

# Dynamic Projection Mapping onto Deforming Non-Rigid Surface Using Deformable Dot Cluster Marker

Gaku Narita, Yoshihiro Watanabe, and Masatoshi Ishikawa

**Abstract**—Dynamic projection mapping for moving objects has attracted much attention in recent years. However, conventional approaches have faced some issues, such as the target objects being limited to rigid objects, and the limited moving speed of the targets. In this paper, we focus on dynamic projection mapping onto rapidly deforming non-rigid surfaces with a speed sufficiently high that a human does not perceive any misalignment between the target object and the projected images. In order to achieve such projection mapping, we need a high-speed technique for tracking non-rigid surfaces, which is still a challenging problem in the field of computer vision. We propose the Deformable Dot Cluster Marker (DDCM), a novel fiducial marker for high-speed tracking of non-rigid surfaces using a high-frame-rate camera. The DDCM has three performance advantages. First, it can be detected even when it is strongly deformed. Second, it realizes robust tracking even in the presence of external and self occlusions. Third, it allows millisecond-order computational speed. Using DDCM and a high-speed projector, we realized dynamic projection mapping onto a deformed sheet of paper and a T-shirt with a speed sufficiently high that the projected images appeared to be printed on the objects.

**Index Terms**—Non-rigid surface tracking, fiducial marker, projection mapping, spatial augmented reality, high-speed vision

## 1 INTRODUCTION

IN recent years, Augmented Reality (AR), which is a technique for literally augmenting the real world by adding visual, acoustic or tactile information to objects, has been attracting much attention. In particular, the technique for projecting registered images onto a target object according to its position and shape is called projection mapping or Spatial Augmented Reality (SAR) [5]. Compared with traditional AR through a hand-held display or head-mounted display (HMD), projection mapping has the advantage that it does not force users to wear any special devices, and the information displayed via projection mapping can be simultaneously shared with many users. Due to these advantages, many applications based on projection mapping have been proposed. For example, projection mapping onto theme park attractions and large buildings in entertainment exhibitions has attracted people all around the world [28]. Projection mapping can also be used in user interfaces [16], [19], prototyping [27], [47], learning assistance [22], [23] and so on.

Existing projection mapping techniques can be classified in terms of the dynamics and shapes of their target object. Since the proposal of the “Shader Lamps” concept by Raskar et al. [39], the target objects in general projection mapping have been limited to static or quasi-static and rigid

objects [4]. To overcome these limitations, many studies on dynamic projection mapping with moving objects have been reported recently [2], [3], [7], [40], [41], [43]. For example, Siegl et al. tracked a white statue using a depth sensor, and images with consistent shape and optimized luminance were projected onto the statue in real time by using multiple projectors [41]. Resch et al. estimated the 3D motion of a rigid object using feature points, and the projected images were corrected at 10 fps [40]. However, the projected images in these approaches are not able to follow rapidly moving objects because the tracking and projection speeds are not high enough.

There are also projection mapping systems specially designed for higher speed. Okumura et al. have developed an optical system, called a Saccade Mirror, that is able to track a ball by rapidly changing the gaze direction at 1,000 Hz using a high-speed camera and galvanometer mirrors [33]. Combining the Saccade Mirror and a projector, Okumura et al. and Sueishi et al. have realized dynamic projection mapping systems for rapidly moving rigid objects [34], [43]. The results of their work revealed that the faster projection mapping remarkably enhanced the degree of immersion. However, the target object in these systems is assumed to be spherical or planar because the pose of the target object relative to the gaze direction of the Saccade Mirror must not change during projection. In addition, the user needs to initialize its position before starting tracking.

In this paper, we focus on dynamic projection mapping onto a non-rigid surface. In this situation, the target motion has more degrees of freedom than a rigid body, resulting in more difficult problems, especially tracking, which must be solved in real-time. Such projection mapping, however, will remarkably expand the possible applications.

- The authors are with the Graduate School of Information Science and Technology, University of Tokyo, Tokyo 113-8654, Japan.  
E-mail: {gaku\_narita, yoshihiro\_watanabe, masatoshi\_ishikawa}@ipc.i.u-tokyo.ac.jp.

Manuscript received 1 Feb. 2016; revised 29 Apr. 2016; accepted 30 May 2016.  
Date of publication 19 July 2016; date of current version 1 Feb. 2017.

Recommended for acceptance by M. C. Lin.

For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.

Digital Object Identifier no. 10.1109/TVCG.2016.2592910

Interesting applications can be found in a number of studies. For instance, Fujimoto et al. have realized projection-based geometrically-correct texture mapping onto a deformable acrylic foam sheet [11]. In their paper, applications such as design support and digital draping are discussed. Every time the object is reshaped this system needs to perform 3D sensing with gray-coded patterns and IR marker recognition, which consume about 8.5 s, so that the system is not capable of operating in real-time. On the other hand, the deformable paper display system proposed in [42] uses a depth sensor to acquire the shape of a target sheet of paper based on a paper deformation model. The system can project, at a maximum frame rate of 25 fps, depth-consistent images that appear to be printed. Applications such as exploring volumetric data and animating virtual paper characters by deforming the paper display were developed in their paper. Punpongson et al. have presented a projection-based AR system that tracks the tangential deformation of an elastic clay object on which invisible textures are written with IR ink at about 30 fps and visualized the deformation using a projector [38]. Applications such as science education and toys for small children are discussed in their paper. However, in these systems, the delay involved with image processing for recognizing the deformation and the delay of the projector cause a misalignment between the target object and the projected image when the target is moving and deforming quickly. This misalignment problem causes difficulties in user interaction and lack of immersion.

So, how should this delay be reduced in order for the human eye not to perceive such misalignments? The answer to this question can be found in [32]. They have developed a projection-based touch-input device which responds with 1 ms latency. In their user experiment, a white square is displayed on a freely moving user's hand while changing the projection latency. The results showed that participants perceived a difference when the projection latency was greater than 6.04 ms on average. This means that in order to realize dynamic projection mapping onto a deforming non-rigid surface without any perceivable misalignment, we need a high-speed projector and high-speed non-rigid surface tracking technique that both achieve a latency of a few milliseconds. In order to solve the former problem, projectors based on a Digital Micromirror Device (DMD) have been proposed. In one approach, which was specially designed for an optical see-through display [26], [51], the binary image is updated in every refresh so as to reduce the alignment errors using sensor feedback. The integrated image is perceived as a low-latency projected image. Also, there is a high-speed projector that can project eight-bit images at 1,000 fps with 3 ms latency from image generation in a computer to projection [50]. This is realized by closely coordinating mirror control based on a DMD and the on/off blinking of a high-brightness LED and can exceed the speed achievable by using DMD control alone.

In this paper, in order to resolve the problem of high-speed non-rigid surface tracking, we propose the Deformable Dot Cluster Marker (DDCM), a novel fiducial marker for high-speed tracking of non-rigid surfaces using a high-frame-rate camera. The DDCM is formed of four types of dot clusters arranged in an array structure. The DDCM has three performance advantages. First, it can be detected even when it is strongly deformed. Second, it realizes robust

tracking even in the presence of external occlusions caused by user interactions and self-occlusions caused by the object's deformation itself. Third, it enables a remarkably high computational speed: less than 2 ms per frame for detection and less than 1 ms per frame for tracking.

Using this DDCM and a high-speed projector, we realized dynamic projection mapping onto a deformed sheet of paper and a T-shirt. As the high-speed projector, we used the one reported in [50]. The results showed the ability to robustly and consistently display projected images on target objects with a speed sufficiently high that the projected images appeared to be printed on the objects.

## 2 RELATED WORK

In this section, we mainly discuss work related to non-rigid surface tracking. The problem of rigid object tracking or registration has been widely studied in the fields of robotics, 3D reconstruction, AR and so on. On the other hand, non-rigid surface tracking is still a challenging problem in the field of computer vision. This is because, compared with rigid objects, the deformation of a non-rigid surface has more degrees-of-freedom and involves self-occlusions as well as external occlusions. In the following discussion, we give the speed performance of the conventional approaches as reference information, although they were not measured on the same hardware.

The texture naturally possessed by the target can be used as a useful cue for deformation detection and tracking. For example, Pilet et al. have proposed a feature-based method which is able to run at 10 fps [37]. This method finds corresponding feature points between the deformed image and the reference image using a wide-baseline matching algorithm [24]. Then, deformed meshes are reconstructed using a well-designed robust estimator and deformation energy [10]. Bellile et al. have proposed a template-based direct method [12], [13]. This method deforms small templates in the reference image using the Thin Plate Spline energy [6] to register them with the deformed image with off-line processing. While these methods target surfaces with arbitrary textures, there is also a method for Near-Regular Textures (NRTs), which consist of regularly repeated textons [25]. In this work, the array-structure of textons are modeled as a Markov Random Field, and robust tracking is achieved even in the presence of occlusions with off-line processing.

Although these texture-based methods have the advantage that the objects can be used directly without any equipment, they face several problems in projection mapping applications. First, the tracking performance strongly depends on the texture density. Second, the dense textures that are needed for robust tracking are not suitable for projection mapping because they degrade the visibility of the projected images. Third, the projected images and the original texture are mixed in the observed image, and tracking may fail as a result. Fourth, texture-based methods generally have high computational cost, and almost all of them involve off-line processing.

The other approach for non-rigid surface tracking is to use a fiducial marker. Compared with texture-based methods, fiducial markers can achieve more robust tracking because we can suitably design their color and shape [45]. In projection mapping applications, IR markers that are

invisible to the human eye, such as those implemented by IR ink or semi-transparent retro-reflective material [30], [35], are suitable not only because invisible markers enhance the degree of immersion, but because an IR filter in front of the camera can cut visible light, and image processing for marker detection in IR region is not disturbed by the projected image. However, the above markers are assumed to be set on planar objects, and they cannot be used to recognize the deformation of a non-rigid object.

There are some markers that can be used to detect and track non-rigid surfaces. For example, there are markers that utilize an array-structure [14], [15], similar to NRT tracking [25]. These markers are designed with regularly repeated squares, such as a chessboard, and are tracked at 10-50 fps. However, they have the limitations that manual set-up is required [14] before starting tracking, and color-coded patterns cannot be implemented as IR markers [15]. On the other hand, there are markers with patterns that have a topological invariant, a value that is not changed under any deformation. Horvath et al. have proposed a honeycomb marker which consists of hexagonal symmetric cells with different shadings [17]. Intersections of hexagonal cells are effectively detected and identified using the combination of shadings around the intersections, which is a topological invariant. This allowed a processing speed of 5 ms per VGA input image. However, their paper does not mention how the marker can be deformed. In addition, it is difficult to implement shading using IR ink or retro-reflective material. Elbrechter et al. have presented a fiducial marker consisting of nested patterns with a hierarchical structure, which are also topologically invariant [9]. This allowed a processing speed of 14.5 ms. However, it is difficult to densely arrange the patterns in the marker, and occlusion robustness is not guaranteed because the size of each pattern is large.

