AR Marker Hiding with Real-time Texture Deformation

Norihiko Kawai* Tomokazu Sato† Yuta Nakashima‡ Naokazu Yokoya§

Nara Institute of Science and Technology

ABSTRACT

Augmented Reality (AR) marker hiding is a technique to visually remove AR markers from a real-time video stream. The conventional approach transforms a background image with a homography matrix calculated on the basis of a camera pose and overlays the transformed image on the AR marker region in the real-time frame, assuming that the AR marker is on a planar surface. However, this approach can 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 under unknown and nonplanar background geometry. The proposed method estimates dense motion in the marker's background based on feature point correspondences around the marker, together with a smooth motion assumption, and deforms the background image according to the dense motion. Experimental results demonstrate the effectiveness of the proposed method in various environments with different background geometries and textures.

Index Terms: H.5.1 [INFORMATION INTERFACES AND PRESENTATION]: Multimedia Information Systems—Artificial, augmented, and virtual realities

1 INTRODUCTION

With the commodification of smartphones, tablets, and portable game consoles, applications that use augmented reality (AR) to overlay various objects on a real-time video stream of a real environment, such as furniture arrangement simulation and AR games with virtual characters, are becoming increasingly available. Such applications must estimate the camera pose in real time, and various approaches have been employed to accomplish this, e.g., simultaneous localization and mapping (SLAM) and AR marker-based approaches [6, 10, 14]. Although SLAM-based approaches [10, 14] have been developed intensively, the marker-based approach [6] is widely adopted because of the ease by which virtual objects can be placed at the user's desired positions and its robustness against various textures and shapes in a scene. However, the marker-based approach requires visible AR markers, which can hinder seamless fusion of the real environment and the virtual objects.

Some methods that attempt to visually remove the AR markers from a real-time video stream without incurring user burden have been proposed to address this problem. Siltanen et al. [15] synthesized a background image by mixing several pixel values around the marker for each frame and replaced the marker region with this image. This simple method works quite quickly but may cause significant visual artifacts, especially for complex textures. Korkalo et al. [11] and Kawai et al. [8] have proposed AR marker hiding based on a planar assumption, in which the AR marker and the background surface are on the same plane. These methods generate background images by applying image inpainting to the marker regions in a certain frame in a real-time video stream and overlay the background images on a newly captured frame by transforming them with a homography matrix. These methods also adjust image intensities in the generated background to reduce discontinuities around the boundaries between the marker regions and their surroundings, which are caused by changes in lighting conditions. However, these methods can suffer from discontinuities on the boundary, because the marker is often placed on a nonplanar surface or it is attached to a thick base, such as cardboard, so that the marker cannot bend.

For AR marker hiding, using diminished reality (DR) technology, which visually removes a variety of real objects in real time, is also a promising approach. Among diminished reality methods, methods that use preliminarily captured background images and those that use image inpainting are possible ones.

The methods in [2] and [12] preliminarily capture multiple background images in which occluding objects are removed or collects images that captures a target scene from the Internet to obtain background images. These methods then overlay the background image on target object regions in the real-time frames to visually remove the target objects. The main challenges in such methods are finding an image that is suitable for the current frame from a number of background images and transforming the background image to seamlessly overlay it on the frame. Cosco et al. [2] selected background images based on view-dependent texture mapping criterion [3] and rendered a background image for the frame assuming that the background geometry is available as a mesh model. Li et al. [12] searched the Internet for images captured from a position that is close to the current frame and transform the image using a homography matrix. To address the previously mentioned challenges, these methods must satisfy one of the following conditions: (i) the background geometry must be known or should be possible to be reconstructed with high accuracy; or (ii) the images must be densely captured from various viewpoints. Unless one of these conditions is satisfied, it is difficult to seamlessly overlay background images on real-time frames.

Methods using image inpainting can be classified into those that generate a background image for each frame [4, 5] and those that generate a background image in a certain frame and geometrically and photometrically adjust for other real-time frames [7]. The former approach can generate a plausible background image for each frame in many cases; however applying image inpainting to each frame can 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 a homography for each plane in each frame. However, this method can also suffer from texture discontinuities on the boundary between the target region and its surrounding if the background geometry assumption is not satisfied.

