Matrix: A Realtime Object Identification and Registration Method for Augmented Reality

Jun Rekimoto
Sony Computer Science Laboratory Inc.
3-14-13, Higashi-gotanda, Shinagawa-ku, Tokyo 141 Japan
Phone: +81-3-5448-4380
Fax: +81-3-5448-4273
Mail: rekimoto@csl.sony.co.jp
http://www.csl.sony.co.jp/person/rekimoto.html

Abstract

This paper introduces a new technique for producing augmented reality systems that simultaneously identify real world objects and estimate their coordinate systems. This method utilizes a 2D matrix marker, a square shaped barcode, which can identify a large number of objects. It also acts as a landmark to register information on real world images. As a result, it costs virtually nothing to produce and attach codes on various kinds of real world objects, because the matrix code are printable. We have developed an augmented reality system based on this method, and demonstrated several potential applications.

1. Introduction

In augmented reality (AR) systems, it is crucial to correctly register real world computer information on the real world image. AR systems normally measure the position and orientation of a device with 3D sensors (either magnetic or ultrasonic); however these sensors often suffer from inaccuracy and limited tracking volume. Recently, vision-based methods of estimating position information from known landmarks in the real world scene have been proposed. Bajura and Neumann used LEDs as landmarks and demonstrated vision-based registration for AR systems[1]. Uenohara and Kanade used template matching for object registration[7]. State et al. proposed a hybrid method of combining landmark tracking and magnetic tracking (they used color markers as landmarks)[5].

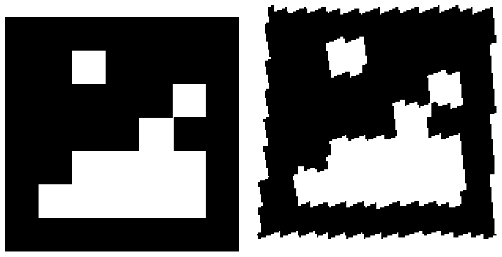


Figure 1. 2D matrix code examples (left: original, right: restored from video)

Aside that the position estimation problem, AR systems also need to determine which information should be overlaid without interfering with the user's real world tasks. For example, it is quite unlikely that a repair worker using AR would need to enter the part ID number to retrieve information. We have reported a system that can automatically identify real world objects using the combination of visual markers and a video camera[4].

This paper describes a new method that satisfies both of the above requirements. Our proposed method identifies real world objects and estimates their position and orientation using a combination of visual markers (Figure 1) and a video camera. The overall information flow of the system is illustated in Figure 2. The 2D-matrix marker our method uses is a square shaped barcode that can identify 2^{16} different objects. We can create and attach codes on a large number of real world objects at virtually no cost, because codes are printable with normal laser or ink-jet printers. The AR system recognizes these codes from captured video images.