On the other hand, the Deformable Random Dot Marker (DRDM) [44] is a spatially dense marker. The DRDM consists of many randomly distributed dots [46]. These dots are tracked frame-by-frame at a maximum frame rate of about 50 fps based on the affine invariant descriptor [29] which is computed from the relative position of each dot. This has the advantage that it is easy to implement the DRDM with IR ink or retro-reflective material; however, there are some problems. First, before starting frame-by-frame tracking, the DRDM should be set to a planar state for initial detection. Second, the affine invariant descriptor is drastically modified even when the relative positions of dots are slightly changed so that tracking is likely to fail in the situation where the marker is strongly deformed between two successive frames. Third, the DRDM tends to cause false-positive tracking, where a wrong ID is assigned to a certain dot, when the object is strongly deformed and the dots become crowded together, and the occurrence of these false-positives cannot be detected. Although each of these deformable markers has its merits and demerits, there is a common problem of computational speed. As mentioned in the introduction, in order to realize dynamic projection mapping at a speed sufficiently high that a human does not perceive any misalignment, millisecond-order tracking is required.

To achieve high-throughput, low-latency tracking, high-frame-rate image capturing is essential. Additionally, with

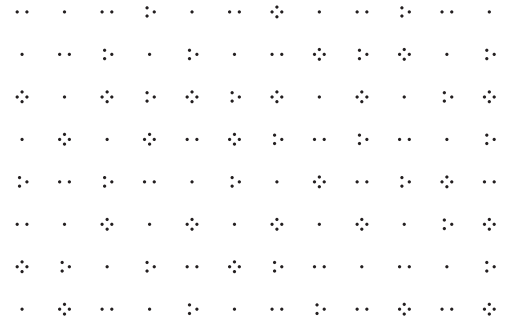


Fig. 1. One example of Deformable Dot Cluster Marker. Dot clusters are distributed in an array structure.

high-frame-rate imaging, the assumption that the displacement of the object between two successive frames is tiny is satisfied, and this enables robust, high-speed tracking by virtue of the simplified image processing. For example, based on this assumption, Ishii et al. proposed an efficient algorithm for tracking of a single target implemented only with a few logical operations and realized tracking at 1,000 fps [18]. This work was extended to tracking of 10 targets [49] and tracking of 1,000 target [48] at 1,000 fps. This type of high-speed multi-target tracking in particular is related to fiducial marker tracking, which can be regarded as tracking of many patterns in a marker. However, their algorithm cannot be applied directly to non-rigid surface tracking because it is not able to detect and identify the marker nor stably track patterns that are sometimes occluded.

Some of these problems have been attempted to be solved by using an array of dot markers arranged on a non-rigid surface [31]. This method enables the tracking of the dot marker array, even in the event of occlusion, at high speed with a performance of about 0.2 ms per frame. However, the method requires an initialization step to recognize all dots, in which the object should be planar, and all dots need to be visible from the camera. Also the tracking sometimes fails because this method is based on only recursive, frame-by-frame tracking. In this paper, we extend this method to solve these problems.

### 3 DEFORMABLE DOT CLUSTER MARKER

#### 3.1 Marker Design

In this section, we describe the design of the Deformable Dot Cluster Marker. Taking into account the related work described above, it is assumed that a marker pattern that satisfies the following three properties is suitable for high-speed, robust tracking of non-rigid surfaces. First, each pattern should have topological invariance which is not changed under any deformation. Second, the patterns should be arranged in an array structure. Third, each pattern should be small because many small patterns can be distributed in the marker, which makes tracking based on the high-speed imaging assumption more robust and faster. In addition to these three properties, it is desirable that the marker be easily implemented with IR ink for projection mapping applications.

From these considerations, we propose the DDCM shown in Fig. 1. The DDCM consists of four types of dot clusters which are distributed in an array structure. Dot clusters are shown in Fig. 2. The dot number in each dot cluster is a



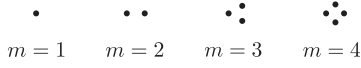



Fig. 2. Four types of dot clusters.  $m$  represents the dot number. Each dot cluster consists of one, two, three or four dots.

topological invariant which is not changed under any deformation. Here, dot numbers are distributed randomly and exclusively. The exclusive distribution means that when we choose an arbitrary  $p \times p$  dot cluster (we call it an identification window), the four series which are generated by  $p^2$  dot numbers in the identification window starting at the four corners, as shown in Fig. 3, are not coincident with any series which is generated in any other identification window. Note that adjacent dot clusters have different dot numbers from each other in order to enhance the robustness of frame-by-frame tracking, described later.

The DDCM can be generated based on a brute-force search. First, we generate a marker pattern in which adjacent dot clusters have different dot numbers. In this step, the dot numbers are decided from top left to bottom right in raster order. At each location, we select the dot number randomly. This selection is performed until the adjacency condition is met. After the whole pattern is generated, we check whether this pattern satisfies the condition of the exclusive distribution. If the condition is met, we employ this pattern as the DDCM. If not, we go back to the first step. Using this algorithm, after around 1 million iterations, we found that the maximum size of a square DDCM was  $19 \times 19$ . In this search, the size of the identification window was assumed to be  $3 \times 3$ .

We also can generate multiple DDCM  changing the arrangement of the dot clusters. Therefore, when multiple markers are attached on different objects, we can track each object by distinguishing the markers. This is an advantage in the projection mapping application because the system can flexibly and easily change the projected images for multiple objects.

### 3.2 Processing Pipeline

The processing pipeline is shown in Fig. 4. The proposed method consists of detection for identifying dot cluster IDs from an individual frame, and frame-by-frame tracking for independently tracking each dot cluster through the image sequence. Detection and frame-by-frame tracking are executed independently and in parallel in separate threads. Every time detection processing is completed, the coordinates of the identified dot clusters are transferred to the frame-by-frame tracking thread as described in Section 3.4.2. Parallel processing of detection and frame-by-frame tracking enables a reduction in the total latency, and their

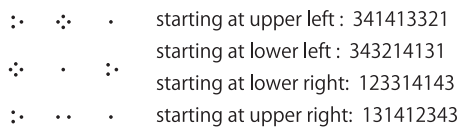


Fig. 3. Examples of sequences generated by dot numbers in a certain identification window,  $p = 3$ . These sequences start at the upper left, lower left, lower right, and upper right dot number. The dot numbers are distributed such that whichever identification windows in the DDCM are chosen, there are no sequences that are coincident with each other.

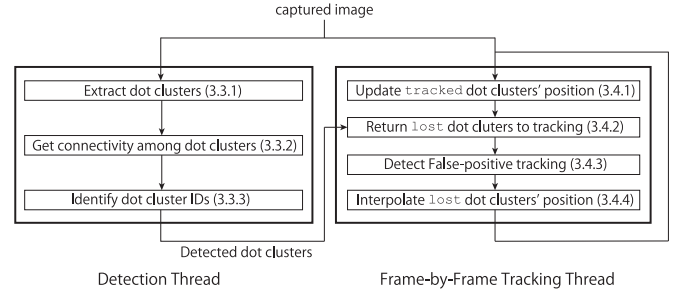


Fig. 4. Processing pipeline. The detection thread identifies dot clusters from an individual frame. The frame-by-frame tracking thread independently tracks each dot cluster through the image sequence. They are executed in parallel in separate threads.

performance levels are complementary. Parallelized tracking similar to our method can be found in some other studies [20], [21], [36].

### 3.3 Detection

In this section, we describe the detection method which identifies dot cluster IDs from individual captured images. Detection consists of three steps: extracting dot clusters, obtaining the connectivity among dot clusters, and identifying dot cluster IDs. Fig. 5 shows the detection pipeline. We discuss the details of these steps below.

#### 3.3.1 Dot Cluster Extraction

First, the captured image is adaptively binarized. In the experiments, the local threshold value was decided from a mean of the pixel values in the neighborhood. Next, the connected components are obtained from the binarized image by contour extraction. Here, we select connected components having an area larger than  $A_{\min}$  and smaller than  $A_{\max}$  in order to remove outliers, such as the background and image noise. We call a selected connected component a keypoint.

Next, the whole image is divided into square grid cells whose width is  $w_{\text{grid}}$ , and the number of keypoints in each grid cell is counted. Here, it can be assumed that the number of keypoints in a grid cell is not so large. For example, even when four dot clusters exist in a grid cell, the total number of keypoints in that grid cell would be 16 at most. Therefore, grid cells containing more than  $n_{\text{kp}}^*$  keypoints are considered to be outliers and are removed in the following procedure.  $n_{\text{kp}}^*$  is a threshold decided beforehand. For the remaining grid cells, we regard the keypoints that are in the same grid cell and within the distance  $d_{\text{cl}}^*$  as a dot cluster. Here,  $d_{\text{cl}}^*$  is a threshold decided beforehand. This grid division is effective not only for removing the outliers but also for reducing the amount of computation by restricting the region in which the distance between keypoints is calculated to a grid cell.

#### 3.3.2 Connectivity Acquisition Among Dot Clusters

Next, we perform Delaunay triangulation for all dot clusters extracted as described in the previous section. Delaunay triangulation is a technique for dividing point clouds into many triangles by connecting the points to each other. It has the property that the generated triangles can be as obtuse as

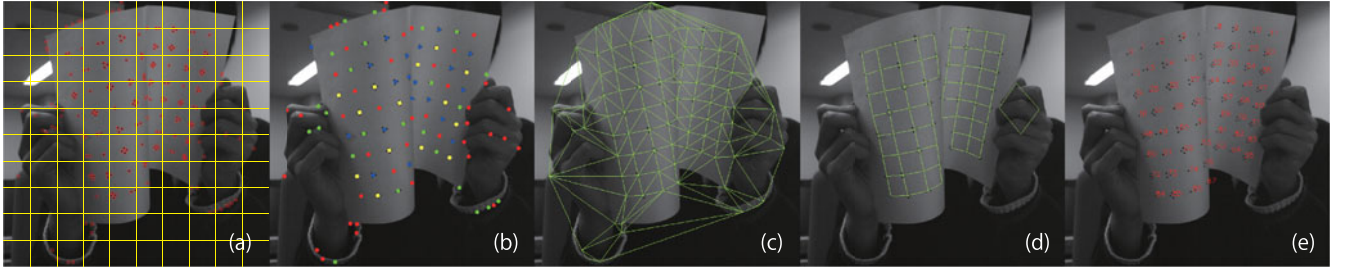


Fig. 5. Detection pipeline. (a) Keypoints are extracted from the binarized image. (b) We regard the keypoints that are in the same grid illustrated in (a) and close to each other as a dot cluster. The red, green, blue, yellow dots in (b) represent  $m = 1, 2, 3, 4$  dot clusters, respectively. (c) Delaunay triangulation for extracted dot clusters is computed. (d) The connectivities among dot clusters are obtained based on the breadth-first search. (e) Finally the IDs of each dot cluster are identified based on dot numbers by the hash table prepared beforehand. The red numbers in (e) represent IDs of identified dot clusters.

possible. There is also an efficient algorithm whose computational cost is  $\mathcal{O}(n \log n)$  for  $n$  input points [8].