In this work, considering a scenario in which users wish to hide markers in AR applications without burden, we propose a method for AR marker hiding. In the proposed method, users can visually remove markers that are possibly fixed to the environment from the real-time video stream with unknown background geometry and marker thickness. The proposed method obtains a background image by capturing a single image before placing a marker or by

*e-mail: norihi-k@is.naist.jp
†e-mail: tomoka-s@is.naist.jp
‡e-mail: n-yuta@is.naist.jp
§e-mail: yokoya@is.naist.jp

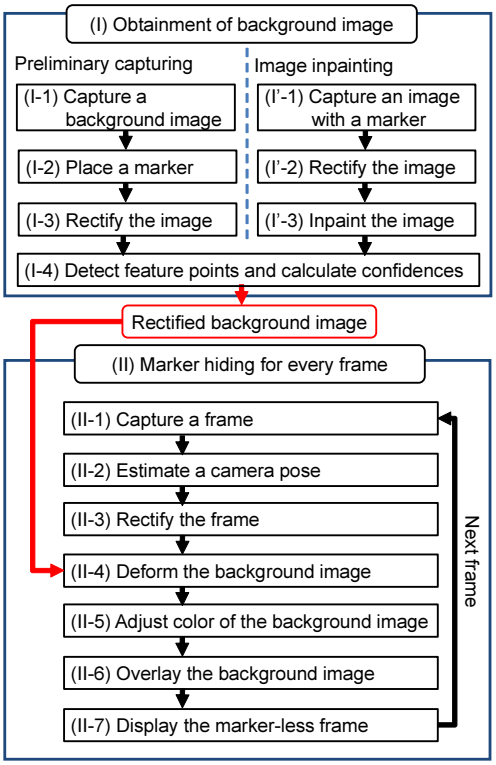

Figure 1: Flow diagram of the proposed marker hiding method.

applying image inpainting to an image with a marker. Given a new frame from the real-time video stream, the proposed method overlays the background image on the marker region with geometric deformation and photometric adjustment. To relax the planar assumption, our method estimates the pixel-wise motion of the marker's background to deform the background image according to the estimated motion and overlays the deformed image on real-time frames. This approach reduces discontinuities on the boundary between the marker region and its surroundings.

2 Overview of AR Marker Hiding

Figure 1 shows a flow diagram of the proposed AR marker hiding method. 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.

In process (I), the proposed method first obtains a background image without a marker by either preliminarily capturing an actual background or by applying an image inpainting technique to the AR marker region. As shown in Fig. 2, when capturing the background image preliminarily, we first capture a background image (I-1) and subsequently place a marker while the camera pose is fixed (I-2). Next, a homography matrix is determined to transform the image with the AR marker in (I-2) so that the marker can be a square. The homography matrix then transforms the background image captured in (I-1) to obtain a rectified background image B in (I-3). Note that for image inpainting, we follow our previous marker hiding method [8]. 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 a square (I'-2). An image inpainting method then generates the rectified background image B in (I'-3). The proposed method then detects feature points around the marker region in the rectified background image B and calculates the degrees of confidence of the feature points (I-4), each of which is an

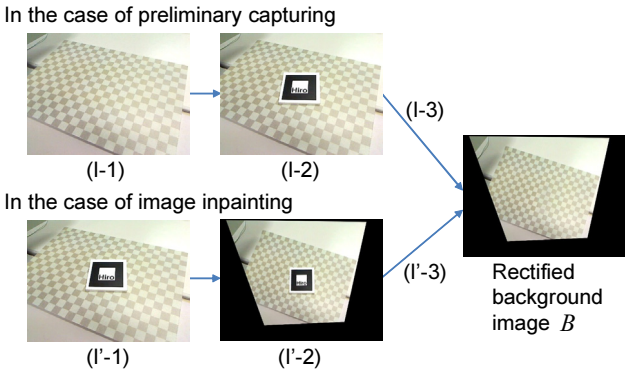

Figure 2: Obtaining rectified background image B.

autocorrelation of an image patch around a feature point.