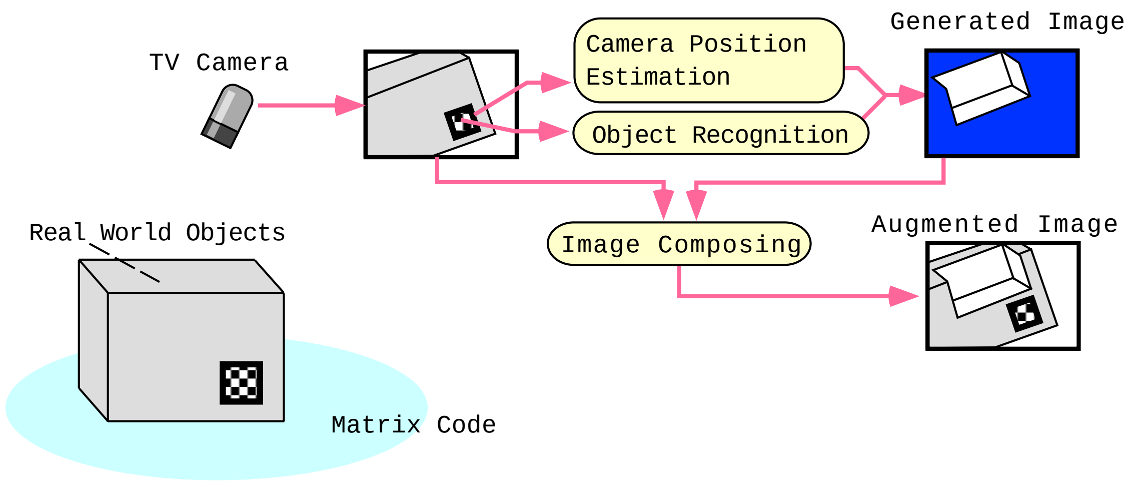


Figure 2. Overview of the proposed vision-based information registration method

By analyzing the distortion of the rectangular shape of the matrix code frame, the system estimates the position and orientation of the video camera. This information is used to correctly overlay annotating information on the captured video image.

2. Recognition Algorithm

The system seeks out 2D matrix codes on incoming video images. It then identifies their code value and estimates the camera position and orientation in relation to the attached position of the matrix code. The entire image processing is performed by the software. The code recognition algorithm takes the following five steps:

(1) Binarization:

First, a captured video image is binarized. (Figure 3-(b)). We use the adaptive thresholding method described in [8]. However a simple binarization using the fixed threshold value also works quite well because the used matrix code has a high enough contrast.

(2) Connected Component Analysis:

Then, the system looks up connected regions of binary-1 (black) pixels (Figure 3-(c)).

For each region found, a heuristic check based on the size and the aspect-ratio of the region bounding rectangle is applied to select code candidate areas.

(3) Code Frame Fitting:

The next step is to find the code frames. For each selected region, a quad-tangle is fitted on the frame of the region using the least-square method.

Then, transformation parameters are calculated based on the positions of the 4 corners of this quad-tangle. Let $(x_i, y_i, 0)$ be a point on the plane of the matrix code and (X_i, Y_i) be a corresponding point on the image plane of the camera. There is a relation between the two:

