Using ARToolKit for 3D Hand Position Tracking in Mobile Outdoor Environments

Wayne Piekarski and Bruce H. Thomas
Wearable Computer Laboratory
School of Computer and Information Science
University of South Australia
Mawson Lakes, SA, 5095, Australia
{wayne, thomas}@cs.unisa.edu.au

Abstract

This paper describes how we have used the ARToolKit to perform three degree of freedom tracking of the hands, in world coordinates, which is used to interact with a mobile outdoor augmented reality computer. Since ARToolKit generates matrices in camera coordinates, if errors occur during the calibration process, it is difficult to extract out real world coordinates. We discuss the problem of making ARToolKit generate world coordinates, and the solutions we developed to meet the requirements for our tracking system.

1 Introduction

We have been performing research into mobile outdoor augmented reality, with examples such as the Tinmith-Metro application [2] shown in Figure 1. In these applications, we have been developing new user interaction techniques, as traditional desktop devices such as mice and keyboards do not work well in mobile outdoor environments. In order to interact with the complex modelling features of Tinmith-Metro, we have developed a glove which is the primary input device for the system. Unfortunately, traditional tracking technology, such as magnetic, ultrasonic, and acoustic systems, are designed for indoor use and are not suitable for placement onto a mobile outdoor backpack. To solve these problems, we have developed a cheap and low cost 3 DOF (degrees of freedom) hand tracker based on the ARToolKit libraries, allowing us to use a video camera mounted on the head to track the motion of the hands of the user as real world X, Y, Z values.

The ARToolKit libraries [1] were developed to support the tracking of simple fiducial markers, allowing applications to appear to place 3D objects onto these markers, and then viewed on a display device – an example is the ARToolKit *simpleTest* application. It should be noted that while the ARToolKit generates 6DOF matrices for each fiducial marker that are sent to OpenGL, these matrices are relative to a special distorted camera frustum model, and not usually in real world coordinates. If the calibration model for the camera is not perfect (the ARToolKit calibration process does not always generate good results) then extra errors are introduced into the results and it is unusable for tracking.

For more information, as well as videos of the system in use outdoors, readers are invited to visit our web site at http://www.tinmith.net

This paper discusses problems with calibration and getting accurate results for use as a tracker, and then discusses the solutions that we used to make the tracker work to our requirements.

2 Gloves and video cameras

We use a set of custom designed gloves to control the system, using metallic contacts on the fingertips, thumb, and palm to detect finger presses. We created two ARToolKit fiducial markers, each 2cm x 2cm in diameter, and glued them to the thumbs of the glove (see Figure 1). We placed the markers on the thumb because this allows us to perform finger presses without moving the position of the marker. The size of the markers is the most appropriate size given that the position tracking is always within arms reach of the head mounted camera.

We use a PGR Firefly camera mounted on the user's head to capture the video and then pass it to ARToolKit. The camera has good dynamic range and can capture the fiducial markers in direct sunlight, at sunset, and in complete darkness with an extra light on the camera. Since the tracker operates in real world coordinates, it is possible to mount the camera anywhere appropriate on the body and transform the coordinates using a scene graph. The camera used for tracking does not have to be the same as that used for the AR overlay, and allows users watching from external VR views to see the locations of the cursors floating in front of the user's avatar.

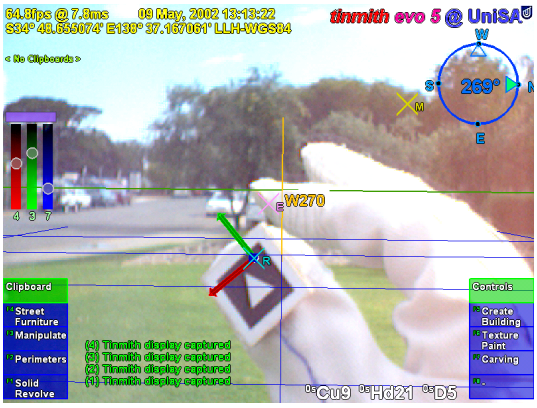

Figure 1 – Tinmith immersive augmented reality display, with gloves and 2 cm² fiducial markers, and 3D overlaid red(X) / green(Y) / blue(Z) cursor axes

In ART02, 1st International Augmented Reality Toolkit Workshop
September 29, 2002 - Darmstadt, Germany - Copyright (C) 2002 IEEE
Please visit http://www.tinmith.net for more information

3 Camera calibration

In applications which completely use the ARToolKit (such as *simpleTest*), the video is first captured by *libARvideo*. Next, recognition of fiducial markers and calculation of camera-space transformations is performed in *libAR*, which is then used to render the final scene using the camera calibration frustum in *libARgsub*.

