Vision, Video, and Graphics (2003), pp. 1–8
P. Hall, P. Willis (Editors)

A Vision-Based Location System Using Fiducials

David J. Johnston and Adrian F. Clark

VASE Laboratory, Department of Electronic Systems Engineering
University of Essex, Colchester, CO4 3SQ, UK
{djjohn,alien}@essex.ac.uk

Abstract

A system for vision-based ego location using 'targets' or 'fiducials' is described. The system is robust and operates on commodity hardware in real time. The accuracy of the system is assessed and found to be good enough to support some augmented reality applications. Two example applications are described: the control of an avatar in a shared virtual environment and a video 'joystick' for manipulating 3D models.

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [Image Processing and Computer Vision]: Scene Analysis

1. Introduction

Augmented Reality (AR) involves mixing real-world and computer-generated imagery seamlessly. For this to be possible, the position and orientation of the camera within the real scene must be known, as must some properties of the camera. Estimating the position and orientation of the camera is a six degrees-of-freedom ("6-dof") problem. In the motion picture industry, where this type of mixing is most widely used, the traditional approach is to instrument the (live action) camera, then make the virtual camera mimic the motions (and zoom changes). Latterly however, computer vision has been exploited to estimate this information from the imagery, though there are some restrictions on camera motion, and performance is normally non-real-time.

The motivation for this research is the production of a *real-time* augmented reality system that runs on commodity hardware (*i.e.*, PC-class machines) and allows the user to roam over a comparatively wide area, up to several hundred metres — what we might call *untethered* AR. Untethered operation can be achieved through the use of wearable computers and wireless networks; but the problem of finding the user's position and orientation remains.[1]

Vision is attractive for this 6-dof position estimation problem because it is inherently untethered. Hence, an obvious approach is to estimate the observer's position relative to known, fixed points. Computer vision is not yet able to cope robustly with arbitrary objects, so the best approach available is to instrument the scene at known places with specially-designed "targets" amenable to real-time processing. These targets are often known as *fiducials*. This paper describes a system that the authors have devised that combines fiducials and appropriate vision processing, and achieves real-time performance. The system, which has been in regular operation for four years, is known as the *vision positioning system* or VPS.

The mathematics that underlie VPS are fairly well-established[2]; yet most published work has concentrated on the pose estimation phase, *i.e.* the geometrical aspects of the problem; there has been little published work on the considerations that allow the principles to be translated into practice. Most important among these is the combination of fiducial design and associated image processing, so this paper emphasises these issues.

There are two well-known systems that are similar to VPS. One was devised by the BBC[3] as an intrinsic part of a virtual studio, in which actors perform in an empty studio and the set is chroma-keyed into place. This has now become a commercial product. These fiducials take the form of concentric black and white circles, each pattern of circles encoding a unique value. This design facilitates real-time processing and permits the centres of the circles to be found accurately. The fiducials are suspended at differing heights from the ceiling of the studio, and broadcast TV cameras have mounted on them upward-pointing cameras that feed

Figure 1: *A room 'instrumented' with ceiling-mounted VPS fiducials*

the positioning system. With care, an accuracy of a few millimetres is achievable.

The other VPS-like system is the *AR Toolkit* from the University of Washington[4]. This system, which was developed at roughly the same time as VPS, uses black patterns (originally Kanji characters) on white cards and works at near real-time on SGI and PC hardware.

The next section considers the design of fiducials. Section 3 then describes how these established *desiderata* are instantiated in the VPS fiducials. Section 4 gives an overview of the processing performed by VPS. Section 5 presents an overview of the estimation of position from fiducial positions, while Section 6 presents results on the accuracy that has been achieved. Example applications of the system are outlined in Section 7 and Section 8 draws conclusions.

