of selected images (Fig. 8). One way to reduce such visual artifacts is to smoothly mixing the colors from multiple images. The proposed method determines the j-th pixel's color using images selected based on precomputed spherical Delaunay triangulation (Renka, 1997) of camera positions $\mathbf{t}_n$, of which example is shown in Fig. 9. Given regained 3D position $\mathbf{p}_j$, we can identify the triangle intersecting the ray from C_{U} to $\mathbf{p}_j$. The j-th pixel is colored by a weighted average of the corresponding pixels' colors in the images that form the triangle. Since exhaustively finding the intersecting triangle for each pixel is computationally expensive, we use the suitable image identified by $\hat{n}_j$ to reduce the possible triangles.

After spherical Delaunay triangulation, which can be done at the offline stage, we store the association between each camera position and the triangles whose vertices include that camera position, so that we can

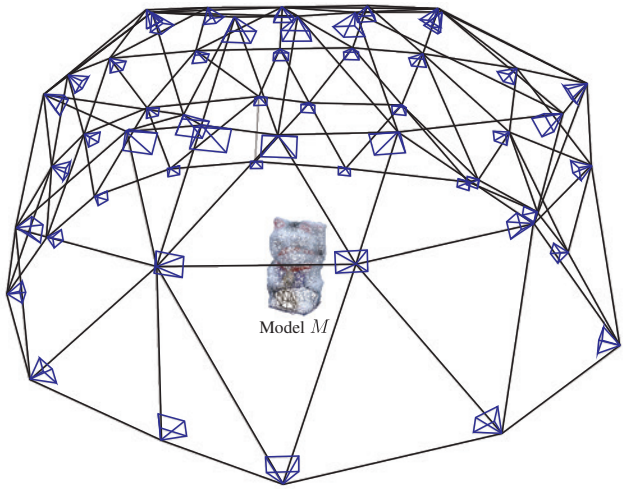


Fig. 9 Example result of spherical Delaunay triangulation with target. Each blue pyramid represents C_n, and each black line is a triangle edge obtained from spherical Delaunay triangulation.

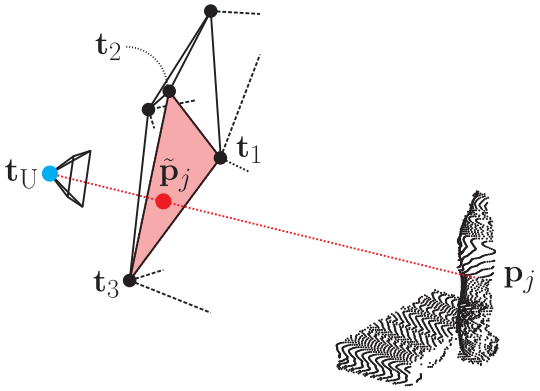


Fig. 10 Our coloring process finds an intersecting triangle represented by pale red. Black dots represent regained positions, i.e., $\mathbf{p}_j$'s. Black and blue circles are camera positions corresponding to C_n's and C_U, respectively.

rapidly identify the triangles that may intersect the line specified by $\mathbf{p}_j$ and $\mathbf{t}_U$.

At the online stage, given suitable image $\hat{n}_j$ selected in the previous section, the proposed method enumerates possible intersecting triangles using the stored association. Let $\tilde{\mathbf{p}}_j$ be the intersecting point on the plane determined by one of the enumerated triangles, whose vertices (i.e., camera positions) are denoted by $\mathbf{t}_1$, $\mathbf{t}_2$, and $\mathbf{t}_3$ (Fig. 10). The barycentric coordinates of $\tilde{\mathbf{p}}_j$ with respect to $\mathbf{t}_1$, $\mathbf{t}_2$, and $\mathbf{t}_3$ is given by

$$\tilde{\mathbf{p}}_j = \alpha\mathbf{t}_1 + \beta\mathbf{t}_2 + \gamma\mathbf{t}_3. \tag{6}$$

