Augmented Reality Marker Hiding with Texture Deformation

Norihiko Kawai, *Member, IEEE,* Tomokazu Sato, *Member, IEEE,* Yuta Nakashima, *Member, IEEE,* and Naokazu Yokoya, *Member, IEEE*

Abstract—Augmented reality (AR) marker hiding is a technique to visually remove AR markers in a real-time video stream. A conventional approach transforms a background image with a homography matrix calculated on the basis of a camera pose and overlays the transformed image on an AR marker region in a real-time frame, assuming that the AR marker is on a planar surface. However, this approach may cause discontinuities in textures around the boundary between the marker and its surrounding area when the planar surface assumption is not satisfied. This paper proposes a method for AR marker hiding without discontinuities around texture boundaries even under nonplanar background geometry without measuring it. For doing this, our method estimates the dense motion in the marker's background by analyzing the motion of sparse feature points around it, together with a smooth motion assumption, and deforms the background image according to it. Our experiments demonstrate the effectiveness of the proposed method in various environments with different background geometries and textures.

Index Terms—Marker hiding, Diminished reality, Texture deformation.

1 Introduction

With the commodification of smartphones, tablets, and portable game consoles, applications using augmented reality (AR) are becoming available to ordinary users, such as furniture arrangement simulation and AR games with virtual characters, overlaying various objects on a real-time video stream capturing a real environment. These applications need to estimate the camera pose in real-time, and various approaches have been employed, e.g., simultaneous localization and mapping (SLAM)- and AR marker-based approaches [1], [2], [3], [4]. Although the SLAM-based approaches [2], [3], [4] have been intensively studied in these days, the marker-based approach [1] is widely and practically used because of its easiness of putting virtual objects on users' desired positions and its robustness against various textures and shapes in the scene. However, the marker-based approach requires the AR markers to be visible, which may hinder seamless fusion of the real environment and virtual objects.

For this problem of the AR marker-based approach, some methods for AR marker hiding have been proposed, which aim to visually remove the markers from a real-time video stream. Siltanen et al. [5] synthesizes a background image by mixing several pixel values around the marker for each frame and replaces the marker region with it. This simple method works quite fast but may cause significant visual artifacts, especially on complex textures. To overcome the problem, Korkalo et al. [6] and Kawai et al. [7] proposed AR marker hiding based on the planar assumption, in which the AR marker and the background surface are both on the same plane. They generate a background image by applying an image inpainting method, which estimates background images behind some objects from their surrounding regions, to a marker region in a certain frame in a real-time video stream, and overlay it on each subsequent frame with transforming it with a homography matrix. They also adjust image intensities in the generated background for reducing discontinuities around the boundary between the marker region and its surrounding, which are caused by changes in lighting conditions. However, these methods may still suffer from discontinuities on the boundary because the marker is often placed on a nonplanar surface or the marker is attached to a thick base, such as cardboard, so that the marker cannot bend.

In this work, considering a scenario in which a user wants to hide a marker in AR applications without much burden (e.g., measuring an accurate shape behind a marker by capturing the scene from multiple viewpoints before placing the marker), we propose a method for AR maker hiding, which is an extension of [8], with which a user can visually remove a marker in the real-time video stream under nonplanar background geometry and thickness of the marker base. Our method obtains a background image by capturing a single image before placing a marker or applying image inpainting to an image with a marker. Given a new frame from the real-time video stream, it overlays the background image on the marker region with geometric deformation and photometric adjustment. In order to relax the planar assumption employed in [6], [7], our method estimates pixel-wise motion of the marker's background to deform the background image according to it and overlays the deformed image on real-time frames. This approach reduces discontinuities on the boundary between the marker region and its surrounding.

The rest of this paper is organized as follows. The next section briefly summarizes related work and our contribu-

- *N. Kawai is with the Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma, Nara, Japan, 630-0192.. E-mail: norihi-k@is.naist.jp*
- *T. Sato, Y. Nakashima and N. Yokoya are Nara Institute of Science and Technology.*

tion. Sections 3 to 6 describe the proposed method. In Section 7, we show experimental results and give discussion. We conclude this paper in Section 8.

2 Related Work and Contribution

For AR marker hiding, using the diminished reality technology, which visually removes a variety of real objects in real time, is a promising approach. On the other hand, real-time texture deformation, which has been tailored for various AR applications overlaying textures on objects with nonplanar geometry, is also a possible approach. In this section, considering the scenario of this study, we first describe some methods for diminished reality and texture deformation. We then summarize the contribution of this paper.

2.1 Diminished Reality

Methods for diminished reality can be classified into three categories: ones using multiple cameras, using preliminarily captured background images, and using image inpainting.

The methods using multiple cameras [9], [10], [11], [12], [13] are mainly designed for applications, such as a see-through system to remove view-blocking buildings for traffic safety [9], [13]. They capture a scene from different camera positions simultaneously, and remove objects in an image taken with one of the cameras using the background images taken with the other cameras. They are basically inapplicable to AR marker hiding because AR markers are usually put on such an object as a table, a wall, and a floor, and thus the background is not visible from any cameras.

As the methods using preliminarily captured background images, the method in [14] uses the previously captured image that contains the background due to motion disparity. In [15], [16], users or system providers preliminarily capture the multiple background images with physically removing occluding objects. The method in [17] collects images capturing the target scene from the Internet to obtain background images. These methods then overlay the background image on target objects' regions in the real-time frames to visually remove the target objects. The main challenges in these methods are how to find an image suitable for a current frame from a number of background images and how to transform the background image for seamlessly overlaying it on the frame. Lepetit et al. [14] calculate 3D positions of feature points in the scene and apply Delaunay triangulation to the projected 2D coordinates of the feature points on each frame. When a triangle contains the target objects, the method searches previously captured images for the nearest frame with the background and replace the texture in the triangle with the background image using a homography transformation. Cosco et al. [15] and Mori et al. [16] select background images on the basis of the view-dependent texture mapping criterion [18], assuming that the background geometry is known or estimated as a mesh model. Takeda et al. [19] presume that a user takes a distant landscape video, and the background image captured in the nearest frame is overlaid on occluders' regions with homography based on SIFT feature matching [20]. Li et al. [17] search the Internet for the images that were captured from the position close to the current frame, and transform the image using a homography matrix. In any case, these methods are required to satisfy one of the following conditions: (i) the background geometry is known or reconstructed with high accuracy, (ii) a background object is distant enough so that its transformation can be approximated by homography, and (iii) the images are densely captured from various viewpoints. Otherwise, it is difficult to seamlessly overlay the background images on real-time frames.

The methods using image inpainting can be further classified into ones that generate a background image for each frame [21], [22] versus one that generates a background image in a certain frame and adjusts it geometrically and photometrically for other real-time frames [23]. The former can generate a plausible background image for each frame in many cases, but applying image inpainting to each frame may cause temporal inconsistency in the target region. Assuming that the background geometry is approximated by multiple local planes, the latter transforms a background image generated for a certain frame using homography for each plane in each frame. However, this method may also suffer from the texture discontinuities on the boundary between the target region and its surrounding if the assumption about the background geometry is not satisfied.

2.2 Texture Deformation

Real-time texture deformation methods using images captured with an RGB camera have been developed for AR applications such as virtual clothing. These methods first set an initial surface mesh to a target object. They then estimate the deformation of a target object and transfer textures to the object according to the deformation. Ehara et al. [24] proposed to learn the relationships between T-shirts' silhouettes and their deformations for applying a texture on a T-shirt. While this method needs a T-shirt with many markers on it for learning deformations, Pilet et al. [25] estimate the surface deformation by tracking feature points without using markers. Hilsmann et al. [26] use optical flow to estimate the deformation of surface instead of feature points.

In these AR applications based on texture deformation, misalignment by several pixels is not a crucial issue because, especially in such AR applications, the texture transferred to the target object is very different from the object's original texture and users usually do not care about such a small misalignment. In diminished reality applications, on the other hand, users obviously notice even small misalignment of textures because it causes the discontinuities in homogeneous textures. Therefore, misalignment can be perceptually more severe for diminished reality applications than AR. Considering this point, the method in [25], for example, is not applicable to diminished reality applications because it can give a few pixel errors of correspondences and thus discontinuities can be significant.

2.3 Our contribution

In the scenario of AR marker hiding, existing diminished reality approaches using preliminarily captured background images can render a background image of scenes with

various textures and geometries from an arbitrary viewpoint because they assume that the geometry is known or accurately reconstructed. However, a user can put a marker on an arbitrary place and thus background geometry is unknown in general. Also, acquiring background geometry using, e.g., muti-view stereo [27], usually takes much time, and its accuracy largely depends on the amount of informative textures in the scene (e.g., it is difficult to reconstruct geometry of scenes with stripe patterns and no texture), which may hinder the users from using it.

