The Journal of the Society for Art and Science, Vol. 14, No. 5, pp. 198-206

Interactive Video Editing for Occluded Object Using Synthetic Aperture Imaging

Takahiro Daimon **Tadahiro Fujimoto** (Member)

Graduate School of Engineering, Iwate University

{h20j56, fujimoto} at cis.iwate-u.ac.jp

Abstract

This paper proposes a new method to enable a user to interactively edit an existing object displayed in a virtual scene on a free-viewpoint video in real time simultaneously while capturing its original real-world scene by a camera array. The proposed method consists of the pure extraction method and the editing method. The pure extraction method is based on synthetic aperture imaging and purely extracts the whole shape of a target object that a user wants to edit on a free-viewpoint video even if the object is partially occluded by other objects. The editing method enables the user to interactively edit the extracted target object by 2D geometric transformations directly on the video in which the object is displayed in front of its background. The background is always displayed as it is in the real-world scene with no influence by the editing in the virtual scene; for example, any empty regions are not caused in the background in the virtual scene by the editing even if the background contains moving objects. Some experimental results show a good ability of the proposed method.

1. Introduction

Many virtual-world imaging techniques have been developed to create a virtual scene to provide us rich experiences. A *Virtual Reality* (*VR*) technique creates a virtual world to walk through in a computer. An *Augmented Reality* (*AR*) technique adds virtual objects to a real-world scene to augment it. Conversely, a *Diminished Reality* (*DR*) technique creates a virtual scene in which unwanted real objects are removed from a real-world scene. A *Mixed Reality* (*MR*) technique combines a real-world scene and a virtual scene to interact with each other. These techniques are included in a more general concept, *Mediated Reality*, in which a real-world scene is altered diversely by a computer. In relation to these techniques, *image-based* or *video-based rendering* techniques have been actively developed. An image-based rendering technique creates a new image of a real-world scene seen from a *virtual free-viewpoint* using multiple images taken from different camera positions without geometric models.

In this paper, we propose a video-based rendering method to provide a combination of AR and DR applications. A user can alter a real-world scene in real time by editing an object in the scene interactively on a *free-viewpoint video* generated from videos taken by multiple cameras of a *camera array*. Our method is based on *light field rendering* and *synthetic aperture imaging* to simulate a single camera with a large aperture lens by a camera array [1-14]. A synthetic aperture imaging method generates a free-viewpoint image, which is called *synthetic aperture image*, so as to focus on a desired object by setting a focal plane at its depth. Even if the object is partially occluded by other objects near the camera array, the occlusion is mitigated on the image such that the whole shape of the "focused" occluded object can be seen through the "blurred" occluding objects. The occluded object is displayed with blurred color impurities of the occluding objects.

Our method has two processes executed on a free-viewpoint video: the *pure extraction* of an object, and the *editing* of the extracted object. In the pure extraction, by focusing on a target object by a focal plane, the object is extracted and solely displayed in front of its background. Even if the object is partially occluded by other objects near the camera array, the whole shape of the occluded object is *purely* displayed without blurred color impurities of the occluding objects. In the editing, the extracted target object can be interactively edited by 2D geometric transformations, such as scaling, rotation, and translation, while its background is always displayed as it is in the real-world scene with no influence by the editing in the virtual scene.

The idea of our method was inspired by many kinds of *image editing* techniques on static images and videos such as *texture synthesis*, *image completion*, *retargeting*, and *reshuffling* [15]. Besides, *background subtraction*, or *foreground extraction*, is a related technique that separates a foreground object from its background on a video [16]. To our knowledge, any previous image editing methods did not allow a user to interactively edit a moving object in a scene directly on a video in real time simultaneously while capturing the scene by cameras. Our method enables interactive real-time editing of a moving object on a video even if it is partially occluded. Usual image editing methods

tackle a laborious problem to fill an empty region in the background caused by removing an object with suitable colors so as to look natural. In our method, the background is always displayed as it is in the real-world scene with no influence caused by editing an object in the virtual scene; any empty regions are not caused in the background in the virtual scene even if the background contains moving objects.

We can expect various applications using the ability of our method to virtually rearrange and remove objects existing in a real-world scene in real time. For example, a remote system, such as teleconference, can be realized in which a user can virtually handle an object existing in a remote place. For various design systems in the 3D space, such as room and landscape design, a designer can virtually rearrange existing objects by trial and error without actually moving them.

2. Related work

A light field of a scene is defined by a set of rays passing through the pixels of images taken from different camera positions. Each ray has the color of the point where the ray hits an object in the scene. Thus, the light field is a data set of colors to describe the scene. The concept of light field rendering was proposed by Levoy et al. [1] and Gortler et al. [2]. They defined a light field as a function of four parameters for a ray passing virtual two planes; the function returns the color of the point hit by the ray. From this idea, Isaksen et al. proposed a synthetic aperture imaging method using a planar camera array to simulate a single camera with a large aperture lens [3]. A synthetic aperture image is generated by averaging images taken by multiple cameras. Its focus and depth of field can be controlled by a focal plane and an aperture filter, and an occluded object can be seen through occluding objects.

While Isaksen's method treated static images, Yang et al. proposed a method to treat videos [4]. Their distributed light field camera system with 64 video cameras generates a synthetic aperture video in real time. Vaish el al. proposed a calibration method for a planar camera array using planar parallax [5], which was improved for a non-planar camera array later [6]. Wilburn et al. developed a large array system with 100 video cameras, which realized various applications including synthetic aperture imaging [7]. Vaish el al. compared four cost functions using stereo, focus, median, and entropy for reconstructing the surface of an object from multiple images when it was partially occluded [8]. Joshi et al. proposed two synthetic aperture imaging methods to track a moving object through occlusions by sweeping a focal plane [9]. Pei et al. also proposed a tracking method using a plane sweep approach by combining synthetic aperture imaging and background subtraction [10]. Their method tracks multiple objects well by detecting their optimal depths and achieves better performance than Joshi's method. Yang et al. proposed a method to display the whole non-planar surface of an occluded object sharply by combining synthetic aperture imaging with multiple layer image fusion [11].

