TRICODES: A BARCODE-LIKE FIDUCIAL DESIGN FOR AUGMENTED REALITY MEDIA

Jonathan Mooser, Suya You, and Ulrich Neumann

CGIT Lab, University of Southern California
mooser@usc.edu, {suya,neumann}@graphics.usc.edu

ABSTRACT

Visual markers, or fiducials, have become one of the most common methods of camera pose estimation in Augmented Reality (AR) media. Many present day fiducial-based AR systems use arbitrary patterns, such as simple line drawings or alpha-numeric characters, and require that an application be "trained" to recognize its pattern set. These techniques work well on a small scale, but as the number of fiducials grows, accuracy and performance degrade. We describe a new fiducial design called TriCodes that, like a barcode, provides a systematic way of printing and identifying a vast library of patterns. We compare TriCodes to the popular ARToolkit package, demonstrating its advantages in the presence of large numbers of fiducials.

1. INTRODUCTION AND RELATED WORK

The principle behind fiducial-based AR is straightforward. A camera attached to the user captures the surrounding environment in real time. The system then uses the locations of fiducials within the captured image to estimate the position and orientation of the camera and, in turn, render virtual media on top of the real world. An early description of camera pose estimation based on identified image points is given by Fischler and Bolles [1]. A similar method is incorporated into the Zhang's camera calibration algorithm [2].

A number of specific fiducial designs and corresponding identification algorithms have been proposed [3, 4, 5]. In some cases, natural features of the environment can be incorporated into the pose estimation process [6] or visual markers may be combined with other tracking technologies [7]. While these methods go beyond the scope of purely fiducial-based AR, they depend on artificial landmarks to begin the process, and thus stand to benefit from improved fiducial identification.

Like the fiducial design described in this paper, a tagging system called Cybercode uses a barcode-like pattern library [8]. Its intended application, however, is somewhat different. Cybercodes are small thumbnail tags that need to be viewed from a close distance and occupy a relatively large portion of the image.

One of the most common frameworks for AR implementation is ARToolkit, an open source, freely downloadable API [9]. It is directly supported by the DART authoring environment [10] and it has been ported to PocketPC OS for handheld development [11]. ARToolkit uses rectangular markers consisting of arbitrary black-and-white or color patterns, with a training step required to teach the system to recognize a particular pattern set. The identification itself uses one of two algorithms. By default, ARToolkit uses a traditional template matching algorithm, which involves finding a correlation coefficient between a detected marker and each member of the training set. The detected marker is identified by the trained pattern with maximal correlation. This method is described by Ababsa and Mallem [4].

Alternately, ARToolkit can be set to use principal component analysis (PCA). This reduces the dimensionality of the training set to improve performance at the cost of accuracy. A description of PCA as applied to pattern recognition can be found in Turk and Pentland's work identifying images of faces and cars [12].

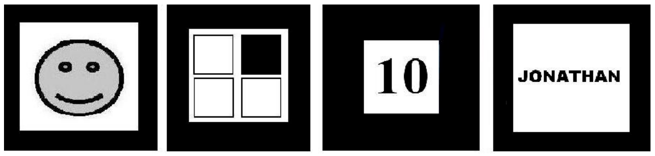

Fig. 1. Examples of markers we used with ARToolkit. These four patterns are the ones used in our experimental results (see Sec. 4), where we refer to them as "face," "squares," "10," and "Jonathan."

In recent years, there has been increasing demand for large scale, multi-user AR multimedia applications. Schmalsteig and Wagner describe an interactive museum guide [13, 14] in which virtual media is superimposed over real exhibits. It is not difficult, in an actual implementation, to imagine the need for thousands or tens of thousands of individually placed markers. Moreover, the fiducials set is necessarily dynamic. As exhibits are added and removed, one needs to create new patterns that are guaranteed not to interfere with the existing ones. Similarly, a multimedia training application for oil refinery employees was developed by Träskbäack and Haller [15]. As with the museum application, growing and changing training content will often mean adding and removing fiducials. Any AR application that annotates a large real-world environment needs the flexibility to continually grow and change.