The procedure to obtain the connectivity among dot clusters based on Delaunay triangulation is shown in Algorithm 1. The connectivity among dot clusters represents which dot clusters are connected to a certain dot cluster in up, down, left, right directions. Here, let  $S(e)$  be a quadrangle consisting of two triangles which have a common edge,  $e$ . We compute  $L_{sq}(S(e))$ , defined in Eq. (1) below, for all edges in Delaunay triangulation. A, B, C, and D in this equation are defined as shown in Fig. 6, and  $\mathbf{n}(\cdot)$  is the normalization operator of a vector. The reason why we compute the direction cosines, not the angles between edges, is to reduce the computational cost

$$L_{sq}(S(e)) := 1 - \frac{1}{3}(\mathbf{n}(\overrightarrow{AB}) \cdot \mathbf{n}(\overrightarrow{AD}))^2 - \frac{1}{3}(\mathbf{n}(\overrightarrow{CB}) \cdot \mathbf{n}(\overrightarrow{CD}))^2 - \frac{1}{3}(\mathbf{n}(\overrightarrow{AC}) \cdot \mathbf{n}(\overrightarrow{DB}))^2. \quad (1)$$

Here,  $L_{sq}(S(e))$  takes a greater value when quadrangle  $S(e)$  is similar to a square. We select quadrangles  $S(e)$  such that  $L_{sq}(S(e))$  is greater than the threshold  $L_{sq}^*$  defined beforehand and sort them in order of decreasing  $L_{sq}(S(e))$ . The sequence of sorted quadrangles is  $\{S^{(n)}\}$  (Algorithm 1, line 8).

Next, we take one quadrangle  $S^{(n)}$  from  $\{S^{(n)}\}$  and obtain the connectivity among dot clusters which are connected to  $S^{(n)}$  based on a breadth-first search (BFS) with the initial node  $S^{(n)}$ . The connectivity is represented as a 2D array  $M^{(n)}$  whose elements are quadrangles. The specific procedure for obtaining the connectivity is as follows. First, we prepare a large 2D array  $M^{(n)}$  and register quadrangles  $S^{(n)}$  with  $M_{0,0}^{(n)}$ . Then, we add  $S^{(n)}$  to queue  $Q$  and let  $S^{(n)}$  be visited (Algorithm 1, lines 14-15). Next, we remove a quadrangle  $S$  from queue  $Q$  and let  $e_l$  ( $l = 0, 1, 2, 3$ ) be the edges of  $S$  in the right, up, left, and down directions, respectively, as in Fig. 7 (Algorithm 1, lines 17-19). There are two types of quadrangle connections, as

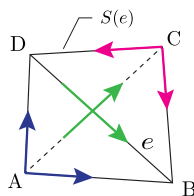


Fig. 6. Calculation of  $L_{sq}(S(e))$ , which becomes greater when quadrangle  $S(e)$  is similar to a square. The direction cosines of blue, red, and green vectors are computed.

shown in Fig. 7. One is the case where quadrangle  $S(e_l)$  is similar to a parallelogram, as shown in Fig. 7a. The other is the case where quadrangle  $S(e_l)$  is similar to an isosceles right triangle, as shown in Fig. 7b. Then, in order to remove falsely extracted dot clusters in the background, as shown in Fig. 5 (c), we confirm whether  $S(e_l)$  meets one of the following two conditions: **Condition (i)** (Eq. (2)) which decides whether the quadrangle is similar to a parallelogram, and **Condition (ii)** (Eq. (3)) which decides whether the quadrangle is similar to an isosceles right triangle (Algorithm 1, line 21). A, B, C, and D in Eqs. (2) and (3) are defined in Fig. 8, and  $\theta_{\parallel}^*, \theta_{\perp}^*$  are the thresholds defined beforehand

$$\text{Condition (i)} : \begin{cases} 1 - (\mathbf{n}(\overrightarrow{AB}) \cdot \mathbf{n}(\overrightarrow{CD}))^2 < \theta_{\parallel}^* \\ 1 - (\mathbf{n}(\overrightarrow{AD}) \cdot \mathbf{n}(\overrightarrow{CB}))^2 < \theta_{\parallel}^* \end{cases} \quad (2)$$

$$\text{Condition (ii)} : \begin{cases} (\mathbf{n}(\overrightarrow{AD}) \cdot \mathbf{n}(\overrightarrow{CD}))^2 < \theta_{\perp}^* \\ 1 - (\mathbf{n}(\overrightarrow{AB}) \cdot \mathbf{n}(\overrightarrow{CB}))^2 < \theta_{\parallel}^* \end{cases} \quad (3)$$

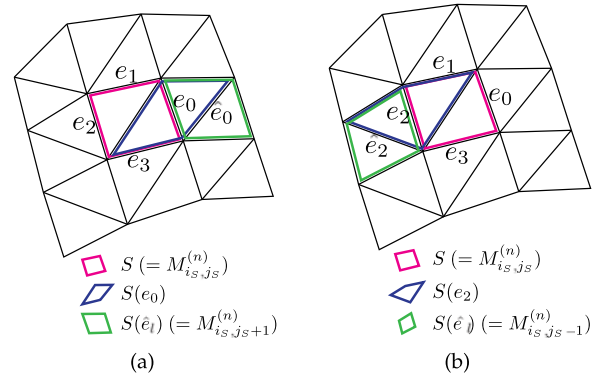


Fig. 7. There are two types of quadrangle connections. (a):  $S(e_0)$  is similar to a parallelogram and meets **Condition (i)**. (b):  $S(e_2)$  is similar to an isosceles right triangle and meets **Condition (ii)**. In both cases,  $S(e_l)$ , which is defined as illustrated in the figure, is similar to a parallelogram.

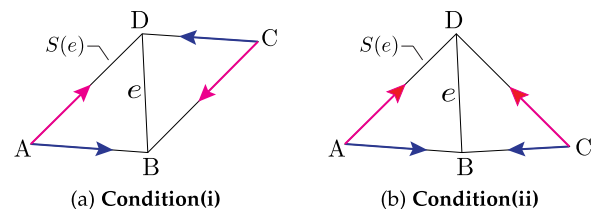


Fig. 8. Calculation of equations in **Conditions (i)** and **(ii)**. The direction cosines of blue and red vectors are computed.

In both cases in Figs. 7a and 7b, the quadrangle  $S(\hat{e}_l)$ , which is defined as illustrated in the figure, is similar to a parallelogram. Then, we also confirm whether  $S(\hat{e}_l)$  meets **Condition (i)** (Algorithm 1, line 23). If  $S(\hat{e}_l)$  acquired in such a way is still not visited, we register it with a suitable index of a 2D array  $M^{(n)}$  (Algorithm 1, lines 25-28). In addition, we add it to queue  $Q$  and let it be visited (Algorithm 1, lines 31-32). This procedure is repeated until  $Q$  becomes empty, and we consequently obtain the connectivity among dot clusters  $M^{(n)}$ .

---

**Algorithm 1.** Obtain the Connectivity Among Dot Clusters

---

```

1: Prepare an empty sequence  $\{S^{(n)}\}$ .
2: for all  $e$  in Delauney triangulation do
3:   Calculate  $L_{sq}(S(e))$ .
4:   if  $L_{sq}(S(e)) > L_{sq}^*$  then
5:     Add  $S(e)$  to  $\{S^{(n)}\}$ .
6:   end if
7: end for
8: Sort  $\{S^{(n)}\}$  such as  $L_{sq}(S^{(0)}) \geq L_{sq}(S^{(1)}) \geq L_{sq}(S^{(2)}) \dots$ .
9:
10: Prepare a sequence of 2D arrays  $\{M^{(n)}\}$ .
11: for  $n = 0$  to end do
12:   if  $S^{(n)}$  is not visited then
13:      $M_{0,0}^{(n)} \leftarrow S^{(n)}$ .
14:     Prepare a queue  $Q$  and add  $S^{(n)}$  to  $Q$ .
15:     Let  $S^{(n)}$  be visited.
16:     while  $Q$  is not empty do
17:       Remove  $S$  from queue  $Q$ .
18:       Let  $(i_S, j_S)$  be the index of  $S$  in  $M^{(n)}$ .
19:       Let  $e_l$  ( $l = 0, 1, 2, 3$ ) be edges of  $S$ .
20:       for  $l = 0$  to  $3$  do
21:         if  $S(e_l)$  satisfies Condition (i) or (ii) then
22:           Let  $\hat{e}_l$  be the edge as shown in Fig. 7.
23:           if  $S(\hat{e}_l)$  is not visited and satisfies Condition (i) then
24:             switch ( $l$ )
25:               case 0:  $M_{i_S, j_S+1}^{(n)} \leftarrow S(\hat{e}_l)$ 
26:               case 1:  $M_{i_S-1, j_S}^{(n)} \leftarrow S(\hat{e}_l)$ 
27:               case 2:  $M_{i_S, j_S-1}^{(n)} \leftarrow S(\hat{e}_l)$ 
28:               case 3:  $M_{i_S+1, j_S}^{(n)} \leftarrow S(\hat{e}_l)$ 
29:             end switch
30:           end if
31:           Add  $S(\hat{e}_l)$  to queue  $Q$ .
32:           Let  $S(\hat{e}_l)$  be visited.
33:         end if
34:       end for
35:     end while
36:   end if
37: end for
38:
39: return  $\{M^{(n)}\}$ 

```

---

Moreover, this BFS is executed several times so that all quadrangles in  $\{S^{(n)}\}$  are used as an initial node for the BFS. Consequently we obtain the sequence of connectivities  $\{M^{(n)}\}$  (Algorithm 1, line 39). Note that the same quadrangle is not registered at multiple connectivities  $M^{(n)}$ . This is because, once a quadrangle is registered, it is regarded as visited and is not registered in subsequent BFSs. Executing

BFSs several times with different initial quadrangles enables us to independently obtain the connectivity without duplication even when dot clusters are separated into several regions, as shown in Fig. 5.