For occlusions, a usual synthetic aperture imaging method displays an occluded object in focus sharply by blurring occluding objects out of focus, not by actually removing them. This relatively mitigates the influence of the occluding objects. Unlike this approach, some methods actually remove occluding objects to display only an occluded object without blurred color impurities of the occluding objects. Lin et al. proposed a method to recover 3D scene layers from a light field of a static scene with occlusions by an iterative optimization algorithm [12]. McCloskey proposed a method to remove a nearby occluding object from distant occluded background in a static image taken by a micro-lens-based light field camera [13]. The method proposed by Yamashita et al. specializes in removing a fence in front of an object in a static scene using multi-focus images [14].

While these methods treat static images, the methods below treat videos. Pei's tracking method can remove an occluding object in front of a tracked object [10]. First, their method detects the optimal depth of the occluding object and its silhouette, which is used to remove its pixels in each camera image. Then, the remaining pixels are used for the occluded tracked object; after detecting its optimal depth, it is displayed purely by synthetic aperture imaging. They did not mention specific performances, such as fps, while the computation complexity was theoretically analyzed in comparison to Joshi's method. Fehrman et al. proposed a method to display an occluded object purely without using synthetic aperture imaging [17]. They used a camera array with five cameras in a planar plus-styled configuration. The central camera is paired with the other four cameras, and the two images of each pair are used to estimate disparity and depth maps by stereo matching. By giving a depth for a desired occluded object, five camera images are combined to display it purely by removing occluding objects using the maps. In experiments, while most occluding objects were removed, some occluding parts remained due to the lack of stereo matching accuracy; particularly many edge parts made noticeable artifacts. The performance was 2 fps with a resolution of 320 by 240 pixels using an Intel Core i7-3610-QM 2.3GHz processor and 8GB memory. Unlike Fehrman's method, our method uses synthetic aperture imaging. While Pei's method uses a plane sweep approach with many focal planes for both an occluding object and an occluded object, our method uses only a single focal plane for an occluded object to distinguish it from occluding objects and display it purely by clustering pixel colors. Our approach is expected to be more efficient and accurate than previous approaches.

3. Synthetic aperture imaging

As shown in Fig. 1, to generate a free-viewpoint image I_v of a target object, each pixel $p_v \in I_v$ has to be given the color of the point $\boldsymbol{X}_v$ on the surface of the object; the point $\boldsymbol{X}_v$ is hit by the ray passing through p_v coming from the free-viewpoint V. If the geometry of the object is known in advance, the point $\boldsymbol{X}_v$ can be easily determined. Then, using camera images I_i of cameras C_i, $i = 1, \cdots, N$, on a camera array, the color $\boldsymbol{C}(p_v)$ of the pixel p_v is given the average of the colors $\boldsymbol{C}(p_i)$ of the pixels $p_i \in I_i$ whose rays hit the point $\boldsymbol{X}_v$.

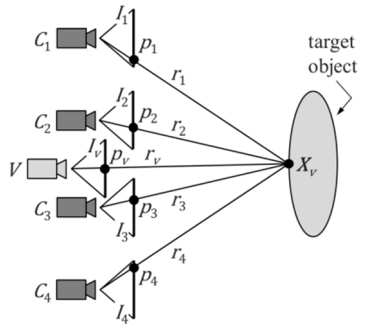
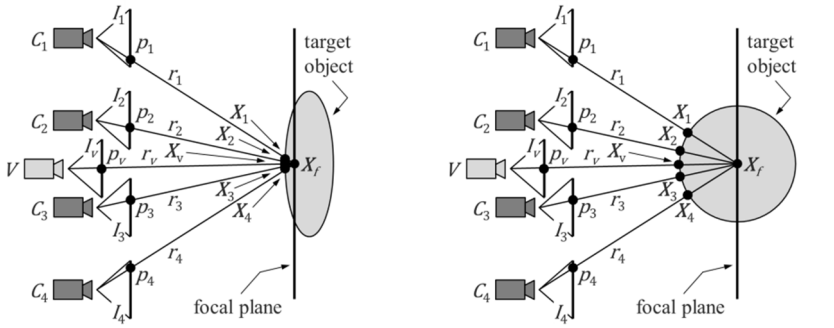


Figure 1. Free-viewpoint image.



(a) (b)

Figure 2. Synthetic aperture image.

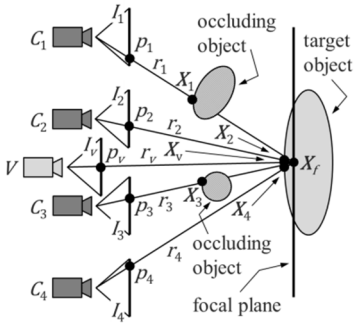
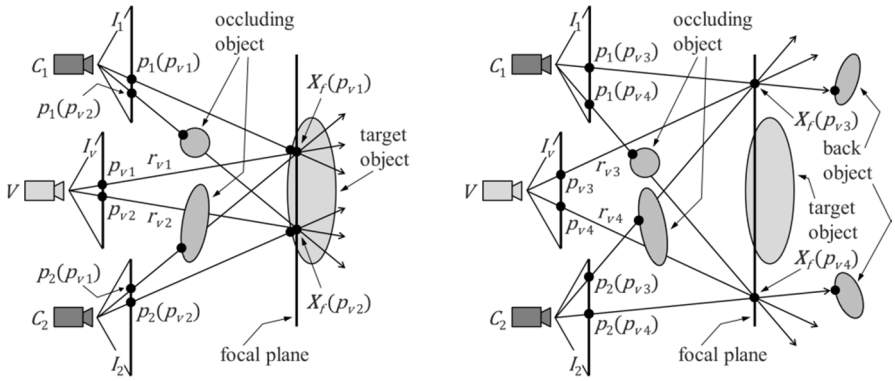


Figure 3. Occlusion problem.



