2009 Conference for Visual Media Production

A PORTABLE PROJECTOR EXTENDED FOR OBJECT-CENTERED REAL-TIME INTERACTIONS

Thitirat Siriborvornratanakul, Masanori Sugimoto

Interaction Technology Laboratory, Department of Electrical Engineering and Information Systems, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan 113-8656
{thitirat, sugi}@itl.t.u-tokyo.ac.jp

Abstract

With the recent growth in projection technologies, interactive applications using portable projectors have received attention from research communities. However, almost all applications are limited to controller-centered interactions and few pay attention to the existence of other objects found in an environment. In this paper, we propose an object-centered interactive system using a single self-contained device. Physical objects are integrated into the system so virtual objects projected by a projector can respond to them in a realistic manner. The device consists of a portable projector, a FireWire camera, a motion sensor and a VGA splitter. A computational framework for the system is presented in the paper. We focus on the three fundamental requirements of the system and their solution. First, a projector and a camera were geometrically calibrated in real time using a motion sensor. Physical objects were then identified and tracked using particle filters. Finally, virtual information was projected using a nonintrusive projection technique. Two basic programs were implemented as proofs of concept, and experiments were conducted to evaluate the accuracy and speed of the proposed system.

Keywords: portable projector, object-centered interaction, real-time projector-camera geometric calibration, nonintrusive projection, multiple-target tracking using particle filters.

1 Introduction

Using a projector in human–computer interaction allows intuitive interaction by directly superimposing virtual information onto a physical surface. In the past decade, a number of interactive applications have been developed that benefit from projection technologies. With the recent growth in portable devices, several new interactive systems have been based on a portable projector. However, in a portable system, devices and architectures are quite limited. Therefore, achieving robust interactions is very challenging.

Many research approaches have proposed an interactive application using a portable projector. However, most of these do not properly integrate objects found inside an environment. Accordingly, interactions are only generated from the point where a controller object (e.g., a user's head, hand, or fingers, or a robot equipped with a marker) is found to the point hinted at by that controller object. Interactions relating to other existing objects are rarely implemented. In this paper, we call this kind of interaction "controller-centered interaction".

The concept of controller-centered interaction helps eliminate several unpredictable factors in a portable system. Nevertheless, it is desirable in practice that a portable system should perform appropriately even in an unknown environment containing unknown objects. To achieve this, we add more information about an environment to our portable system. The concept of controller-centered interaction is extended to object-centered interaction. The word "object" in this context refers to a common object that is not prepared, resides in an environment and is neither a controller object nor a surface.

To be more specific, the concept of object-centered interaction helps merge physical objects found in an actual environment into virtual worlds created for projection, so that interactions between them can be generated in a realistic manner. This concept solves the problem of inappropriate projections onto a surface occupied by objects, especially in a portable system. When objects inside an environment are recognized, it is now up to the system to determine how they should be treated.

In our previous work [26], a mobile device was used for minor object-centered interactions. However, four color markers are required on a surface and the system cannot actually project interactions onto the surface because of the intrusive-projection problem. The aim of this paper is to explore a novel aspect of a single self-contained projection device that supports object-centered interactions. We are also interested in differentiating objects found in an environment and interacting with them uniquely.

Our proposed system is based on a projector-camera paired system, also known as a pro-cam system. There are three principal aspects to our proposed system. The first is a real-time geometric calibration to transform camera coordinates to projector coordinates and vice versa. The second is multiple-target tracking for identifying and tracking objects that reside in an environment. Finally, nonintrusive projection guarantees that projected interactions will be visible to humans but invisible to the camera. This is necessary to avoid interference (as seen by the camera) that may lead to an incorrect analysis of the environment.

2 Related work

As discussed, our achievement is related to three types of problem. In the following section, we explain recent advances in interactive projector systems and then discuss related research in the three areas.

Interactive projector systems: Cotting and Gross [7] introduced an environment-aware display system that automatically avoids projections onto nonsurface objects. Their system performs real-time object detection, but it is limited to fixed projectors and cannot differentiate between detected objects. Thus, the same interaction is provided to all objects. In Cao et al. [5,6], interactive mobile projector systems were developed. Their systems project static annotations onto registered objects. However, it requires a user to register each object manually and includes a camera mounted separately in a workspace. In CoGAME [13], images projected from a handheld projector act as a robot controller. The only object handled by this system is a robot attached to three IR LEDs; any other objects are disregarded completely. The latest SixthSense prototype [19] is close to an ideal interactive portable system and offers meaningful interactions with different objects found in the real world. However, it focuses on controller-centered interactions where the controller objects are fingers equipped with color markers.

Real-time pro-cam geometric calibration: When a projector and a camera are rigidly fixed to each other, some assume that the geometric registration between them is roughly constant [4]. However, as the angle of a projector moves from the perpendicular, this approach cannot guarantee good geometric registration. Projecting a known pattern onto a surface is a classic approach to solve this problem that gives precise calibrations for both planar surfaces [10,23,24] and irregular surfaces [2,22,28]. However, the computational cost is high for a complex surface, and patterns must be re-projected when a component of the system (e.g., a projector, camera or surface) moves. A real-time approach that does not interrupt normal projection was proposed in [16] by attaching four laser-pens to a pro-cam system. Although detecting bright laser points sounds easier than detecting points projected by a projector, locating small laser points in a messy camera image is still difficult. In this paper, we chose to apply a real-time approach inspired by [9] using only one additional motion sensor. Real-time geometric calibration was achieved with a single self-contained device.