We can identify the triangle that intersects the ray from $\mathbf{t}_U$ to $\mathbf{p}_j$ by finding one that satisfies both of the following conditions.

$$\begin{cases} \alpha, \beta, \gamma \geq 0 \\ \alpha + \beta + \gamma = 1 \end{cases}. \tag{7}$$

Color $\mathbf{c}_j$ of the j-th pixel of the novel view image is determined by

$$\mathbf{c}_j = \alpha\mathbf{c}_1 + \beta\mathbf{c}_2 + \gamma\mathbf{c}_3, \tag{8}$$

where $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{c}_3$ are the pixels' colors corresponding to $\mathbf{p}_j$ in images associated with $\mathbf{t}_1$, $\mathbf{t}_2$, and $\mathbf{t}_3$, respectively.

Basically, this coloring process works well; however, it can suffer from the following two problems:

1. It can fail in finding intersecting triangles because (i) the target on the floor cannot be shot from the position below the floor or (ii) the camera identified by $\hat{n}_j$ is not included in the actual intersecting triangle, which can happen if Delaunay triangulation gives obtuse triangles or $\mathbf{p}_j$ is invisible in the image that gives the smallest similarity value, $s_{n,j}$. This failure in finding the triangles is indicated by negative values of at least one of α, β, and γ, i.e.,

   $$\alpha < 0, \quad \beta < 0, \quad \text{or} \quad \gamma < 0 \tag{9}$$

2. The 3D point, $\mathbf{p}_j$, can be invisible in the three images that correspond to the vertices of the intersecting triangle due to occlusion, which is indicated by

   $$v_{n,j} = 0 \tag{10}$$

   for indices n corresponding to those three images.

Considering these problems, we take the following strategy for coloring pixels: If Eq. (9) or (10) holds true, letting $\mathbf{c}'_j$ be the color of the pixel corresponding to $\mathbf{p}_j$ in $\hat{I}'_j$, the j-th pixel's color is given by $\mathbf{c}_j = \mathbf{c}'_j$. Otherwise, we use Eq. (8).

6 Implementation

At the offline stage, we used VisualSFM (Wu, 2013, 2011, 2007) for estimating camera poses of images in S and CMPMVS (Jancosek and Pajdla, 2011) for 3D model reconstruction. For spherical Delaunay triangulation, we used implementation in (Renka, 1997). Foreground extraction was manually done using Adobe Photoshop CS5/CC.

Our implementation of the online stage was built upon OpenGL on Windows 7 OS. For real-time and stable camera position estimation, we used Metaio SDK (Metaio GmbH, 2014). VDGM and most part of VDTM-based color assignment were implemented on CUDA to achieve real-time AR image generation. More specifically, we implemented all processes except for ones related to Eq. (8). This is because Eq. (8) uses all input images and the sizes of the input images are usually large to be stored in the GPU memory. Our method is