(a) (b)

Figure 4. Relationship of synthetic pixels, camera pixels, and objects.

However, if the object geometry is unknown, the above approach is not available because the object surface point $\boldsymbol{X}_v$ cannot be obtained. One solution is to use a synthetic aperture imaging technique. In this technique, by using a focal plane given by a user, all camera images I_i are projected onto the focal plane and their colors are averaged to generate a synthetic aperture image I_v. On the image I_v, objects near the focal plane appear sharply in focus, and objects far from the plane are blurred out of focus. As shown in Fig. 2, the point $\boldsymbol{X}_f$ on the focal plane projected to a pixel $p_v \in I_v$ can be obtained without knowing the object geometry. By assuming the point $\boldsymbol{X}_f$ to be on the object surface, the color $\boldsymbol{C}(p_v)$ can be obtained from the colors $\boldsymbol{C}(p_i)$ of the pixels $p_i \in I_i$ whose rays r_i intersect the point $\boldsymbol{X}_f$. Let $\boldsymbol{X}_i$ be the points at which the rays r_i hit the object surface. As shown in Fig. 2 (a), if the object surface is near the focal plane, the points $\boldsymbol{X}_v$ and $\boldsymbol{X}_i$ are close to each other near the point $\boldsymbol{X}_f$. Then, the average of the colors $\boldsymbol{C}(p_i)$ is valid for the color $\boldsymbol{C}(p_v)$, which makes the point $\boldsymbol{X}_v$ to be displayed sharply in focus. On the other hand, as shown in Fig. 2 (b), if the object surface is far from the focal plane, the points $\boldsymbol{X}_v$ and $\boldsymbol{X}_i$ are far from each other. Then, the colors $\boldsymbol{C}(p_i)$ do not validly contribute to the color $\boldsymbol{C}(p_v)$, which makes the point $\boldsymbol{X}_v$ to be displayed blurredly out of focus.

4. Pure extraction of occluded object

4.1 Occlusion problem

When occluding objects exist between a camera array and a target object to focus on, the target object is partially occluded from the camera array and each camera captures a part of the target object if the baselines of the cameras are large relative to the sizes of the occluding objects. Then, on a synthetic aperture image I_v generated as explained in section 3, the occluding objects out of focus are blurred, and the target object in focus is seen through the blurred occluding objects. The pureness of the target object on I_v depends on the degree of the occlusion. When the occlusion is severe, the target object appears impurely or almost disappears. The color $\boldsymbol{C}(p_v)$, $p_v \in I_v$, becomes the mixture of the colors $\boldsymbol{C}(p_i)$, $p_i \in I_i$, of the target object and the occluding objects, as shown in Fig. 3. The more pixels p_i capture the target object, the more purely the target object is displayed.

4.2 Pure extraction method

Our method extracts a partially occluded target object by removing blurred color impurities of occluding objects on a synthetic aperture image. Suppose that the surface of a target object is approximately planar and a focal plane is placed close to the surface. Let $\boldsymbol{X}_f(p_v)$ be the point at which the ray of a *synthetic pixel* p_v of a synthetic aperture image I_v intersects the focal plane. Let $p_i(p_v)$, $i = 1, \cdots, N$, be the *camera pixels* of camera images I_i whose rays intersect the

point $\boldsymbol{X}_f(p_v)$. Each camera pixel $p_i(p_v)$ captures one of the three kinds of objects: a target object, an occluding object closer to the camera array than the target object, and a back object farther from the camera array than the target object. A far back object is treated as a part of the background. The rays r_v of some synthetic pixels p_v intersect the target object while the other rays do not intersect due to the finite size of the target object. Fig. 4 shows these two cases (a) and (b). Fig. 4 (a) shows the case (a); a ray r_v intersects a target object, and each of the N camera pixels $p_i(p_v)$ captures either the target object or an occluding object. In Fig. 4 (a), for synthetic pixels p_{v1}, p_{v2}, the pixels $p_1(p_{v1})$, $p_2(p_{v2})$ capture the target object while $p_1(p_{v2})$, $p_2(p_{v1})$ capture occluding objects. The camera pixels $p_j(p_v)$, $\mathrm{j} \in \mathrm{J} \subseteq \{1, \cdots, N\}$, capturing the target object have similar colors $\boldsymbol{C}(p_j(p_v))$ from nearby points on the target object. The other camera pixels $p_k(p_v)$, $\mathrm{k} \in \mathrm{K} = \{1, \cdots, N\} - J$, capturing occluding objects have dissimilar colors $\boldsymbol{C}(p_k(p_v))$ from points far from each other in general. Fig. 4 (b) shows the case (b); a ray r_v does not intersect a target object, and each of the N camera pixels $p_i(p_v)$ captures either an occluding object or a back object. In Fig. 4 (b), for synthetic pixels p_{v3}, p_{v4}, the pixels $p_1(p_{v4})$, $p_2(p_{v3})$ capture occluding objects while $p_1(p_{v3})$, $p_2(p_{v4})$ capture back objects. In this case, all the colors $\boldsymbol{C}(p_i(p_v))$ from far points are generally not similar.