On the other hand, image inpainting-based methods require no preliminary preparations but the planar assumption in the background geometry. This strong assumption often results in discontinuities of textures on the boundary between the marker region and its surrounding.

Real-time texture deformation methods can overlay textures on nonplanar objects by estimating their geometry from images. Since these methods are designed for AR applications, misalignment between the real objects and overlaid objects may be acceptable; however, in diminished reality applications, it is not acceptable because even small misalignment is noticeable. In addition, since the deformation methods use a mesh composed of sparse points and triangles, the resolution of mesh also has influence on the misalignment.

Although the proposed method works well for a little limited types of background geometries compared to the approaches using preliminarily captured background images with known or accurately reconstructed geometry, it can achieve AR marker hiding without measuring the background geometry. The main contributions of this paper are as follows:

(1) We introduce texture deformation for AR marker hiding to deal with nonplanar background geometry, marker thickness, and a variety of textures.
(2) We modify a feature detection method and propose a tracking method for AR marker hiding, which provide a guide for compensating texture displacement without knowledge of background geometry.
(3) We design an energy function for real-time and pixel-wise texture deformation with maintaining continuity of texture. Our energy function can be quickly minimized using GPUs.

3 Overview of AR Marker Hiding

Figure 1 shows the flow diagram of our AR marker hiding. The proposed method consists of two processes: (I) obtaining a background image and (II) hiding an AR marker in each frame in a real-time video stream. Note that we describe our method assuming that our target AR marker is a square one which is used for some AR libraries such as ARToolkit [1]. However, any shape of the marker is acceptable as long as a homography matrix for rectifying an image is appropriately calculated.

In process (I), our method first obtains a background image without a marker by either preliminarily capturing an actual background or applying an image inpainting method to the AR marker region. As illustrated in Fig. 2, when preliminarily capturing the background image, we first capture a background image (I-1) and subsequently

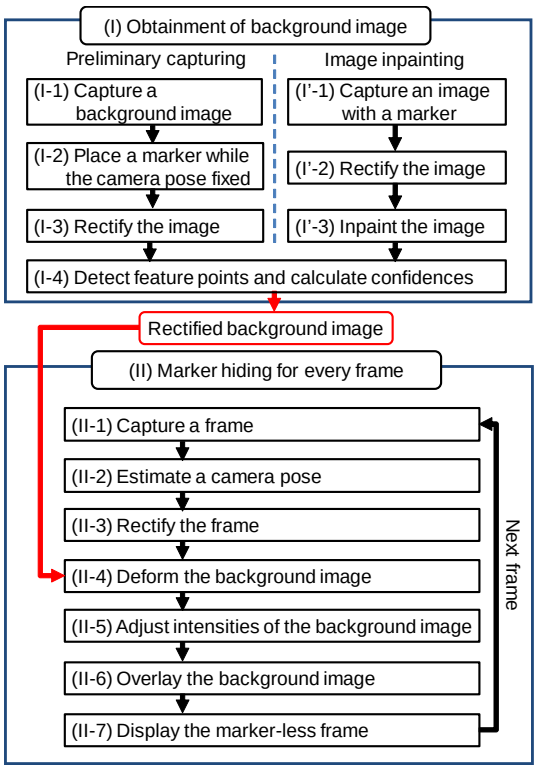


Fig. 1. Flow diagram of the proposed marker hiding.

place a marker while the camera pose fixed (I-2). Next, it finds the homography matrix to transform the image with the AR marker so that the marker can be square. The homography matrix then transforms the background image captured in (I-1) to obtain a rectified background image B (I-3). For the image inpainting case, we follow our previous marker hiding method [7]. Specifically, we first place a marker and capture an image with a marker (I'-1). The captured image is rectified so that the marker can be square (I'-2). An image inpainting method then generates the rectified background image B (I'-3). Our method then detects feature points around the marker region in rectified background image B and calculates the confidences of the feature points (I-4).

In process (II), our method visually removes the marker in a real-time video stream. It first acquires a frame from the real-time video stream (II-1) and estimates the camera pose using the marker in the frame (II-2). The frame is rectified as in process (I) using the homography matrix that is calculated so that the marker's shape and size in the rectified frame can be the same as those in the image obtained in process (I) (II-3). Next, we find the pixels in the rectified frame that correspond to the feature points detected in B (II-4), deforming B on the basis of the feature point correspondences (II-4). Poisson blending [28] then adjusts RGB intensities of the deformed image to compensate for the difference in intensities between the rectified frame and the deformed background image (II-5). The background image is transformed using the inverse of the homography matrix obtained in (II-3) and is overlaid on the marker in the original frame (II-6). Finally, the resulting frame is displayed (II-7). It should be noted that processes (I) and (II) can be

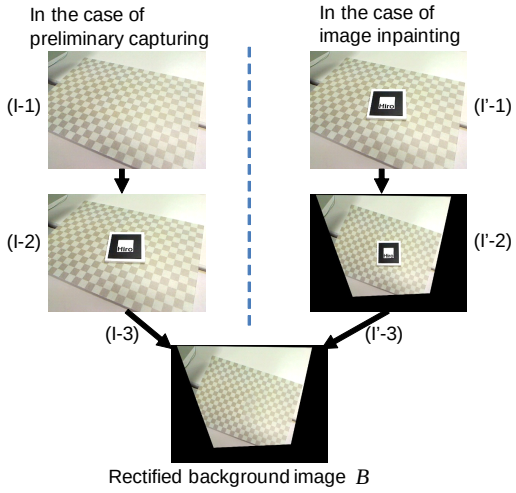


Fig. 2. Obtainment of rectified background image B.

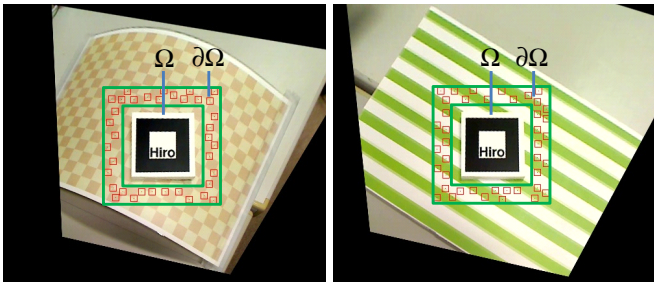


Fig. 3. Examples of detected feature points in region $\partial\Omega$.

done simultaneously for the image inpainting case as in [7].

In the following sections, we describe feature point detection and confidence calculation in (I-4), real-time background image deformation (II-4), and intensity adjustment of the background image (II-5).

4 Detection of Feature Points and Calculation of Confidence

In (I-4), we first determine marker region Ω, which contains the AR marker region and its surrounding so that Ω is slightly larger than the actual marker region in the rectified background image. We also determine the marker's surrounding region $\partial\Omega$ which is the relative complement of Ω in its dilated region by l pixels, as shown in Fig. 3. We then detect feature points in $\partial\Omega$ and calculate the confidence for them.

A number of methods have been proposed so far for feature point detection, such as Harris [29], GFTT [30], SIFT [20], SURF [31], and FAST [32]. These methods use a certain criterion for each pixel and extract a pixel as a feature point if, e.g., the criterion for that pixel exceeds a preliminarily determined threshold. However, such methods often result in failure to extract feature points useful for background image deformation. For example, GFTT [30] with a certain threshold could not extract feature points around corners in the bottom-right region of Fig. 4 because the contrast is relatively low and thus the criterion values for pixels in this

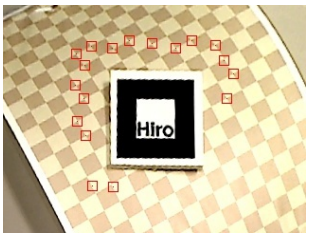


Fig. 4. Example of feature detection using a certain threshold.

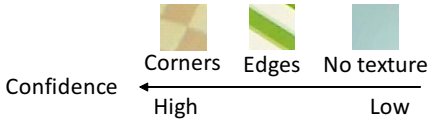


Fig. 5. Confidence of feature point.

region do not exceeds the threshold. In addition, feature points by these methods are not uniformly distributed in general but concentrated in regions with rich texture. Also, they basically detect feature points only at or around corners, and few feature points are detected on straight edges because they are not unique (i.e., the aperture problem).

In order to alleviate discontinuities in textures, our feature detector should satisfy the following requirements.

- Feature points should distribute as uniformly as possible.
- The number of feature points should be sufficiently large for accurate motion interpolation.
- Feature points should be detected not only on corners as shown in Fig. 3(left) but also even on straight edges if there are no corners around them as shown in Fig. 3(right).

In addition, each feature point should have a confidence as a measure of uniqueness as shown in Fig. 5 for our texture deformation described later.

