ARPushPush: Augmented Reality Game in Indoor Environment[*]

Kiyoung Kim, Minkyung Lee, Youngmin Park, Woontack Woo
GIST U-VR Lab.
Gwangju 500-712, S.Korea
+82-62-970-315
{kkim, mlee, ypark, wwoo}@gist.ac.kr

Jongweon Lee
Department of Digital Contents
Sejong University
Seoul 143-747, S.Korea
+82-2-3408-3798
jwlee@sejong.ac.kr

ABSTRACT
In this paper, we propose ARPushPush, a novel augmented reality game for indoor environment, which uses vision-based tracking and user's hand gestures. The systems using trackers are too expensive and the calibration between a camera and a tracker, which is indispensable for vision-based interaction, is not easy. The proposed system only uses two cameras for tracking markers and augmenting virtual objects, respectively. The tracking camera detects ARToolKit markers attached on the ceiling and calculates the pose of user's head on the fly [11]. The HMD camera detects markers attached to the back of user's hand for interaction and gives augmented view to a user. The transformation matrix between two cameras is updated when markers are visible in each camera. To support wide working area effectively, we propose an algorithm to train multiple markers on the ceiling. In the ARPushPush based on our framework, a user acts to move blocks augmented in a virtual maze to predefined places. The proposed framework allows a user to work in wide area, to interact naturally, and to collaborate with other users.

Keywords
Augmented Reality, ARToolKit, HCI, Game, VR, Hand gestures

1. INTRODUCTION
Augmented Reality (AR) is a technology which provides more enhanced immersion by seamless merging of real and virtual worlds. And it also provides realism through interaction with augmented objects [1]. Tracking and interaction technologies are essential for implementation of AR applications. Especially, the recent developments in tracking technologies guarantee wider AR environment.

Many researchers have developed AR systems which enlarge user's working area to wide indoor environment. Foxlin et al. proposed 'Vis-tracker' which tracks user's head pose in indoor environment with gyro sensors and vision sensors [2]. 'Vis-tracker' estimates user's head pose using markers on the ceiling and the wall. Galantay et al. proposed 'Living-room' which emphasized interaction [3]. ARToolKit markers are attached densely on the ceiling and the wall. It helps robust tracking of user's head pose. Piekarski et al. proposed a hybrid tracking system which can be applied to indoor and outdoor environment [4]. The estimation of user's head pose is done by using GPS outdoors and ARToolKit markers indoors. Thomas et al. developed 'ARQuake' game with first-person view with Pierkarski's system [6]. Recently, Nakazato et al. proposed IR marker system which is robust against the light condition [5]. However, it has lower frame rate than the others. Previous systems which put tracking devices to practical use in indoor environment provide accurate tracking results of user's head pose.

However, the previous systems using trackers are too expensive and the calibration between a camera and a tracker, necessary to support vision-based interaction, is not easy. Since arrangement of all markers are required, it is hard to cover the experimental environment with markers dynamically. Additionally, markers attached on the wall, not the ceiling, obstacle user's natural view. As a result, immersion of user's to the AR environment is reduced.

To overcome these problems, we propose a vision-based AR system designed for entertainment applications in indoor environments. The system uses vision-based tracking method and user's hand gestures to interact with augmented objects. First, we attach a marker tracking camera to the Head Mounted Display (HMD) in order to enhance vision-based interaction. Second, we attach multiple markers to the ceiling in arbitrary order to avoid burden of arrangement and occlusion problems. The tracking camera detects ARToolKit markers on the ceiling and calculates the pose of user's head on the fly. The HMD camera estimates absolute poses by using the calibration result. The calibration result between two cameras is updated when markers are visible in each camera. Third, we attach an interaction marker to the back of user's hand. Combining of an interaction marker and user's hand gesture makes a certain interaction function for controlling games and blocks. ARPushPush is proposed to show usefulness of the proposed framework. In the game, a user acts to move virtual blocks in augmented maze to predefined places.

[*] This research is supported by Immersive Contents Research Center (ICRC) and Realistic Broadcasting Research Center (RBRC) in GIST.

The proposed system enlarges user's working area to indoor environments. It allows users intuitive interaction with augmented objects with the help of hand gestures. Also the occlusion problems, which mean user's hands occlude markers, are solved by using markers on the ceiling. Thus, it increases user's immersion through continuous virtual object augmentation.