From the above, our pure extraction method needs the two requirements. First, for each synthetic pixel p_v, if its ray r_v intersects the target object, several of the N camera pixels $p_i(p_v)$ need to capture the target object without being occluded by occluding objects. Second, the surfaces of all objects in a scene need to have a variety of colors such that, for each synthetic pixel p_v, it does not happen that some of the N camera pixels $p_i(p_v)$ casually capture similar colors coming from far points. Then, by considering the cases (a) and (b) above, the pure extraction method extracts a target object purely by determining which camera pixels capture the target object as follows.

(a) Among the N camera pixels $p_i(p_v)$, $i = 1, \cdots, N$, for a synthetic pixel p_v, if several camera pixels $p_j(p_v)$, $\mathrm{j} \in \mathrm{J} \subseteq \{1, \cdots, N\}$, have similar colors $\boldsymbol{C}(p_j(p_v))$ to each other, the pixels $p_j(p_v)$ are judged to capture the target object. Then, the pixel p_v should capture the target object, and its color $\boldsymbol{C}(p_v)$ is given the average of the colors $\boldsymbol{C}(p_j(p_v))$. The remaining camera pixels $p_k(p_v)$, $\mathrm{k} \in \mathrm{K} = \{1, \cdots, N\} - J$, are judged to capture occluding objects, and the colors $\boldsymbol{C}(p_k(p_v))$ are discarded.

(b) If few camera pixels have similar colors, all the N camera pixels $p_i(p_v)$ are judged to capture occluding objects and back objects, not the target object. Then, the synthetic pixel p_v should not capture the target object, and its color $\boldsymbol{C}(p_v)$ is given no color.

For the above procedure, it is needed to specify how to determine similar colors among the N colors. There is no reference color obtained in advance to compare with the N colors. Thus, our method utilizes a clustering technique to distinguish similar colors from the others. This clustering technique separates the N colors into some clusters by the distances between the colors in a color space, and the largest cluster having the most colors becomes a candidate cluster for a target object. This clustering technique uses an agglomerative hierarchical method [18], which is one of well-known clustering methods and iteratively constructs clusters. In this method, first, each color is put in a cluster. Then, in each iteration step, two clusters are merged into one cluster. By using the centroid method [18], the two clusters to merge are determined by the Euclidian distance between the centroids of the colors in the two clusters. In the original method, the merging iteration proceeds until only one cluster remains. However, in our method, the iteration stops when a candidate cluster for a target object is obtained, even if more than one cluster remains.

The pure extraction algorithm is described as follows. After step 0, steps 1 to 7 are executed for every synthetic pixel p_v independently.

Step.0 : Define thresholds L_{th} and N_{th}. The value L_{th} is a Euclidian distance to decide whether the cluster merging should be continued. The value N_{th} is the number of colors in a cluster to decide whether a candidate cluster should be accepted for a target object.

Step.1 : Put each color $\boldsymbol{C}(p_i(p_v))$, $i = 1, \cdots, N$, in a cluster A_i, and set the centroid of A_i to the coordinate of the color $\boldsymbol{C}(p_i(p_v))$ in a color space, such as the RGB space. Let $\boldsymbol{A}_{all}$ be the set containing all the clusters.

Step.2 : Let $L(A_s, A_t)$ be the Euclidian distance between the centroids of two clusters A_s and A_t. Obtain the distance $L(A_s, A_t)$ for every pair of clusters A_s, $A_t \in \boldsymbol{A}_{all}$, $A_s \neq A_t$. Then, select the pair of clusters A_{m1}, A_{m2} having the minimum distance $L_{min} = L(A_{m1}, A_{m2})$ among all the distances.

Step.3 : If $L_{min} > L_{th}$, stop the merging iteration and go to Step 6.

Step.4 : Merge the clusters A_{m1} and A_{m2} by moving all the colors in A_{m2} into A_{m1} and removing A_{m2} from $\boldsymbol{A}_{all}$. Then, calculate the centroid of all the colors in A_{m1}.

Step.5 : If only one cluster remains in $\boldsymbol{A}_{all}$, then go to Step 6. Otherwise, go to Step 2.

Step.6 : Among all the clusters in $\boldsymbol{A}_{all}$, select the largest cluster containing the most colors as a candidate cluster A_{can} for a target object. Let N_{can} be the number of the colors in A_{can}.

Step.7 : If $N_{can} \geq N_{th}$, accept the candidate cluster A_{can} as a set of the colors of the target object. Then, give the average of the colors in A_{can} to the color $\boldsymbol{C}(p_v)$. If $N_{can} < N_{th}$, do not accept A_{can}, and give no color to $\boldsymbol{C}(p_v)$.

This algorithm generates a free-viewpoint image I_v on which only the target object is purely extracted without blurred color impurities of occluding objects. If a pixel $p_v \in I_v$ is given a color of the extracted target object, the colors of not less than N_{th} camera pixels $\boldsymbol{C}(p_j(p_v))$, $\mathrm{j} \in \mathrm{J} \subseteq \{1, \cdots, N\}$, contribute to the color $\boldsymbol{C}(p_v)$; the set of camera indices J depends on the pixel p_v. This means that if every part of the target object is captured by any combination of not less than N_{th} cameras then the target object is perfectly extracted by synthesizing the captured colors. As described in section 4.1, a usual synthetic aperture imaging method degrades the pureness of a target object as the occlusion becomes severe. On the other hand, our method

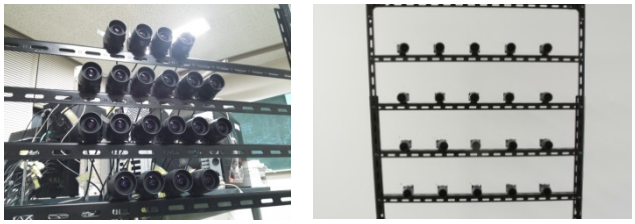
can extract a target object purely even for severe occlusion if every part of the target object is captured by N_t cameras, $N_t \geq N_{th}$, and not less than N_t cameras do not casually capture similar colors of far points on occluding objects. When the occlusion is not severe, our method works well by setting N_{th} to more than $(1/2)N$ even if occluding objects have a uniform color.