Considering these requirements, our proposed method employs GFTT [30] with some modifications as our feature point detector. GFTT provides an eigenvalue for each pixel as a criterion, which is the smaller one out of two eigenvalues according to the direction of texture, and we use it as the confidence as well. Specifically, to achieve uniform distribution of feature points and not to miss useful feature points, we first calculate the criterion values for all the pixels in $\partial\Omega$, and then sequentially test each pixel in descending order of their criterion values if it satisfies the condition that the distance between the pixel and any feature point obtained so far is larger than L (where L is a constant). If a pixel satisfies the condition, we use the pixel as a feature point. Finally, we normalize each feature point's criterion value so that the highest value can be one, and use it as the confidence for that feature point. It should be noted here that we do not use a preliminaryly determined threshold for detecting feature points unlike common use of feature detectors. Hereinafter, we denote the k-th feature point in $\partial\Omega$ in background image B by $\mathbf{x}_k^B$.

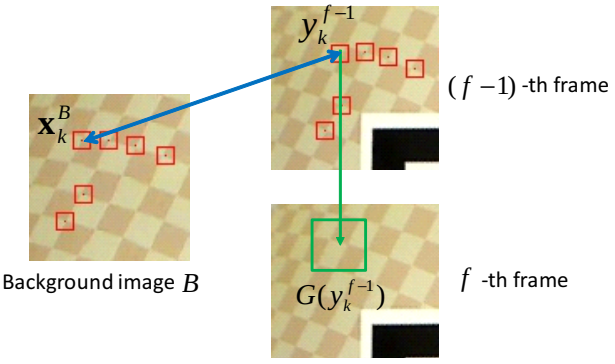


Fig. 6. Illustration of determination of correspondences.

5 Deformation of Background Image Based on Motion Interpolation

Process (II-4) first makes correspondences between feature points selected in the marker's surrounding region $\partial\Omega$ in rectified background image B and pixels in the rectified frame. This process then interpolates the motion in the marker region Ω and its surrounding region $\partial\Omega$ using the correspondences and deforms the rectified background image on the basis of the interpolated motion. In the following sections, we describe the background image deformation for the f-th frame.

5.1 Correspondence in marker's surrounding region

We determine pixel $\mathbf{y}_k^f$ in the rectified f-th frame corresponding to feature point $\mathbf{x}_k^B$ in region $\partial\Omega$ in rectified background image B, basically by finding a region in the f-th frame similar to the local patch around $\mathbf{x}_k^B$. Considering that our method is in a sufficiently high frame rate, we presume that $\mathbf{y}_k^f$ is in the region $G(\mathbf{y}_k^{f-1})$ centered at $\mathbf{y}_k^{f-1}$ as shown in Fig. 6. Region $G(\mathbf{y}_k^{f-1})$ can be small because, even without the planner assumption, a point sufficiently close to the marker remains at almost the same position regardless of the camera motion thanks to the rectification of input images.

Here, assuming the motions of neighboring feature points are similar, our method finds pixel $\mathbf{y}_k^f$ corresponding to $\mathbf{x}_k^B$ from the limited search region $G(\mathbf{y}_k^{f-1})$ in descending order of $\mathbf{x}_k^B$'s confidence by

$$\mathbf{y}_k^f = \underset{\mathbf{y}' \in G(\mathbf{y}_k^{f-1})}{\arg\max} \frac{NCC(\mathbf{x}_k^B, \mathbf{y}')}{1 + \sum_{\mathbf{x}_l^B \in M_{\mathbf{x}_k^B}} D_{\mathbf{s}}(\mathbf{t})}, \tag{1}$$

where $\mathbf{s} = \mathbf{x}_k^B - \mathbf{x}_l^B$ and $\mathbf{t} = (\mathbf{y}' - \mathbf{x}_k^B) - (\mathbf{y}_l^f - \mathbf{x}_l^B)$. $NCC(\mathbf{x}_k^B, \mathbf{y}')$ is the normalized cross correlation between the patch centered at pixel $\mathbf{x}_k^B$ in rectified background image B and the patch centered at pixel $\mathbf{y}'$ in the rectified f-th frame. $M_{\mathbf{x}_k^B}$ is the set of the feature points whose confidences are higher than $\mathbf{x}_k^B$ in the region around feature point $\mathbf{x}_k^B$. This means that $M_{\mathbf{x}_k^B}$ contains the feature points to which corresponding pixels have been already determined in the rectified f-th frame. $D_{\mathbf{s}}(\mathbf{t})$ is a cost term based on the distance between feature points in rectified background image B and the difference in shift vectors of the feature points in current frame f for preventing neighboring feature points from moving in a different way (i.e., preventing wrong correspondences) and is defined as follows:

$$D_{\mathbf{s}}(\mathbf{t}) = \begin{cases} 0 & (d(\|\mathbf{s}\|) > \|\mathbf{t}\|) \\ \kappa & (\text{otherwise}) \end{cases}, \tag{2}$$

where $d(\|\mathbf{s}\|)$ is a monotonically increasing function, which gives higher value as $\|\mathbf{s}\|$ gets larger. For example, we use $d(\|\mathbf{s}\|) = \|\mathbf{s}\|/10$ in our experiments. The cost function allows the difference in the shift vector $\|\mathbf{t}\|$ to become larger as distance $\|\mathbf{s}\|$ between feature points becomes more distant.

5.2 Motion Interpolation

Using $\mathbf{y}_k^f$ corresponding to feature point $\mathbf{x}_k^B$, we interpolate the shift vectors from rectified background image B to the rectified f-th input frame for all pixels in the marker region Ω and the marker's surrounding region $\partial\Omega$ in B, and deform rectified background image B based on them. It should be noted here that we need the shift vectors of not only pixels in Ω but also pixels in $\partial\Omega$ because pixels in $\partial\Omega$ in B may be partially occluded by the marker in the f-th frame.

Specifically, on the basis of the assumption that pixels move in a similar way to feature points with high degrees of confidence around them and adjacent pixels' motions are highly correlated, we estimate the motion of each pixel in $\Omega \cup \partial\Omega$ in B by minimizing the following energy function.

$$E = \sum_{i \in \Omega \cup \partial\Omega} \sum_k \omega_{i,k} \|\mathbf{u}_i - \mathbf{u}_k\|^2 + \alpha \sum_{(i,j) \in A} \|\mathbf{u}_i - \mathbf{u}_j\|^2, \tag{3}$$

where the summation over k is calculated for all indexes of feature points, and A is a set of adjacent pixel pairs in $\Omega \cup \partial\Omega$. $\mathbf{u}_i$ is the shift vector for pixel i. Shift vector $\mathbf{u}_k$ for feature point $\mathbf{x}_k^B$ is given by $\mathbf{u}_k = \mathbf{y}_k^f - \mathbf{x}_k^B$. Weight $\omega_{i,k}$ is calculated on the basis of the distance between feature point $\mathbf{x}_k^B$ and pixel $\mathbf{x}_i^B$ as well as the confidence $C(\mathbf{x}_k^B)$ of feature point $\mathbf{x}_k^B$ as follows:

$$\omega_{i,k} = C(\mathbf{x}_k^B) \exp\left(-\frac{\|\mathbf{x}_i^B - \mathbf{x}_k^B\|^2}{\sigma^2}\right). \tag{4}$$

Minimization of E is equivalent to solving a symmetric and positive-definite linear system obtained by setting its partial derivatives with respect to the horizontal and vertical components of $\mathbf{u}_k$ to zero. Since the system's coefficient matrix is sparse, we can solve it by the conjugate gradient method for sparse systems, which works efficiently on GPUs (an example CUDA implementation for the conjugate gradient method is provided by NVIDIA and the information is available at [33]).

Finally, we deform the rectified background image B by projecting each pixel in image B on the basis of the obtained shift vectors and linearly interpolating pixel values. Our implementation uses texture mapping provided in OpenGL.

6 Intensity Adjustment

The intensity of texture usually varies between the rectified background image B and the input frames because of

Fig. 7. AR marker used in experiments.

illumination change and soft shadows due to a user. Thus, we adjust the RGB intensities in B for smoothly overlaying it on the marker region in the input frame. Conventionally, the intensity change is calculated for a down-sampled image [6] or for image grids [7] in order to reduce the calculation cost. However, these methods work well only if each down-sampled pixel or each grid covers the spatially same area in the original image throughout all input frames, which is satisfied only when the background geometry is planar. Therefore, our method employs pixel-wise Poisson blending [28] for adjusting intensities. Pixel values in red, green, and blue channels are independently adjusted. The Poisson equation used in this blending can be solved with the same way as the motion interpolation in the previous section. Therefore, we can again utilize the computational capability of GPUs.

7 Experiments

We carried out experiments to visually remove an AR marker using a PC with Windows 10, Core i7-990X 3.46 GHz CPU, 12 GB of memory, and a GeForce GTX Titan GPU. We used a USB camera (Logicool Qcam Pro 9000) to capture the real-time input video stream, each of whose frame is 640 $\times$ 480 pixels. We used ARToolkit [1] for camera pose estimation and a square AR marker whose edge length is 80 mm, attached to a relatively thick object whose edge length is 95 mm and thickness is 7 mm as shown in Fig. 7. Table 1 shows parameter values used in the experiments. In the following sections, we first compare experimental results obtained by the propose method with those by baseline methods to demonstrate the effectiveness of the proposed method. We then validate each component of the proposed method, and investigate the influence of parameters. We finally describe the limitations by showing some examples and future possibilities.