Multiple-target tracking: Particle filtering [14] has become one of the most popular visual tracking algorithms over the past decade. One problem with this approach is that particle filtering can track only one target at a time. Additional algorithms are required to make particle filtering work with multiple targets. In former approach, full knowledge of the true targets (including when and where they appear and disappear) must be provided [18]. Some simplify the approach by limiting the maximum number of true targets [15]. Unlike these, however, [20] proposed a hybrid approach that can track an unlimited number of sensors. This approach is very close to our requirement, but cannot be applied directly. The computational costs of sensor-based tracking and image-based tracking differ too greatly. In this paper, we modify several proposals in [20] and achieve a new tracking approach that is more suited to image-based tracking.

Nonintrusive projection: This topic is a subset of the embedded imperceptible-pattern projection problem. Lately, prototypes of an infrared projector were proposed in [1,17] to project infrared and visible light simultaneously. An infrared pattern is fixed by using an internal mask inside a projector in [1], but is variable in [17]. Unfortunately, the work of Lee et al. [17] requires many internal changes inside a Digital Light Processing (DLP) projector that can be accomplished only by a commercial manufacturer. While an infrared projector is under investigation, there are existing solutions proposed for this problem. For the office of the future [25], embedding structured light into a DLP projector can be achieved by significant changes to the projection hardware. However, this implementation is impossible unless it is incorporated into the design of the projector or full access to the projection hardware is available. In [12,21,29], a code image is projected at high speed with its neutralized image, which integrates the coded patterns invisibly because of limitations of the human visual system. According to these papers, projecting and capturing at 120 Hz can guarantee a hidden code. Commonly available projectors usually perform projections at a maximum rate of 87 Hz. For this paper, we apply an approach proposed in [8], which can be achieved using an off-the-shelf DLP projector.

3 Proposed framework

The configuration of our system is illustrated in fig. 1. A camera and a motion sensor are fixed to a DLP projector, forming a single self-contained device. A VGA splitter is added for pro-cam synchronization (see Section 3.3 for details).

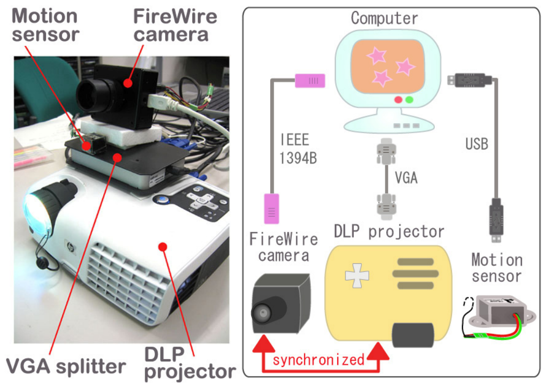

Figure 1: System configuration.

As shown in fig. 2, the framework supporting our approach consists of five main steps. First, the projection area appearing inside a camera image is located (*Calibration*).

Then, an appropriate algorithm is applied to detect objects for interaction *(Detection)*. Next, contours of the detected objects are sent to the tracker, so that each object can be labeled according to its previous state *(Tracking)*. After this, an individual interaction is assigned to each object following its previous status *(Interaction)*. A *nonintrusive projection* technique is applied in this step to guarantee that all interactions are drawn using the correct colors. Up to this point, all calculations are performed using camera coordinates. Finally, another calibration is performed to convert every interactive projection to projector coordinates *(Calibration)*. As noted in fig. 2, the detection step and the interaction step are performed using different algorithms (depending on the application), so a detailed explanation of them is not considered here. The following sections discuss the calibration, tracking and nonintrusive projection used in our system.

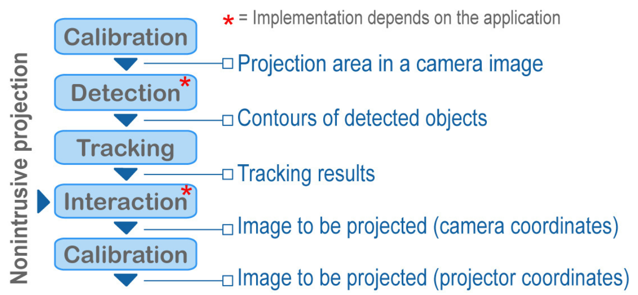
* = Implementation depends on the application
Calibration
Projection area in a camera image
Detection*
Contours of detected objects
Tracking
Tracking results
Interaction*
Image to be projected (camera coordinates)
Calibration
Image to be projected (projector coordinates)
Nonintrusive projection

Figure 2: Overall procedural flow.

3.1 Real-time pro-cam geometric calibration using a motion sensor

This section describes geometric calibration between a projector and a camera using a motion sensor. A projector, camera and motion sensor are fixed together so that their relative positions and orientations cannot be changed. Two tilt sensors fixed to a projector were first proposed in [24]. Acquiring the tilt angles from both sensors in real time allows the correct estimation of the world horizontal and vertical directions without using markers. Dao et al. [9] extended the sensor-based idea to an accelerometer combined with a digital compass. Their system measures the inclined angle of a projector directly in both vertical and horizontal axes, and then creates an interactive game by using a real-time keystone correction feature.