5. Editing of extracted object

We want to provide an application for a user to create a virtual scene from a real-world scene so that a desired target object is deformed and rearranged, and unwanted objects are removed by interactive editing operations. This is considered as a kind of combination of AR and DR applications. Thus, our method allows a user to edit an extracted target object purely displayed in front of its background in a virtual scene; other unwanted objects, including occluding objects, are not displayed, and the background is always displayed as it is in the real-world scene with no influence by the editing. This is realized by extracting the background as well as the target object. Suppose that the surface of the background is approximately planar. In the same way as the target object, the pure extraction method is applied to the background by placing another focal plane close to its surface. The whole background is treated as another target object, and objects in front of the background, including the target object to edit, are treated as occluding objects. As a result, the background is extracted purely by removing all objects in front of the background such that the extracted background covers the whole free-viewpoint image with no empty regions. Objects moving near the background are treated as parts of the background.

After the pure extraction method extracts a target object and its background, our editing method starts. First, the extracted target object and the extracted background are put on 2D image layers respectively. Then, according to the editing by a user, a geometric transformation is applied to the target object layer to transform the target object. Currently, a user can interactively scale, rotate, and translate the target object by 2D affine transformations in real time. The editing works well even if the target object is moving, as long as its surface is kept on its focal plane. Finally, the edited target object layer is overlaid on the background layer to generate a free-viewpoint image. As the two layers are independent from each other, the background is not influenced by the editing of the target object even if the background contains moving objects. In particular, any empty regions are not caused by the editing because the background layer always covers the whole image.

A free-viewpoint video is generated using videos captured by all the cameras in real time. The generation of one *free-viewpoint frame image* starts by getting the latest frame images from all the cameras immediately after completing the previous free-viewpoint frame image. One free-viewpoint frame image is generated by the three processes: the pure extraction of a target object and its background, the editing of the target object, and the overlaying of the edited target object on the background. Consecutive free-viewpoint frame images are generated in response to interactive editing operations. A resulting free-viewpoint video shows a virtual scene different from a real-world scene.



(a) (b)

Figure 5. Camera array.

6. Equipment and implementation

Our camera array is shown in Fig. 5. It consists of twenty Firefly MV video cameras made by Point Grey arranged in a 2D planar array configuration. The arrangement of the cameras can be changed arbitrarily, as shown in Fig. 5 (a) and (b). In the experiments of section 7, the arrangement of (b) was used, and each camera captured a video with a resolution of 640 by 480 pixels at 15 fps. All the cameras are connected to one PC with five four-port USB 2.0 interface cards. The PC has an Intel Core i7-3770 3.4GHz processor, a GeForce GTX 670 2GB graphics card, and 8GB memory. The intrinsic and extrinsic parameters of the cameras were obtained by OpenCV calibration functions.

Our software was written in C using Visual Studio 2010 on Windows 7 64bit. Graphics libraries OpenGL and GLSL, an image processing library OpenCV, and a camera library FlyCapture 2.0 were used. In particular, the fragment shader programing by GLSL greatly accelerated our software. The camera parameters obtained by OpenCV calibration functions were used to arrange the N cameras of the camera array in a virtual world made by OpenGL. After starting our software, first, a user interactively places a focal plane close to the surface of a background in the virtual world; the focal plane is implemented by an OpenGL polygon. In the same way, the user also places another focal plane for a target object to edit. The user can move this focal plane anytime so as to select an arbitrary target object to edit. Then, free-viewpoint frame images are generated. To generate one free-viewpoint frame image, first, the latest frame images captured by the N cameras are converted to OpenGL textures. Next, the N *camera image textures* are projected and mapped onto the two focal planes from the camera viewpoints by OpenGL and GLSL projection mapping functions. Then, the two focal planes are rendered for a free-viewpoint by involving the pure extraction method and the editing method, and finally a free-viewpoint frame image is displayed. The two focal planes are rendered separately, and the rendered images are converted to textures respectively. The resulting two *focal plane textures* are used as the two 2D image layers described in section 5. When each focal plane is rendered, the N camera image textures mapped on the focal plane are used for the pure extraction method to extract the