### 3.3.3 Identification of Dot Cluster IDs

Based on the connectivities among dot clusters  $\{M^{(n)}\}$ , each dot cluster ID is identified. We focus on a  $p \times p$  identification window and compute the following key  $K_{win}$  from  $p^2$  dot numbers  $m_{i_k}$  ( $k = 1, 2, \dots, p^2$ ) of dot clusters  $i_k$

$$K_{win} := \sum_{k=1}^{p^2} (m_{i_k} - 1) \cdot 4^{k-1}. \quad (4)$$

Each identification window has a unique key  $K_{win}$ . Therefore, we can prepare a hash table using this key so that the IDs of  $p^2$  dot clusters included within this window are referred to at high speed. This identification is repeated at all possible window locations while allowing overlap of the windows, similarly to the method of Fujimoto et al. [11]. The possible locations can be extracted from the connectivity stored in  $M^{(n)}$  because this array contains information about the adjacency relationship of quadrangles. Additionally, this procedure is executed through all connectivities in  $\{M^{(n)}\}$ .

As a result, the ID of each dot cluster is assigned multiple times. Here, the ID which gets the most number of identifications is finally output, like voting. This voting is able to remove the outliers which are not removed in the previous processing.

## 3.4 Frame-by-Frame Tracking

In this section, we describe frame-by-frame tracking for independently tracking each dot cluster through the image sequence. This method deals with two values at time  $t$ :  $s_i(t)$ , which represents the tracking state of dot cluster  $i$ , and  $\mathbf{x}_i(t)$ , which represents its image coordinate.  $s_i(t)$  takes two values:  $s_i(t) = \text{tracked}$  means that dot cluster  $i$  is observable from a camera and is being correctly tracked by the following method, and  $s_i(t) = \text{lost}$  means that dot cluster  $i$  is lost because of user interaction, the deformation itself, image noise and so on.

Frame-by-frame tracking updates variables  $s_i(t), \mathbf{x}_i(t)$  based on  $s_i(t-1), \mathbf{x}_i(t-1)$  and the current camera image  $I(t)$ . There are four steps in frame-by-frame tracking: updating of tracked dot clusters' positions, tracking recovery of lost dot clusters, detecting false-positive tracking, and interpolation of lost dot clusters. Note that all processing except the interpolation of lost dot clusters can be parallelized for each dot cluster.

### 3.4.1 Updating of Tracked Dot Clusters' Positions

As mentioned in Section 2, under the condition that a high-frame-rate camera is used, we are able to confidently assume that the displacement of a dot cluster between two successive frames is small. Then, we focus on a square region of interest (ROI) with width  $w_{ROI}(t)$  whose center is the position of the tracked dot cluster in the previous frame,  $\mathbf{x}_i(t-1)$ , as shown in Fig. 9. The ROI is adaptively binarized, and the number of connected components is



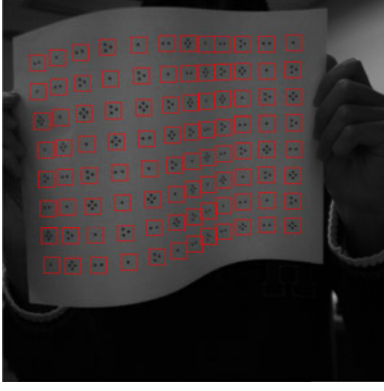


Fig. 9. Updating of tracked dot clusters' positions based on ROIs which are illustrated as red squares.

counted based on contour extraction. If the number of connected components is equal to the dot number  $m_i$ , the position  $\mathbf{x}_i(t)$  is updated to the centroid of the binarized ROI. Otherwise the tracking state is updated with  $s_i(t) = \text{lost}$ .

Here, the ROI width  $w_{\text{ROI}}(t)$  is defined adaptively as follows using the pre-defined coefficient  $\alpha_{\text{ROI}}$  and the mean distance between all tracked dot clusters connected to each other at time  $t$ ,  $\bar{d}(t)$

$$w_{\text{ROI}}(t) := \alpha_{\text{ROI}} \times \bar{d}(t). \quad (5)$$

An adaptive width enables a dot cluster to be in an ROI even when the size of the dot clusters are changed when the distance between the camera and the marker is changed or the marker is stretched.

### 3.4.2 Tracking Recovery of Lost Dot Clusters

This section describes two ways in which the lost dot clusters are re-tracked.

First, we mention a method using the array structure of the DDCM. We estimate the position of a lost dot cluster  $i$ . As shown in Fig. 10, let  $j_l$  ( $l = 0, 1, 2, 3$ ) be the four dot clusters which are connected to dot cluster  $i$  in the right, up, left, and down directions. Also, let  $j'_l$  ( $l = 0, 1, 2, 3$ ) be the four dot clusters that are next to  $j_l$ . Here, let  $N_i^{(2)}(t)$  represent the combination  $(j_l, j'_l)$  such that the tracking states of both  $j_l$  and  $j'_l$  are tracked. The position of dot cluster  $i$ ,  $\hat{\mathbf{x}}_i(t)$ , is estimated as follows. Note that  $|\cdot|$  represents the number of members

$$\hat{\mathbf{x}}_i(t) = \frac{1}{|N_i^{(2)}(t)|} \sum_{(j_l, j'_l) \in N_i^{(2)}(t)} \left( 2\mathbf{x}_{j_l}(t) - \mathbf{x}_{j'_l}(t) \right). \quad (6)$$

Similarly to the previous section, we focus on a square ROI with width  $w_{\text{ROI}}(t)$  whose center is  $\hat{\mathbf{x}}_i(t)$ . If the number of connected components in the binarized ROI is equal to the dot number  $m_i$ , the tracking state is updated with  $s_i(t) = \text{tracked}$  and the centroid of the binarized ROI is substituted for the position of  $i$ ,  $\mathbf{x}_i(t)$ ; otherwise the tracking state lost is maintained.

Next, we describe the method using dot clusters identified in the detection thread. As shown in Fig. 4, the coordinates of dot clusters having tracking state  $s_i(t) = \text{lost}$ , identified in the detection thread, are transferred to the

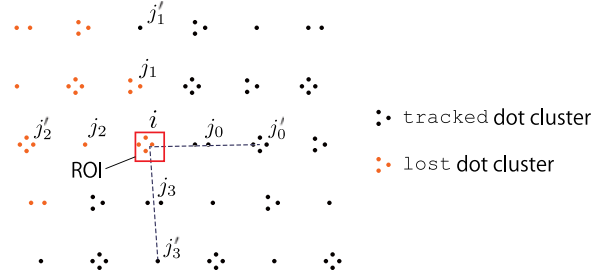


Fig. 10. Tracking recovery based on array structure of DDCM. The position of a lost dot cluster is estimated by the linear extrapolation in Eq. (6). Then, we focus on the ROI whose center is the extrapolated point.

frame-by-frame tracking thread. Then the coordinates are substituted for  $\mathbf{x}_i(t)$ , and the tracking states are updated to  $s_i(t) = \text{tracked}$ . **Although this detection-based tracking recovery needs all dot clusters in the identification window to be observable, it is able to recover ones that are not connected to the tracked dot cluster.**

### 3.4.3 False-Positive Tracking Detection

When the object is strongly deformed so that self-occlusion occurs, false-positive tracking could occur mainly in the region where dot clusters are crowded. False-positive tracking means to track a different dot cluster or an outlier such as the background. In order to detect false-positive tracking, it is effective to check whether each dot cluster keep a suitable distance between itself and the connected dot clusters. Then, we focus on the normalized mean distance between connected dot clusters,  $\hat{d}_i(t)$ , defined as follows. Here,  $N_i^{(1)}(t)$  represents the tracked dot clusters which are connected to dot cluster  $i$  in the right, up, left, and down directions

$$\hat{d}_i(t) := \frac{1}{|N_i^{(1)}(t)|\bar{d}(t)} \sum_{j \in N_i^{(1)}(t)} \|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|. \quad (7)$$

If  $\hat{d}_i(t)$  is greater than the pre-defined threshold  $\hat{d}^*$ , we can assume that false-positive tracking occurs at dot cluster  $i$ , and the tracking state is updated to  $s_i(t) = \text{lost}$ .

Also, there is the case where more than one dot cluster in the original marker are identified at the same dot cluster position detected in the input image. This false-positive cannot be detected by using  $\hat{d}_i(t)$ . Then, if more than one dot cluster are falsely tracked as the same dot cluster (i.e.,  $\mathbf{x}_{i_1} = \mathbf{x}_{i_2} = \dots$ ), the tracking states of these dot clusters  $i_1, i_2, \dots$  are changed to lost.

### 3.4.4 Interpolation of Lost Dot Clusters

In order to generate more natural graphics for projection mapping, we need to interpolate the lost dot clusters. The interpolation is computed by minimizing the following deformation energy [10]:

$$\min_{\mathbf{x}_i(t) \text{ s.t. } s_i(t) = \text{lost}} \frac{1}{2} \sum_{(i,j,k) \in E} \|\mathbf{x}_i(t) - 2\mathbf{x}_j(t) + \mathbf{x}_k(t)\|^2. \quad (8)$$

Here,  $E$  represents three colinearly connected dot clusters (see [37] for detail). This energy requires that all distances between the connected dot clusters should be as equal as

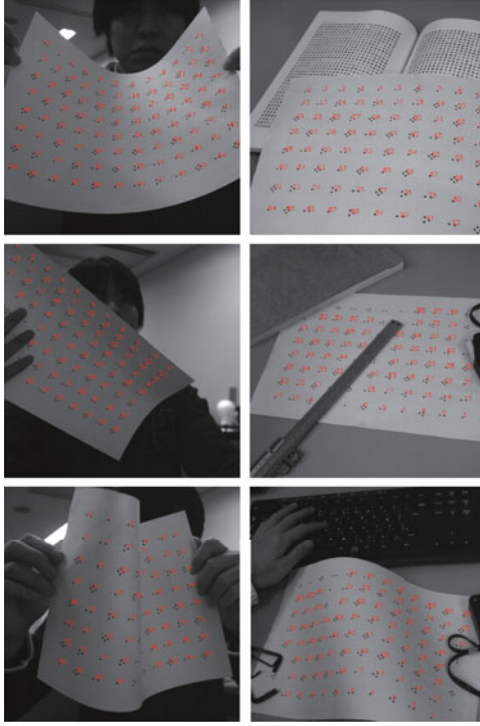


Fig. 11. Detection results for various deformations. The red numbers represent IDs of identified dot clusters. The DDCM could be successfully identified even when it was strongly deformed or there were some objects around and on it.

possible. The minimization can be efficiently solved as a linear problem using Cholesky decomposition.