In process (II), the proposed method visually removes a marker from 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 obtained on the basis of the camera pose. In process (II), the homography matrix is set so that the marker's size can be the same in the rectified frame and the image obtained in process (I) (II-3). Next, the proposed method finds the pixels in the rectified frame that correspond to the feature points detected from B (II-4), and then deforms B based on the feature point correspondences (II-4). Poisson blending [13] is then used to adjust the color of the deformed image to compensate for the difference in luminance 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 from the real-time video stream (II-6). Finally, the resulting frame is displayed to the user (II-7). Note that processes (I) and (II) can be performed simultaneously for an image inpainting case as in [8].

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

3 Detection of Feature Points and Confidence Calculation

In process (I-4) in Fig. 1, the proposed method first determines marker region Ω, which includes the AR marker but 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 Ω's dilated region by l pixels (Fig. 3). We then detect the feature points in $\partial\Omega$ and calculate their degrees of confidence.

To alleviate discontinuities in textures, the proposed feature point detector must satisfy the following requirements. (i) Feature points should be distinguishable from their surroundings to determine reliable correspondences. (ii) Feature points should distribute uniformly, and feature points on straight edges are acceptable if no corners exist around them. (iii) The number of detected feature points should be sufficiently large for accurate motion interpolation. Considering these requirements, the proposed method employs the following algorithm for feature point detection, which also provides the degree of confidence for each feature point as a measure of distinguishability.

Specifically, as shown in Fig. 3(left), the proposed feature point detector applies the Laplacian of Gaussian (LoG) filter to the recti-

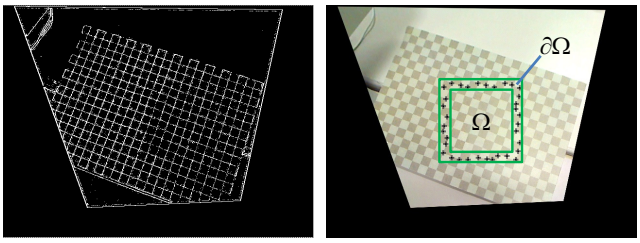

Figure 3: Examples of our feature point detection. Left images show edges extracted with LoG and right images show detected feature points in region $\partial\Omega$.

fied background image B and finds zero crossing points as feature point candidates. Then, the degree of confidence is calculated for each candidate in $\partial\Omega$. The degree of confidence $C(\mathbf{x}_i)$ is calculated for pixel i at $\mathbf{x}_i$ based on autocorrelation of the local patch centered at $\mathbf{x}_i$ as follows:

$$C(\mathbf{x}_i) = \frac{|N_{\mathbf{x}_i}| - \sum_{\mathbf{x}' \in N_{\mathbf{x}_i}} NCC^{(B,B)}(\mathbf{x}_i, \mathbf{x}')}{|N_{\mathbf{x}_i}|}, \tag{1}$$

where $N_{\mathbf{x}_i}$ is a set of pixels in the local patch centered at $\mathbf{x}_i$ excluding the pixel at $\mathbf{x}_i$ and $|N_{\mathbf{x}_i}|$ is the number of pixels in $N_{\mathbf{x}_i}$. $NCC^{(B,B)}(\mathbf{x}_i, \mathbf{x}')$ is the normalized cross correlation of pixel values between local patches $N_{\mathbf{x}_i}$ and $N_{\mathbf{x}'}$ in rectified background image B and is defined as follows:

$$NCC^{(B,B)}(\mathbf{x}_i, \mathbf{x}') = \frac{\sum_c \sum_{\mathbf{p} \in W} I_c^B(\mathbf{x}_i + \mathbf{p}) I_c^B(\mathbf{x}' + \mathbf{p})}{\sqrt{\sum_c \sum_{\mathbf{p} \in W} I_c^B(\mathbf{x}_i + \mathbf{p})^2} \sqrt{\sum_c \sum_{\mathbf{p} \in W} I_c^B(\mathbf{x}' + \mathbf{p})^2}}, \tag{2}$$

where $I_c^B(\mathbf{x}_i)$ is the pixel value in channel c (one of red, green, and blue channels of the pixel in the RGB color space) of the pixel at $\mathbf{x}_i$ in rectified background image B, and $\mathbf{p}$ is a shift vector in local patch W. With this definition, a pixel whose patch has low correlation with its surrounding patches has a high degree of confidence.

Next, as shown in Fig. 3(right), the feature point detector picks feature points from candidates using their degrees of confidence while maintaining their uniform distribution. For this, the feature point detector determines candidates as feature points if the candidates satisfy the following conditions in descending order of their degrees of confidence.