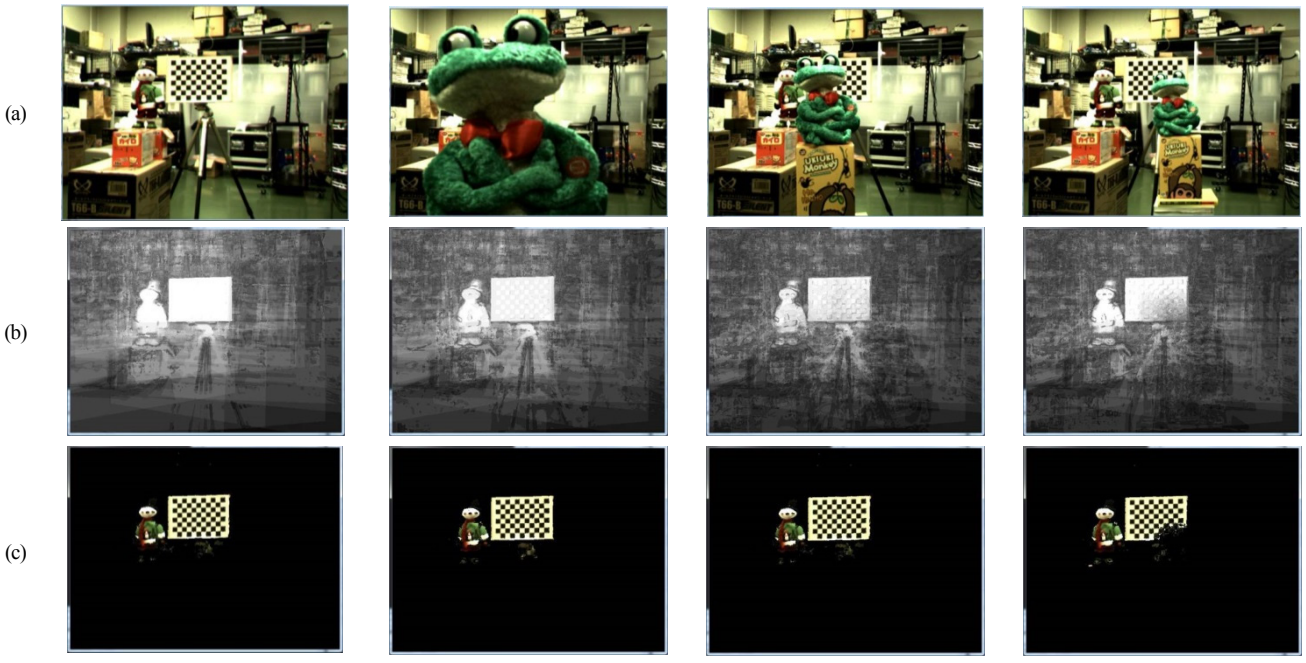


Figure 6. Pure extraction of two target objects (1) with no occluding object, (2) with an occluding object near the camera array, (3) with an occluding object between the camera array and the target objects, and (4) with an occluding object near the target objects.

target object or the background by the algorithm in section 4.2. In the algorithm, the color $\boldsymbol{C}(p_v)$ of every synthetic pixel p_v is determined independently. Thus, it was implemented using GLSL fragment shader programing so as to efficiently process all the synthetic pixels p_v in parallel on GPU. After applying the pure extraction method, one focal plane texture has the extracted target object, and the other has the extracted background. Then, the former is transformed geometrically by an editing operation given by the user, which was implemented using GLSL texture matrices. Finally, a free-viewpoint frame image is rendered by overlaying the transformed focal plane texture of the target object on the focal plane texture of the background.

7. Experimental result

7.1 Experiment of pure extraction method

Fig. 6 shows experimental results by the pure extraction method in four cases (1) to (4). In RGB color space, each of R, G, and B values was normalized to $[0, 1]$, and the threshold L_{th} was set to 0.3. The threshold N_{th} was set to 16 in relation to the number of the cameras, N, being 20. In each case, the image (c) is a free-viewpoint image showing only two target objects extracted by the pure extraction method using twenty images captured by the camera array, while the image (a) is an image captured by a single camera whose viewpoint is the same as the free-viewpoint of (c). The greyscale image (b) shows, for each pixel of the image (c), the number of colors, N_{can}, in a candidate cluster A_{can} in the algorithm. The greyscale color value changes linearly with N_{can}; the color is pure black when N_{can} is zero, and it is pure white when N_{can} is equal to N. Each image in Fig. 6 is a frame image in a video with a resolution of 640 by 480 pixels. Our software extracted the target objects and generated free-viewpoint frame images at about 4 fps. Fig. 6 (1) shows a case in which the chessboard and the doll are target objects given a common focal plane and any occluding object does not exist in front of them. As shown in (1-c), the target objects are extracted purely. The chessboard is extracted almost perfectly; in (1-b), the whole region of the chessboard is almost colored in pure white. This means that the surface of the chessboard is focused almost perfectly by all the N cameras. The doll is extracted partially because its surface is not planar and does not fit the focal plane; the face and upper body are extracted while the lower body is not extracted enough. This is explained by the image (1-b) in which the regions in focus are white and those out of focus are darker; besides, the regions other than the target objects are also darker. The regions out of focus are not extracted by the threshold N_{th} as shown in (1-c). Fig. 6 (2), (3), and (4) show cases in which the frog works as an occluding object in front of the two target objects at different depths. As shown in (2,3,4-a), the frog is put near the camera array in (2), roughly in the middle between the camera array and the target objects in (3), and near the target objects in (4). First, in case (2), the target objects in (2-c) are extracted almost as purely as those in (1-c) are. They are not covered with blurred color impurities of the frog. The frog blocks the views of only a few cameras and the majority of the N cameras capture the target objects, which removes the colors of the frog. This is explained by (2-b) in which the extracted regions of the target objects are nearly white and slightly darker than (1-b). Next, in case (3), the target objects in (3-c) are also extracted almost as purely as those in (1-c) and (2-c) are. However, the image (3-b) is darker than