2. Fiducial *desiderata*

If one is attempting to determine one's position relative to a set of fiducials at known positions (Figure 1) in real time, this immediately imposes rather stringent requirements:

1. It must be possible for vision-based processing to identify fiducials unambiguously. False alarms, even for as little as a single frame, cause the AR system to produce unacceptable results and hence the false alarm rate must be zero. (This is usually very difficult to achieve with a computer vision system.)
2. Each fiducial must be unique in order to calculate position and orientation correctly. It is helpful, but not essential, for a fiducial to encode a number.
3. The information encoded on the fiducial must be amenable to real-time image processing from a range of angles of observation.
4. It should be possible to estimate position and orientation on a frame-by-frame basis, *i.e.* without any requirement for temporal processing.
5. Fiducials should be inexpensive to produce: the authors' research lab employs well over 150 of them, for example.
6. Monochrome patterns facilitate the use of chroma-keying, while grey-levels in patterns are best avoided because of problems introduced by the automatic gain controls of cameras.
7. Finally, it is helpful if a single fiducial can yield more than one known position: this reduces the number that must be visible to determine position and orientation or, for a given number of visible fiducials, improves accuracy.

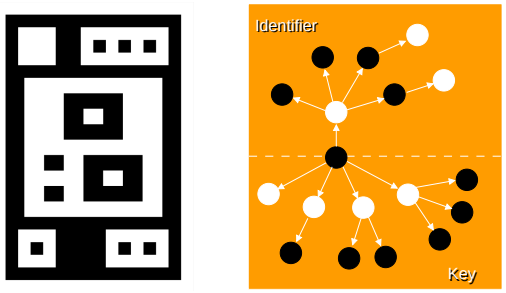
Of these, only the first two are absolutely necessary; the other features enhance their practical value. As we shall see, the VPS fiducial design scores on each of these points.

3. The VPS fiducial design

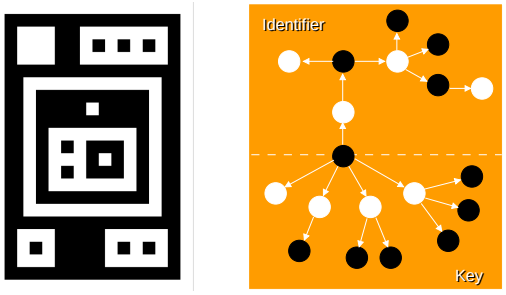
The authors' first exploration into pose estimation using fiducials employed circular bar-codes and spirals. However, processing these types of patterns required a combination of edge- and region-based vision techniques, which was slow and lacked robustness. It was found that region-based processing was generally the more reliable approach, so the decision was taken that the VPS fiducials should be entirely region-based.

Bearing in mind this point and the *desiderata* outlined above, VPS fiducials comprise a set of black and white regions. These regions are normally rectangular though, as we shall see, non-rectangular regions can be employed equally effectively. Although there are some subtleties involved, what VPS (Section 4) does is form the region-adjacency graph (RAG) of the image; this is a fairly simple image processing operation, easily achievable in real time.[5] The RAG is then searched to find any fiducials.

Examples of two VPS fiducials and their corresponding RAGs are shown in Figure 2. It can be seen that the outer