This paper is organized as follows: In chapter 2, we describe the background of the proposed AR system. Also, we explain calibration method, multiple marker training method, and interaction algorithm. In chapter 3 we will show experimental results, and finally, conclusions and future works are presented in chapter 4.

2. THE PROPOSED SYSTEM

2.1 System Configuration

The proposed AR system, as shown in Figure 1, adopts vision-based tracking using ARToolKit markers, instead of expensive tracking devices. It consists of two vision system. One is to track multiple markers attached on the ceiling, the other is to augment virtual objects based on the reference marker.

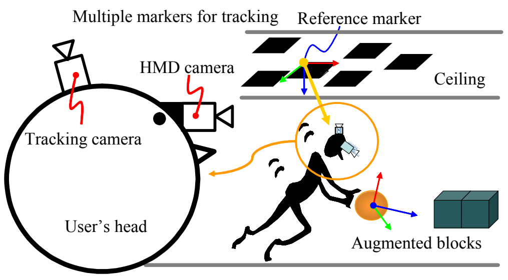

Figure 1. Configuration of the proposed AR system

Relative positions of markers attached on the ceiling in arbitrary order are trained in off-line process. In on-line process, absolute pose of a tracking camera relative to the reference marker is estimated by exploiting marker transformations stored during off-line process. Relative transformation between an HMD camera and a tracking camera is calculated by solving *AX=XB* problem [7]. Multiplying obtained transformation *X* to pose of a tracking camera, we get the transformation matrix of the HMD camera relative to the reference marker. Through this procedure, virtual objects are represented in the reference marker coordinate. Figure 2 shows the overall flow of ARPushPush game system including two processes.

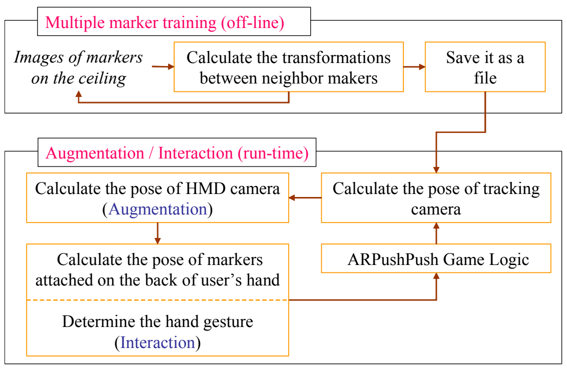
Multiple marker training (off-line)
Images of markers on the ceiling
Calculate the transformations between neighbor makers
Save it as a file
Augmentation / Interaction (run-time)
Calculate the pose of HMD camera (Augmentation)
Calculate the pose of tracking camera
Calculate the pose of markers attached on the back of user's hand
ARPushPush Game Logic
Determine the hand gesture (Interaction)

Figure 2. Overall flow of ARPushPush system structure including off-line process and run-time process

ARPushPush is AR version of a popular game PushPush. In PushPush game, a mission is to move the blocks to the pre-defined spaces. The logic of ARPushPush is the same as original PushPush game. There is a playground which includes several roads. The blocks are located on the road according to the game rules.

2.2 Calibration between a tracking camera and an HMD camera

In this section, we describe a method which calibrates rigid-body transformation between two cameras in real time. Two cameras cannot have correspondent image points because of the structural limitation as shown in Figure 1. Thus, we cannot obtain the rigid-body transformation between two cameras with previous approaches, such as a method using calibration pattern [8]. Additionally, it is impossible to reduce the errors occurred whenever a user moves.

The relationship between a tracking camera and an HMD camera is represented as rigid-body transformation *X* including rotation and translation. Since two cameras are fixed on the same body, *X* is preserved whenever a user moves. Each camera movement from *(i)th* frame to *(i+1)th* frame is represented as rigid-body transformations A_i and B_i, respectively.

$$X = \begin{bmatrix} R_x & t_x \\ 0 & 1 \end{bmatrix} \quad A_i = \begin{bmatrix} R_{a,i} & t_{a,i} \\ 0 & 1 \end{bmatrix} \quad B_i = \begin{bmatrix} R_{b,i} & t_{b,i} \\ 0 & 1 \end{bmatrix} \tag{1}$$

where, *R* represents 3×3 rotation matrix, *t* means 3×1 translation matrix.