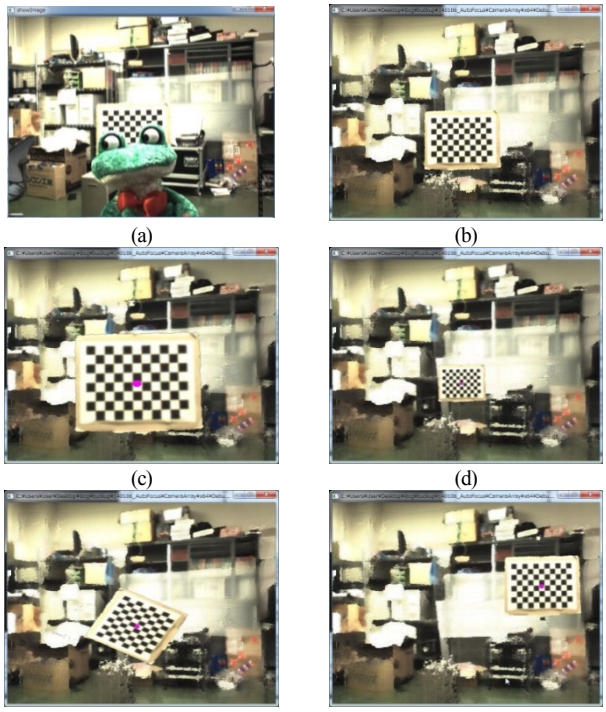


Figure 7. Interactive editing for a static target object.

(2-b) because the frog blocks the views of more cameras. Nevertheless, the extracted parts of the target objects are successfully extracted because these parts are captured by not less than N_{th} cameras. Finally, in case (4), the chessboard in (4-c) is partially extracted, and its lower right part is removed. This is caused by the fact that the frog is too close to the chessboard and the removed part is captured by less than N_{th} cameras, which is found in (4-b). The experiments above show that the pure extraction method works according to the degree of occlusion for the views of cameras; the part of a target object captured by not less than N_{th} cameras is successfully extracted.

7.2 Experiment of editing method

Fig. 7 shows an experimental result by the editing method. A real-world scene is shown in (a), which is a frame image of a video captured by a single camera. The chessboard is a target object occluded by the frog. The images (b) to (f) are frame images of a free-viewpoint video for the same viewpoint as (a). The video was generated in real time by our software for interactive editing by a user simultaneously while capturing the real-world scene by the camera array. Frame images of the video were generated about 3 fps with a resolution of 640 by 480 pixels. In (b) to (f), the chessboard and its background are extracted by the pure extraction method respectively, and other objects, including the frog, are removed. The extracted background is not displayed clearly because the irregular non-planar surface of the whole background consisting of many objects does not fit its focal plane enough. The image (b) is given no editing operation. The images (c) to (f) are given editing operations to the chessboard: scaling up in (c), scaling down in (d), rotation in (e), and translation in (f). The pink point on the chessboard in each image was given by the user as the center of each geometric transformation. Each operation transforms only the chessboard while it is kept extracted purely. The background is not influenced by the editing, and any empty regions are not caused.

Fig. 8 shows another experimental result. The images (a) to (e) are frame images selected from an experimental video on which the PC monitor was captured during the execution of our software; the images are arranged in temporal order from (a) to (e). Our software displays two windows on a PC monitor; the left window shows a video of a real-world scene captured by a single camera, and the right window shows a free-viewpoint video of a virtual scene generated by our method on which a user is allowed to edit objects interactively using a mouse and a keyboard. A user can move a free-viewpoint on the right window although the free-viewpoint was fixed in this experiment to compare videos on both windows by the same viewpoint, which was given by a fixed single camera of the left window. A user can edit objects in a scene directly on a free-viewpoint video in real time simultaneously while capturing the scene by a camera array. In the real-world scene shown in the left windows in Fig. 8, while a doll is dancing at the center as shown in (a) to (e), a man enters into the scene, stands near a shelf in the background, and swings his arms as shown in (d) and (e). In the virtual scene, as shown in the right windows of (a) to (c), the doll and the background are extracted, and the doll is edited as a target object by two editing operations; the doll is scaled down in (b), and is translated onto the shelf in (c). As a result, as shown in the right windows of (d) and (e), after the man entered, he swings his arms under the small doll dancing on the shelf. By comparing the left and right windows in (c), it is found that the background is displayed as it is in the real-world scene with no influence by the editing of the doll in the virtual scene; the background region that appears by translating the doll is completed with appropriate colors in the right window, while the region continues to be occluded by the doll in the left window. In (d) and (e), the man is treated as a part of the background. In addition to the background region described above in (c), the whole region of the moving man is also completed in the right windows, while the region is partially occluded by the doll in the left windows. This shows that the background is displayed as it is in the real-world scene even if it contains a moving object. A careful comparison between the left and right windows shows that the free-viewpoint video in the right window does not perfectly synchronize with the video in the left window because of some delay caused by the process to generate a free-viewpoint frame image after getting frame images from all the cameras. The free-viewpoint video of Fig. 8 was generated about 3 fps with a resolution of 640 by 480 pixels as well as Fig. 7. This performance is not enough for real-time interaction, although its resolution is higher than the resolution of 320 by 240 pixels in many other applications. Besides, as well as Fig. 7, the extracted background, including the man, is not displayed clearly because the irregular surface of the background does not fit its focal plane enough. These problems should be solved in future. The video of Fig. 8 is found at the journal site and the following

Figure 8. Interactive editing for a moving target object in a dynamic scene.

site:
http://www-cg.cis.iwate-u.ac.jp/~fujimoto/demo/video-edit-ASnico2015/video-edit-ASnico2015.wmv

8. Conclusion

We proposed a new method to enable a user to interactively edit an existing object displayed in a virtual scene on a free-viewpoint video in real time simultaneously while capturing its original real-world scene by a camera array. Our pure extraction method based on synthetic aperture imaging can extract the whole shape of a target object purely even if it is partially occluded by other objects. By purely extracting a target object and its background, our editing method enables a user to create a virtual scene by editing the target object interactively, while the background is displayed as it is in the real-world scene with no influence by the editing even if it contains moving objects.