(I) The distance between the candidate and any feature point obtained so far should be greater than L_1, where L_1 is a constant.

(II) The degree of confidence C of the candidate should exceed P_2% of that of an already selected feature point if the distance between the candidate pixel and the feature point is less than L_2, where P_2 and L_2 are constants.

(III) The degree of confidence C of the candidate should be greater than P_3% of that of the first feature point, which has the largest degree of confidence among the feature points, where P_3 is a constant

Condition (I) prevents the distribution of feature points from becoming overly concentrated. Condition (II) inhibits a candidate with a relatively greater degree of confidence from being selected when there is another feature point with a greater degree of confidence near the candidate. Condition (III) selects a candidate with a relatively low degree of confidence when there are no other feature points with greater degree of confidence around it to prevent the distribution of feature points from becoming too sparse. Note that the k-th feature point is denoted by $\mathbf{x}_k$ in the following.

4 Deformation of Background Image Based on Motion Interpolation

Process (II-4) first determines correspondences between the feature points selected in the marker's surrounding region $\partial\Omega$ in the 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 based on the interpolated motion. In the following sections, we describe the background image deformation of the f-th frame.

4.1 Correspondence in marker's surrounding region

Pixel $\mathbf{y}_{f,k}$ is determined in the rectified f-th frame corresponding to feature point $\mathbf{x}_k$ in region $\partial\Omega$ in the rectified background image B, essentially by finding a region in the f-th frame that is similar to the local patch around $\mathbf{x}_k$. Note that naïvely scanning the entire image is inefficient. Considering that the proposed method should find correspondences at a sufficiently high frame rate, the temporal consistency of the pixels in the f-th and the $(f-1)$-th frames corresponding to $\mathbf{x}_k$ can be leveraged. Based on this temporal consistency, the proposed method presumes that $\mathbf{y}_{f,k}$ is in the region $G(\mathbf{y}_{f-1,k})$ centered at $\mathbf{y}_{f-1,k}$. Region $G(\mathbf{y}_{f-1,k})$ can be small because, even without the planer assumption, a point sufficiently close to the marker after rectification in (II-3) remains at nearly the same position regardless of the camera motion. Based on this, the proposed method finds pixel $\mathbf{y}_{f,k}$ corresponding to $\mathbf{x}_k$ in descending order of $\mathbf{x}_k$'s degree of confidence as follows:

$$\mathbf{y}_{f,k} = \underset{\mathbf{y}' \in G(\mathbf{y}_{f-1,k})}{\arg\max} \frac{NCC^{(B,f)}(\mathbf{x}_k, \mathbf{y}')}{1 + \sum_{\mathbf{x}_l \in M_{\mathbf{x}_k}} D_{\mathbf{s}}(\mathbf{t})}, \tag{3}$$

where $\mathbf{s} = \mathbf{x}_k - \mathbf{x}_l$ and $\mathbf{t} = \mathbf{y}' - \mathbf{y}_{f,l}$. Here $NCC^{(B,f)}(\mathbf{x}_k, \mathbf{y}')$ is the normalized cross correlation between the patch centered at pixel $\mathbf{x}_k$ in rectified background image B and the patch centered at pixel $\mathbf{y}'$ in the rectified f-th frame (see Eq. (2)). $M_{\mathbf{x}_k}$ is the set of the feature points whose degrees of confidence are greater than $\mathbf{x}_k$ in a certain region centered at the feature point $\mathbf{x}_k$. This means that $M_{\mathbf{x}_k}$ contains the feature points to which corresponding pixels have been found in the rectified f-th frame. $D_{\mathbf{s}}(\mathbf{t})$ is a cost term based on the difference in shift vectors and the distance between feature points, which is defined as follows:

$$D_{\mathbf{s}}(\mathbf{t}) = \begin{cases} 0 & (d(\|\mathbf{s}\|) > \|\mathbf{t}\|) \\ \kappa & (\text{otherwise}) \end{cases}, \tag{4}$$

where $d(\|\mathbf{s}\|)$ is a monotonically increasing function that gives higher value as $\|\mathbf{s}\|$ increases. For example, we use $d(\|\mathbf{s}\|) = \|\mathbf{s}\|/10$ in our experiments. $D_{\mathbf{s}}(\mathbf{t})$ is a cost function that encourages a feature point to move in a similar manner to the neighboring feature points. The cost function allows the difference in the shift vectors $\|\mathbf{t}\|$ to become greater as the distance $\|\mathbf{s}\|$ between feature points increases.