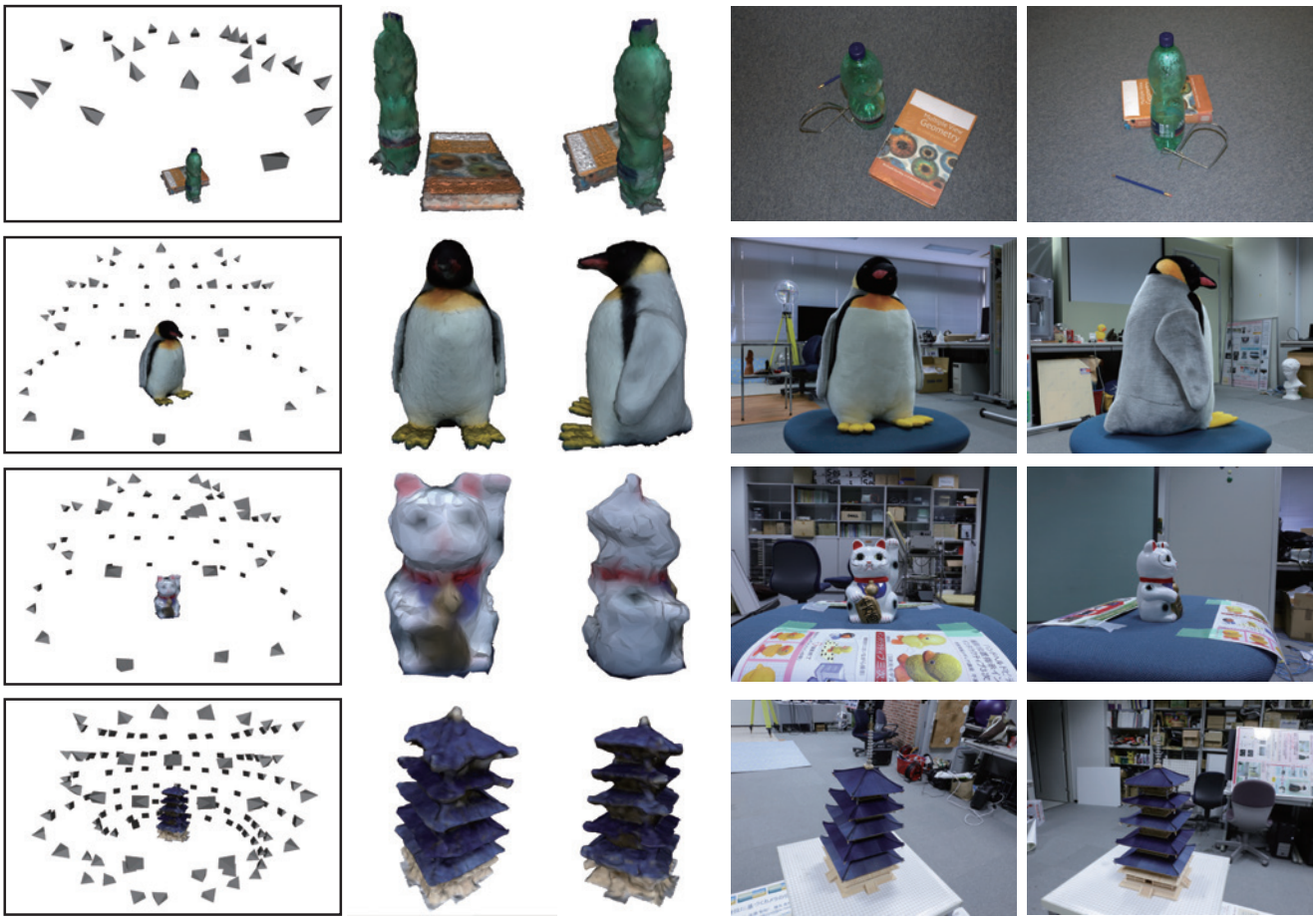


Fig. 11 From top to bottom: datasets Bottle and Book borrowed from (Jancosek and Pajdla, 2011), Stuffed Penguin, Ceramic Cat, and Japanese Tower. Left most column is reconstructed 3D models with estimated camera positions of input images. Second and third columns are reconstructed 3D models and the others are example input images.

implemented on a PC with Intel Core i7-2600 CPU at 3.4 GHz, 16 GB RAM, and NVIDIA GeForce GTX 550 Ti with 1 GB GPU memory.

The parameter θ_{JE} for finding jump edge depends on scaling, viewpoints, and the shape of targets; therefore, it may be determined adaptively. In our implementation, however, we empirically set a constant value $\theta_{\mathrm{JE}} = 0.005$ regardless of viewpoints and targets, because we found that it did not cause severe visual artifacts. The parameter θ_{VC} also depends on the scaling. For simplicity, our implementation evaluates Eq. (3) in the world coordinate system, not in depth obtained from an ordinary rendering pipeline. The parameter θ_{VC} was set to 0.3.