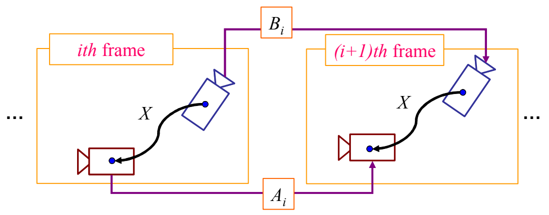

Figure 3. The relationship between a tracking camera and an HMD camera in *(i)th* frame and *(i+1)th* frame, A_i, B_i mean rigid transformation between *(i)th* frame and *(i+1)th* frame of an HMD camera and tracking camera, respectively.

With the proposed system, rigid-body transformation of a tracking camera B_i is known through tracking markers on the ceiling. And, the transformation A_i can be calculated by tracking markers on the back of user's hand. The transformation relationship among A_i, B_i, and *X* is represented as follows:

$$A_i X = X B_i \tag{2}$$

Equation (2) is separated into two parts. Rotation matrices and translation matrices can be separated as follows:

$$R_{a,i} R_x = R_x R_{b,i} \qquad (R_{a,i} - I) t_x = R_x t_{b,i} - t_{a,i} \tag{3}$$

To get R_x in equation (3), we use the clue that the eigenvalues of $R_{a,i}$ and $R_{b,i}$ is the same. That is, the relationship between $R_{a,i}$ and $R_{b,i}$ is similarity transform.

$$R_x^T R_{a,i} R_x = R_{b,i} \quad (\because R_x^T = R_x^{-1}) \tag{4}$$

We can derive following equation with equation (4)

$$e_{a,i} = R_x e_{b,i} \tag{5}$$

where, $e_{a,i}$ and $e_{b,i}$ is eigenvectors of $R_{a,i}$ and $R_{b,i}$, respectively. Thus, rotation matrix of X is obtained by calculating R_x which minimizes equation (6).

$$f = \sum_{i=0}^{n} \| e_{a,i} - R_x e_{b,i} \|^2 \tag{6}$$

where, n is the number of image motions. Minimizing equation (6) is well described in [7].

If R_x is known, we can calculate t_x with easy by using equation (3). Since the scale of translation is determined by real length of marker size, we do not need to care about scale effect. Thus, equation (3) is solved in linear method, such as SVD (Singular Value Decomposition). Equation (3) can be written in equation (7).

$$\begin{bmatrix} (R_{a,i} - I) & -R_x u_{b,i} + t_{a,i} \\ \dots & \dots \end{bmatrix} \begin{bmatrix} t_x \\ 1 \end{bmatrix} = 0 \tag{7}$$

The calibration process between an HMD camera and a tracking camera is summarized as follows:

Step 1: Gather at least 2 pairs of rigid transformations of tracking camera and an HMD camera $\{A_i, B_i\}$
Step 2: Calculate rotation matrix of X minimizing equation (6)
Step 3: Calculate translation matrix of X solve equation (7) using SVD method
Step 4: (Optional) Nonlinear optimization

2.3 Training method for multiple markers

There is a problem when we attach multiple markers to the ceiling. Even though ARToolKit supports the efficient routine that tracks multiple markers, it requires the exact transformation matrices between multiple markers [11]. In our case, attaching markers to the ceiling is hard work with predefined exact transformations. Thus, we need the training method that calculates transformations by using only marker information.

In this paper, we propose a method which resolves problems of previous works by measuring pose relations between initial markers. At first, markers are attached on the ceiling in arbitrary positions, and then select an origin marker in marker group. With the reference, the pose relationship among neighboring markers is calculated.

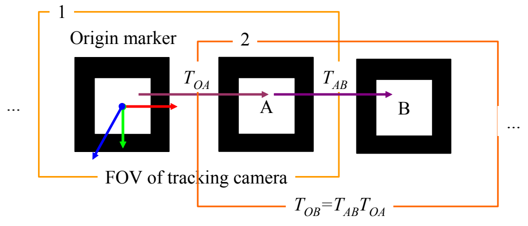

Figure 4. Procedure to extract relative relationship between markers on the ceiling, 1 is the first step and 2 is the second step.