$$X_i = \frac{a_1 x_i + a_2 y_i + a_3}{a_7 x_i + a_8 y_i + 1}$$

$$Y_i = \frac{a_4 x_i + a_5 y_i + a_6}{a_7 x_i + a_8 y_i + 1}$$

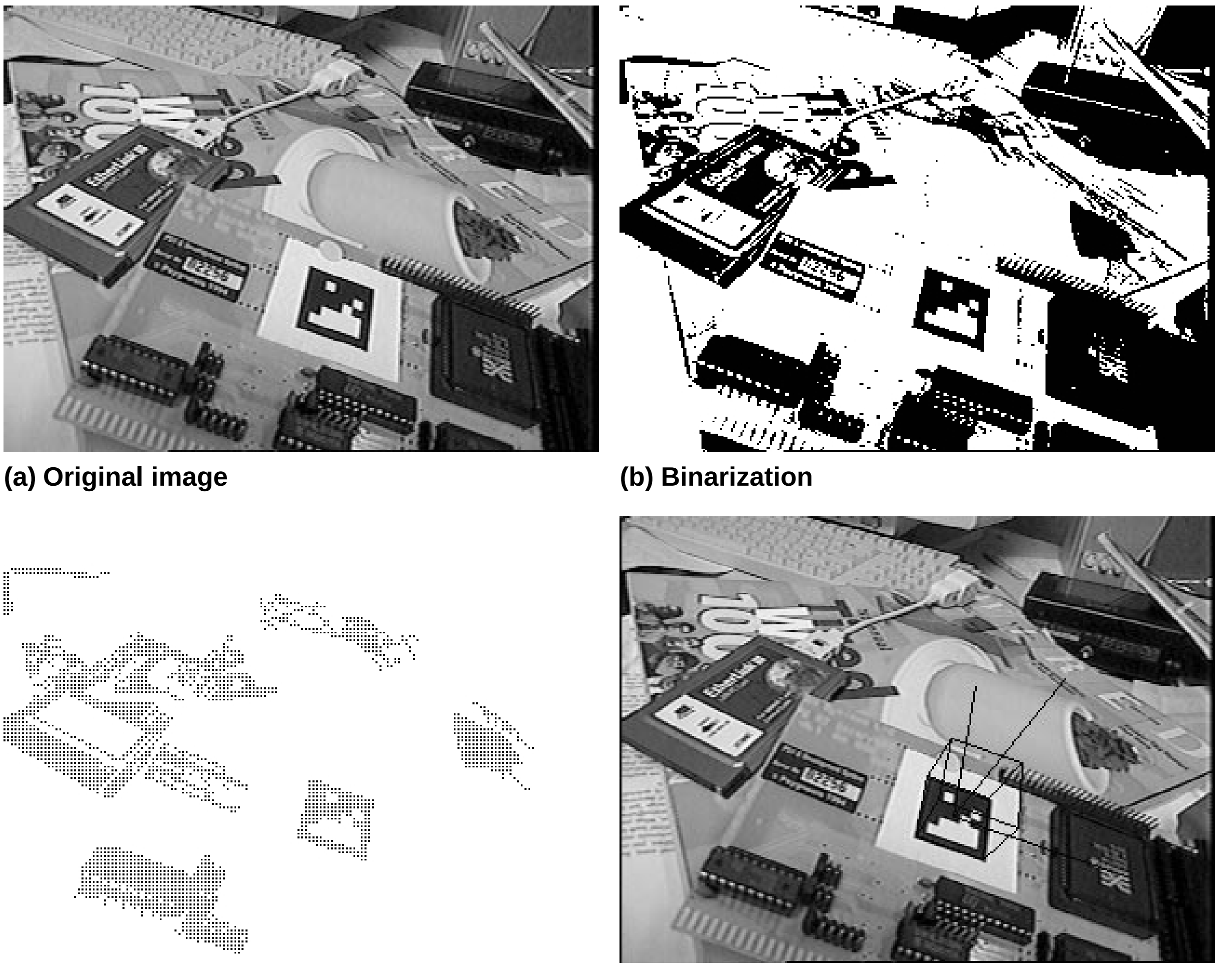
where $a_1, \ldots, a_8$ are parameters to be solved, representing the camera‘s intrinsic and extrinsic (motion and rotation) parameters. If we have four pairs of $(x_i, y_i, 0)$ and (X_i, Y_i), these parameters can be determined by solving:

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -X_1 x_1 & -X_1 y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -X_2 x_2 & -X_2 y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -X_3 x_3 & -X_3 y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -X_4 x_4 & -X_4 y_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -Y_1 x_1 & -Y_1 y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -Y_2 x_2 & -Y_2 y_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -Y_3 x_3 & -Y_3 y_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -Y_4 x_4 & -Y_4 y_4 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix}$$

This set of parameters restores the effect of rotation, motion and perspective transformation of the camera and maps distorted code image on the camera plane to the normalized matrix code space.

(4) Decoding and Error Check:

Using the obtained parameters, the system projects the corresponding image region to the code rectangle space



(a) Original image

(b) Binarization

(c) Connected components analysis

(d) Coordinate system estimation

Figure 3. The matrix code recognition process

(Figure 1, right). The number of black and white pixels that are projected within the area of each cell determine each cell bit. Finally, the CRC-error check is applied on the decoded bits, and the certificated cell value is regarded as a recognized code ID.

(5) Camera Position and Pose Estimation:

The recognized code frame is also used for camera pose estimation. From 4 known coplanar (real world) points on the image plane, it is possible to calculate a matrix representing the translation and rotation of the camera at a real-world coordinate. We use 4 corners of the 2D-code frame as these reference points.