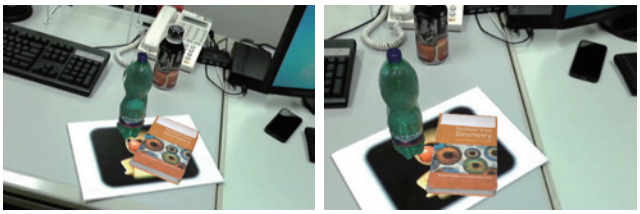
7 Results

This section demonstrates the visual quality of AR images generated from four datasets using our proposed method. Figure 11 shows the reconstructed 3D models and some input images in the datasets, and Table 1 summarizes the specification of the datasets. The Bottle and Book dataset borrowed from (Jancosek and Pajdla, 2011) is challenging because it has only 24 images, which are captured from above the targets, and because the bottle is translucent. The Stuffed Penguin dataset is relatively easy because the 3D model was faithful to the target's shape. The 3D model for Ceramic Cat shows significant missing volumes in the back of its head as can be seen in Fig. 11. We consider this is because specular reflection on its ceramic surface. Japanese Tower is another challenging dataset because it can suffer from excessive jump edges.

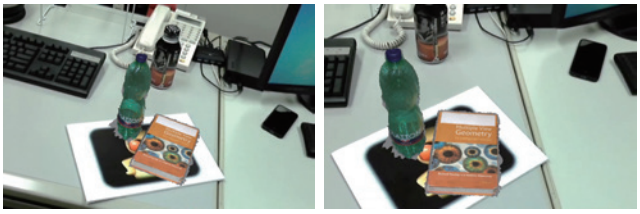
Table 1 Specification of datasets.

	Number of images	Number of faces
Bottle and Book	24	36051
Stuffed Penguin	61	7926
Ceramic Cat	52	3363
Japanese Tower	125	62856

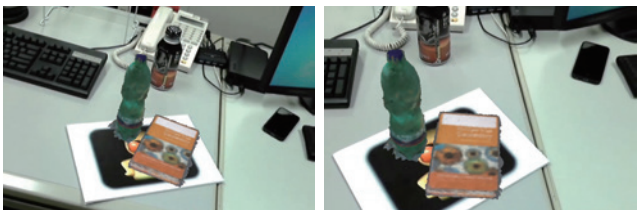
For comparison, we implemented two baselines, i.e., model-based and VDTM-based methods. The model-



Proposed method



VDTM-based method



Model-based method

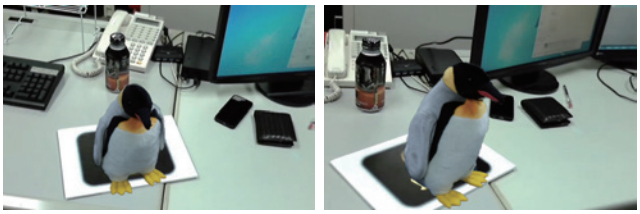
Fig. 12 AR images from Bottle and Book.

Proposed method

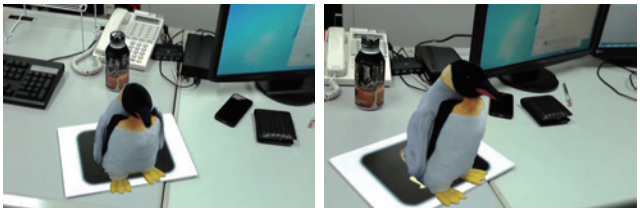
VDTM-based method

Model-based method

Fig. 14 AR images from Ceramic Cat.



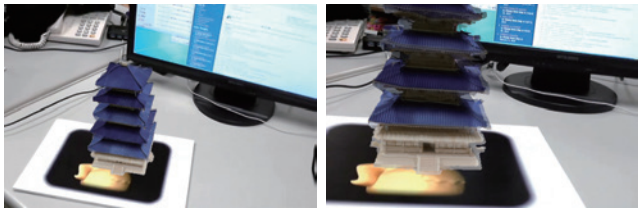
Proposed method



VDTM-based method

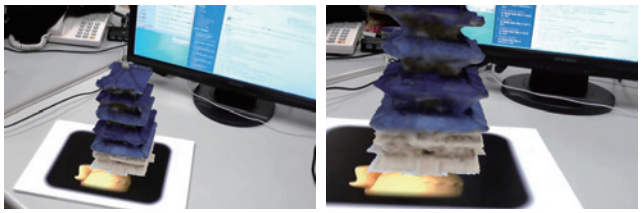
Model-based method

Fig. 13 AR images from Stuffed Penguin.