(a) Fiducial 22



(b) Fiducial 294

Figure 2: *Examples of fiducials and their RAGs*

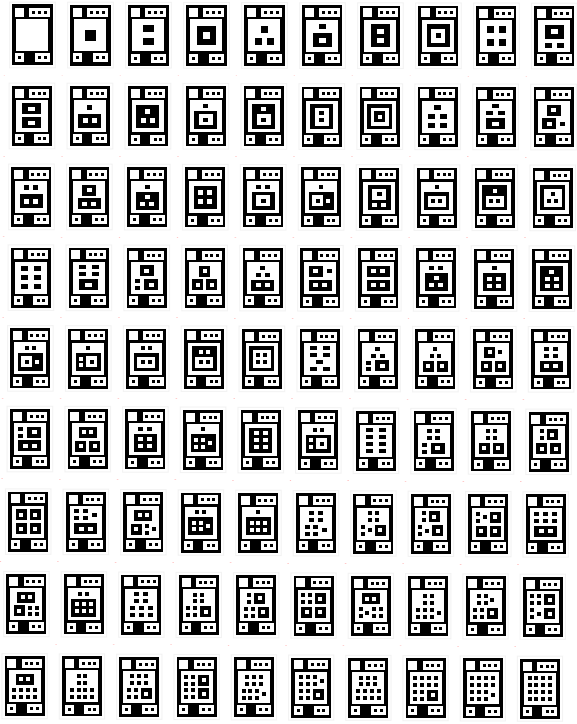


Figure 3: *The complete set of targets for a* 9×9 *grid*

part of every fiducial contains the same pattern of regions, termed the *key*. This consists of a black region, inside which are four white regions containing zero, one, two and three black regions respectively. This pattern is sufficiently unusual that it does not occur in normal scenes — unlike, say, patterns of concentric circles. Indeed, in roughly four years of operation within and outside the research laboratory, VPS has *never* experienced a false alarm. A further advantage of this scheme is that each fiducial provides *four* locations (the parts of the key) that can be used in the estimation process.

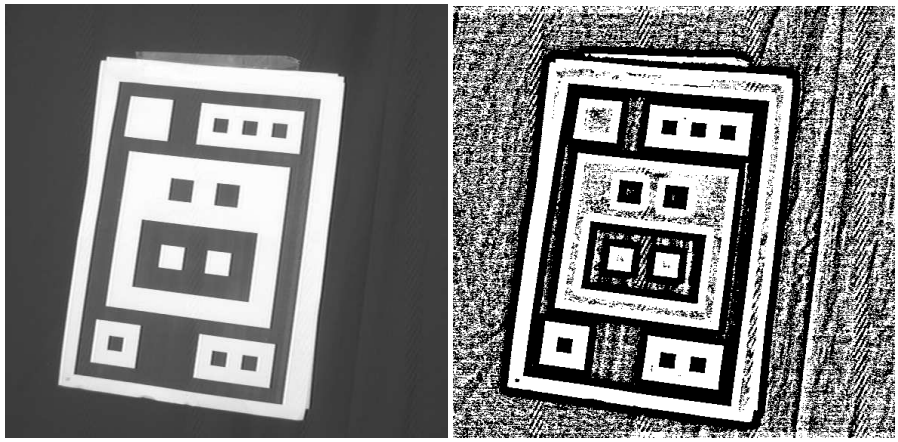
The central part of each fiducial encodes a unique number in its RAG. In principle, an arbitrarily large set of distinguishable RAGs can be produced, simply by recursively subdividing each region. There is, of course, a practical limit to the number of sub-divisions that can be achieved; the authors normally work with an 11×11 grid which yields in excess of 200 distinct targets (those in Figure 2 are from this set). However, to give the reader an idea of what is achievable, the entire set of 9×9 targets is presented in Figure 3.

The targets are actually generated as graphs, rather than images, and then rendered. One rendering is as an image, usually in the form of POSTSCRIPT. These rendered images are then printed onto A4-sized paper using a laser printer, making for inexpensive targets. The second rendering is as a C structure (see Figure 4), which ensures that the printed targets match the program representation.

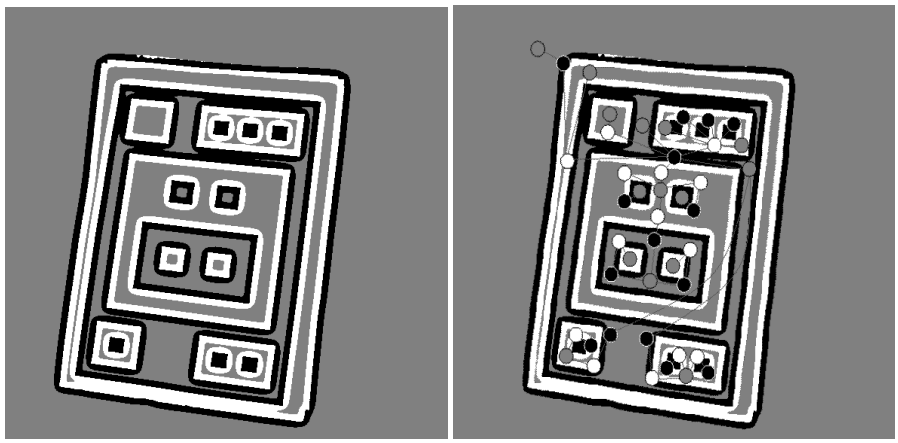
```
typedef struct entry {
  int page;
  char *ascii_graph;
} PageEntry;

PageEntry Pages[] =  {
  {0,"()"},
  {1,"(())"},
  {2,"(()())"},
  {3,"((()))"},
  . . .
  {304,"(((()()()()()))()())"},
  {305,"(((()()()()())()()))"}
};
```