2. THE TRICODE DESIGN

The motivation behind TriCodes is to facilitate the creation, organization, and identification of large, dynamic fiducial sets. Each new pattern represents a unique code built from a small alphabet of easily readable elements. Like ARToolkit markers, each TriCode is surrounded by a black frame for initial detection. The region inside that frame is then divided into a 3 by 3 grid. The upper-left cell always contains a solid black square, used for establishing the overall orientation of the marker. The remaining eight cells each contain a single code element: a solid black isosceles right triangle pointing in one of eight directions. We arbitrarily assign an element pointing up and to the right a value of 0, an element pointing straight up a value of 1 and so on counterclockwise until an element pointing due right is assigned a value of 7.

Because each element can take on any of eight values, it effectively encodes three bits. Two elements are set aside as a redundancy checksum, leaving six independent elements for a total of $6 \times 3 = 18$

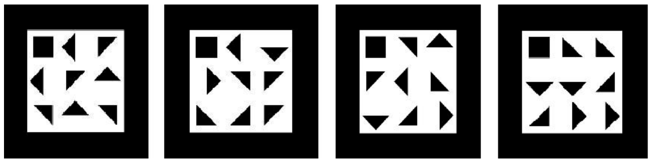

Fig. 2. Examples of TriCodes. The first, reading from the top line down after the square, has a value of 3-2-3-2-1-0 with a checksum of 1-0. The second has a value of 3-5-7-0-2-4 with a checksum of 6-2, etc.

bits for a complete "codespace" of $2^{18} \approx 260,000$ unique patterns.

The checksum is included to make the identification process robust to misidentification of individual elements. When a new TriCodes is created, the values of the last two elements are automatically generated from the first six in such a way that any two patterns differ by at least three elements. That means that a misidentification in one element will result in a code that still differs by two elements from any other possible code. Thus, when the system identifies a marker and compares its value to that of some known marker, we declare a positive identification as long as seven of the eight elements match.

3. DETECTION AND IDENTIFICATION PROCESS

Each captured image is initially scanned for any polygonal shape that may represent a valid TriCode (detection). Each candidate marker is then separately analyzed to attempt to read its code (identification).

The detection process is relatively similar to that used by ARToolkit [9]. The entire image is initially converted to grayscale and its average pixel intensity value is computed. We then create a bilevel version of the image by setting each pixel to 1 or 0 based whether or not its intensity is above or below that average (Fig. 3b). Because TriCodes consists of solid black regions against a white background, this kind of crude segmentation is sufficient. Next the perimeter of each contiguous region of black pixels is traced to produce a chain representing an object border (Fig. 3c). A set of line segments are then fit to these points, so that all border pixels are within some minimum distance of a line segment. If the resulting polygon is sufficiently large and has exactly four sides, it is considered a candidate marker and passed on to the identification process.

Identification begins by building a 48x48 pixel rectified mini-image of the interior of the detected polygon (Fig. 4a,b). The idea is to transform the polygon so that it looks as it would if it were being viewed from straight ahead. Assuming that the four corners of any marker in its own reference frame are at $(0,0)$, $(0,1)$, $(1,1)$, and $(0,0)$, object space and image space can be related by a homography:

$$\begin{bmatrix} \alpha u \\ \alpha v \\ \alpha \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where (x,y) is a point on the marker plane and (u,v) is its projection onto the image plane. A simple algorithm for deriving $\mathbf{H}$ is given by Fischler and Bolles [1]. We can now build the mini-image by iterating through its pixels, setting each to the value of the corresponding pixel in the original grayscale image. The pixel intensity values are then normalized to account for varying illumination conditions:

$$z_i = \frac{p_i - \mu_p}{\sigma_p}$$

where z_i is the normalized intensity value of the i^{th} pixel, p_i its original value, μ_p the average intensity of the entire mini-image, and σ_p the standard deviation.

If a candidate marker is, in fact, a TriCode, we know the approximate location of each element within the rectified mini-image. Like the original pattern itself, we can think of the mini-image as a grid of nine cells, each containing exactly one code element. The first step is to rotate the mini-image until the solid black square appears in the upper left corner (Fig. 4a,b). Its presence is verified by summing the intensity values of pixels in the upper left cell weighted by 3-pixel Gaussian around its center. After normalization, the average intensity value is 0, so if the upper left cell contains the black square, most of the pixels will be negative, as will be their weighted sum. Through experimentation, we have found that a safe threshold is -0.5. Any sum greater than that indicates the upper left cell of the mini-image does not contain a solid black square, in which case, the mini-image is rotated 90° and re-tested. If, after three rotations, the square cannot be found, the polygon under consideration is considered unidentifiable as a TriCode and is discarded.