A sensor eliminates the need for markers but still allows a single self-contained device, so we decided to apply the sensor approach to our system. Our purpose is to obtain updated transformations between camera coordinates and projector coordinates in real time. A real-time keystone correction is not applied here, but it can be performed if required.

In this paper, a NEC/TOKIN MDP-A3U9S 3D motion sensor is used with a data update rate of 125 Hz. The relative pitch and roll angles are calculated from three acceleration values read from the sensor. Setting the reference angles is simple: the projector is moved until the images appear rectangular on a surface, and then a key or button is pressed. The reference can be reset whenever significant errors occur in the calibration. Five consecutive samples of the 3D accelerations acquired from the sensor are averaged in real time before being used to compute the relative pitch and roll angles. Averaging adds a delay to the calibration but is recommended for smoother calibration.

This approach requires offline calibration. However, the calibration data are compatible with the system if there is no change in the relative positions or orientations of the three devices. Suppose that the offline calibration is done by N sample images captured from the camera and all sample images share a set of n points to be calibrated. A set of calibration data provided by one sample image is written as $(p, r, x_1, y_1, x_2, y_2, \cdots, x_n, y_n)$. Let p and r refer to the relative pitch and roll angles; (x_i, y_i) represents the 2D coordinates of an i^{th} observed point in the sample image. For N sample images captured from different angles and orientations, we have:

$$A=\begin{bmatrix} p_{(1)} & r_{(1)} & 1 \\ p_{(2)} & r_{(2)} & 1 \\ \vdots & \vdots & \vdots \\ p_{(N)} & r_{(N)} & 1 \end{bmatrix}, B=\begin{bmatrix} x_{1(1)} & y_{1(1)} & x_{2(1)} & y_{2(1)} & \cdots & x_{n(1)} & y_{n(1)} \\ x_{1(2)} & y_{1(2)} & x_{2(2)} & y_{2(2)} & \cdots & x_{n(2)} & y_{n(2)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{1(N)} & y_{1(N)} & x_{2(N)} & y_{2(N)} & \cdots & x_{n(N)} & y_{n(N)} \end{bmatrix}.$$

An adjustment matrix (β) is obtained by using linear least squares to solve:

$$\left(A^T A\right)\beta = A^T B. \quad (1)$$

At any time τ during online calibration, (x_i, y_i) is updated by Equation (2) using the relative pitch and roll angles calculated as explained earlier.

$$\left[x_{1(\tau)}, y_{1(\tau)}, x_{2(\tau)}, y_{2(\tau)}, \cdots, x_{n(\tau)}, y_{n(\tau)}\right] = \left[p_{(\tau)}, r_{(\tau)}, 1\right]\beta \quad (2)$$

Following the above explanation, the first calibration step written in fig. 2 is performed. The system is able to locate the projection area appearing inside a camera image, even though the position and orientation of a projector (relative to a surface) are not known.

To achieve a more precise calibration, n compensation values are computed at startup for each (x_i, y_i). The compensation values are obtained by finding differences between each (x_i, y_i) pair computed by our sensor approach and its corresponding (x_i, y_i) pair actually found on a camera image. During online calibration, the compensation values calculated since startup are added to those values calculated from the sensor approach. Our experiments showed that this compensation actually increased accuracy in the calibration using very few additional computations.

The other calibration step shown in fig. 2 is performed by applying perspective transform, as proposed in [27]. We chose this approach instead of the full calibration approach proposed in [24] because of its lighter computational load. The transformation from camera coordinates to projector coordinates is updated in real time by referring to (x_i, y_i) in Equation (2). After a response image is generated in camera coordinates (i.e., after the *Interaction* step in fig. 2), it is warped to projector coordinates. In this way, when the response image (in projector coordinates) is projected, the geometry appears on the surface as required. Note that when using this calibration approach, a surface must be planar but may be slanted.

3.2 Improved multiple-target tracking using particle filters

As mentioned previously, our tracking is based on an approach proposed in [20] with modifications suited to image-based tracking. An input to our tracker is a single-channel binary image created by the *Detection* step shown in fig. 2, with white contours inside the input image referring to target objects to be tracked. The maximum number of target objects varies in time according to the detected number of targets. Hence, computations are incurred only where necessary. The outputs from our tracker are the clustering particles and object identification. An attractive aspect of this approach is that it performs a consistency check. Moreover, it offers strategies to avoid premature initialization or finalization that may arise from misdetection in a few input images.

Migration from the approach of [20] to our approach is straightforward. However, while the approach of Ng et al. measures and stores single values from a sensor, our approach stores sets of points that form the contours found in the input image. Two major changes were made to the deterministic clustering algorithm.

First, we introduce a new approach to compute a normalized distance between two contours found at different times. The normalized distance is an important key to judge the accuracy of the clustering algorithm and must be defined carefully to suit each data format. Our experiments found a combination of three representative values that best distinguish whether two contours originate from the same target object. As shown in fig. 3, the representative values are the intersection area, the approximate minimum distance, and the absolute difference between the sizes of the contour.