Proposed method

VDTM-based method



Model-based method

Fig. 15 AR images from Japanese Tower.

based method uses only automatically reconstructed 3D models with colors that were preliminarily assigned to each vertices by (Jancosek and Pajdla, 2011), which is frequently used in AR applications (Azuma, 1997). Using this method, we show reconstructed 3D models themselves. The VDTM-based method applies VDTM to each pixel in the depth map. This is slightly modified version of the original method by Debevec et al (1998), which applies texture to each face of the 3D mesh models. We consider that our modified version gave almost

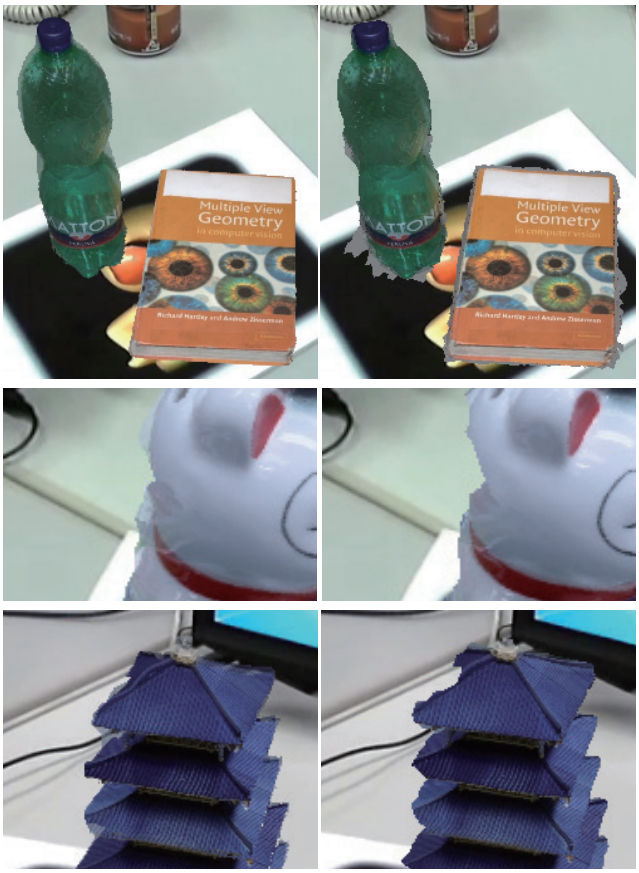


Fig. 16 Close views of targets by proposed method (left) and by model-based method with VDTM (right). From top to bottom: Bottle and Book shown in Fig. 12, Ceramic Cat shown in Fig. 14(b), and Japanese Tower shown in Fig. 15(d).

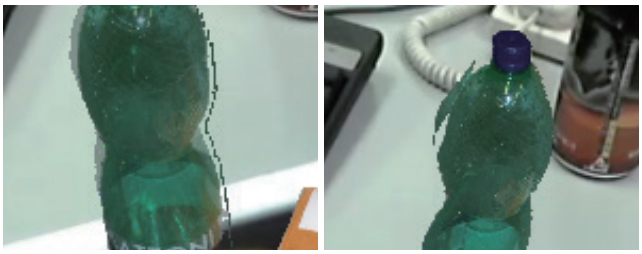


Fig. 17 Visual artifacts in our proposed method.

the same visual quality as the original one. In addition, we did not use the foreground extraction results for this method to emphasize its effectiveness.