## 4 EVALUATION

This section shows evaluations of the DDCM and DRDM [44], including the detection accuracy, tracking accuracy based on parallel processing shown in Fig. 4, and the processing time. For the DRDM [44], we used the open source code provided in [1]. Experimental conditions were as follows. The camera used was a Photron IDP-Express R2000 (monochrome, 1,000 fps, 512×512 pixels). The computer was equipped with dual processors, both Intel Xeon E5-2687 (3.10 GHz). The DDCM and DRDM [44] markers were drawn on A4 paper (297×210 mm). The size of the identification window in the DDCM was  $p = 3$ . The number of dot clusters in the DDCM was  $12 \times 8 = 96$ , and the number of dots in the DRDM was 120. Also, the parameters for the detection and frame-by-frame tracking with the DDCM were set as follows:  $A_{\min} = 1.0[\text{px}^2]$ ,  $A_{\max} = 15.0[\text{px}^2]$ ,  $w_{\text{grid}} = 32[\text{px}]$ ,  $n_{\text{kp}} = 15$ ,  $d_{\text{cl}}^* = 13.0$ ,  $L_{\text{sq}}^* = 0.95$ ,  $\theta_{\parallel}^* = 0.05$ ,  $\theta_{\perp}^* = 0.05$ ,  $\alpha_{\text{ROI}} = 0.6$ , and  $\dot{d}^* = 1.4$ .

### 4.1 Evaluation of Detection Accuracy

This section shows the evaluation of the detection accuracy. Fig. 11 shows examples of detection under various deformations. The results show that DDCM can be successfully identified even when it is strongly deformed or there are some objects around it and on it.

We also built the experimental setup shown in Fig. 12 for quantitative evaluation. In this setup, the short sides of a sheet of paper were fixed to a frame. By changing the

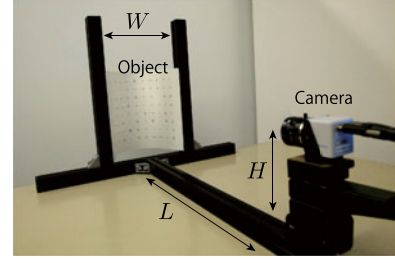


Fig. 12. Experimental setup for evaluation of the detection accuracy. By changing the width  $W$ , we can control the degree of deformation of the paper. When the paper was flat, the distance between the camera and the paper was  $L = 49[\text{cm}]$ , the height of the camera was  $H = 12.5[\text{cm}]$ , and the focal length of the lens was  $f = 8[\text{mm}]$ .

distance between the frames, the degree of deformation was changed. When the paper was flat, the distance between the camera and the paper was  $L = 49[\text{cm}]$ , the height of the camera was  $H = 12.5[\text{cm}]$ , and the focal length of the lens was  $f = 8[\text{mm}]$ . Fig. 13 shows the actual deformations of the paper which were captured from above the paper. The width  $W$  for each deformation is also described in the figure.

In Fig. 14, the detection rates for the DDCM and DRDM [44] are compared. For the DDCM, the detection rate means the number of correctly identified dot clusters relative to the total number of dot clusters, 96. For the DRDM, the detection rate means the number of correctly identified dots relative to the total number of dots, 120. In the DDCM, the detection was successfully achieved even when the paper deformation was large. This is because the adjacency relationship of dot clusters based on Delaunay triangulation is correctly recognized when the deformation occurs. In the case of the DRDM, detection successfully worked only for the dots near the center of the marker, and successful cases were limited to small deformation. With the DRDM, the affine invariants are stored only for the flat target in the database in advance, and it is assumed that the target is set

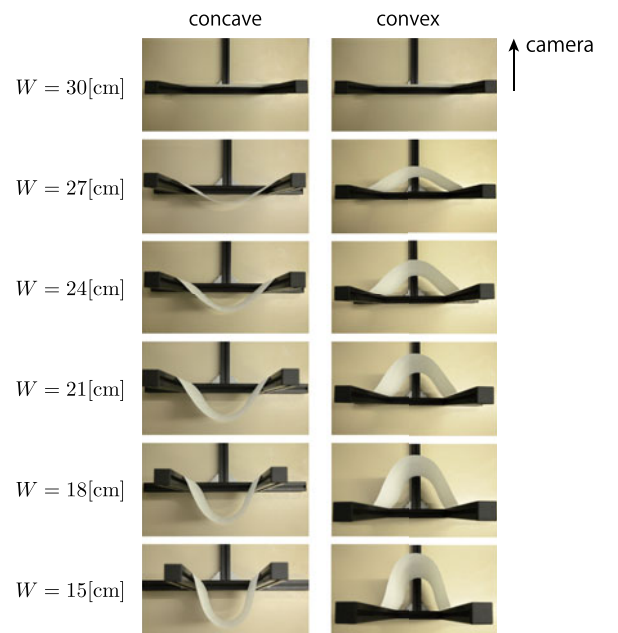


Fig. 13. The paper deformations for evaluating the detection accuracy. The images were captured from above the paper. The camera for detection was set pointing in the direction shown in the figure.



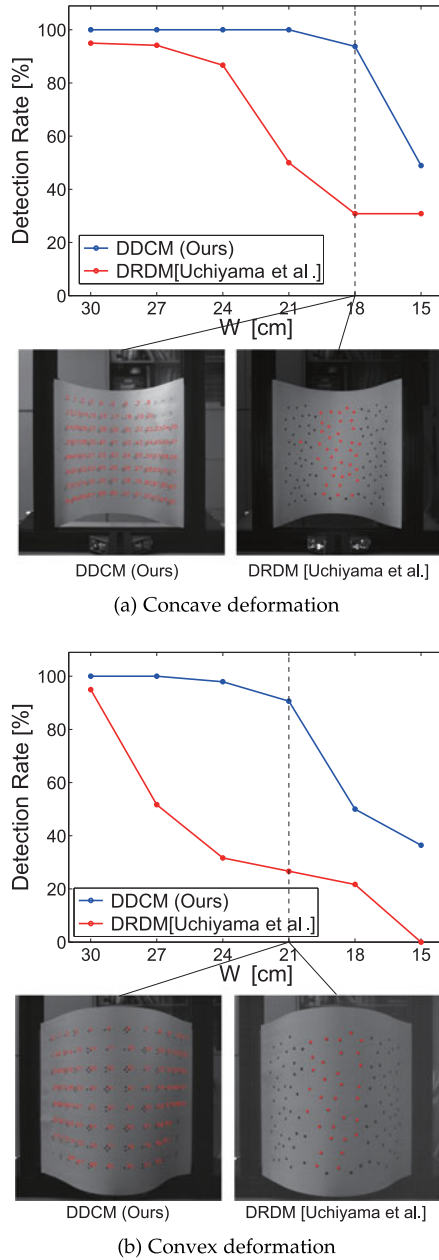


Fig. 14. Comparison of the detection rates between the DDCM and DRDM [44] in the concave and convex cases. The red numbers in the DDCM results represent IDs of identified dot clusters. The red dots in the DRDM results show identified dots.

flat in the initial step, as described in Section 2. Therefore, it was difficult to achieve matching of feature values calculated from the deformed paper, as shown in these results.

## 4.2 Evaluation of Tracking

We evaluated the whole algorithm in which the detection and the frame-by-frame tracking were executed in parallel, as shown in Fig. 4. In the evaluation, we used dynamic scenes with high-speed motion, including (i) S-deformation, (ii) U-deformation, (iii) waving deformation, and (iv) occlusion by a hand. In (i) S-deformation, the paper was deformed from the flat shape to a state in which it formed a letter “S” and returned to the flat shape again. This motion was repeated two times per second. In (ii) U-deformation, the paper was deformed to a state in which it formed a letter “U”, which was

convex to the camera, and returned to the flat shape. This motion was repeated one time per second. These motions are examples of motions with self-occlusion. In (iii) waving deformation, a corner of the paper was held by a hand, and the other corner was moved backward and forward. This motion was repeated two times per second. This motion is an example of a motion involving large deformation. In (iv) occlusion by a hand, a hand was moved up and down to occlude half of the paper. This motion was repeated one time per second. This is an example of a motion with external occlusion.

Input images for the DDCM were video images captured at 1,000 fps. This frame rate was decided because the calculation time per frame for the frame-by-frame tracking in the DDCM was within 1 ms, as shown in Fig. 19 and as described in the next section. Also, since the detection required 2 ms per frame, as shown in Fig. 19, the data transfer to the frame-by-frame tracking thread from the detection thread was updated every two frames in the successive images captured at 1,000 fps.

Input images for the DRDM [44] were two types of video images captured at 50 fps and 1,000 fps, respectively. The reason why we used 50 fps images was because the DRDM [44] required about 20 ms at the best performance, as shown in Fig. 19 in the same calculation environment as that used with the DDCM. Also, in order to check the performance in the case where the DRDM [44] calculation time is improved in the future, we employed 1,000 fps images. The 50 fps images were prepared by selecting every 20th image from the images acquired at 1,000 fps.

For tracking evaluation, various methods are used in related work. In this paper, we used the F-measure. The F-measure is obtained as a harmonic average of Recall and Precision, as follows:  $F\text{-measure} := (2 \text{ Recall} \times \text{Precision}) / (\text{Recall} + \text{Precision})$ .

Recall is defined as  $\text{Recall} := TP / (TP + FN)$ . In this case, this value is the ratio of the number of correctly recognized dot clusters to the total number of dot clusters observed in the captured image. Also, Precision is defined as  $\text{Precision} := TP / (TP + FP)$ . In this case, this value is the ratio of the number of the correctly recognized dot clusters to the number of dot clusters whose state  $s_i(t) = \text{tracked}$ . The Recall and Precision in the DRDM evaluation were defined in a similar way. Since Recall and Precision have a tradeoff relationship generally, the F-measure is considered to be a good value for comprehensive evaluation.

The results are shown in Figs. 15, 16, 17, 18. In these figures, the red dots show the dot clusters whose states were tracked. As shown in these figures, the DDCM achieved good tracking performance in dynamic scenes with external occlusions and self-occlusions. Also, compared with the results obtained when applying only detection, the whole algorithm using both detection and frame-by-frame tracking showed better accuracy. In our method, although the detection requires all dot clusters belonging to the identification window to be observed, the frame-by-frame tracking can track each dot cluster independently. This mutual complementation allowed robust and high-speed tracking of the deformation of a non-rigid object.