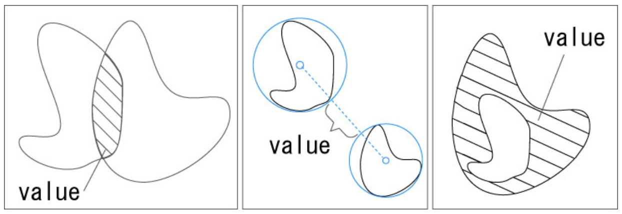

Figure 3: Three representative values for computing the normalized distance between two contours.

Two contours from different times are most likely to be clustered together if the intersection is large and both the approximate minimum distance and the absolute difference between the sizes of the contour are small. The normalized distance is calculated by a weighted average approach. In the current implementation, weights of 3, 2 and 1 are used for the intersection, the minimum distance and the absolute difference of sizes, respectively. This approach worked well in both simulation and real-time camera capturing experiments, and was able to deal with both static objects and dynamic objects whose positions or shapes changed over time. Considering that changes to an object do not increase sharply in sequential capturing, we set a threshold for the normalized distance to consider whether to group two contours.

Second, we introduce a new clustering iteration. The successive scan approach proposed by [20] is compatible with our proposed normalized distance. However, computational loads were significantly higher than for other calculations in tracking. The $\Theta(\eta^2)$ iteration, where η is the number of contours detected concurrently, is not good for interactive applications, especially those using image processing. We closely examined related factors and tried to make this scan faster. Dynamic data structures were used to store all possible combinations of the normalized distance. We hoped that reducing the number of calculations in the latter iterations might increase the overall speed. Unfortunately, the cost of dynamic data structures was too high with large datasets. Consequently, the overall speed barely increased.

Later, we developed an approach that can perform clustering in $\Theta(\eta)$. The overall speed increases remarkably in this approach. Additional data storage, using a linked list of linked lists, is illustrated in fig. 4. Each element of the outer linked list contains three pieces of information about its corresponding inner linked list: the number of elements, a pointer to the first element, and a pointer to the last element. The inner chain formed by one inner linked list represents a trajectory of one object being tracked by the tracker.

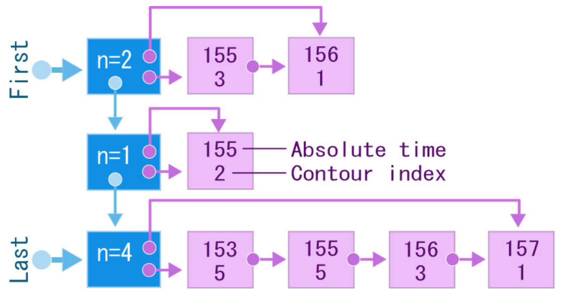

Figure 4: Linked list of linked lists for fast clustering.

Using this data storage method, clustering starts by eliminating inner linked list elements whose corresponding contour is no longer stored in the tracker's buffer. This is achieved by checking the first element of all inner linked lists. An element is removed from its inner linked list if its absolute time is too old compared with the current absolute time and the size of the tracker's buffer. Because elements in each inner linked list are ordered by a unique absolute time, there is no need to check the rest of the inner linked list elements. After elimination, every contour found in the current input image is processed. The normalized distances are calculated between each new input contour and the contours found at the end of all inner linked lists. If any input contour is considered to belong to an existing inner chain, it is added to the end of that chain. Otherwise, a new outer linked list element is initialized with an inner linked list containing that input contour.

A comparison between the original successive scan approach and our approach is shown in fig. 5. All experiments shared the same set of 200 continuous input images; other parameters except the buffer size were fixed. Apparently, the successive scan approach spent more time calculating when the buffer size grew larger, while the computational cost of our approach barely increased, even though the buffer became four times larger.

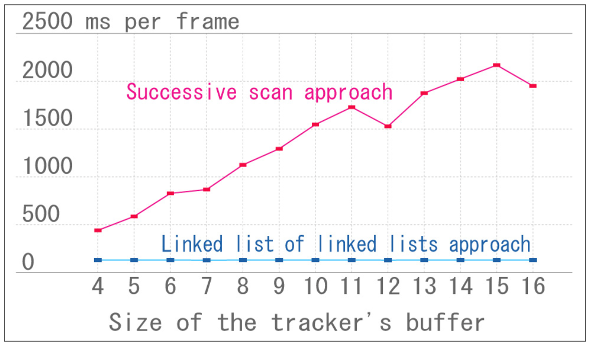

Figure 5: Comparison between the original successive scan approach and our linked list of linked lists approach.

By applying these two modifications, we are able to cluster all contours available in the buffer. A region of interest (ROI) is created for each inner chain if and only if the number of contours in that chain reaches a persistent threshold. Finally, we continue the ROI classification as explained in [20]. Particle filtering is performed inside each individual ROI using the propagation function

$$p_t = p'_t + V, \qquad (3)$$

where p'_t and p_t refer to 2D coordinates of a particle before and after propagation, and V is a random velocity set from our experiment. In this way, we are able to identify and track an unknown number of objects efficiently using particle filters. Note that all tracks share the same number of particles in our implementation.

3.3 Nonintrusive projection

As mentioned in the introduction, it is important that a real-time environment analysis not have any interference from any projected contents. We apply an approach proposed in [8] for three reasons: it requires no internal change to a projector or camera, it can apply to any off-the-shelf DLP projector, and it supports embedded variable light patterns in the future without further hardware modifications.