7.1 Comparison of experimental results

To demonstrate the effectiveness of the proposed method, we tested our method under various environments as follows.

TABLE 1
Parameters and values used in experiments.

Input image	640×480 pixels
Marker size in a rectified image	80×80 pixels
Marker region Ω	140×140 pixels
Width l of surrounding region $\partial\Omega$	15 pixels
L	15 pixels
Search range G	5×5 pixels
Size of patch	11×11 pixels
κ, α, σ	0.001, 1, 10

Scene A A curved background geometry with a grid pattern (Fig. 8).
Scene B A planar background geometry with a stripe pattern (Fig. 9).
Scene C A curved background geometry with a stripe pattern (Fig. 10).
Scene D A messy desk with a book, some sheets of paper with figures and texts, and a blank notebook (Fig. 11).
Scene E An uneven surface on aligned books (Fig. 12).
Scene F An uneven surface of stones (Fig. 13).

To obtain background images, we used our previous approach [7] for scenes A, B, C, and F, which applies the method for image inpainting [34] to a rectified image with a marker. We preliminarily captured a single background image for scenes D and E.

In the experiments, we compare the results by our method (d) in Figs. 8-13 with those by the baseline approach, which just uses a homography matrix to transform the background image (b). To confirm the effectiveness of the intensity adjustment, we also show the results by our method without intensity adjustment (c). (a) and (e) show input frames, and the rectified frames with the feature tracking results by the proposed method. (f) shows the deformed images of the rectified background images in the marker and its surrounding regions. The first row in each figure shows the result when a camera did not move a lot from the initial pose. The second and third rows show the results when the camera moves. We discuss the results for each scene in detail.

Figure 8 shows experimental results for scene A. We can see the texture's appearance changes around the marker in the rectified image due to camera motion and the curved geometry as shown in (e). Thus, (b) exhibits large discontinuities in texture around the boundary. In (c), the edges in the grid pattern are successfully connected on the boundary, but brightness differs between the hidden and its surrounding regions. On the other hand, our method did not yield geometric and photometric discontinuities in texture. Our method deformed the background image as shown in (f), on the basis of the tracking of feature points in the marker's surrounding region.

In scene B shown in Fig. 9, appearance changes in the texture do not seem to be significant from (d) compared to scene A, but displacement of the texture actually occurs because of the thickness of the marker base. Thus, we can see the discontinuous straight lines in (b) without deformation. Noticeable difference in brightness can be observed without intensity adjustment in (c). Our method did not produce significant visual artifacts as shown in (d). For such a stripe pattern as in this scene, our method does not always make accurate correspondences between the feature points detected in the rectified background image and pixels in the input frame because of the aperture problem. Therefore, these inaccurate correspondences excessively deformed the background image as shown in (f). However, this deformation compensated for the displacement in the direction orthogonal to the stripes.

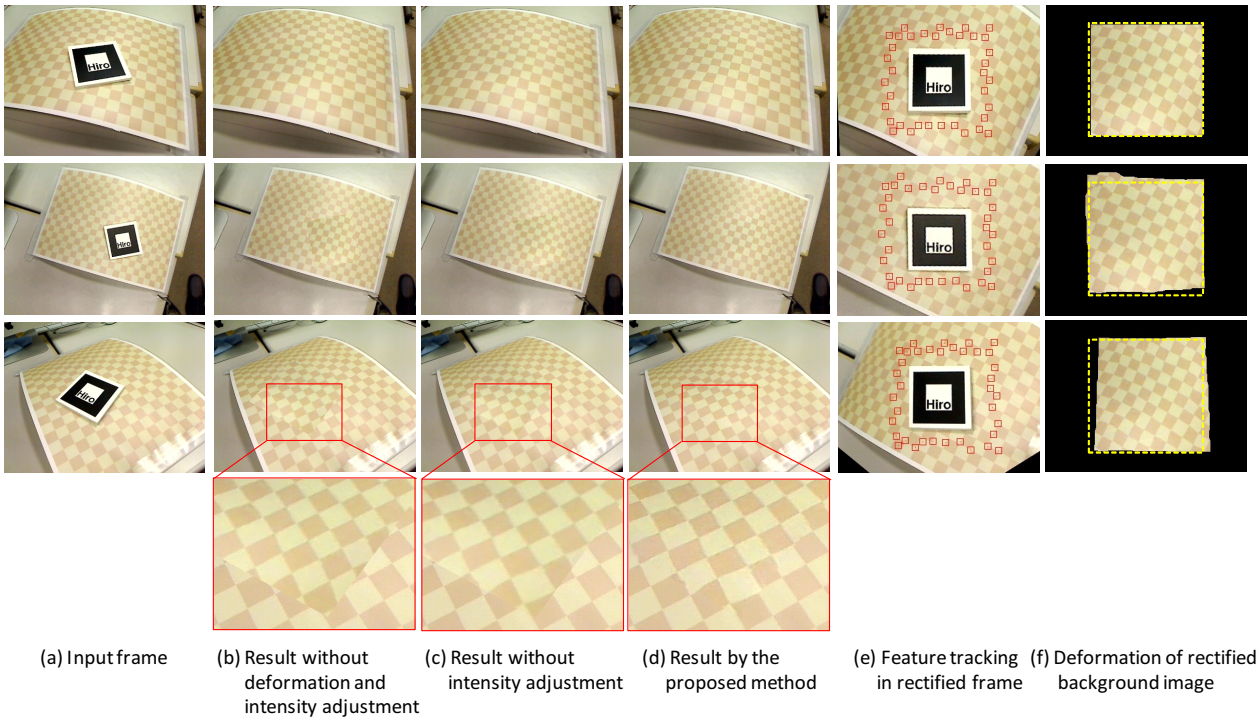


Fig. 8. Experimental results for scene A with a curved background geometry with a grid pattern.