All the markers on the ceiling cannot be included in FOV (Field Of View) of a camera at the same time. Thus, we need an algorithm to calculate relative positions between maximum numbers of markers which can be recognized in one image. For example, shown in Figure 4, we get an image which includes the origin marker and A marker. Then, we calculate T_{OA}, relative rigid transformation matrix, using the camera pose matrices at each marker. If we get an image which only includes marker A and B, then first we calculate T_{AB} with same method, and then we apply T_{OA} calculated before. After the movement of a user, it should be recalculated. After repeating the procedure, finally, we can get the relationship between whole markers on the ceiling.

Practically, if we calculate the pose relationship to one origin, errors are increased according to the distance. Thus, we need to have accurate camera calibration and optimization method using geometry to calculate the pose relationship. However, the optimization method, such as bundle adjustment, is too expensive and hard to implement, thus we introduce very simple and robust method to select good transformation matrix quickly [12].

We assume that each marker on the ceiling has parallel normal vectors. Thus, we use a dot product value of each marker as a threshold. The algorithm of determining good transformation matrix of two markers is as follows:

```
Step 1: Determine the initial threshold value $\tau$
Step 2: Get the camera poses of each marker, A and B
Step 3: Calculate the relative transformation between two markers, $T_{AB}$ as shown in Figure 4
Step 4: Calculate dot product of normal vector of A and B: $dot(n_A, n_B)$
Step 5: If $dot(n_A, n_B) > \tau$, then go to Step 2. else go to step 6.
Step 6: Save the transformation matrix
        Move tracking camera to neighbor marker
        Update $\tau$ & Go to step 2
```

2.4 Interaction with hand gestures

We utilize user's hand gestures with the marker attached on the back of user's hand in order to support intuitive, various, and stable interaction methods. Since the marker is always visible when a user tries to interact with virtual objects, we can combine hand gestures with markers information to provide various interaction functions. We choose the distinct gestures which is very easy to be determined by their shape. Each hand gesture and its function are shown in Figure 5.

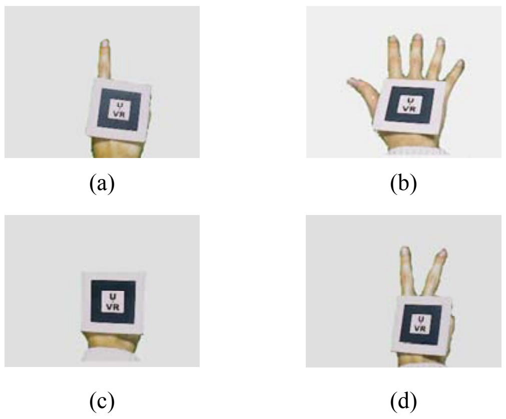

Figure 5. Object control gestures (a) Start/Resume the game (b) Select/Move augmented objects (c) Pause the game (d) Show the game minimap on the screen

Tracking of hand region is done by tracking the interaction marker attached on the back of user's hand. Based on the tracked marker information, we apply heuristic factor to determine search window including user's hand region. After determining search window, we perform hand segmentation using HSV color space. And then, we determine contour of hand to count convex and concave points. Finally, we determine the function of user's gestures. For example, if a user makes V posture as shown in Figure 6, then the number of concave points should be two and the number of convex points should be one. Simple procedure of determining interaction function is shown in Figure 6.

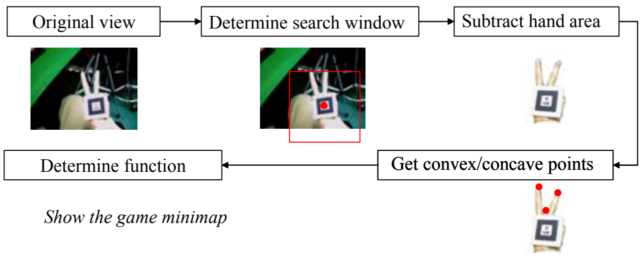

Figure 6. Simple method to combine user's gesture information with interaction functions

3. Experiment

3.1 Experimental Setup

We used a Flea camera for vision-based tracking. The Flea camera adopted IEEE1394 interface, and CCD cell from SONY [9]. And it supports 640×480 size image with 60 frames per sec (fps). It attached on the rigid body hat for the convenience, as shown in Figure 7. We assume that an HMD and the flea camera attached on the hat strictly, so that it guarantees same movement regardless of user's head movement. We can apply various kinds of lenses to Flea. We adopted 6mm camera lens after considering the height of the ceiling. We used a Trivisio video see-through HMD to get mixed user's view [13].