On the other hand, the DRDM performance using 50 fps images was not stable, except for case (iv). The displacement of dots between two successive frames was large, causing

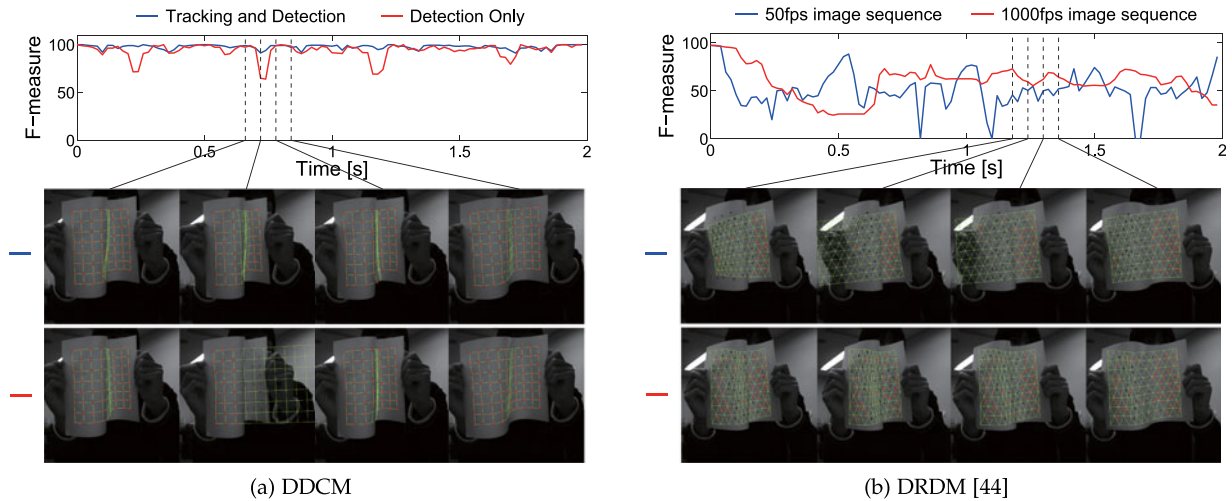


Fig. 15. (i) Tracking Performance comparison of the DDCM and DRDM [44] in the case of S-deformation. The paper is deformed to a shape like a letter “S” two times per second.

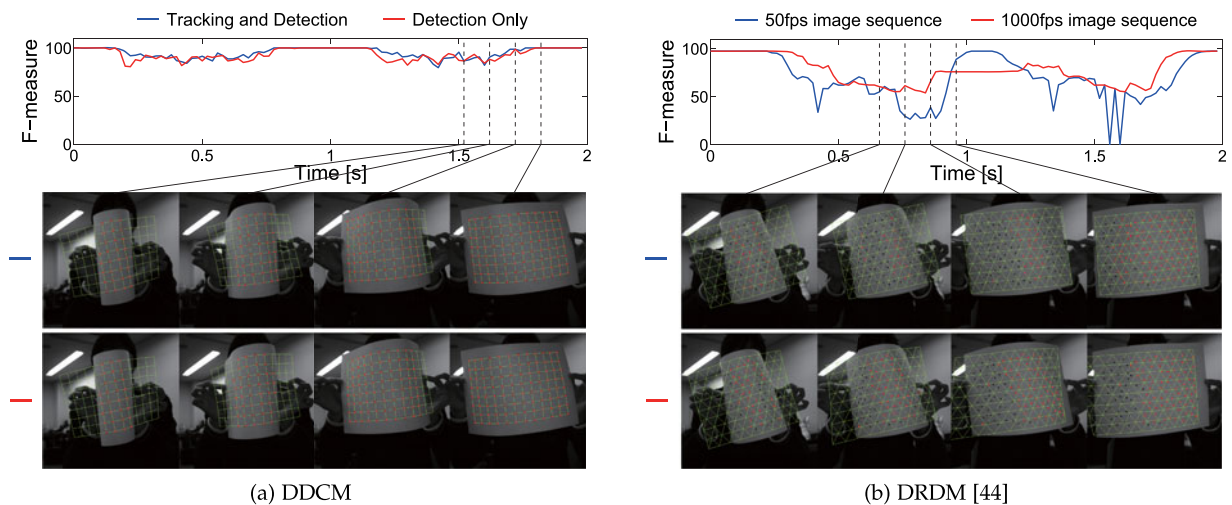


Fig. 16. (ii) Tracking Performance comparison of the DDCM and DRDM [44] in the case of U-deformation. The paper is deformed to a shape like a letter “U”, in which the shape was convex towards the camera, one time per second.

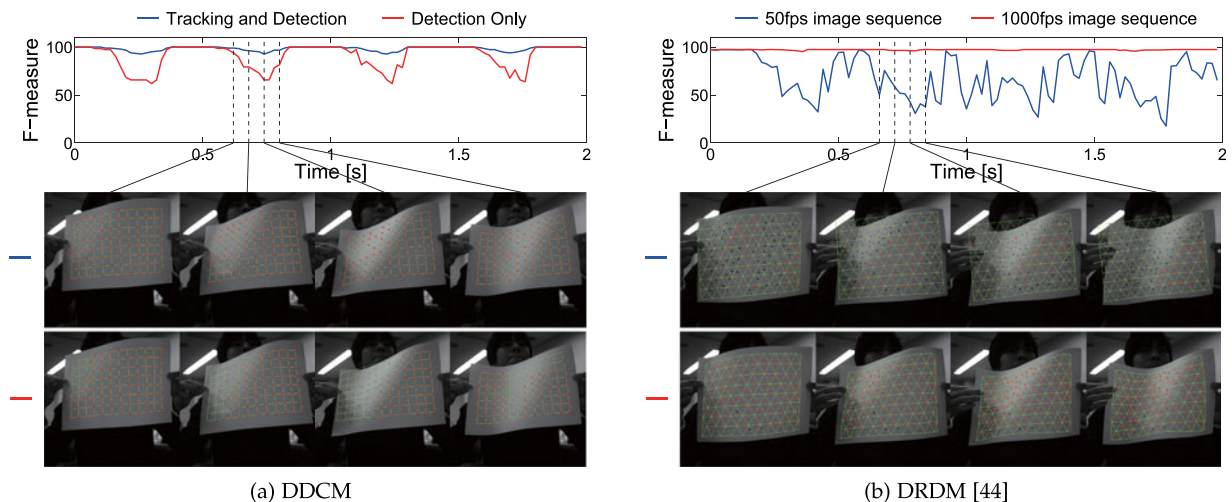


Fig. 17. (iii) Tracking performance comparison of the DDCM and DRDM [44] in the case of waving deformation. One of the short edges of the paper was held by the hand, and the other edge was moved backward and forward two times per second.

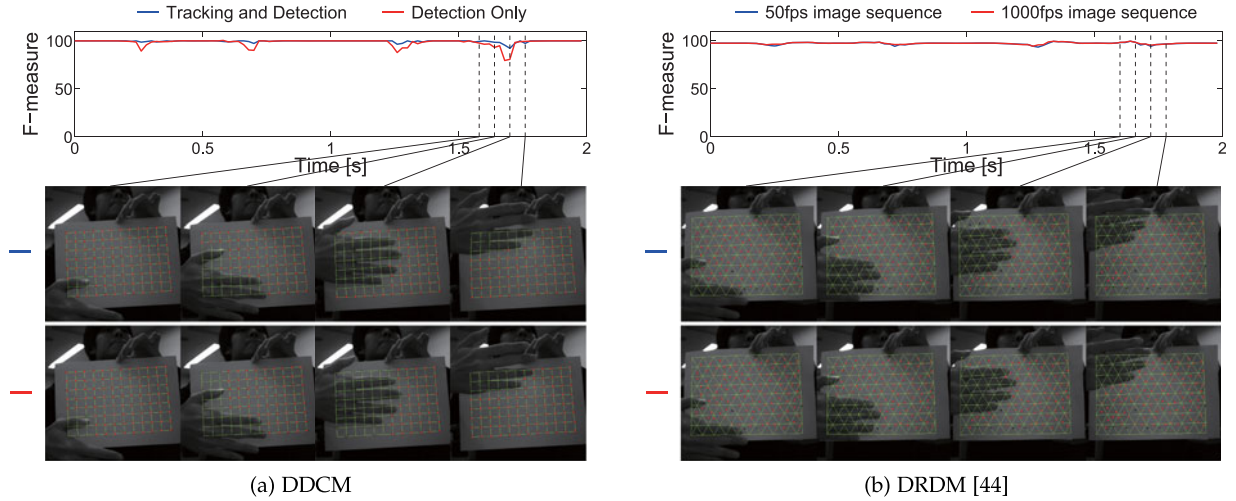


Fig. 18. (iv) Tracking performance comparison of the DDCM and DRDM [44] in the case of occlusion by a hand. The hand moved up and down and occluded half of the paper one time per second.

failure to match the affine invariants. Compared with the results obtained at 50 fps, in 1,000 fps results, improved tracking accuracy can be found, especially in case (iii). This is mainly due to the reduced displacement between two frames. However, even when we used the 1,000 fps images for the DRDM, cases (i) and (ii) including self-occlusion resulted in failure. In the DRDM, false-positives tended to occur in situations where the dots were dense and close together. DRDM cannot be used to detect such a situation, and frame-by-frame tracking continues without solving the false-positive problem. This is considered to cause a performance degradation.

### 4.3 Evaluation of Calculation Time

We measured the calculation time for five scenes including a scene in which the object on which the marker was displayed was held flat in the front of the camera and the four cases (i)-(iv) described in the previous section. The frame-by-frame tracking with the DDCM was parallelized using OpenMP. Two threads were used for the parallel operation. Also the calculation time was obtained as the average value of 5,000 cycles. The results are shown in Fig. 19. The DRDM required about 20 ms when the object was held flat. It required more

time in the cases where the object was moving and deforming. On the other hand, with the DDCM, the detection and the frame-by-frame tracking required less than 2 and 1 ms, respectively, for all scenes. This performance is considered to be high enough for dynamic projection mapping where the human eye cannot perceive the geometric misalignment between the real-world object and the projected image.

## 5 DYNAMIC PROJECTION MAPPING ONTO DEFORMING NON-RIGID SURFACE

### 5.1 System Configuration

Fig. 20 shows the system configuration for projection mapping. The camera and the CPU were the same as those used in the evaluation setup described above. Also, we used a high-speed projector developed by Watanabe et al. [50]. The developed high-speed projector projects 8-bit images at up to 1,000 fps with a minimum delay of 3 ms. The camera and the projector were arranged in such a way that both light axes were aligned through a hot mirror (Edmund Optics TS Hot Mirror 45 degree). Also, the homography transformation from the camera image plane to the projector image plane was obtained in calibration performed in advance. The

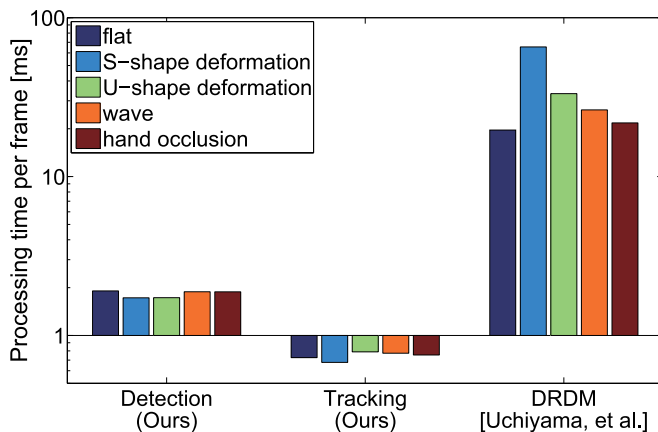


Fig. 19. Comparison of the calculation time of the DDCM and DRDM [44]. Our DDCM method achieved less than 2 ms/frame detection and less than 0.8 ms/frame frame-by-frame tracking in all five cases.