Figures 12–15 show example AR images generated by the proposed method and by the baselines. Compared with the VDTM-based method, our results demonstrate that the proposed method reduce visual artifacts due to missing and excessive volumes as well as false high-frequency components in the target boundaries as in the close views shown in Fig. 16. For a viewpoint that is not very close to one of C_n's, the proposed method produced blurry boundaries around the bottle cap and the middle of the bottle in Fig. 16(top left). This is because the proposed method uses three input images to color a single pixel in a novel view image of the target. Each pixel of the generated image is a weighted sum of pixels in these three image, and in general, the target boundaries are not consistent with each other due to the inaccurate 3D models and camera poses.

Reduction of missing volume is significant for the Ceramic Cat dataset: As shown in Fig. 11, the reconstructed 3D model for Ceramic Cat has large missing volume in its back (Fig. 16(middle right)). VDGM fills this missing part of the target (although it is not completely filled), which can be seen in Fig. 16(middle left). For the Japanese tower dataset, the missing volume around the top roof of the tower in the left image of Fig. 16(bottom) was also partly filled. The spire on the top of the roof, however, cannot be regained because it is elongate and is almost completely missing in the reconstructed 3D model.

One of the limitations of the proposed method is that it may give artifacts due to the use of VDGM and VDTM as shown in Fig. 17(left): To expand the target region, VDGM calculates the depth values in Ω'_{E} using Eq. (1), which forces these depth values to be smooth. As results, the 3D points corresponding to the calculated depth values can be accidentally projected to the foreground region of the one of the selected images, generating the artifacts. Another limitation that produced the artifacts in Fig. 17(right) is due to a small number of input images. As described in Section 5.3, the coloring strategy of the proposed method uses only one suitable image when it cannot find an intersecting triangle. In this case, the proposed method cannot reduce the artifacts due to selected image transition as in Fig. 8. Together with the first limitation, the proposed method can produce false shape in Fig. 17(right).

Figures 18(a)–(d) show the timing results. Instead of showing the averaged time of each process, we plotted the graph of number of pixels versus elapsed time per frame for each dataset with a single sequence because it depends much on the target size. As can be seen from Fig. 18(a), the overall timing linearly increased due to VDGM and VDTM-based color assignment. Figure 18(d) implies that VDTM-based color assignment depends on the number of input images. The dominant process was VDTM-based color assignment, followed by VDGM. The depth map generation was done in an almost constant time.

8 Conclusion

In this paper, we have proposed an image-based rendering method for AR image generation, which modifies

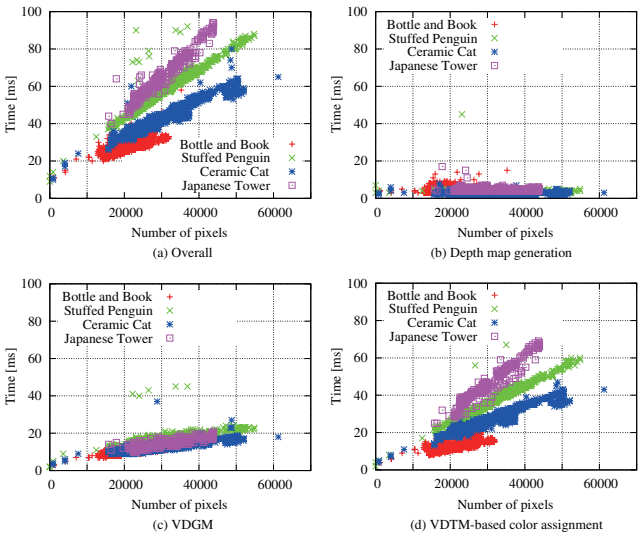


Fig. 18 Timing results.

the geometry (i.e., a rough 3D model) of a target object for a given viewpoint by solving the discrete Poisson equation. Such a view-dependent approach is beneficial because it does not require to construct a sufficiently accurate 3D model of the target object. Our results have demonstrated that the proposed method successfully reduces the visual artifacts due to inaccuracy of the 3D model. For a small number of input images and target object region that consists of less than about 20,000 pixels, the proposed method worked in real-time with our GPU implementation, and the elapsed time for AR image generation increased linearly with respect to the target object size. Our future work includes to remedy the limitations mostly due to false depth values produced by VDGM, and the use of input images in the VDGM process as a hint to limit the regions to be expanded.