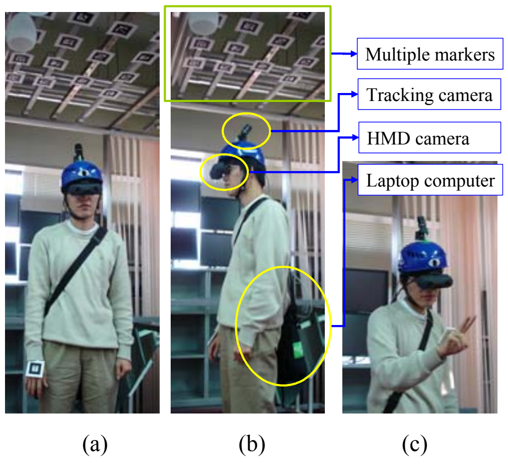

Figure 7. Experimental equipments for a user (Tracking camera, an HMD camera, Laptop computer) (a) front view (b) side view (c) a user is playing the ARPushPush

Figure 8 shows the experimental room environment and marker arrangement. When a user plays ARPushPush, the user's view direction is often close to the floor. Thus, considering that situation, we made tracking camera be tilted around 10 degree so that its direction is orthogonal to the ceiling when a user looks at a floor.

We attached 60 markers on the ceiling. The size of a marker is 100mm×100mm. And the height of the room is 3.0m.

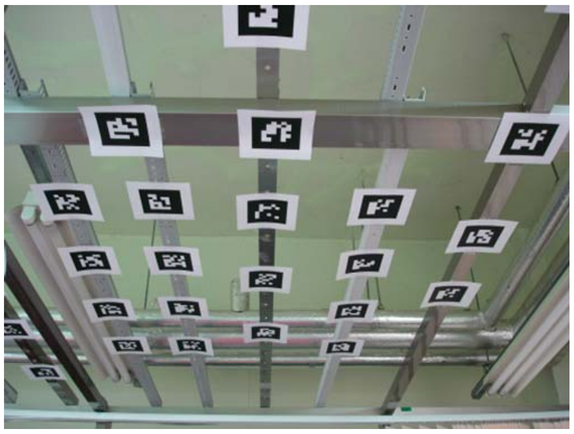

Figure 8. Multiple markers attached on the ceiling for tracking user's head pose

PointGrey's Flea depends on the driver of manufacturer. Thus, we modified existing ARToolKit to develop our proposed system. We used Microsoft Visual C++ 6.0 with ARToolKit 2.65 and OpenCV beta 4.0 [10].

3.2 Experimental Results

We performed multiple markers training explained in section 2.3. We set the initial threshold value to τ = 0.99. Reconstructed and saved multiple markers are shown in Figure 9.

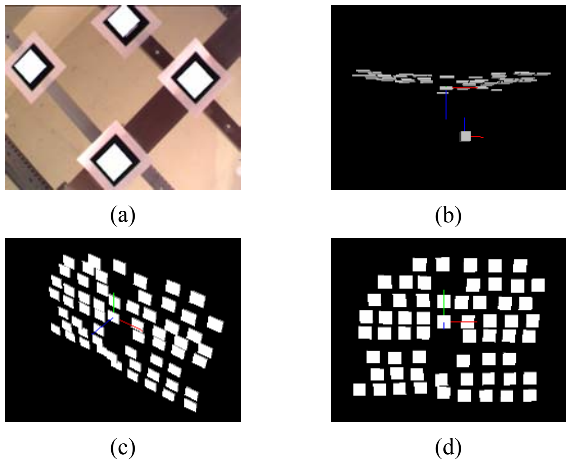

Figure 9. Training of multiple marker and reconstructed multiple markers (a) after saving trained markers, we augmented simple white objects (b) (c) (d) multiple markers stored as a file are rendered using OpenGL

From the scene of rendered markers, we can know that errors are slightly increasing according to distance from the reference marker.

We analyzed the results of training method of multiple markers. As shown in Figure 10, when we used the normal vector constraint, we got better results than we used no constraint. Normal vector error, Y axis of Figure 10, is defined as follows.

$$Normal\ vector\ error = 1 - (n_1 \cdot n_2) \tag{8}$$

where, n_1 and n_2 is the neighbor normal vectors.