The chosen approach requires a DLP projector and a camera with an external trigger feature. Synchronization between the projector and the camera is necessary and is performed by tapping the vertical sync signal (5 V, 60 Hz) from the computer to the projector. By using the tapped signal as a trigger, our camera remains synchronized to the projector. The synchronization setup is shown in fig. 6. Because our DLP projector uses a VGA input, a VGA splitter is used to split the input signal.

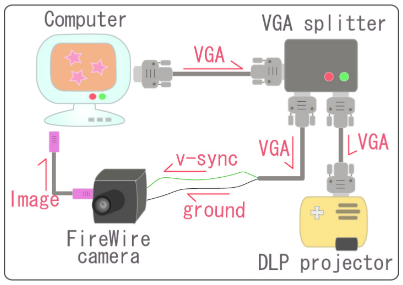

Figure 6: Projector-camera synchronization.

The following devices were used for the pro-cam synchronization: a HP MP2225 DLP projector with D-sub connector, a Dragonfly Express camera connecting through a FireWire 800 port (aka. IEEE1394B port), and an ELECOM VSP-A2 VGA splitter. The camera is equipped with a Tamron 13VM308AS lens.

To understand the overall characteristics of the color wheels inside our DLP projector, we projected single-color images (corresponding to the colors of each available color wheel of the projector) at maximum intensity. Fig. 7 was created by allowing the synchronized camera to sense those projected colors with different synchronization delay times. Note that our DLP projector has five color wheels: red, yellow, green, white and blue. The extra yellow wheel offers richer reds and brighter yellows in projection.

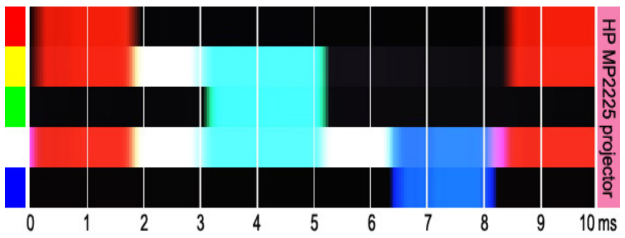

Figure 7: Color-wheel sequence of the HP MP2225.

To achieve this technique, the shutter of the camera is set to open for only 0.3 ms, which is too fast for the camera to see things properly unless there is light from the projector (see fig. 8-B). Hence, we need to illuminate the environment while projecting nonintrusive interactive responses. From fig. 7, we selected red and white with a 1 ms delay to perform nonintrusive projection. We used both colors at their maximum intensity; yellow was not chosen to avoid possible interdependent color channels. When a projected image contains only two selected colors, the camera that is synchronized with a 1 ms delay will see an environment completely lit up with red light, as depicted in fig. 8-C. Note that the intensity of fig. 8-B was enhanced to allow the environment to be seen here.

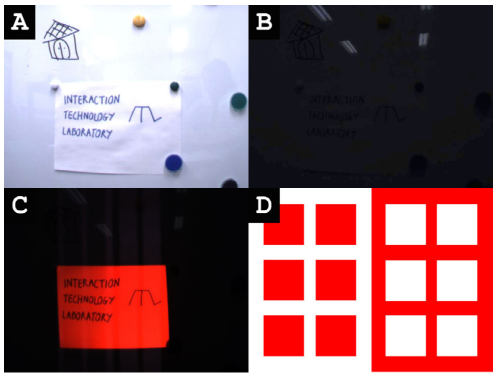

Figure 8: (A) is an environment seen by a camera with normal settings. (B) is the environment (A) seen by the synchronized camera. (C) is the environment (A) seen by the synchronized camera when the image (D) is being projected from the DLP projector.

3.4 Proofs of concept

To prove the proposed framework, we implemented two basic programs. Detection (step 2 of fig. 2) is performed using the 2D Gabor filter [11], which is known to be useful in segregating textural regions. Input to the detection is a captured grayscale image, which is then convolved with the 2D Gabor function

$$G_{\lambda,\Theta,\varphi}(x,y) = e^{-((x'^2+\gamma^2 y'^2)/2\sigma^2)} \cos\left(2\pi\frac{x'}{\lambda}+\varphi\right), \tag{4}$$

where

$$x' = x\cos\Theta + y\sin\Theta \qquad y' = -x\sin\Theta + y\cos\Theta$$

The angle Θ is set to zero in our programs. For uniform planar surfaces, the area whose Gabor filter response magnitude is low will be considered the target object. In other words, the target objects for our programs are textural objects or surfaces that have either nonuniform reflections or discontinuities.

In the first program, we randomly assigned three different animations to each target object. Animations were attached to the corresponding object and followed them even if the object or the projector was moving. The purpose of using animation is to demonstrate the continuity of the interaction provided for each object. The second program provided the same animations as the first program, and a clock was drawn in an object-free location. The clock always appeared in the largest object-free area using the dynamic target area approach discussed in [26]. The location and size of the clock were adjusted adaptively to fit with the current object-free area, which may change dynamically because of movements of either the projector or the objects.

Fig. 9 shows these programs in action on an actual surface. These images were shot from a separate camera in order to show how a human actually saw the projected interactions on a surface. Using the proposed framework, objects were initialized and finalized automatically when they entered or left the projection area. Each object was tracked separately in real time so that its corresponding animation was updated correctly in a continuous manner. Because each object has a unique label in the proposed framework, a series of interactions can be assigned to objects in a specific or random order. This should be useful for creating an interactive game, adding a story, or for virtual communication among objects (both physical and virtual).