Acknowledgment This work is partly supported by Japan Society for the Promotion of Science KAKENHI No. 23240024.

References

Azuma R (1997) A survey of augmented reality. Presence: Teleoperators and Virtual Environments 6(4):355–385

Bastian JW, Ward B, Hill R, Hengel AVD, Dick AR (2010) Interactive modelling for AR applications. In: Proceedings of IEEE International Symposium on Mixed and Augmented Reality, pp 199–205

Buehler C, Bosse M, McMillan L, Gortler S, Cohen M (2001) Unstructured lumigraph rendering. In: Proceedings of ACM SIGGRAPH, pp 425–432

Chaurasia G, Duchene S, Sorkine-Hornun O, Drettakis G (2013) Depth synthesis and local warps for plausible image-based navigation. ACM Transactions Graphics 32(3):30:1–30:12

Davis A, Levoy M, Durand F (2012) Unstructured light fields. Computer Graphics Forum 31(2):305–314

Debevec P, Yu Y, Borshukov G (1998) Efficient view-dependent image-based rendering with projective texture-mapping. In: Proceedings of Rendering Techniques, pp 105–116

Furukawa Y, Curless B, Seitz SM, Szeliski R (2010) Towards internet-scale multi-view stereo. In: Proceedings of 2010 IEEE Conference Computer Vision and Pattern Recognition, pp 1434–1441

Gortler SJ, Grzeszczuk R, Szeliski R, Cohen MF (1996) The lumigraph. In: Proceedings of ACM SIGGRAPH, pp 43–54

Irani M, Hassner T, Anandan P (2002) What does the scene look like from a scene point? In: Proceedings of 7th European Conference Computer Vision, pp 883–897

Jancosek M, Pajdla T (2011) Multi-view reconstruction preserving weakly-supported surfaces. In: Proceedings of 2011 IEEE Conference Computer Vision and Pattern Recognition, pp 3121–3128

Kato H, Billinghurst M (1999) Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: Proceedings of 2nd IEEE/ACM International Workshop on Augmented Reality, pp 85–94

Klein G, Murray D (2007) Parallel tracking and mapping for small AR workspaces. In: Proceedings of 6th IEEE/ACM International Symposium on Mixed and Augmented Reality, pp 225–234

Kolev K, Brox T, Cremers D (2012) Fast joint estimation of silhouettes and dense 3D geometry from multiple images. IEEE Transactions Pattern Analysis and Machine Intelligence 34(3):493–505

Levoy M, Hanrahan P (1996) Light field rendering. In: Proceedings of ACM SIGGRAPH, pp 31–42

Metaio GmbH (2014) Metaio SDK. http://www.metaio.com/products/sdk/

Renka R (1997) STRIPACK: Delaunay triangulation and Voronoi diagram on the surface of a sphere. ACM Transactions Mathematical Software 23(3):416–434

Rother C, Kolmogorov V, Blake A (2004) Grabcut: Interactive foreground extraction using iterated graph cuts 23(3):309–314

Seitz SM, Curless B, Diebel J, Scharstein D, Szeliski R (2006) A comparison and evaluation of multi-view stereo reconstruction algorithms. In: Proceedings of 2006 IEEE Conference Computer Vision and Pattern Recognition, pp 519–528

Wu C (2007) SiftGPU: A GPU implementation of Scale Invaraint Feature Transform (SIFT). http://ccwu.me/vsfm/

Wu C (2011) VisualSFM: A visual structure from motion system. http://ccwu.me/vsfm/

Wu C (2013) Towards linear-time incremental structure from motion. In: Proceedings of International Conference on 3D Vision, pp 127–134