Our method also tries to minimize the following constraint during estimation, to ensure an estimated coordinate system orthogonal (this method is partially based on the algorithm described in [6]):

$$E = (\vec{v}_0 \cdot \vec{v}_1)^2 + (\vec{v}_1 \cdot \vec{v}_2)^2 + (\vec{v}_2 \cdot \vec{v}_3)^2 + (\vec{v}_3 \cdot \vec{v}_0)^2 + (\vec{v}_4 \cdot \vec{v}_5)^2 \rightarrow min$$

where $\vec{v}_0 \ldots \vec{v}_3$ are the orientation vectors of the four edges, and $\vec{v}_4 \ldots \vec{v}_5$ are two diagonals of the code frame. Since we can represent $\vec{v}_0 \ldots \vec{v}_5$ by using $\vec{n}$, which is a normal vector of the matrix code plane, the above equation can be replaced by:

$$E(\vec{n}) \rightarrow min.$$

We use the downhill simplex method [3] to determine $\vec{n}$ that minimizes E. Once $\vec{n}$ is calculated, we can re-calculate vectors $\vec{v}_0 \ldots \vec{v}_5$ based on it. A point in the real world $(x, y, z)^T$ corresponds to the point in the camera coordinate system $(X, Y, Z)^T$:

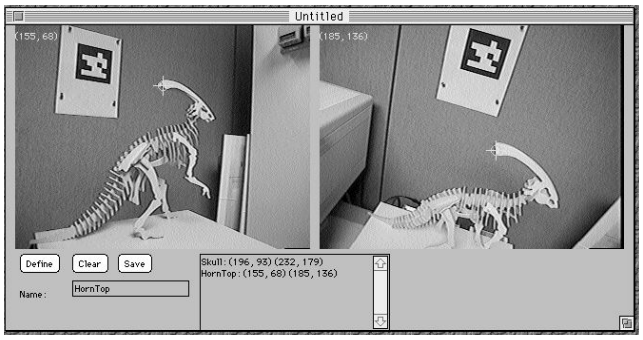
$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} & & & \\ \vec{e}_x & \vec{e}_y & \vec{e}_z & \vec{e}_t \\ & & & \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

where $\vec{e_x}, \vec{e_y}, \vec{e_z}$ represent camera rotation coefficients, and $\vec{e_t}$ represents camera motion, as:

$$\begin{aligned} \vec{e_x} &= N(\vec{v_5} - \vec{v_4}), \\ \vec{e_y} &= N(\vec{v_4} + \vec{v_5}), \\ \vec{e_z} &= N(\vec{n}), \\ \vec{e_t} &= dist \times N(\vec{p}), \end{aligned}$$

where $dist$ is the distance from the camera center to the center of the matrix code, $\vec{p}$ is a vector from the camera center to the center of the matrix code on the image plane, and $N(\vec{v})$ is a normalization function.

Once the transformation matrix become known, it is easy to overlay spatially correct annotation information, and computer graphics on the real world video images.



HornTop

Skull: (196, 93) (232, 179)
HornTop: (155, 68) (185, 136)

Figure 4. A Tool for defining annotating information

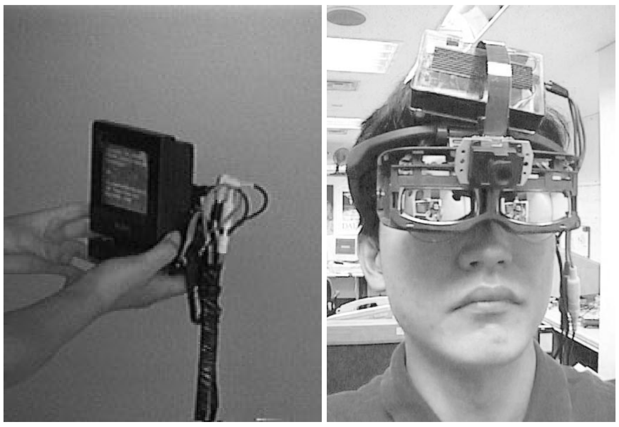
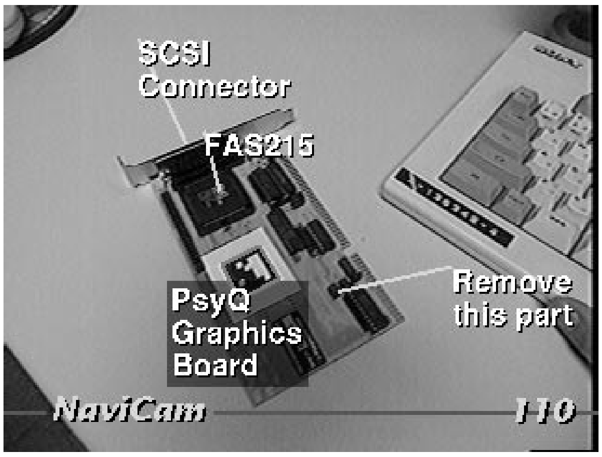