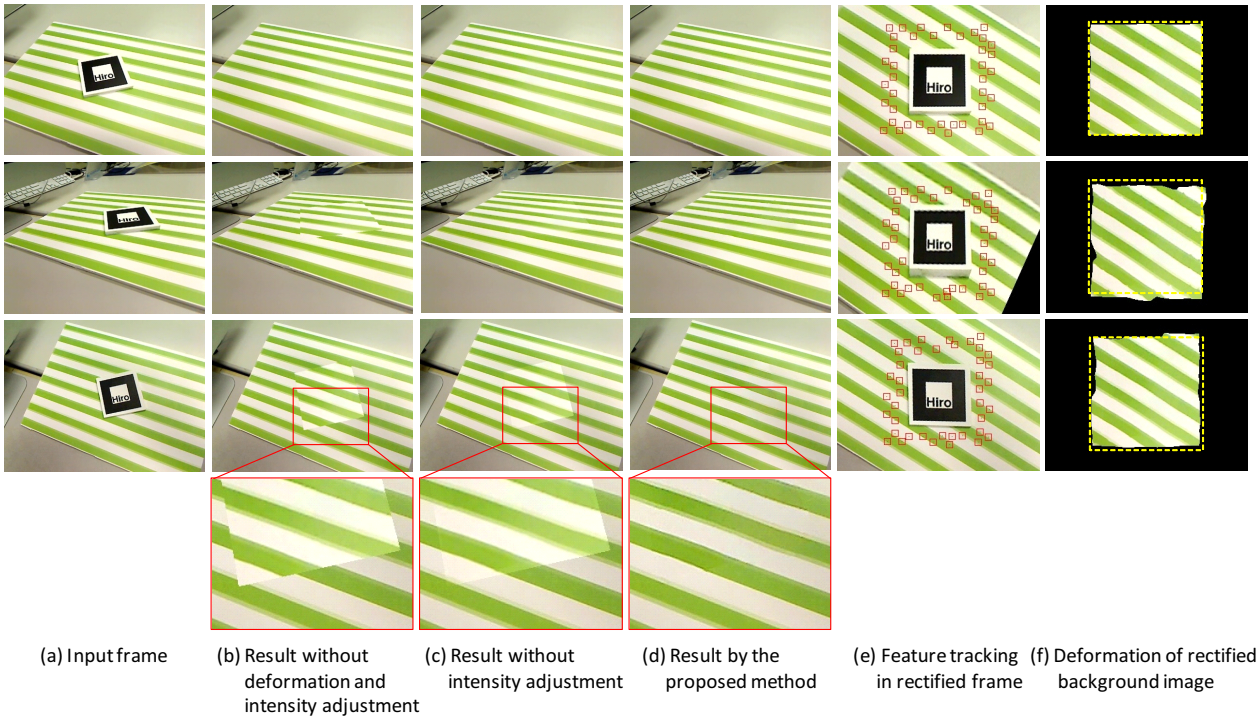
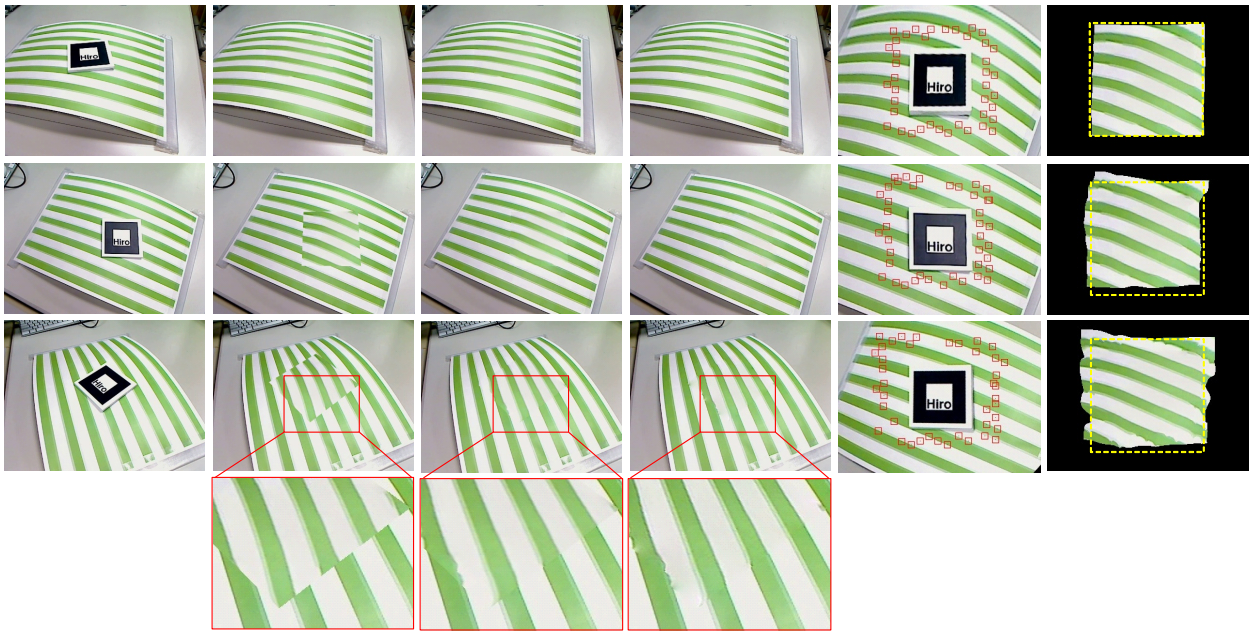


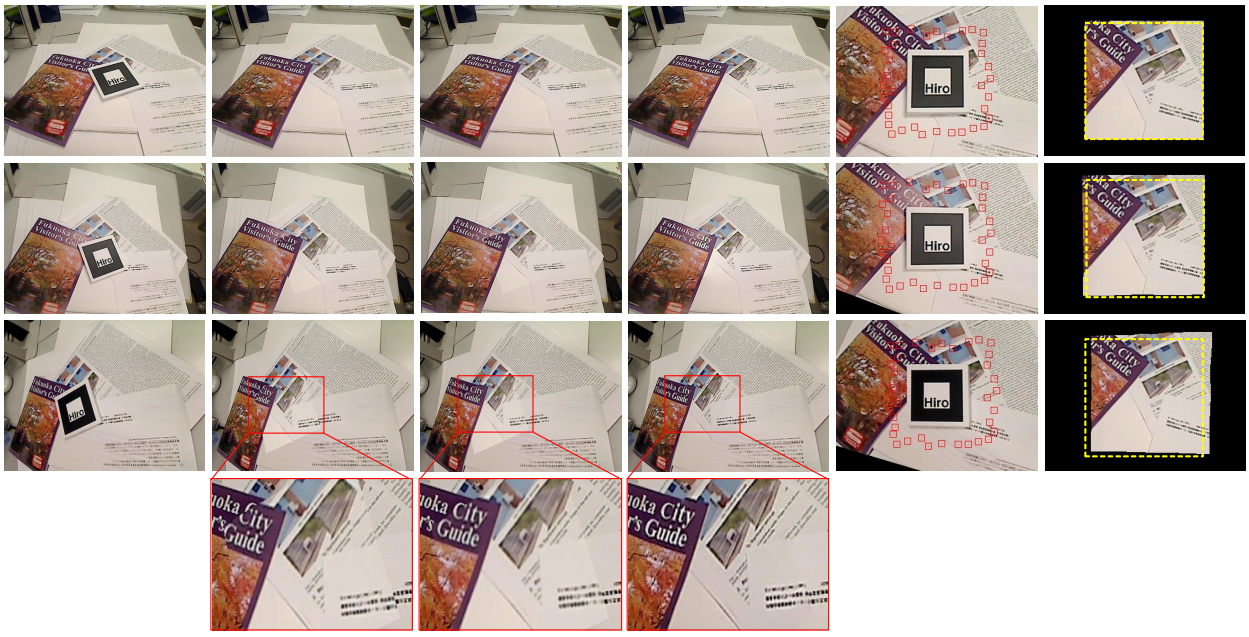
Fig. 9. Experimental results for scene B with a planar background geometry with a stripe pattern.

In scene C in Fig. 10, the marker is placed on a curved geometry that is the same as scene A but with a stripe pattern as in scene B. As with scene B, our method produced more plausible texture than those without deformation and intensity adjustment as shown in (b), (c), and (d). However, we can also see small distortions in the stripes compared with the results for scene B in Fig. 9. If the background geometry is planar, these stripes are straight regardless of camera poses in the rectified images as in scene B, but otherwise, they get curved as in the first row of (e), and the curvature varies according to camera poses as in the second and third rows. To compensate the difference in curvature, we may need more dense feature points as well as accurate correspondences, although for this scene, associating feature points accurately seems almost infeasible because of the aperture problem.



(a) Input frame (b) Result without deformation and intensity adjustment (c) Result without intensity adjustment (d) Result by the proposed method (e) Feature tracking in rectified frame (f) Deformation of rectified background image

Fig. 10. Experimental results for scene C with a curved background geometry with a stripe pattern.



(a) Input frame (b) Result without deformation and intensity adjustment (c) Result without intensity adjustment (d) Result by the proposed method (e) Feature tracking in rectified frame (f) Deformation of rectified background image

Fig. 11. Experimental results for scene D with a messy desk with a book, some sheets of paper with figures and texts, and a blank notebook.

In scene D (Fig. 11), the marker is placed on a book, a blank notebook, and sheets of paper containing figures and texts on a desk. Due to partially overlapping these items, the surface is slightly uneven. As seen in (e), feature points were distributed almost uniformly on textured regions, edges, and texture-less regions. In (b), discontinuities in the texture became significant as the camera moved away from the initial position. On the other hand, although feature point tracking in texture-less regions is extremely difficult, our proposed method's result still looks plausible because the proposed method uses the confidence of each feature point to deform the background image.

In scene E shown in Fig. 12, the marker is placed on aligned books where there are slight misalignments between books. In addition, we changed illumination color continuously while the camera moved for this scene. In

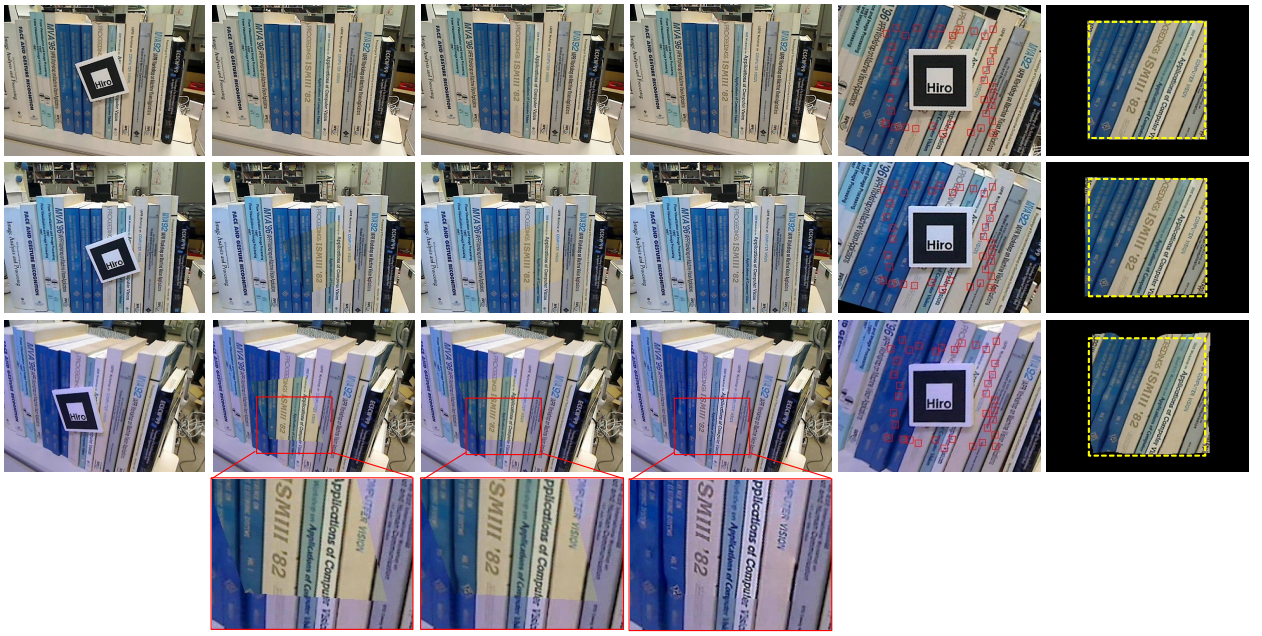


Fig. 12. Experimental results for scene E with an uneven surface on aligned books.