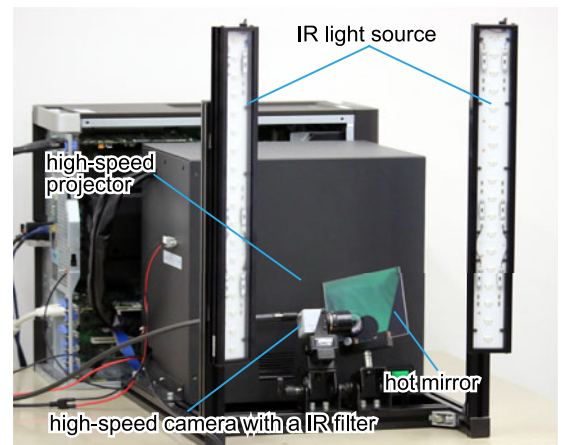


Fig. 20. System configuration for projection mapping. The camera and the projector were arranged in such a way that both light axes were aligned through a hot mirror.



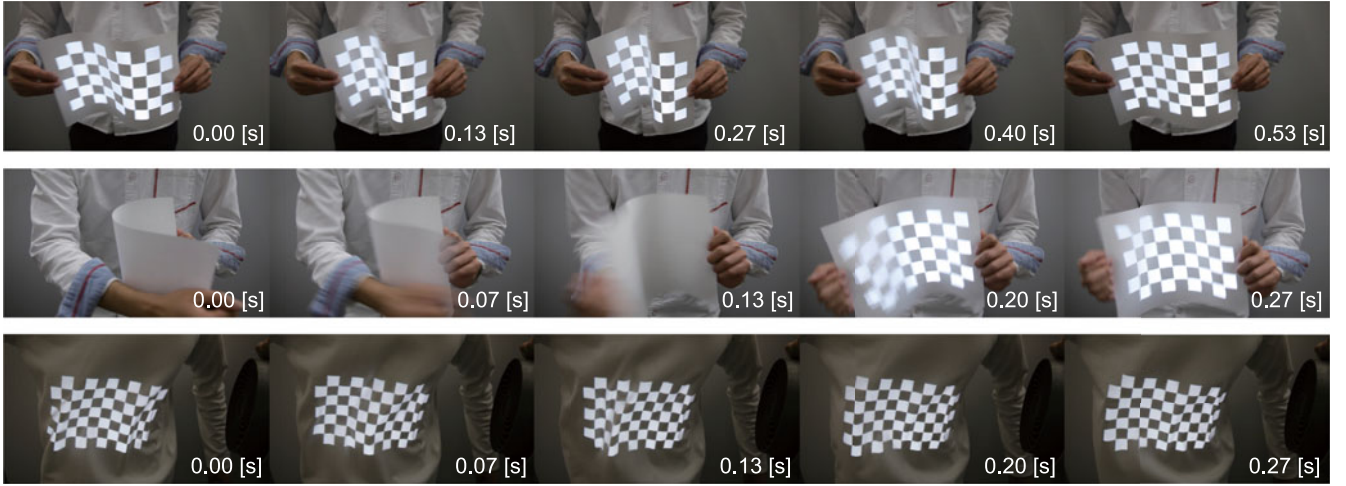


Fig. 21. The sheet of paper and T-shirt on which a chessboard pattern is projected. In the top five figures, the sheet of paper was rapidly deformed like a letter “S”. In middle five figures, detection of the DDCM and projection were completed immediately when the DDCM appeared in the field of view of the camera. In the bottom five figures, a T-shirt was made to flutter by wind, and exhibited more complicated deformation than a sheet of paper.

graphic board in the computer was an NVIDIA Quadro K600. The projection images were generated on its GPU and were sent to the high-speed projector via the main memory.

As the projected target, we used sheet of A4 paper and a T-shirt. On these target surfaces, the DDCM was drawn by using IR ink. As the IR ink, we used LDP LLC IR1PenSm. This ink absorbs only near-infrared light with wavelengths from 793 to 820 nm and is almost invisible to the human eye. A filter for blocking visible light (LDP LLC XNite71537, cut wavelength 715 nm) was set in front of the camera. Also, we used an infrared light source (CCS HL2L2-450X45IR-DF-W, peak wavelength 860 nm) so that the camera observes the IR ink as black.

The projected image was generated as follows. Preliminarily, the original projected image was divided into a grid. The grid-point configuration was the same as the dot-cluster configuration in the DDCM. Therefore, their correspondences were given in advance. For example, in this demonstration, the grid consisted of  $12 \times 8$  grid points. In the online processing, the obtained dot-cluster coordinates in the camera image plane were converted to coordinates in the projector image plane using the homography transformation described above. Using the converted coordinates, we constructed the transformed grid. The original image was transformed for projection mapping so as to fit the grid. We implemented this transformation using the texture mapping technique, assuming that each rectangle in the grid was planar.

## 5.2 Demonstrations

In order to confirm and visualize that the projected images rapidly and correctly tracked the moving and deforming target objects, a chessboard pattern was projected onto the sheet of paper and the T-shirt. The projection mapping results described here were captured by a 30 fps camera. The top row in Fig. 21 shows the situation where the sheet of paper was rapidly deformed, like the letter “S”. This result shows that the correctly deformed chessboard pattern was projected as if the pattern was printed on the paper. The second row in Fig. 21 shows that the DDCM was detected immediately when it appeared in the viewing field of the camera. The third row in Fig. 21 shows the situation

where a chessboard pattern was projected on the T-shirt, which was fluttering in wind blown from a fan. This result shows that the DDCM can be applied to more complicated deformation than that of a sheet of paper.

Fig. 22 shows the situation where the sheet of paper was rapidly moved horizontally. This figure reveals that high-throughput projection was achieved because many chessboard patterns overlap in the figure. Actually the projection throughput was about 350 fps in our system configuration. The total latency of our system, which is difficult to measure, can be estimated to be about 6.6 ms from individual processing times: frame-by-frame tracking takes about 0.8 ms, generation of projection images and their transfer to main memory takes about 2.8 ms, and the delay of the projector itself is about 3 ms [50]. This latency is close to the value of 6.04 ms which is necessary for projection mapping without any perceivable misalignment [32].

Finally, we discuss possible applications of our projection mapping. The left column of Fig. 23 shows a situation where a movie is projected on a freely moving sheet of paper. As in this figure, we can use paper-like material as a deformable display with low latency. Also, we can register a large number of input operations in association with their corresponding deformations, and this can be used as a user interface with high degree-of-freedom of inputs. Application involving high-speed interactions, such as games, can also be realized

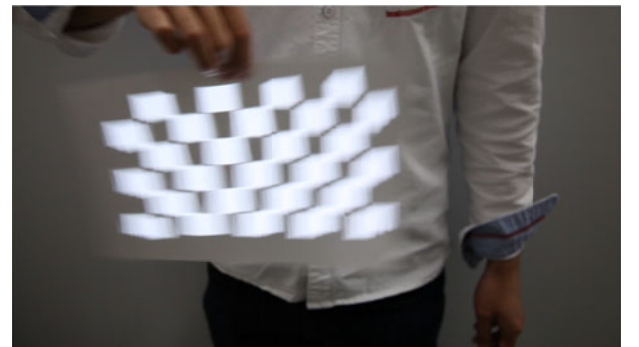


Fig. 22. The paper is rapidly moving horizontally. Many chessboard pattern overlap in this figure, and high-throughput projection was achieved.



Fig. 23. Application examples of our projection mapping. The left column shows that a movie was projected onto a freely moving sheet of paper. The right column shows that animated graphics and a logo were projected onto a plain T-shirt, like a try-on system.

because our system has low latency. The right column of Fig. 23 shows a situation where animated graphics and a logo are projected on a plain T-shirt. A try-on system in which customers can put on clothes with a variety of patterns and a prototyping support system for clothing designers can be realized with our projection mapping.

## 6 DISCUSSION

The evaluation showed that our method could satisfy the requirements for both high speed and robustness. Both performance measures are considered to exceed those of the conventional approaches. On the other hand, there are still some limitations. The first limitation is related to recognition of the number of dots in a cluster. If finer shading or deformation occur, dots are connected in the observed image. Also, noise may form false dot clusters in the image. Consequently, if the numbers of dots in all clusters in the identification window are incorrectly recognized, the tracking could fail. Although the evaluation showed that these situations were expected to be rare, a better implementation must be explored. The second limitation is related to occlusion issues in the projection mapping. The present implementation projects images onto the occluded regions because we interpolate the lost dot clusters, as described in Section 3.4.4. However, in the case of occlusion by a hand, it is desirable not to project onto the hand region. Although we can modify the system so as not to project onto such regions by assuming that the lost dot clusters are caused by occlusion, it is difficult to segment regions exactly along the occluded object because the projected image is controlled by the position of dot clusters. Also, it is difficult to distinguish such occluded regions because the dot clusters can be lost due to the incorrect recognition described above.

## 7 CONCLUSION

In this paper, in order to realize dynamic projection mapping onto a deforming non-rigid surface with a speed sufficiently high that a human does not perceive any misalignment

between the target object and the projected images, we have proposed the Deformable Dot Cluster Marker, a novel fiducial marker for high-speed tracking of non-rigid surfaces using a high-frame-rate camera. The DDCM consists of four types of dot clusters arranged in an array structure. The DDCM has three performance advantages. First, it can be detected even when it is strongly deformed. Second, it realizes robust tracking even in the presence of external occlusions caused by user interactions and self-occlusions caused by the object's deformation itself. Third, it enables a remarkably high computational speed: less than 2 ms per frame for detection and less than 0.8 ms per frame for tracking.

Using this DDCM and a high-speed projector [50], we realized dynamic projection mapping onto a sheet of paper and a T-shirt. The results showed robust and consistent display of projected images onto target objects with sufficient speed that the projected images appeared to be printed on the objects.

The limitation of our projection mapping is that the projection area is limited to the angles of view of the camera and projector. By combining our technique with a high-speed gaze controller [33], we could realize projection mapping onto a deforming object which moves rapidly in a wide area.