4.2 Motion Interpolation

Using $\mathbf{y}_{f,k}$ corresponding to feature point $\mathbf{x}_k$, the proposed method interpolates the shift vectors from the 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 the rectified background image B based on the shift vectors. Note that the shift vectors of not only pixels in Ω but also pixels in $\partial\Omega$ are required, because pixels in $\partial\Omega$ in B may be occluded by the marker in the f-th frame.

Specifically, based on the assumption that pixels move in a similar manner to feature points with high degrees of confidence around

them and the motion of adjacent pixels is highly correlated, the proposed method estimates the motion of each pixel in $\Omega \cup \partial\Omega$ in B by minimizing the following energy function:

$$E = \sum_{i \in \Omega \cup \partial\Omega} \sum_{k} \omega_{i,k} \|\mathbf{u}_i - \mathbf{u}_k\|^2 + \alpha \sum_{(i,j) \in A} \|\mathbf{u}_i - \mathbf{u}_j\|^2, \quad (5)$$

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

$$\omega_{i,k} = \max_k C(\mathbf{x}_k) \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_k\|^2}{\sigma^2}\right), \quad (6)$$

where σ is a constant. Minimization of E in Eq. (5) 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. The system's coefficient matrix is sparse; thus it can be solved by the conjugate gradient method for sparse systems, which works efficiently on GPUs (example implementation is provided in [1]).

Finally, rectified background image B is deformed by forward-projecting each pixel in image B based on the obtained shift vectors and linearly interpolating pixel values. Note that our implementation uses texture mapping provided in OpenGL.

5 Experiments

To demonstrate the effectiveness of the proposed method, we performed experiments to visually remove an AR marker using a PC with a Windows 7, Core i7-990X 3.46 GHz CPU, 12 GB memory, and a GeForce GTX Titan GPU. We used a USB camera (Logicool Qcam Pro 9000) to capture real-time input video streams, each of whose frames consists of 640 × 480 pixels. We used ARToolkit[6] for camera pose estimation and a square AR marker with edge length of 80 mm, which was attached to a relatively thick object with edge length of 95 mm and thickness of 7 mm, as is shown in Fig. 4. The proposed method was tested under the following three environments:

Scene A A curved background geometry with grid-patterned texture (Fig. 5).

Scene B A planar background geometry with stripe texture (Fig. 6).

Scene C A step background geometry with grid-patterned texture (Fig. 7).

Table 1 shows the parameter values used in experiments. To obtain background images, we used our previous approach [8] for scenes A and B, which applies an image inpainting method [9] to a rectified image with a marker. Initially, we captured a single background image for scene C.

In the experiments, we compared the results obtained by the proposed method ((d) in Figs. 5-7) with those obtained by a conventional approach that uses a homography matrix to transform the background image with color adjustment ((b) in Figs. 5-7). To confirm the effectiveness of the color adjustment, we also show the results obtained by the proposed method without color adjustment ((c) in Figs. 5-7). (a) and (e) in Figs. 5-7 show the input frames and the rectified frames with the tracking results obtained by the proposed method. (f) in Figs. 5-7 shows the deformed images of the rectified background images in the marker and its surrounding regions. The first row in each figure shows the results when the camera was mostly static. The second and third rows show the results when camera motion was in motion. In the following, we discuss the results obtained for each scene in further detail.

Figure 4: AR marker used in experiments.

Figure 5 shows the experimental results for scene A. As can be seen, the appearance of the texture changes around the marker in the rectified image because of camera motion and the curved geometry as shown in (e). Thus, (b) demonstrates large discontinuities in the texture around the boundary. In (c), the edges in the grid pattern are successfully connected on the boundary; however brightness between the marker and its surrounding regions differs. Conversely, the proposed method did not yield geometric and photometric discontinuities in the texture. The proposed method deformed the background image (Fig. 5(f)) based on the tracking of feature points in the marker's surrounding region.