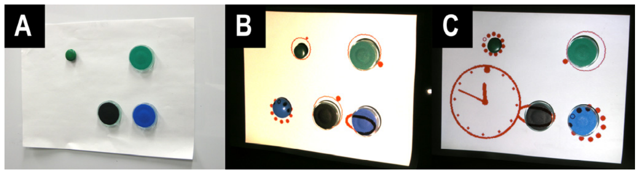

Figure 9: Snapshots of the test environment (A), the first program (B), and the second program (C).

4 Experimental results & evaluations

In this section, we discuss the experiments conducted to evaluate the accuracy and speed of the proposed framework. All experiments were performed using a Dell Inspiron 1150 Mobile Intel® Pentium® 4 laptop with a processor running at 2.80 GHz.

4.1 Accuracy

We measured the accuracy of the pro-cam geometric calibration by comparing a ground truth projection area to a program-generated projection area. The four corners of the projection area were used as calibrated points (i.e. $n = 4$ in Equation (2)). Two measurements were used, as illustrated in fig. 10: a percentage of the overlapped area and a mean Euclidean distance. Experiments were conducted on three different approaches: our calibration without compensation, our calibration with compensation, and the static projection. The last approach assumed that the projection area was unchanged since startup, and is added here for reference. We performed an experiment using 20 test points and plotted the graphs shown in fig. 10. For one test point, two measurement values were computed for the three approaches using eight different calibration data lengths (N). The image resolution was set to 640×480 pixels in the experiment.

From fig. 10, it is obvious that our proposed technique with

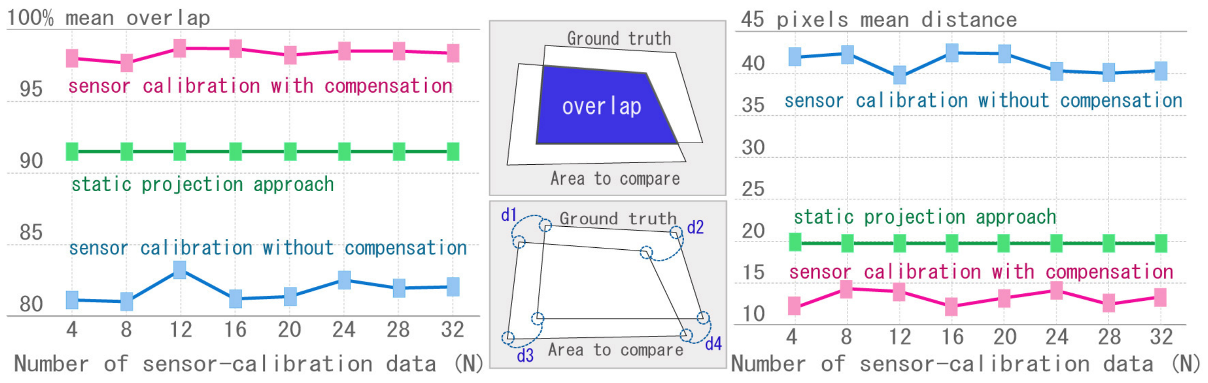

Figure 10: The number of calibration data (N) vs accuracy of pro-cam geometric calibration of three approaches. (Left) shows an overlap percentage between a ground truth and each approach. (Right) shows a mean Euclidean distance between a ground truth and each approach.

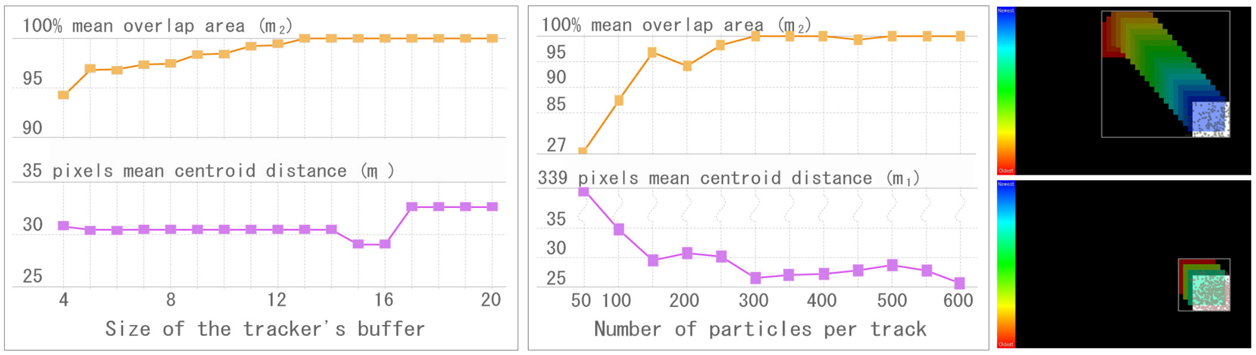

Figure 11: Accuracy of tracking vs size of the tracker's buffer (left) and the number of particles per track (middle). (Right) shows trajectory and ROI created by the tracker: (top) buffer = 20 and particles = 200; (bottom) buffer = 4 and particles = 600.