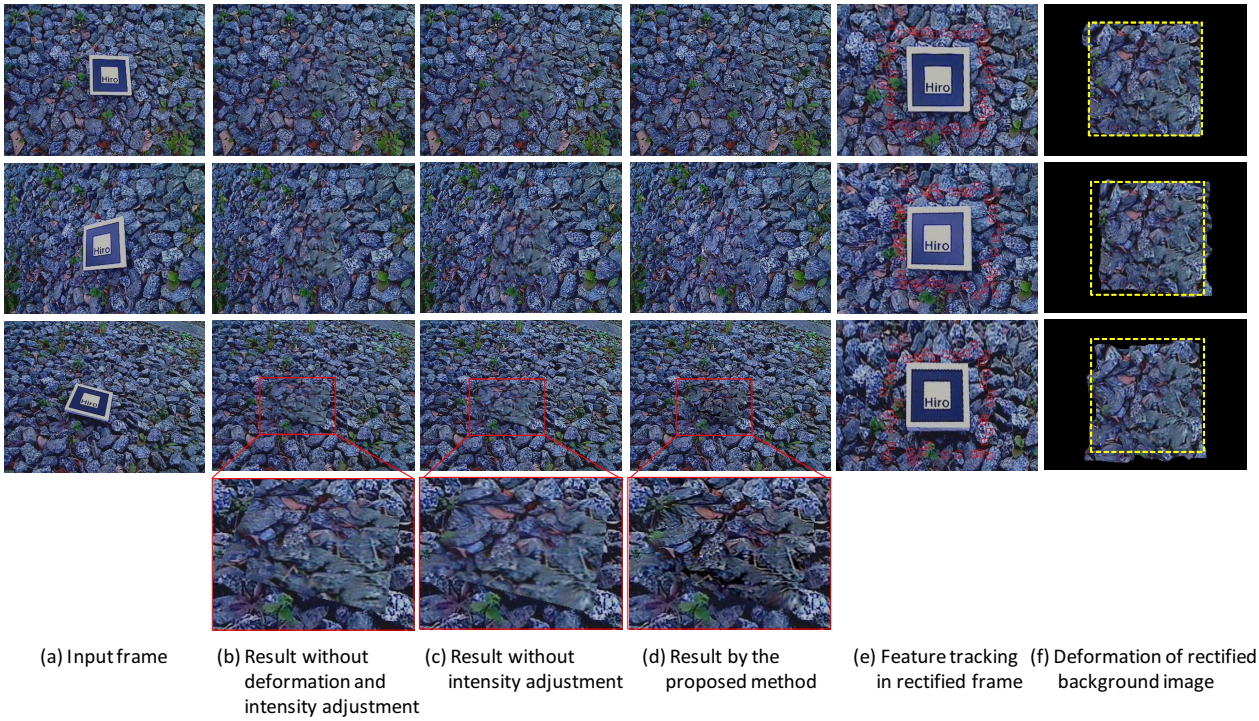


Fig. 13. Experimental results for scene F with an even surface of stones.

(b), discontinuities in texture occur as with other scenes. In (c), the geometric discontinuities were compensated but the differences in colors are more noticeable than other scenes when we did not adjust intensities. In the results in (d), the geometric and photometric discontinuities were successfully compensated by the proposed method. However, we also confirmed one problem of the proposed method that the inconsistency of image quality occurs between the generated image on the marker and its surrounding when the input image blurs due to the camera motion because the generated image does not blur even if the input image blurs.

In scene F shown in Fig. 13, the marker is placed on stones with various shapes and sizes. Since the background has random texture patterns, the overlaid texture on the marker is harmonious with its surrounding at the first glance even in (b) when we see the marker-removed scene

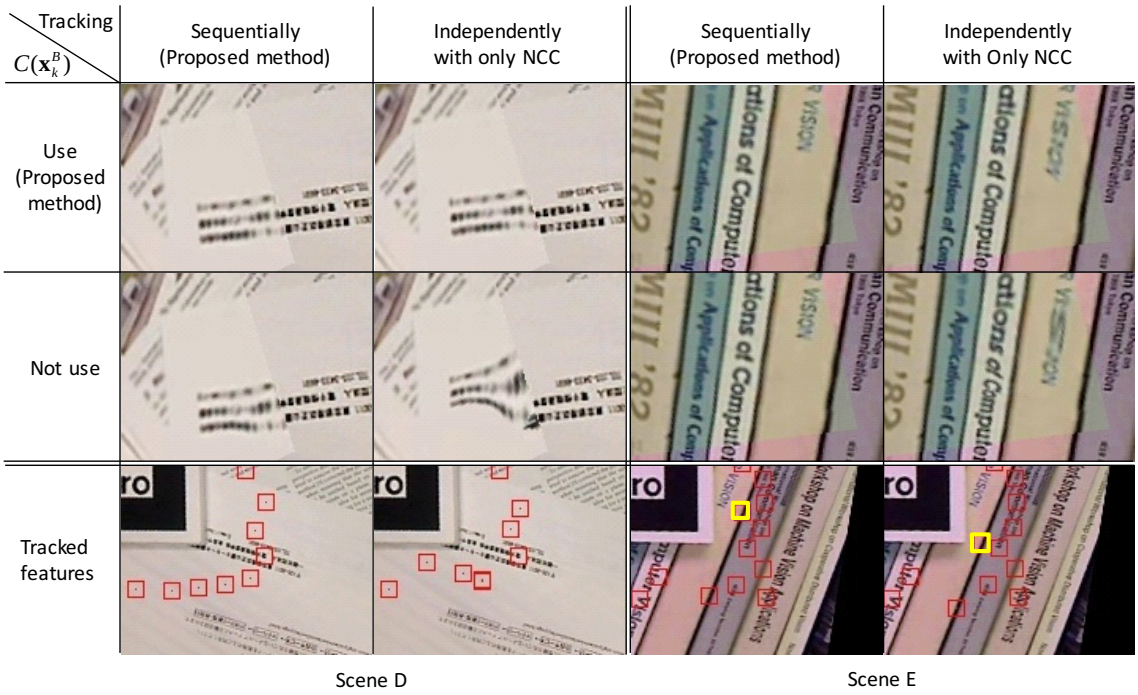


Fig. 14. Experimental results for scenes D and E for validating each component.

frame by frame. However, when we see the scene as a video stream, we can perceive it as if a stone-textured plane is floating. In the results by the proposed method, we can see that the background stones exist on the same surface as the surrounding one as shown in (d).

The numbers of detected feature points for scenes A, B, C, D, E, and F were 37, 41, 41, 36, 37, and 38, respectively. The frame rates for the scenes were 5.2 fps, 5.2 fps, 5.1 fps, 5.1 fps, 5.0 fps, and 5.1 fps, respectively.

7.2 Validity of each component of our method

This section validates each component of our method, i.e., feature tracking and motion interpolation (texture deformation). Specifically, we evaluated scenes D and E under the combinations of following two types of variations.

- Feature points are tracked sequentially using Eq. (1) (the proposed method) or independently using only NCC.
- The confidence $C(\mathbf{x}_k^B)$ is used in Eq. (4) for motion interpolation (our method) or not (i.e., all feature points equally influence the deformation).

Figure 14 shows close-ups of the results and tracked features for both cases where feature points are sequentially or independently tracked. In order to clearly show the deformation, in these figures, we did not apply intensity adjustment. In the case of independent tracking without confidence for scene D, feature points in texture-less regions moved in different directions. As a result, the texture was excessively distorted. The confidence alleviated the distortion, but there are still texture discontinuities on the boundary. In the case of sequential feature tracking by our proposed method, the texture was distorted a little bit when the confidence was not used, because feature point tracking was still a little inaccurate in the texture-less regions. The confidence was able to compensate for the influence of such inaccurate feature points on deformation.