Compared with scene A, the appearance changes in the texture of scene B (Fig. 6) do not seem to be significant (Fig. 6(e)); however displacement of the texture occurs because of the thickness of the marker base. Thus, the discontinuous straight lines can be seen in (b) without deformation. Noticeable difference in brightness can also be observed without color adjustment in (c). The proposed method yields natural results without significant visual artifacts, as is shown in (d). For the stripe texture in this scene, the proposed method does not always yield 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 deform the background image excessively, as is shown in (f). However, this excessive deformation does not cause visual artifacts if the textures in the marker region and the surrounding region are the same, because the proposed method can compensate for the displacement in a direction orthogonal to the stripes.

In scene C shown in Fig. 7, the marker was leaned against a step. The vertical plane is visible in the rectified background image, as is demonstrated in the first row of (f); however, it can become invisible in the input frame depending on the camera pose, as is demonstrated in the input frames in (a) and rectified frames (e). 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 third row of

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_1, L_2	11 pixels, 31 pixels
P_2, P_3	80%, 1%
Range for calculating confidence N	3×3 pixels
Search range G	5×5 pixels
Size of patch W	11×11 pixels
Range for interinfluence of feature points$M$	50×50 pixels
κ, α, σ	0.001, 1000, 25

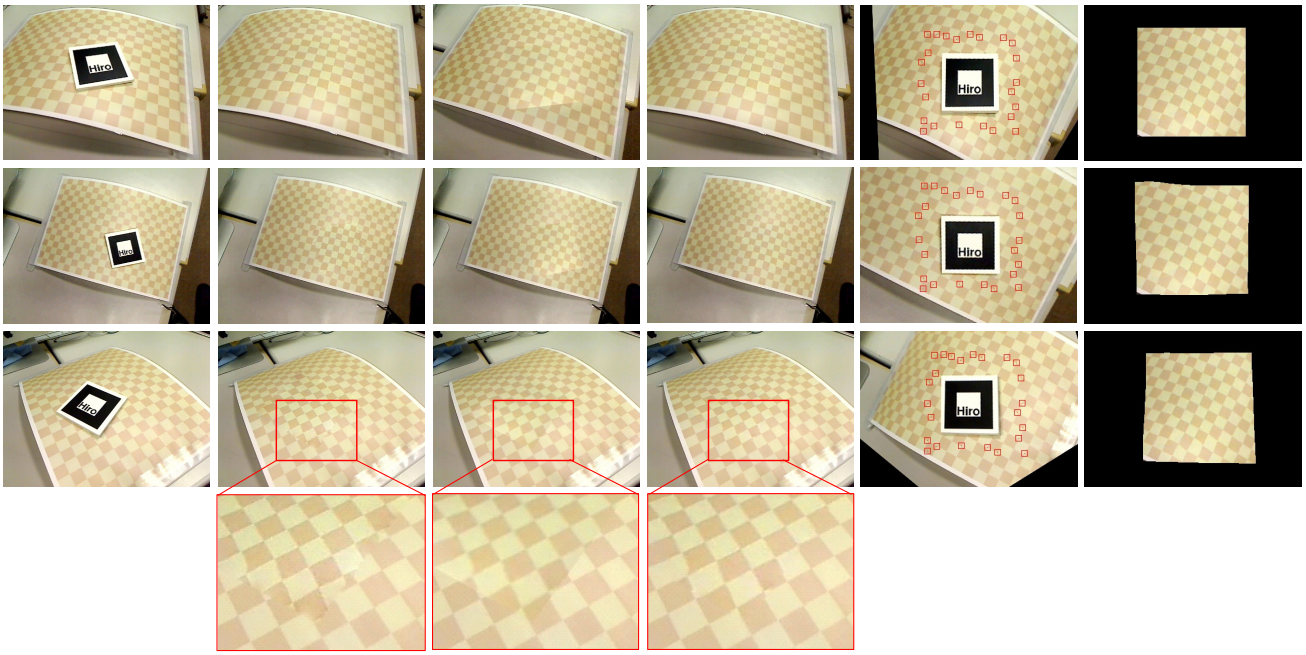

Figure 5: Experimental results for scene A with a curved shape and a grid texture.