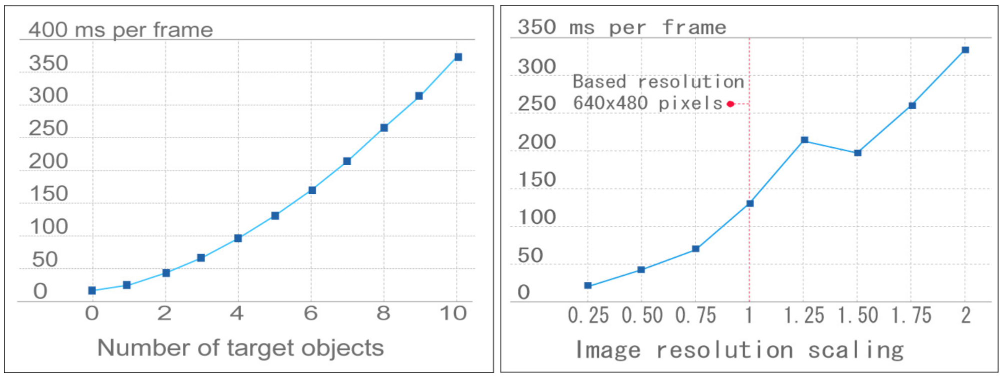

Figure 12: Speed of multiple-target tracking vs the number of target objects (left) and image resolution (right).

compensation gave the best geometric calibration in both measurements. The number of sensor-calibration data (*N*) showed no significant impact on accuracy. However, the static projection approach did better than our technique without compensation. A separate experiment using a different set of pro-cam showed that we could achieve high-precision calibration (a maximum of 12 pixels mean Euclidean distance and minimum of 91% mean overlap area) without compensation by fixing a projector and a camera so their optical axes were as close to coaxial as possible. Nevertheless, the approach with compensation still offered better calibration, according to the separate experiment.

The accuracy of the multiple-target tracking was tested using

200 simulated images that showed one square (100×100 pixels) moving with constant velocity in both the X-axis and the Y-axis. Resolution of the images was fixed at 640×480 pixels. If the detection was accomplished and there was only one target object tracked, two measurements (m_1 and m_2) were applied in the experiments: the Euclidean distance between a centroid of the square and a centroid of particles, and a percentage of the overlap area between the square and a rectangle bounding all particles. We observed the impact of two factors upon tracking accuracies, as shown in fig. 11. The first factor is the size of the tracker's buffer (fig. 11-left) and the second factor is the number of particles per track (fig. 11-middle).

From fig. 11-left and fig. 11-middle, both factors show a slight impact on tracking accuracies. In the middle image, 50 particles were not sufficient to track the 100×100 pixels square. Therefore, the m_1 value increased significantly and the m_2 value dropped sharply compared with the remainder of the middle image. This is the normal behavior of particle filters, where the number of particles must be assigned carefully to ensure full coverage of the object. Fig. 11-right shows the trajectory of tracking and the ROI in two different settings. Note that the ROI is created by the tracker and is not the bounding rectangle used to compute m_2 .

For the nonintrusive projection, we project images containing only two chosen colors. Therefore, the synchronized camera always sees projected images as a completely red image (as shown in fig. 8-C).

4.2 Speed

The proposed pro-cam calibration includes two steps, as described earlier. While the first step, which locates the projection area inside a captured image, took 20 ms to calculate Equation (2), the compensation added little computational load. The second step, which refers to the transformation update and image warping, required 40 ms to create a 640×480 pixels projected image in projector coordinates.

We closely investigated five factors that might affect the speed of the proposed tracking. These factors include the number of target objects (5), the size of the tracker's buffer (4), the number of particles per track (200), the image resolution (640×480 pixels) and the size of the tracked object (118×118 pixels). The numbers given here in brackets are the default values for that factor. Five experiments were conducted for each factor using 200 simulated images that represented perfect detection results for static squares.

From the experiments, two factors (the number of target objects and the image resolution) showed significant effects on tracking speed, as shown in fig. 12. The other three factors contributed less than 5 ms per frame to the average computational times. The experiment on the size of the tracker's buffer is shown in fig. 5.

The nonintrusive projection has no computational cost in this implementation. This is because the two nonintrusive colors are chosen offline, and the projected image is created by using one color as the background color and the other color as the content color.

5 Conclusion and future work

In this paper, we investigated three fundamental requirements of a portable projector that is extended from conventional controller-centered interactions to object-centered interactions. First, a real-time pro-cam geometric calibration is performed using a motion sensor to achieve a single self-contained device. Second, multiple-target tracking is introduced using particle filters to keep track of objects whose appearance and disappearance are unknown. The tracking allows projective interactions among physical and virtual objects. In addition, each physical object is associated with its individual interaction according to the identification feature of the tracking. Finally, we apply a hardware-based approach using a DLP projector so that projected content does not intrude on the environment analysis.

The object-centered interaction concept proposed in this paper will help register physical objects in the virtual world. Virtual information projected by a projector is able to appear on and in response to physical objects (if any) in the most appropriate manner. In the future, we plan to investigate a high-precision detection algorithm that benefits from the proposed framework and offers a robust portable system. Furthermore, we are interested in adding a classification to the framework in order to interact with objects in a meaningful manner.