Currently, only a single focal plane can be used to extract a target object. It should be improved so as to use multiple focal planes to edit more than one target object independently. In addition, currently, multiple objects whose surfaces are on a focal plane are extracted together, as shown in Fig. 6, regardless of user's intention. It should be also improved so that a user can extract individual objects separately. Besides, it is desired to use non-planar focal surfaces to fit irregular object and background surfaces. Currently, a target object to edit and objects in its background are allowed to move as long as their surfaces are kept on their focal planes. The tracking of the extracted objects moving freely without such a restriction is also a challenging problem [9, 10]. The current 2D affine transformations should be extended to 3D geometric transformations. More efficient processing is required for real-time high performance.

Reference

[1] M. Levoy and P. Hanrahan, Light Field Rendering, Proc. of ACM SIGGRAPH 1996, pp. 31-42, 1996.

[2] S. J. Gortler, R. Grzeszczuk, R. Szeliski and M. F. Cohen, The Lumigraph, Proc. of ACM SIGGRAPH 1996, pp. 43-54, 1996.

[3] A. Isaksen, L. McMillan and S. J. Gortler, Dynamically Reparameterized Light Fields, Proc. of ACM SIGGRAPH 2000, pp. 31-42, 2000.

[4] J. C. Yang, M. Everett, C. Buehler and L. McMillan, A Real-Time Distributed Light Field Camera, Proc. of the 13th Eurographics Workshop on Rendering 2002, pp. 77-86, 2002.

[5] V. Vaish, B. Wilburn, N. Joshi and M. Levoy, Using Plane + Parallax for Calibrating Dense Camera Arrays, Proc. of IEEE Computer Vision and Pattern Recognition 2004, pp. 2-9, 2004.

[6] V. Vaish, G. Garg, E. Talvala, E. Antunez, B. Wilburn, M. Horowitz and M. Levoy, Synthetic Aperture Focusing using a Shear-Warp Factorization of the Viewing Transform, Proc. of IEEE Workshop on Computer Vision and Pattern Recognition 2005, 2005.

[7] B. Wilburn, N. Joshi, V. Vaish, E. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz and M. Levoy, High Performance Imaging Using Large Camera Arrays, Proc. of ACM SIGGRAPH 2005, pp. 765-776, 2005.

[8] V. Vaish, R. Szeliski, C. L. Zitnick, S. B. Kang and M. Levoy, Reconstructing Occluded Surfaces using Synthetic Apertures: Stereo, Focus and Robust Measures, Proc. of IEEE Computer Vision and Pattern Recognition 2006, pp. 2331-2338, 2006.

[9] N. Joshi, S. Avidan, W. Matusik and D. J. Kriegman, Synthetic Aperture Tracking: Tracking through Occlusions, Proc. of IEEE Inter-

national Conference on Computer Vision 2007, pp. 1-8, 2007.
[10] Z. Pei, Y. Zhang, T. Yang, X. Zhang and Y.-H. Yang, A Novel Multi-object Detection Method in Complex Scene using Synthetic Aperture Imaging, Pattern Recognition, vol. 45, pp. 1637-1658, 2012.
[11] T. Yang, X. Zhang, L. Ran, R. Yu and R. Xi, Camera Array Synthetic Aperture Focusing and Fusion based Hidden Object Imaging, Intelligent Science and Intelligent Data Engineering, vol. 7202, pp. 660-667, 2012.
[12] Y. Lin, I. Tosic and K. Berkner, Occlusion-aware Layered Scene Recovery from Light Fields, Proc. of IEEE International Conference on Image Processing 2013, pp. 295-299, 2013.
[13] S. McCloskey, Masking Light Fields to Remove Partial Occlusion, Proc. of IEEE International Conference on Pattern Recognition 2014, pp. 2053-2058, 2014.
[14] A. Yamashita, A. Matsui and T. Kaneko, Fence Removal from Multi-Focus Images, Proc. of IEEE International Conference on Pattern Recognition 2010, pp. 4532-4535, 2010.
[15] C. Barnes, E. Shechtman, A. Finkelstein and D. B Goldman, PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing, Proc. of ACM SIGGRAPH 2009, pp. 24(1)-(12), 2009.
[16] A. Sobral and A. Vacavant, A Comprehensive Review of Background Subtraction Algorithms Evaluated with Synthetic and Real Videos, Computer Vision and Image Understanding, vol. 122, pp. 4-21, 2014.
[17] B. Fehrman and J. McGough, Handling Occlusion with an Inexpensive Array of Cameras, Proc. of IEEE Southwest Symposium on Image Analysis and Interpretation 2014, pp. 105-108, 2014.
[18] H. Mucha and H. Sofyan, "9. Cluster Analysis", http://sfb649.wiwi.hu-berlin.de/fedc_homepage/xplore/tutorials/xaghtmlframe142.html, 2003.

Takahiro Daimon received a BE and an ME in information engineering from Iwate University in 2012 and 2014, respectively. His research interests include computer graphics and computer vision. He currently works at SAXA, Inc.

Tadahiro Fujimoto is currently an associate professor in the Department of Computer and Information Sciences at Iwate University. His research interests include computer graphics and computer vision. He received a BE in electrical engineering, and an ME and Ph.D. in computer science from Keio University in 1990, 1992, and 2000, respectively. He worked at Mitsubishi Research Institute from 1992 to 1995. He was a research associate in the Department of Computer and Information Sciences at Iwate University from 1999 to 2002, and a lecturer from 2002 to 2005. He is a member of SAS Japan, IEICE Japan, IPS of Japan, IEEE and ACM.