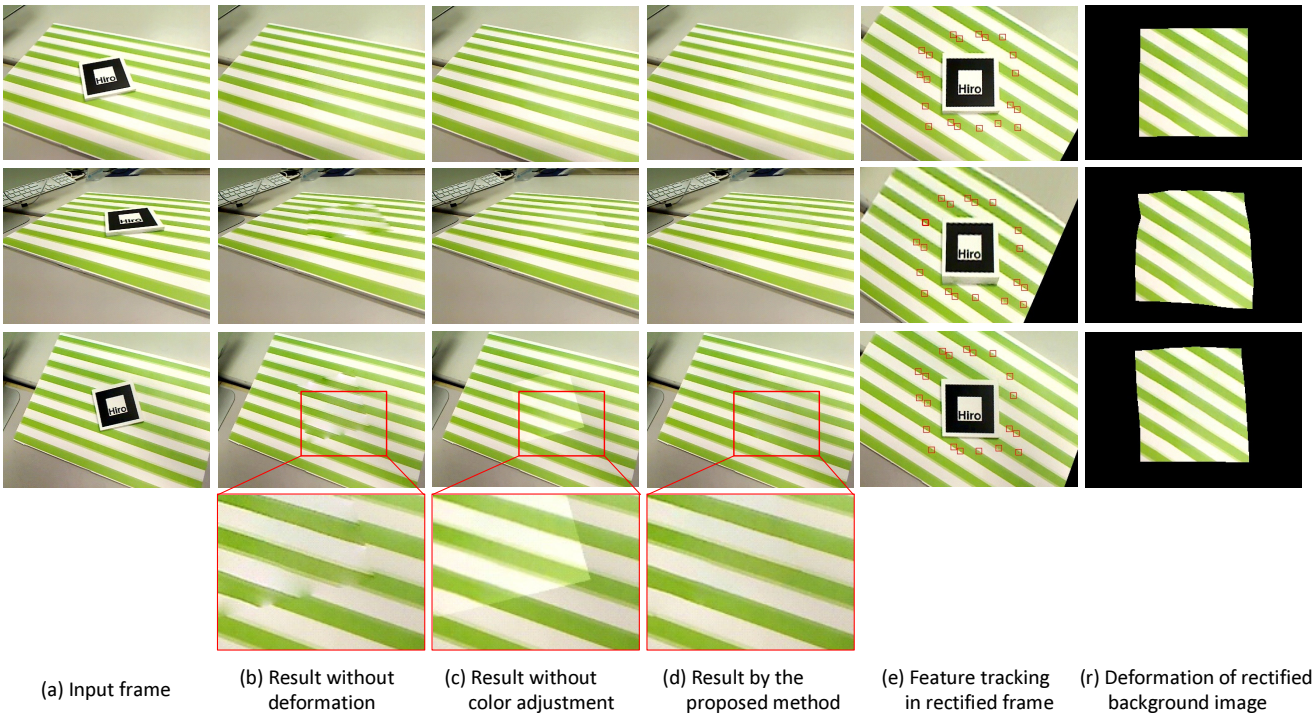

Figure 6: Experimental results for scene B with a planar shape and a stripe texture.

Fig. 7. However, the proposed method generated more continuous texture than the method without deformation on the boundary between the marker and its surroundings excluding the region of the vertical plane.

Note that the numbers of detected feature points for scenes A, B, and C were 24, 20, and 34, respectively, and the frame rates for the scenes were 4.5, 4.2, and 4.4 fps, respectively.