References

[1] K. Akasaka, R. Sagawa, and Y. Yagi. A sensor for simultaneously capturing texture and shape by projecting structured infrared light. In *Proc. of 3DIM*, 2007.

[2] J.V. Baar, T. Willwacher, S. Rao, and R. Raskar. Seamless multi-projector display on curved screens. In *Eurographics Workshop on Virtual Environments (EGVE)*, 2003.

[3] P. Beardsley, C. Forlines, R. Raskar, and J. VanBaar. Handheld projectors for mixing physical and digital textures. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[4] S. Borkowski, O. Riff, and J.L. Crowley. Projecting rectified images in an augmented environment. In *Proc. of the IEEE International Workshop on Projector-Camera systems (PROCAMS)*, 2003.

[5] X. Cao and R. Balakrishnan. Interacting with dynamically defined information spaces using a handheld projector and a pen. In *Proc. of the ACM Symposium on User Interface Software and Technology (UIST)*, 2006.

[6] X. Cao, C. Forlines, and R. Balakrishnan. Multi-user interaction using handheld projectors. In *Proc. of ACM Symposium on User Interface Software and Technology (UIST)*, 2007.

[7] D. Cotting and M. Gross. Interactive environment-aware display bubbles. In *Proc. of ACM Symposium on User Interface Software and Technology (UIST)*, 2006.

[8] D. Cotting, M. Naef, M. Gross, and H. Fuchs. Embedding imperceptible patterns into projected images for simultaneous acquisition and display. In *Proc. of IEEE and ACM ISMAR*, 2004.

[9] V.N. Dao, K. Hosoi, and M. Sugimoto. A semi-automatic realtime calibration technique for a handheld projector. In *Proc. of ACM VRST*, 2007.

[10] M. Fiala. Automatic projector calibration using self-identifying patterns. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[11] S.E. Grigorescu, N. Petkov, and P. Kruizinga. Comparison of texture features based on gabor filters. *Journal of IEEE Image Processing*, 2002.

[12] A. Grundhofer, M. Seeger, F. Hantsch, and O. Bimber. Dynamic adaptation of projected imperceptible codes. In *Proc. of IEEE and ACM ISMAR*, 2007.

[13] K. Hosoi, V.N. Dao, and M. Sugimoto. CoGAME: Manipulation by projection. In *Proc. of ACM SIGGRAPH Emerging Technologies*, 2007.

[14] M. Isard and A. Blake. CONDENSATION-conditional density propagation for visual tracking. *Journal of Computer Vision (IJCV)*, 1998.

[15] M. Jaward, L. Mihaylova, N. Canagarajah, and D. Bull. Multiple object tracking using particle filters. In *Proc. of IEEE Aerospace Conference*, 2006.

[16] A. Kushal, J.V. Baar, R. Raskar, and P. Beardsley. A handheld projector supported by computer vision. In *Proc. of the IEEE Asian Conference on Computer Vision (ACCV)*, 2006.

[17] J.C. Lee, S. Hudson, and P. Dietz. Hybrid infrared and visible light projection for location tracking. In *Proc. of ACM Symposium on User Interface Software and Technology (UIST)*, 2007.

[18] J. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 1993.

[19] P. Mistry, P. Maes, and L. Chang. WUW – Wear Ur World – A wearable gestural interface. In *Proc. of the CHI '09 Extended Abstracts on Human Factors in Computing Systems*, 2009.

[20] W. Ng, J. Li, S. Godsill, and J. Vermaak. A hybrid approach for online joint detection and tracking for multiple targets. In *Proc. of IEEE Aerospace Conference*, 2005.

[21] H. Park, M.H. Lee, B.K. Seo, Y. Jin, and J.I. Park. Content adaptive embedding of complementary patterns for nonintrusive direct-projected augmented reality. In *Proc. of HCI*, 2007.

[22] R. Raskar, J.V. Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. iLamps: Geometrically aware and self-configuring projectors. In *Proc. of ACM SIGGRAPH*, 2003.

[23] R. Raskar, J.V. Baar, and J.X. Chai. A low-cost projector mosaic with fast registration. In *Proc. of the IEEE Asian Conference on Computer Vision (ACCV)*, 2002.

[24] R. Raskar and P. Beardsley. A self correcting projector. In *Proc. of the IEEE Computer Vision and Pattern Recognition (CVPR)*, 2001.

[25] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modelling and spatially immersive displays. In *Proc. of ACM SIGGRAPH*, 1998.

[26] T. Siriborvornratanakul and M. Sugimoto. Clutter-aware dynamic projection system using a handheld projector. In *Proc. of the IEEE International Conference on Control Automation Robotics and Vision (ICARCV)*, 2008.

[27] R. Sukthankar, R.G. Stockton, and M.D. Mullin. Smarter presentations: Exploiting homography in camera-projector systems. In *Proc. of IEEE International Conference on Computer Vision (ICCV)*, 2001.

[28] W. Sun, X. Yang, S. Xiao, and W. Hu. Robust checkerboard recognition for efficient nonplanar geometry registration in projector-camera systems. In *Proc. of the IEEE International Workshop on Projector-Camera systems (PROCAMS)*, 2008.

[29] M. Waschbusch, S. Wurmlin, D. Cotting, F. Sadlo, and M.H. Gross. Scalable 3D video of dynamic scenes. *The Visual Computer*, 2005.