The remaining eight regions of the mini-image are now analyzed. The hypotenuse of each triangular element always passes through the center of its cell, so we apply an orientation filter based on a Canny edge detector [16] at that point. This is a somewhat unique application of an edge detector; rather than processing an entire image searching for edges, there are just a few precise locations where the intensity gradient is measured. Specifically, we measure the gradient at a point, (x,y), on the image, I.

$$\begin{aligned} \nabla I_x &= \sum [G_x(i,j) I(x+i, y+j)] \\ \nabla I_y &= \sum [G_y(i,j) I(x+i, y+j)] \end{aligned}$$

Where the sums are taken over the range $-3 \leq i,j \leq 3$. G_x and G_y are gradient-of-Gaussian filters in the x and y directions.

$$\begin{aligned} G_x(i,j) &= G(i+0.5, j) - G(i-0.5, j) \\ G_y(i,j) &= G(i, j+0.5) - G(i, j-0.5) \\ G(i,j) &= \frac{1}{2\pi\sigma^2} e^{-(i^2+j^2)/2\sigma^2} \end{aligned}$$

We convert ∇I_x and ∇I_y to an orientation and magnitude

$$\begin{aligned} |\nabla I| &= \sqrt{\nabla I_x^2 + \nabla I_y^2} \\ \nabla I_\theta &= \arctan\left(\frac{\nabla I_y}{\nabla I_x}\right) \end{aligned}$$

The value of the gradient magnitude is taken to be the maximum over a small window around the center of a given cell, (x_0, y_0).

$$|\nabla I| = \max_{-1 \leq i,j \leq 1} (|\nabla I(x_0+i, y_0+j)|)$$

So for each element, the gradient filter is applied nine times. If $|\nabla I|$ is below some minimum threshold, no edge has been detected in the current cell and its element cannot be identified. Identification is robust to at least one unreadable element, as explained in Section (2), so this may not render the overall code unreadable. If $|\nabla I|$ is sufficiently large, the element is assigned a value based on the value of ∇I_θ. Whichever of the eight possible orientations is closest to ∇I_θ is taken to be the orientation of the element under investigation. After all eight elements have been read, their values are concatenated to produce the complete code.

Knowing the internal parameters of the camera, we can derive the camera pose and projection matrix from the same homography

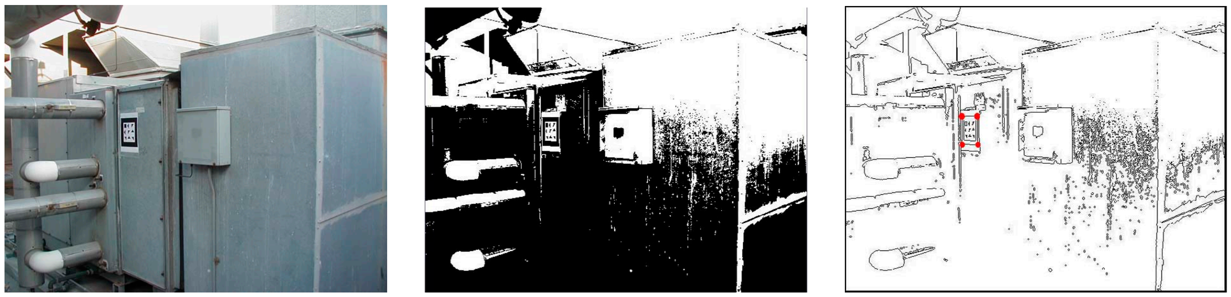

Fig. 3. (a) An input image of an exterior industrial setting, with a TriCode placed amongst other objects. The purpose here is to provide real-time data for AC ducts. (b) After conversion to bilevel. (c) Edges are marked as the boundaries of contiguous black areas. One chain of edges has its corners marked, indicating that it is the right shape and size to be passed to the identification process.