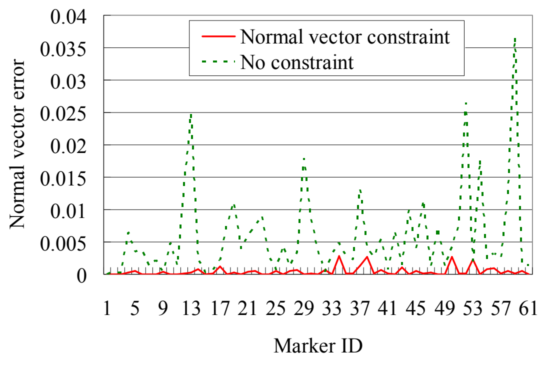

Figure 10. Results of multiple markers training using normal vector constraint

To show the usefulness of the proposed system, we took a usability test on immersion with 9 volunteers. Table 1 shows the results of quantitative measurement of immersion reducing factors. 5 reducing factors were investigated when users were playing the ARPushPush.

Table 1. Quantitative measurement to analyze immersion reducing factors, average and standard deviation

Immersion reducing factors	Average (Max = 5)	STD
Unstable augmentation	4.2	0.8
Unreliable interaction	2.4	0.7
Low frame rate	4.5	0.5
Unrealistic augmented objects	1.6	0.7
Lack of feed-back method	1.6	0.7

Volunteers pointed out unstable augmentation and low frame rate as the strongest factors which reduce immersive feeling. Even though we trained markers well, light condition and sparse distribution of markers made unstable augmentation according to user's fast head movement. And volunteers were more sensitive in visual factors than the other factors.

4. Conclusions and Future works

In this paper, we proposed a vision-based AR system designed for entertainment applications in indoor environments. The system used vision-based tracking method and user's hand gestures to interact with augmented objects. We brought the popular game PushPush into AR environment. The proposed framework provided seamless augmentation through letting markers out of user's view and convenient interaction scheme to manipulate augmented objects with natural hand gestures. And we introduced very simple and fast algorithm to calculate relationships between multiple markers. In the experiment, we showed that our training method reduces normal vector errors effectively. It can be easily applied to various AR applications. As remaining works, we have to improve accuracy of vision-based tracking in order to provide stable augmentation results to users. And we will apply 3D vision techniques with 3D HMD to provide more realism to users.

5. REFERENCES

[1] R. Azuma, "*A Survey of Augmented Reality*", Presence: Teleoperators and Virtual Environments 6, 4, 355 – 385, August. 1997

[2] E. Foxlin, and N. Leonid, "*VIS-Tracker: A Wearable Vision-Inertial Self-Tracker*", IEEE Virtual Reality, Los Angeles, Mar. 2003

[3] M. Engeli, R. Galantay, and J. Torpus, "*living-room Interactive, Space-Oriented Augmented Reality"*, ACM Multimedia, MM'04, New York, USA, October. 2004

[4] W. Piekarski, B. Avery, B. H. Thomas, and P. Malbezin, "*Integrated Head and Hand Tracking for Indoor and Outdoor Augmented Reality*", IEEE Virtual Reality Conference, Chicago, Il, Mar. 2004

[5] Y. Nakazato, M. Kanbara, and N. Yokoya, *"Discreet markers for user localization"*, IEEE International Symposium on Wearable Computers, p.172-173, Nov. 2004

[6] B. Thomas , B. Close , J. Donoghue , J. Squires , P. Bondi, M. Morris, and W. Piekarski, "*ARQuake: An Outdoor/Indoor Augmented Reality First Person Application*", IEEE International Symposium on Wearable Computers, p.139, October. 2000

[7] F. Dornaika, and R. Chung, "*Stereo geometry from 3D ego-motion streams",* IEEE Transactions on Systems, Man, and Cybernetics, Part B 33(2), p.308-323, 2003

[8] Z. Zhang, "*Flexible camera calibration by viewing a plane from unknown orientations*," International Conference on Computer Vision, vol. 1, pp. 666-673, 1999

[9] Point Grey Research Inc., http://www.ptgrey.com, 2002

[10] Intel OpenCV Library, http://www.intel.com/research/mrl/research/openCV

[11] ARToolKit Library, http://www.hitl.washington.edu/research/shared_space/download

[12] R. Hartley and A. Zisserman, "Multiple view geometry in computer vision", Second edition, Cambridge University Press, March 2004

[13] Trivisio Company, http://www.trivisio.com