Figure 5. Hand-held and head-mounted configurations of video-based prototype AR systems

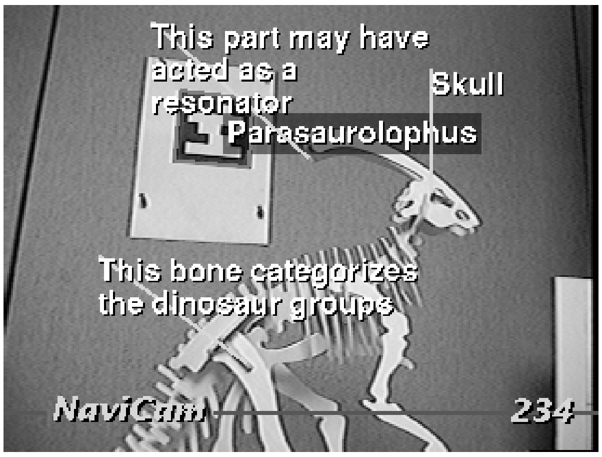
3. An Information Authoring Tool

Based on the method described in the previous section, a user can view real world objects with spatially registered digital annotation. However, from the system developer's point of view, defining such annotating information is not a trivial task. It requires accurate 3D position measurements, and doing it manually is often time consuming.

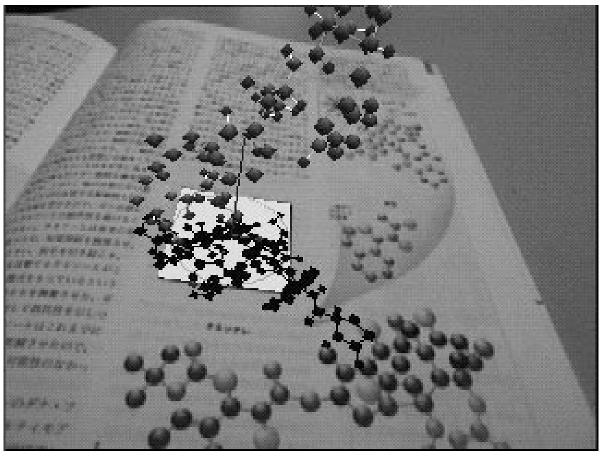
To deal with this situation, we have also developed an information authoring tool for our method. This tool determines the position of a 3D point in the real world based on the stereo vision method. The user first prepares two still images of the target object (both come with the same attached 2D matrix code in the view). Then the user clicks on the two corresponding points on the two images. The system recognizes the 3D coordinate systems of both of the images by using the 2D matrix, and determines the 3D position based on the pair-of 2D points. Figure 4 shows a



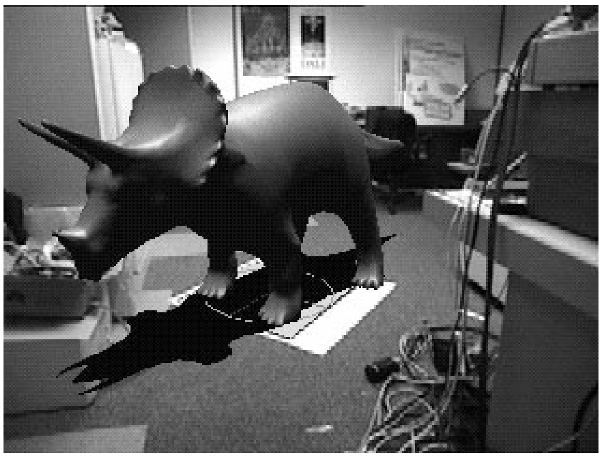
(a) An annotated circuit board


This part may have acted as a resonator
Skull
Parasaurolophus
This bone categorizes the dinosaur groups
NaviCam
234

(b) A dinosaur skeleton with
3D annotations



(c) A 3D molecular model pops-up
from the book



(d) Triceratops in the Laboratory
(The matrix code is spreaded on the floor)

Figure 6. Examples of Information Overlay

screen image of the authoring tool.