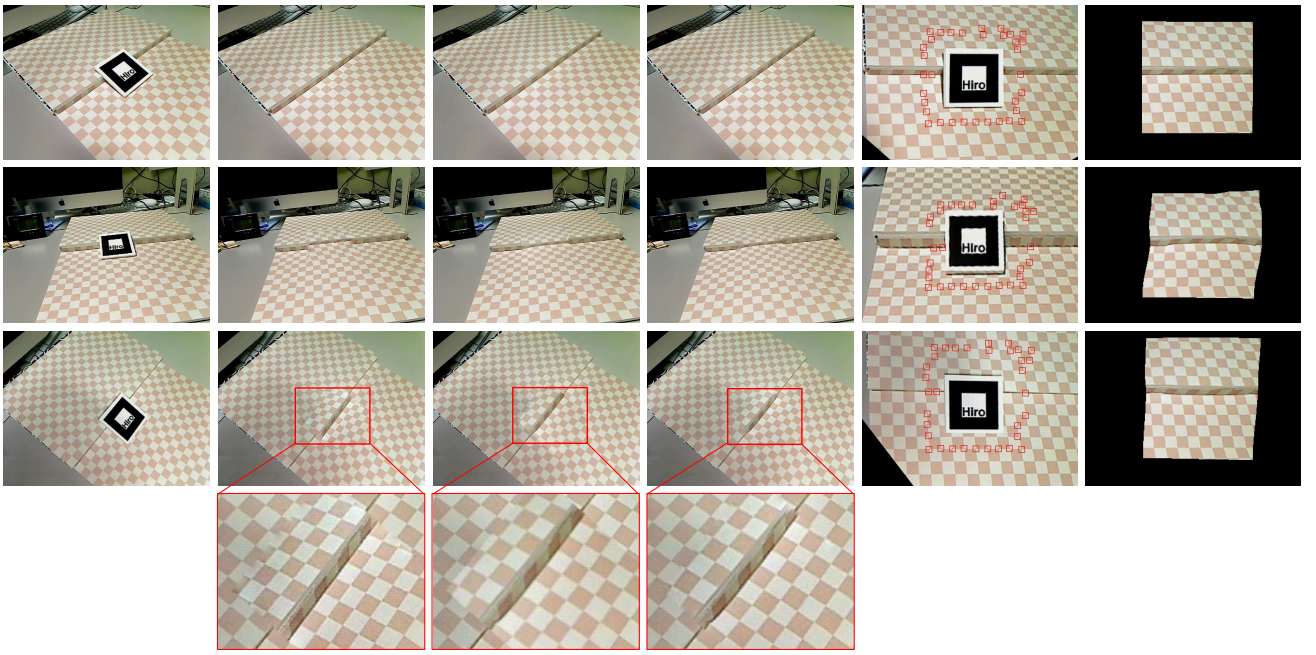

Figure 7: Experimental results for scene C with a step background geometry with grid-patterned texture.

6 Conclusion

This paper has proposed an AR marker hiding method based on real-time deformation of a background image. The proposed feature point detection and tracking algorithm provides uniformly distributed feature points, which is desirable for motion interpolation. The proposed method can achieve real-time pixel-wise deformation using an energy function that can be efficiently minimized using GPUs.

In experiments, we confirmed that the proposed texture deformation-based AR marker hiding method can generate visually natural images even for a thick AR marker and for nonplanar background geometries. Note that we have also obtained good results for a scene with a stripe texture, which suffers from the aperture problem when detecting feature points. However, the proposed method could not handle scenes with relatively complex geometry, which causes occlusions according to camera motion, because the proposed method assumes that the motion of adjacent pixels is similar. Future work includes AR marker hiding for a variety of background geometries at higher frame rates. In addition, we plan to apply real-time texture deformation to DR techniques that visually remove various objects from a real-time video stream.

Acknowledgements

This research was supported in part by the Ministry of Internal Affairs and Communications SCOPE No. 152107001 and the Japan Society for the Promotion of Science KAKENHI No. 15K16039.

References

[1] Nvidia developer zone (http://docs.nvidia.com/cuda/cuda-samples/#conjugategradient).

[2] 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*, pages 99–102, 2009.

[3] 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*, pages 11–20, 1996.

[4] 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*, pages 207–212, 2010.

[5] J. Herling and W. Broll. High-quality real-time video inpainting with pixmix. *IEEE Trans. Visualization and Computer Graphics*, 20(6):866–879, 2014.

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

[7] N. Kawai, T. Sato, and N. Yokoya. Diminished reality considering background structures. In *Proc. Int. Symp. Mixed and Augmented Reality*, pages 259–260, 2013.

[8] 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*, 1(4):343–353, 2013.

[9] N. Kawai and N. Yokoya. Image inpainting considering symmetric patterns. In *Proc. Int. Conf. Pattern Recognition*, pages 2744–2747, 2012.

[10] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Int. Symp. Mixed and Augmented Reality*, pages 225–234, 2007.

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

[12] 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*, pages 11–19, 2013.

[13] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Trans. Graphics*, 22(3):313–318, 2003.

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

[15] S. Siltanen. Texture generation over the marker area. In *Proc. Int. Symp. Mixed and Augmented Reality*, pages 253–254, 2006.