In scene E, our method's tracking and independent tracking made different tracking results for the feature point marked in yellow in Fig. 14, which is on the edge of two book spines, because of the aperture problem. In the case of independent tracking, this feature point made the letters on the spine too stretched. The use of confidence alleviated the artifact because the confidence of this feature point was lower than those of other feature points on the corners around it. In the case of our method's tracking, the results with/without the confidence are plausible because the tracking of the feature point on the edge were accurate. Note that the stretched letters on the book spine are not behind the marker as seen in the bottom row of Fig. 14; however, they are in region Ω, which is a region slightly larger than the actual marker region to handle occlusion due to the thick marker.

These results successfully demonstrated that sequentially tracking feature points in descending order of the confidence using Eq. (1) and using the confidence $C(\mathbf{x}_k^B)$ for motion interpolation (texture deformation) are effective especially when feature points on texture-less and edge regions are close to those on corners, since feature points on corners can be accurately tracked and their motion can be propagated to their surrounding regions.

7.3 Influence of parameters

In this section, we investigate the influence of parameters α and σ in Eqs. (3) and (4) for motion interpolation using the frame shown in the third row in Fig. 8 in Scene A.

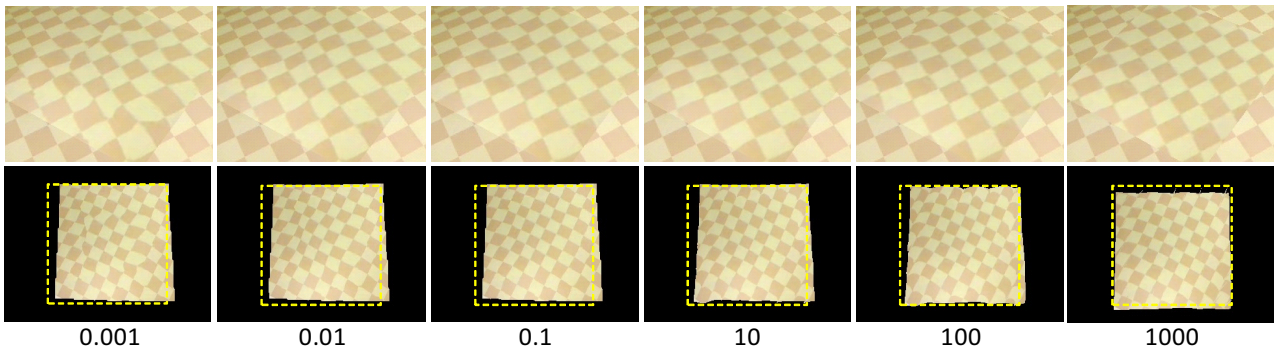


Fig. 15. Experimental results with different α.

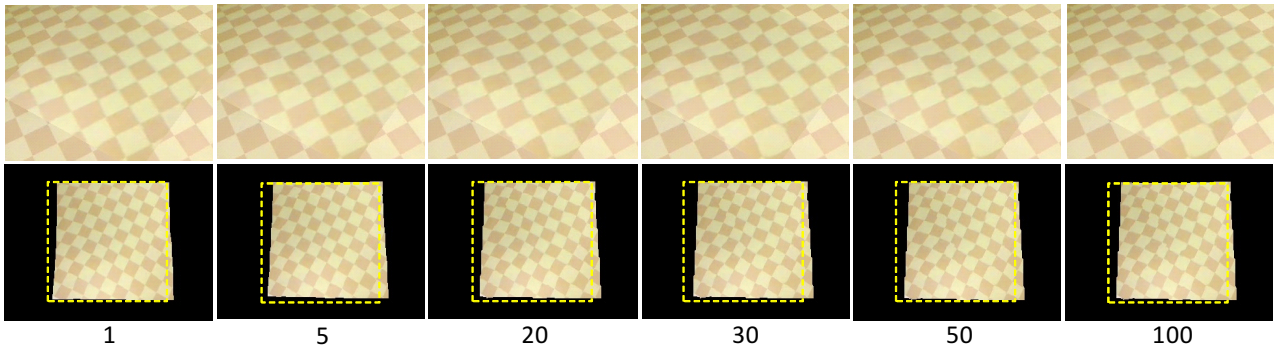


Fig. 16. Experimental results with different σ.

Figures 15 and 16 show close-ups of results before intensity adjustment (top) and their corresponding deformations of rectified background images with different α and σ (bottom), respectively.

For small α, pixels only around the feature points moved, and those near the center of the marker did not at all. This is because, the energy with small α is hardly affected by different motion in adjacent pixels. As a result, the texture between the center and the boundary of Ω was distorted. For too large α, all pixels coherently moved since different motion in adjacent pixels is severely penalized. This resulted in discontinuity around the edge. For α between 0.1 and 10, we did not observe big difference in the results. These results confirm that results are not sensitive to small changes in α.

For smaller σ, the motion of feature points tends to give less influence on pixels far from them. As a result, we found small discontinuities on the boundary for $\sigma = 1$. When σ is too large, the motion of feature points gives influence even on pixels quite far from them. In such a case, pixels near the center of Ω are influenced by feature points far from them. As a result, texture near the center is severely distorted if the motions of feature points at a side are quite different from those at other sides as shown in the results for $\sigma = 50$ and 100. The results for $\sigma = 5$ and 20 did not make a noticeable difference; therefore, our method is not sensitive to small changes in σ.

7.4 Limitations and future possibilities

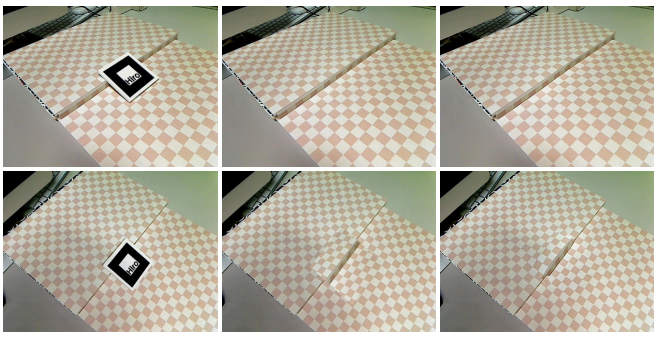
7.4.1 Background geometry

Figure 17 shows experimental results for a scene with a step background geometry. In this scene, the vertical plane becomes invisible in the input frame depending on the camera pose as shown in the input image at the second row in the figure. Even in such a case, the vertical plane still remains on the marker region though other regions in the result were improved by the proposed method as in (b) and (c). Figure 18 shows experimental results for a scene that consists of two textured planes that are orthogonal each other. In the results by the proposed method, edges between planes were not completely corrected as shown in (c). Since the proposed motion interpolation method assumes smooth motion, the proposed method cannot handle such discontinuity in shift vectors. This results in visual artifacts as shown in the results of Figs. 17 and 18.

To alleviate this problem, we need to develop texture deformation that can handle non-smooth geometry but still works in real time. Spatially varying weight α in Eq. (3) for encouraging adjacent pixels to move in a similar direction is a possible approach. Since roughly estimating the background geometry using dense SLAM, such as DTAM [3], only adds a relatively small burden on the users and short computational time, the combination of texture deformation and rough background geometry may also enhance the capability for dealing with a variety of background geometries for which our current implementation fails without excessively spoiling the advantage of our method.

7.4.2 Feature tracking

Since our algorithm for feature point detection and tracking cannot find the correct corresponding points in the input frame once they lost, which sometimes occurs because of rapid camera motion, the tracking failure results in deformation based on inaccurate correspondences.



(a) Input frame (b) Result without deformation and intensity adjustment (c) Result by the proposed method

Fig. 17. Experimental results for a scene with occlusion.

(a) Input frame (b) Result without deformation and intensity adjustment (c) Result by the proposed method

Fig. 18. Experimental results for a scene with non-smooth change in background geometry.

A possible solution to this problem is to consider the relationship between camera poses and the degrees of deformation. For example, if we can store the deformation parameters given a camera pose when the background image is accurately deformed with correct correspondences, we can use them for recovering tracking failure.

7.4.3 Computational cost

As described in the previous section, the frame rate of our proposed method is still lower than a standard video rate. For applications that run on light-weight portable devices, we should reduce the computational cost.

One possible solution is to use the stored deformation parameters described above not only for tracking recovery but also for computational cost reduction. Since they can give good initial parameters for minimizing the energy function in Eq. (3), the frame rate may improve as the number of the stored deformation parameters increases. Another solution is to reduce the number of parameters in Eq. (3). Instead of using all pixels in $\Omega \cup \partial\Omega$ as parameters for our energy minimization, adaptively sampling some pixels depending on texture complexity in the background image may reduce the computational cost while preserving the quality of deformation.