## REFERENCES

- [1] (2011). Deformable Random Dot Markers (UCHIYAMARKERS 2.0). [Online]. Available: <http://limu.ait.kyushu-u.ac.jp/~uchiya/me/code/UCHIYAMARKERS2/index.html>
- [2] H. Asayama, D. Iwai, and K. Sato, "Diminishable visual markers on fabricated projection object for dynamic spatial augmented reality," in *Proc. SIGGRAPH Asia Emerg. Technologies*, 2015, pp. 7:1–7:2.
- [3] A. Bermano, P. Br schweiler, A. Grundh fer, D. Iwai, B. Bickel, and M. Gross, "Augmenting physical avatars using projector-based illumination," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 189:1–189:10, Nov. 2013.
- [4] O. Bimber, D. Iwai, G. Wetzstein, and A. Grundhofer, "The visual computing of projector-camera systems," *Comput. Graph. Forum*, vol. 27, no. 8, pp. 2219–2245, 2008.
- [5] O. Bimber and R. Raskar, *Spatial Augmented Reality: Merging Real and Virtual Worlds*. Boca Raton, FL, USA: CRC Press, 2005.
- [6] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 6, pp. 567–585, Jun. 1989.
- [7] J. Chen, T. Yamamoto, T. Aoyama, T. Takaki, and I. Ishii, "Simultaneous projection mapping using high-frame-rate depth vision," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2014, pp. 4506–4511.
- [8] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Berlin, Germany: Springer, 2008.
- [9] C. Elbrechter, R. Haschke, and H. Ritter, "Bi-manual robotic paper manipulation based on real-time marker tracking and physical modelling," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 1427–1432.
- [10] P. Fua and Y. G. Leclerc, "Object-centered surface reconstruction: Combining multi-image stereo and shading," *Int. J. Comput. Vision*, vol. 16, no. 1, pp. 35–56, 1995.
- [11] Y. Fujimoto "Geometrically-correct projection-based texture mapping onto a deformable object," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 4, pp. 540–549, Apr. 2014.
- [12] V. Gay-Bellile, A. Bartoli, and P. Sayd, "Deformable surface augmentation in spite of self-occlusions," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, Nov. 2007, pp. 235–238.
- [13] V. Gay-Bellile, A. Bartoli, and P. Sayd, "Direct estimation of non-rigid registrations with image-based self-occlusion reasoning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 87–104, Jan. 2010.
- [14] I. Guskov, "Efficient tracking of regular patterns on non-rigid geometry," in *Proc. 16th Int. Conf. Pattern Recognition*, 2002, vol. 2, pp. 1057–1060.



- [15] I. Guskov, S. Klivanov, and B. Bryant, "Trackable surfaces," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2003, pp. 251–257.
- [16] D. Holman, R. Vertegaal, M. Altosaar, N. Troje, and D. Johns, "Paper windows: Interaction techniques for digital paper," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2005, pp. 591–599.
- [17] Z. Horváth, A. Herout, I. Szentandrás, and M. Zachariás, "Design and detection of local geometric features for deformable marker fields," in *Proc. 29th Spring Conf. Comput. Graph.*, 2013, pp. 073:73–073:80.
- [18] I. Ishii, Y. Nakabo, and M. Ishikawa, "Target tracking algorithm for 1 ms visual feedback system using massively parallel processing," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1996, vol. 3, pp. 2309–2314.
- [19] M. Khalilbeigi, R. Lissermann, W. Kleine, and J. Steimle, "FoldMe: Interacting with double-sided foldable displays," in *Proc. 6th Int. Conf. Tangible Embedded Embodied Interaction*, 2012, pp. 33–40.
- [20] K. Kim, V. Lepetit, and W. Woo, "Keyframe-based modeling and tracking of multiple 3D objects," in *Proc. 9th IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2010, pp. 193–198.
- [21] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, Nov. 2007, pp. 225–234.
- [22] B. Kocev, F. Ritter, and L. Linsen, "Projector-based surgeon-computer interaction on deformable surfaces," *Int. J. Comput. Assisted Radiology Surgery*, vol. 9, no. 2, pp. 301–312, 2013.
- [23] D. Kondo and R. Kijima, "Poster: Free form projection display: Virtual image located inside real object," in *Proc. IEEE Symp. 3D User Interfaces*, Mar. 2008, pp. 159–160.
- [24] V. Lepetit, P. Laguerre, and P. Fua, "Randomized trees for real-time keypoint recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognition*, Jun. 2005, vol. 2, pp. 775–781.
- [25] W.-C. Lin and Y. Liu, "A lattice-based MRF model for dynamic near-regular texture tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 5, pp. 777–792, May 2007.
- [26] P. Lincoln, "From motion to photons in 80 microseconds: Towards minimal latency for virtual and augmented reality," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 4, pp. 1367–1376, Apr. 2016.
- [27] M. Marner and B. Thomas, "Augmented foam sculpting for capturing 3D models," in *Proc. IEEE Symp. 3D User Interfaces*, Mar. 2010, pp. 63–70.
- [28] M. Mine, D. Rose, B. Yang, J. van Baar, and A. Grundhofer, "Projection-based augmented reality in Disney theme parks," *Computer*, vol. 45, no. 7, pp. 32–40, Jul. 2012.
- [29] T. Nakai, K. Kise, and M. Iwamura, "Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval," in *Proc. 7th Int. Conf. Document Anal. Syst.*, 2006, pp. 541–552.
- [30] Y. Nakazato, M. Kanbara, and N. Yokoya, "Localization of wearable users using invisible retro-reflective markers and an IR camera," in *Proc. SPIE Stereoscopic Displays Virtual Reality Systems XII*, vol. 5664, Jun. 2005, [Online]. Available: <http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=857648>
- [31] G. Narita, Y. Watanabe, and M. Ishikawa, "Dynamic projection mapping onto a deformable object with occlusion based on high-speed tracking of dot marker array," in *Proc. 21st ACM Symp. Virtual Reality Softw. Technol.*, 2015, pp. 149–152.
- [32] A. Ng, J. Lepinski, D. Wigdor, S. Sanders, and P. Dietz, "Designing for low-latency direct-touch input," in *Proc. 25th Annu. ACM Symp. User Interface Softw. Technol.*, 2012, pp. 453–464.
- [33] K. Okumura, H. Oku, and M. Ishikawa, "High-speed gaze controller for millisecond-order pan/tilt camera," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 6186–6191.
- [34] K. Okumura, H. Oku, and M. Ishikawa, "Lumipen: Projection-based mixed reality for dynamic objects," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2012, pp. 699–704.
- [35] H. Park and J.-I. Park, "Invisible marker based augmented reality system," in *Proc. SPIE Visual Commun. Image Process.*, Jul. 2006, Art. no. 59601.
- [36] Y. Park, V. Lepetit, and W. Woo, "Multiple 3D object tracking for augmented reality," in *Proc. 7th IEEE/ACM Int. Symp. Mixed Augmented Reality*, 2008, pp. 117–120.
- [37] J. Pilet, V. Lepetit, and P. Fua, "Fast non-rigid surface detection, registration and realistic augmentation," *Int. J. Comput. Vision*, vol. 76, no. 2, pp. 109–122, 2007.
- [38] P. Punpongsanon, D. Iwai, and K. Sato, "Projection-based visualization of tangential deformation of nonrigid surface by deformation estimation using infrared texture," *Virtual Reality*, vol. 19, no. 1, pp. 45–56, 2014.
- [39] R. Raskar, G. Welch, K.-L. Low, and D. Bandyopadhyay, "Shader lamps: Animating real objects with image-based illumination," in *Rendering Techniques (Eurographics Series)*, S. Gortler and K. Myszkowski, Eds. Berlin, Germany: Springer, 2001, pp. 89–102.
- [40] C. Resch, P. Keitler, and G. Klinker, "Sticky projections—A model-based approach to interactive shader lamps tracking," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 3, pp. 1291–1301, Mar. 2015.
- [41] C. Siegl, et al., "Real-time pixel luminance optimization for dynamic multi-projection mapping," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 237:1–237:11, Oct. 2015.
- [42] J. Steimle, A. Joridt, and P. Maes, "Flexpad: Highly flexible bending interactions for projected handheld displays," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2013, pp. 237–246.
- [43] T. Sueishi, H. Oku, and M. Ishikawa, "Robust high-speed tracking against illumination changes for dynamic projection mapping," in *Proc. IEEE Virtual Reality*, Mar. 2015, pp. 97–104.
- [44] H. Uchiyama and E. Marchand, "Deformable random dot markers," in *Proc. 10th IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2011, pp. 237–238.
- [45] H. Uchiyama and E. Marchand, "Object detection and pose tracking for augmented reality: Recent approaches," in *Proc. 18th Korea-Japan Joint Workshop Frontiers Comput. Vision*, Feb. 2012, pp. 1–8.
- [46] H. Uchiyama and H. Saito, "Random dot markers," in *Proc. IEEE Virtual Reality Conf.*, Mar. 2011, pp. 35–38.
- [47] J. C. Verlinden, A. de Smit, A. W. J. Peeters, and M. H. van Gelderen, "Development of a flexible augmented prototyping system," *J. WSCG*, vol. 11, no. 1–3, 2003. [Online]. Available: [http://wscg.zcu.cz/wscg2003/wscg\\_program.htm#full](http://wscg.zcu.cz/wscg2003/wscg_program.htm#full)
- [48] Y. Watanabe, T. Komuro, and M. Ishikawa, "A high-speed vision system for moment-based analysis of numerous objects," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2007, vol. 5, pp. V-177–V-180.
- [49] Y. Watanabe, T. Komuro, S. Kagami, and M. Ishikawa, "Multi-target tracking using a vision chip and its applications to real-time visual measurement," *J. Robot. Mechatronics*, vol. 17, no. 2, pp. 121–129, 2005.
- [50] Y. Watanabe, G. Narita, S. Tatsuno, T. Yuasa, K. Sumino, and M. Ishikawa, "High-speed 8-bit image projector at 1,000 fps with 3 ms delay," in *Proc. Int. Display Workshops*, 2015, pp. 1064–1065.
- [51] F. Zheng, et al., "Minimizing latency for augmented reality displays: Frames considered harmful," in *Proc. 2014 IEEE Int. Symp. Mixed Augmented Reality*, Sep. 2014, pp. 195–200.



**Gaku Narita** received the BE and ME degrees from the University of Tokyo, Japan, in 2014 and 2016, respectively. His research interests include computer vision, virtual reality, and projection mapping.



**Yoshihiro Watanabe** received the BE, ME, and PhD degrees in information science and technology from the University of Tokyo, Japan, in 2002, 2004, and 2007, respectively. He is currently a lecturer in the Graduate School of Information Science and Technology, University of Tokyo. His research interests include high-speed vision, computer vision, virtual reality, computer-human interaction, and digital archiving.



**Masatoshi Ishikawa** received the BE, ME, and DrEng degrees in mathematical engineering and information physics from the University of Tokyo, Japan, in 1977, 1979, and 1988, respectively. He is currently a professor in the Graduate School of Information Science and Technology, University of Tokyo. His current research interests include robotics, sensor fusion, high speed vision, visual feedback, dynamic image control, and meta perception.

► For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).