4. Results and Applications

We have implemented the above algorithm on top of the head-mounted and hand-held augmented reality systems [4], running on the SGI O2 workstation (MIPS-R10000 175 MHz). The average screen update rate was from about 15 Hz to 30 Hz, depending on the complexity of displaying information.

With the hand-held configuration, the user holds a hand-held device consisting of a palmtop display and a video camera (Figure 5), which provides a computer augmented view of the real world (i.e., video see-through). With the head-mounted configuration, the user wears the Sony GlassTron head-mounted display, with a miniature video camera (the user sees the real world as the video images). When the system identifies a real world object from the attached code, the corresponding 3D overlay information is retrieved from the database. Using the estimated camera position, this information can correctly be superimposed on the video image.

The 3D annotation data are stored on the local server, and when the user first encounters a new matrix code ID, the system automatically downloads the corresponding annotation information. We are currently using an internal 3D data format; however, it should be fairly a straightforward process to incorporate other standardized 3D graphics format such as VRML or DXF.

Figure 6 shows several examples of information overlay using 2D matrix codes. Examples include: Machine maintenance aid (Figure 6-a), a museum guide (b), paper document augmentation (c), and virtual sculptures superimposed on the real world (d). We are also planning to apply this technique to interior simulation, to enable the user to inspect a 3D furniture model superimposed on a real room image. It is also quite appealing to pick up an objet with a matrix marker (then annotation information appears on the user's view), and rotate it in your hands. The user can actually feel the virtual 3D graphics.

Our method can also be applied to 2D overlay applications. In particular, it is useful to link digital information to the printed materials. Koike and Kobayashi's EnhancedDesk [2] is an application based on this matrix code method. This system determines overlay information for paper on a desk because of on its printed 2D matrix code. The position and the orientation of the paper are also obtained by recognizing the code.

5. Conclusion

We have presented a new method for realizing an augmented reality system by using printed 2D matrix codes. This method relays only on a video images, without requiring other 3D sensors. We have also demonstrated several potential applications for our proposed method.

Acknowledgments

We are in debt to Mario Tokoro and Toshi Doi for their continuing support of our research.

References

[1] M. Bajura and U. Neumann. Dynamic registration correction in augmented-reality systems. In *Virtual Reality Annual International Symposium (VRAIS) '95*, pages 189–196, 1995.

[2] H. Koike and M. Kobayashi. Retrieving and manipulating digital information on EnhancedDesk. In *Proc. of APCHI '98*, 1998.

[3] W. Press, S. Teukoisky, W. Vetterling, and B. Flannery. *Numerical recipes in C*. Cambridge University Press, 1992.

[4] J. Rekimoto and K. Nagao. The world through the computer: Computer augmented interaction with real world environments. In *Proceedings of UIST'95*, pages 29–36, Nov. 1995.

[5] A. State, G. Hirota, D. T. Chen, W. F. Garrett, and M. A. Livingston. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *SIGGRAPH'96 Proceedings*, 1996.

[6] A. Takahashi, I. Ishii, H. Makino, and M. Nakashizuka. A high accuracy realtime 3D measuring method of marker fo VR interface by monocular vision. In *3D Image Conference '96*, pages 167–172, 1996. in Japanese.

[7] M. Uenohara and T. Kanade. Real-time vision based object registration for image overlay. *Journal of the Computers in Biology and Medicine*, pages 249–260, 1995.

[8] P. Wellner. Interacting with paper on the DigitalDesk. *Communication of the ACM*, 36(7):87–96, Aug. 1993.