8 Conclusion

This paper has proposed an AR marker hiding method based on deformation of a background image. Our modified feature point detection and proposed tracking algorithm gives feature points that are desirable for motion interpolation. We achieved fast pixel-wise deformation by designing an energy function that can be efficiently minimized using GPUs.

In the experiments, we have confirmed that our texture deformation-based AR marker hiding method can generate continuous textures on the marker region even for a thick AR marker and for nonplanar background geometry. It should be noted that we have also obtained good results for the scene with a stripe pattern, which causes the aperture problem when tracking feature points. However, our method could not deal with scenes in which occlusions occur according to the camera motion and non-smooth geometry change exists because we assume that the motions of adjacent pixels are similar. Future work includes to establish AR marker hiding for a variety of background geometries at a higher frame rate. In addition, we will apply texture deformation to diminished reality which visually removes various objects from a real-time video stream.

Acknowledgments

This work was partially supported by Grants-in-Aid for Scientific Research No. 15K16039 from the Japan Society for the Promotion of Science (JSPS).

References

[1] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Proc. Int. Workshop Augmented Reality*, 1999, pp. 85–94.

[2] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Int. Symp. Mixed and Augmented Reality*, 2007, pp. 225–234.

[3] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. Int. Conf. Computer Vision*, 2011, pp. 2320–2327.

[4] T. Schöps, J. Engel, and D. Cremers, "Semi-dense visual odometry for AR on a smartphone," in *Proc. Int. Sympo. Mixed and Augmented Reality*, 2014, pp. 145–150.

[5] S. Siltanen, "Texture generation over the marker area," in *Proc. Int. Symp. Mixed and Augmented Reality*, 2006, pp. 253–254.

[6] O. Korkalo, M. Aittala, and S. Siltanen, "Light-weight marker hiding for augmented reality," in *Proc. Int. Symp. Mixed and Augmented Reality*, 2010, pp. 247–248.

[7] N. Kawai, M. Yamasaki, T. Sato, and N. Yokoya, "Diminished reality for AR marker hiding based on image inpainting with reflection of luminance changes," *ITE Trans. Media Technology and Applications*, vol. 1, no. 4, pp. 343–353, 2013.

[8] N. Kawai, T. Sato, Y. Nakashima, and N. Yokoya, "AR marker hiding with real-time texture deformation," in *Proc. IEEE Int. Symp. Mixed and Augmented Reality Workshops (Int. Workshop on Diminished Reality as Challenging Issue in Mixed and Augmented Reality)*, 2015, pp. 26–31.

[9] P. Barnum, Y. Sheikh, A. Datta, and T. Kanade, "Dynamic seethroughs: synthesizing hidden views of moving objects," in *Proc. Int. Symp. Mixed and Augmented Reality*, 2009, pp. 111–114.

[10] A. Enomoto and H. Saito, "Diminished reality using multiple handheld cameras," in *Proc. ACCV'07 Workshop Multi-dimensional and Multi-view Image Processing*, 2007, pp. 130–135.

[11] S. Jarusirisawad, T. Hosokawa, and H. Saito, "Diminished reality using plane-sweep algorithm with weakly-calibrated cameras," *Progress in Informatics*, no. 7, pp. 11–20, 2010.

[12] S. Zokai, J. Esteve, Y. Genc, and N. Navab, "Multiview paraperspective projection model for diminished reality," in *Proc. Int. Symp. Mixed and Augmented Reality*, 2003, pp. 217–226.

[13] T. Tsuda, H. Yamamoto, Y. Kameda, and Y. Ohta, "Visualization methods for outdoor see-through vision," in *Proc. Int. Conf. Artificial Reality and Telexistence*, 2005, pp. 62–69.

[14] V. Lepetit and M.-O. Berger, "An intuitive tool for outlining objects in video sequences: Applications to augmented and diminished reality," in *Proc. Int. Symp. Mixed and Augmented Reality*, 2001, pp. 159–160.
[15] F. I. Cosco, C. Garre, F. Bruno, M. Muzzupappa, and M. A. Otaduy, "Augmented touch without visual obtrusion," in *Proc. Int. Symp. Mixed and Augmented Reality*, 2009, pp. 99–102.
[16] S. Mori, F. Shibata, A. Kimura, and H. Tamura, "Efficient use of textured 3D model for pre-observation-based diminished reality," in *Proc. IEEE Int. Symp. Mixed and Augmented Reality Workshops (Int. Workshop on Diminished Reality as Challenging Issue in Mixed and Augmented Reality)*, 2015, pp. 32–39.
[17] Z. Li, Y. Wang, J. Guo, L.-F. Cheong, and S. Z. Zhou, "Diminished reality using appearance and 3d geometry of internet photo collections," in *Proc. Int. Symp. Mixed and Augmented Reality*, 2013, pp. 11–19.
[18] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach," in *Proc. SIGGRAPH96*, 1996, pp. 11–20.
[19] K. Takeda and R. Sakamoto, "Diminished reality for landscape video sequences with homographies," in *Int. Conf. Knowledge-Based and Intelligent Information and Engineering Systems*, 2010, pp. 501–508.
[20] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
[21] J. Herling and W. Broll, "Advanced self-contained object removal for realizing real-time diminished reality in unconstrained environments," in *Proc. Int. Symp. Mixed and Augmented Reality*, 2010, pp. 207–212.
[22] ——, "High-quality real-time video inpainting with pixmix," *IEEE Trans. Visualization and Computer Graphics*, vol. 20, no. 6, pp. 866–879, 2014.
[23] N. Kawai, T. Sato, and N. Yokoya, "Diminished reality based on image inpainting considering background geometry," *IEEE Trans. Visualization and Computer Graphics*, vol. 22, no. 3, pp. 1236–1247, 2016.
[24] J. Ehara and H. Saito, "Texture overlay for virtual clothing based on pca of silhouettes," in *Proc. Int. Sympo. Mixed and Augmented Reality*, 2006, pp. 139–142.
[25] J. Pilet, V. Lepetit, and P. Fua, "Fast non-rigid surface detection, registration and realistic augmentation," *Int. J. Computer Vision*, vol. 76, no. 2, pp. 109–122, 2007.
[26] A. Hilsmann and P. Eisert, "Tracking and retexturing cloth for real-time virtual clothing applications," in *Proc. Int. Conf. Computer Vision/Computer Graphics CollaborationTechniques (MIRAGE)*, 2009, pp. 94–105.
[27] Y. Furukawa and C. Hernndez, "Multi-view stereo: A tutorial," *Foundations and Trends in Computer Graphics and Vision*, vol. 9, no. 1-2, pp. 1–148, 2015.
[28] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 313–318, 2003.
[29] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Fourth Alvey Vision Conference*, 1988, pp. 147–151.
[30] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1994, pp. 593–600.
[31] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.
[32] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. European Conf. Computer Vision*, vol. 1, 2006, pp. 430–443.
[33] "Nvidia developer zone (http://docs.nvidia.com/cuda/cuda-samples/#conjugategradient)."
[34] N. Kawai and N. Yokoya, "Image inpainting considering symmetric patterns," in *Proc. Int. Conf. Pattern Recognition*, 2012, pp. 2744–2747.

Norihiko Kawai received his B.E. degree in informatics and mathematical science from Kyoto University in 2005. He received his M.E, and Ph.D. degrees in information science from Nara Institute of Science and Technology (NAIST) in 2007 and 2010, respectively. He was a research fellow of the Japan Society for the Promotion of Science and postdoctoral fellow at the University of California at Berkeley in 2010-2011. He has been an assistant professor at NAIST since 2011.

Tomokazu Sato received his B.E. degree in computer and system science from Osaka Prefecture University in 1999. He received his M.E. and Ph.D. degrees in information sciences from Nara Institute of Science and Technology (NAIST) in 2001 and 2003. He was an assistant professor at NAIST in 2003-2011. He was a visiting resercher in Czech Technical University in Prague in 2010-2011. He has been an associate professor at NAIST since 2011.

Yuta Nakashima received the B.E. and M.E. degrees in communication engineering from Osaka University, Osaka, Japan in 2006 and 2008, respectively, and the Ph.D. degree in engineering from Osaka University, Osaka, Japan, in 2012. He is currently an Assistant Professor at Graduate School of Information Science, Nara Institute of Science and Technology (NAIST). He was a visiting scholar at Carnegie Mellon University in 2015-2016. His research interests include video content analysis using probabilistic and statistical approaches. He is a member of the IEEE, the ACM, and the IEICE.

Naokazu Tokoya received his B.E., M.E., and Ph.D. degrees in information and computer sciences from Osaka University in 1974, 1976, and 1979, respectively. He joined Electrotechnical Laboratory (ETL) of the Ministry of International Trade and Industry in 1979. He was a visiting professor at McGill University in 1986-1987 and has been a professor at Nara Institute of Science and Technology (NAIST) since 1992. He has also been a vice president at NAIST since April 2013.