3.1 ARToolKit internal operation

When a pattern is recognised, a matrix transformation is returned from *arGetTransMat()*, which defines the marker's position and orientation (6DOF) relative to the camera, in the camera's calibrated coordinate system. The camera calibration data is used by this function to modify the results to compensate for the camera's unique properties. Due to distortions in the camera, and errors in the calibration process, the coordinate system is not the typical orthogonal coordinate system normally used in 3D graphics.

If this camera-space matrix is used to draw an object with *libARgsub*, the view frustum used in OpenGL will be the same as that of the camera model, and so the image will appear to be rendered at the correct location. An interesting side effect of these transformations is that no matter how poor the camera calibration is, the 3D object overlaid on the fiducial marker will always be correct since the incorrect calibration model used in *arGetTransMat()* is reversed when drawing using the camera as the view frustum in *libARgsub*.

When we tried to take the matrix calculated in *arGetTransMat()* and put it into a scene graph, errors occurred in the results for two reasons. The first was that the camera calibration was not being used to render the display and compensate for any camera distortions. Secondly, because the calibration was not good enough and did not model the camera accurately, extra errors were introduced.

After calibration of a camera, ARToolKit generates information for the camera, such as Figure 3. The values which define the coordinate system of the camera (and we also found introduces the most errors) were the X and Y values for the centre pixel of the camera. In the default supplied file, the Y axis is 48 pixels from the centre of the camera, and under testing we found most cameras had centre's that were reasonably in the middle. This was an error introduced during the calibration process which we are correcting in this paper.

3.2 Calibration solution

We take the matrix which is generated in the ARToolKit calibration procedure (such as Figure 3) and read it in from a binary file, with the layout shown in Figure 2. We only want to adjust the centre point and leave the focal point and scale values untouched because they are reasonably correct and less noticeable. Figure 4 shows the calculations needed to create an orthogonal camera model with the centre of the image being the axis of the camera (see Figure 5).

This technique is only useful for correcting errors caused by the ARToolKit calibration on cameras which have the centre of the image at approximately the axis of the camera. In distorted cameras, this technique could produce results which are worse than the uncorrected version, and so should be used with care and the results carefully monitored to ensure that it is being used in an appropriate fashion.

```
/* (a) Actual struct */
typedef struct {
  int xsize, ysize;
  double mat[3][4];
  double dist_factor[4];
} ARParam;
```

```
/* (b) Components explained */
typedef struct {
  int cam_width, cam_height;
  double matrix [3][4];
  double centre_x, centre_y;
  double focal, size;
} ARParam;
```

Figure 2 – (a) Actual C code definition for ARParam camera distortion model (b) Cleaned up C structure which has easier to understand components

```
camera = (640, 480)        | 780.54   0.54 304.64 0.00 |
centre = (317.5, 192.0)    |   0.00 762.30 208.68 0.00 |
focal  = 26.300000         |   0.00   0.00   1.00 0.00 |
size   = 1.009989          |   0.00   0.00   0.00 1.00 |
```

Figure 3 – ARToolKit default camera_para.dat file, with error x=2.5, y=48.0

```
centre_x = cam_width / 2.0;
centre_y = cam_height / 2.0;
matrix[0][2] = cam_width / 2.0;
matrix[1][2] = cam_height / 2.0;
```

Figure 4 – Pseudocode to make axes of input camera model orthogonal, forcing it to match those of a perfectly symmetrical camera model

```
camera = (640, 480)        | 780.54   0.54 320.00 0.00 |
centre = (320.0, 240.0)    |   0.00 762.30 240.00 0.00 |
focal  = 26.300000         |   0.00   0.00   1.00 0.00 |
size   = 1.009989          |   0.00   0.00   0.00 1.00 |
```

Figure 5 – Corrected version of camera_para.dat, with clean orthogonal axes, suitable for use as a tracker without using the camera model as the frustum

3.3 Conclusion

When used for 3D position tracking the results of the hand tracker are quite good and the registration is within the area of the target. However, detecting 3D rotation is more error prone since large changes in orientation only give small changes in perspective on the image, resulting in very jittery output in the order of 20-30 degrees. As a result, we currently only use this tracker mainly for position tracking, using it as a 3D cursor on the HMD. By combining two cursors it is possible to derive a high quality rotation using their relative positions.

Using ARToolKit as a general purpose tracker with some adjustments has allowed us to integrate hand tracking into our outdoor modelling applications. This opens up a number of new areas of applications for ARToolKit.

4 References

[1] Kato, H. and Billinghurst, M. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In *2nd Int'l Workshop on Augmented Reality,* pp 85-94, San Francisco, Ca, Oct 1999.

[2] Piekarski, W. and Thomas, B. Tinmith-Metro: New Outdoor Techniques for Creating City Models with an Augmented Reality Wearable Computer. In *5th Int'l Symposium on Wearable Computers,* pp 31-38, Zurich, Switzerland, Oct 2001.