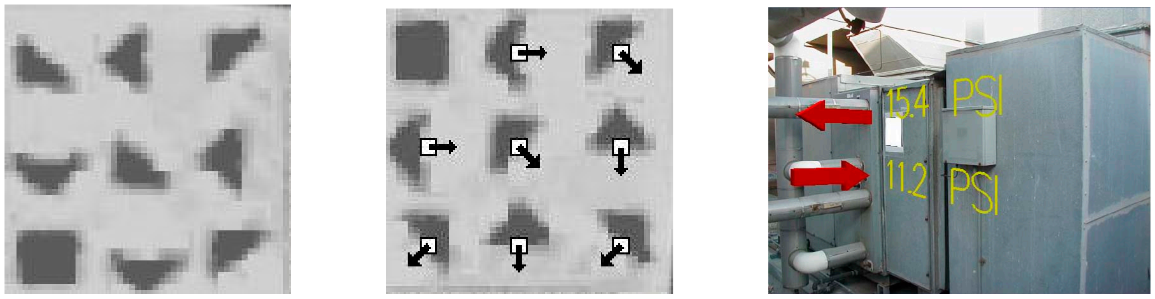

Fig. 4. (a) The rectified mini-image before the black box has been rotated into the upper left corner. (b) After rotation. Markers indicate the centers of the orientation filters and measured orientations. (c) Using the identification and the marker's corner positions, the system can render the correct virtual media content in the desired location.

H used to create the rectified mini-image [2]. The system can now accurately render virtual media content on top of its image of the real world (Fig. 4c).

4. COMPARISON TO TEMPLATE MATCHING AND PCA

TriCodes offer several key advantages over template matching and PCA based pattern recognition. The first is one of performance. TriCode identification as described in Sec. 3 can be performed in constant time. Eight elements need to be read every time, each requiring exactly nine applications of a fixed size gradient filter. The database of known codes can be effciently indexed by their 18-bit values, and a newly identified code can thus be retrieved quickly.

This is not the case with template matching, where each identification effectively requires the computation of a large dot product for each pattern in the training set. Thus if there are N trained fiducials, identification takes $O(N)$ time. For small fiducial sets, identification will still generally run at interactive speeds, but as the set grows, there will necessarily be some point at which identification time grows longer than desired. In the case of PCA recognition, the situation is more complicated. Time complexity is not a function of the number fiducials, but of the number of eigenvectors, which may be fixed or variable depending on implementation. If it is fixed, then identification can be completed in constant time like TriCodes. This, however, comes at a considerable cost in reliability, as our experimental results show.

We compared the accuracy of TriCode identification to template matching and PCA as implemented by ARToolkit (ver. 2.71.2). Accuracy here is defined as the fraction of times that a system returns the correct identification of a marker that it detects in an image. For TriCodes, a positive identification means that at most one of the eight elements is misread. For template matching or PCA, it means that the system found the correct identification to be more likely than any other possible identification. We tested these algorithms by running ARToolkit on the four fiducials in Fig. 1 amidst an increasingly large training set (up to 60 total patterns). For PCA tests we set a maximum of 20 eigenvectors. In all tests, the camera was positioned so that the fidcucial had a diagonal length between 80 and 100 pixels. Given the size of the markers and the focal length of the camera, this corresponds to a distance of approximately one meter. The angle between the marker plane and the image plane was confined to a range of 45° to 60°.

Under these test conditions, our algorithm correctly identified TriCodes with over 99.5% accuracy. The identifability a given TriCode fiducial is unaffected by the existence of other fiducials, so this values remains constant as the database grows. Template matching and PCA exhibited strong accuracy as well when using a relatively small training set of 10 patterns, (Figs. 5 and 6). However, introducing more fiducials into the training set increased the likelihood of interference between similar patterns, and accuracy dropped significantly. Using PCA identification, some patterns became almost completely unidentifiable. The exact accuracy values that we saw in our results are dependent on a number of factors, including camera resolution, illumination conditions, and the precise design of the patterns in the training set. It may be possible, with the right choice of fiducials, to see good accuracy with 60 patterns or even more. But this is ultimately unpredictable. Our results indicate that two fiducials may give vastly different results under identical test conditions. Thus, without extensive testing we do not know which patterns will interfere with each other and compromise accuracy. One of the key motivations behind TriCodes is to provide a consistent way of adding new fiducials to an existing application without this risk.

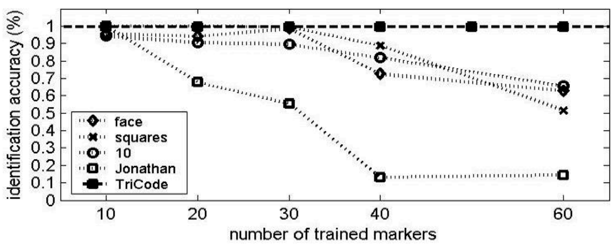

Fig. 5. **TriCodes vs. ARToolkit using template matching.** The results show the identification accuracy of four patterns (Fig. 1) as the total number of trained markers increases.

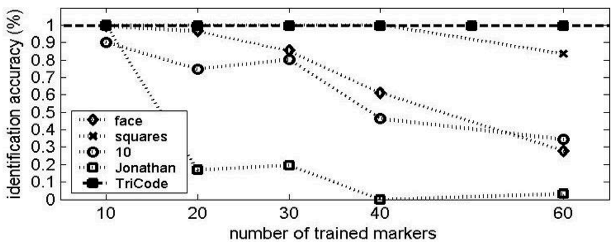

Fig. 6. **TriCodes vs. ARToolkit using PCA recognition.**

5. CONCLUSION

In this paper, we present a novel fiducial design that can easily be generated on a black-and-white printer. As AR applications grow, so will the need for large, dynamic sets of fiducials. Authors and developers working on different parts of a large-scale application will want to independently create new fiducials that are guaranteed not to interfere. TriCodes are specifically designed to address that need, eliminating the process of drawing patterns and testing them for possible interference. The identification process is fast, and is shown to be more accurate than many prevalent fiducial recognition algorithms.

6. ACKNOWLEGEMENTS

This study was funded by the Center of Excellence for Research and Academic Training on Interactive Smart Oilfield Technologies (CiSoft); CiSoft is a joint University of Southern California-Chevron initiative.

7. REFERENCES

[1] Martin A. Fischler and Robert C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381 – 395, 1981.

[2] Zhengyou Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330 – 1334, 2000.

[3] Ulrich Neumann, Suya You, Youngkwan Cho, Jongweon Lee, and Jun Park, "Augmented reality tracking in natural environments," in *International Symposium on Mixed Realities*, 1999.

[4] Fakhr-eddine Ababsa and Malik Mallem, "Robust camera pose estimation using 2d fiducials tracking for real-time augmented reality systems," in *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, 2004, pp. 431–435.

[5] David Claus and Andrew W. Fitzgibbon, "Reliable fiducial detection in natural scenes," in *ECCV 2004*, 2004, pp. 469 – 480.

[6] Bolan Jiang, Suya You, , and Ulrich Neumann, "Camera tracking for aumented reality media," in *IEEE International Conference on Multimedia and Expo 2000*, Jul 2000, pp. 1637 – 1640.

[7] Andrei State, Gentaro Hirota, David T. Chen, William F. Garrett, and Mark A. Livingston, "Superior augmented reality registration by integrating landmark tracking and magnetic tracking," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 429 – 438.

[8] Jun Rekimoto and Yuji Ayatsuka, "CyberCode: designing augmented reality environments with visual tags," in *DARE '00: Proceedings of DARE 2000 on Designing augmented reality environments*. 2000, pp. 1–10, ACM Press.

[9] Hirokazu Kato, "www.hitl.washington.edu/artoolkit," 2005.

[10] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter, "DART: a toolkit for rapid design exploration of augmented reality experiences," in *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, 2004, pp. 197 – 206.

[11] Daniel Wagner and Dieter Schmalstieg, "ARToolkit on the PocketPC platform," Tech. Rep., Vienna University of Technology, 2003.

[12] Matthew Turk and Alex Pentland, "Eigenfaces for recognition," *Journal of Conitive Neuroscience*, vol. 3, no. 1, pp. 71 – 86, 1991.

[13] Dieter Schmalstieg and Daniel Wagner, "A handheld augmented reality museum guide," in *Proceedings of IADIS International Conference on Mobile Learning 2005 (ML2005)*, June 2005.

[14] Daniel Wagner, Thomas Pintaric, Florian Ledermann, and Dieter Schmalstieg, "Towards massively multi-user augmented reality on handheld devices," in *Proceedings of Third International Conference on Pervasive Computing, Pervasive 2005*, May 2005.

[15] Marjaana Träskbäack and Michael Haller, "Mixed reality training application for an oil refinery: user requirements," in *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, 2004, pp. 324 – 327.

[16] J Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679 – 698, 1986.