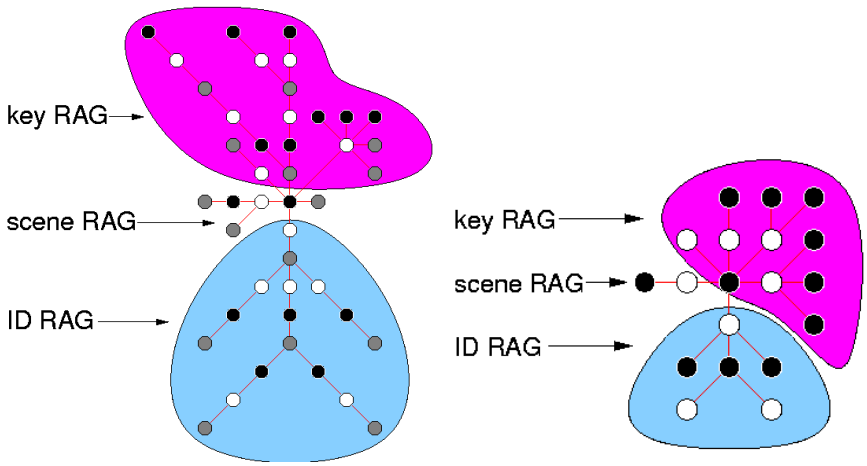
Figure 4: *Rendering VPS RAGs as a C structure*



(a) Original image (b) Result of bi-level thresholding



(c) Result of tri-level thresholding (d) RAG superimposed on processed image



(e) RAG with rationalised layout (f) RAG with grey nodes removed

Figure 5: *The processing of fiducials*

4. Processing images of VPS fiducials

The image processing procedure essentially comes down to converting an original grey-scale image of a fiducial (Figure 5a) to its corresponding RAG, then matching that RAG against one of a known target. However, there are a number of subtleties, expounded in this section. Firstly, thresholding into black and white using a fixed threshold produces poor results (Figure 5b) because illumination typically varies within an image. This suggests the use of some form of adaptive filtering. The particular scheme, believed to be novel, that has been found most effective is

$$o(x,y) = \begin{cases} 0 & \text{(black)}, \quad i(x,y) < \mu - \alpha\sigma \\ 2 & \text{(white)}, \quad i(x,y) > \mu + \alpha\sigma \\ 1 & \text{(grey)}, \quad \text{otherwise} \end{cases}$$

where $i(x,y)$ is the value of the pixel indexed by (x,y), $o(x,y)$ is the resulting filtered pixel, μ and σ are the mean and standard deviation respectively of the region around (x,y), and α is a threshold. This splits the image into regions that are definitely dark and light, leaving the rest — those pixels with values close to the local mean — marked as ambiguous (Figure 5c).

The next step is to assign different labels to black, white and grey regions. This process, called variously 'connected component labelling' and 'blob-finding' in the literature, is well-established[5]. The RAG describing black, white and grey connected regions is then constructed in a similar way to the connected components themselves (Figure 5d and e). This graph is then 'slimmed' as follows:

- merge all white neighbours of each grey area with each other;
- merge all black neighbours of each grey area with each other;
- remove grey areas from the RAG (Figure 5f).

These operations are performed on the graph, not on the image, and hence are rapid.

Having obtained a black-and-white graph of the entire scene in this way, one must then locate those sub-graphs that correspond to fiducials. This is achieved by searching for the key RAG (Section 3). (In fact, these RAGs are actually trees; nevertheless, standard graph-searching techniques can be used.) Having located the key RAG, the fiducial's unique identifier lies in an adjacent graph connected via a black node.

Although the image processing stage involves a number of steps, each of them is simple. The robustness of the scheme is evidenced by the fact that, in over four years of operation in many different environments, no pattern in the surroundings has been mistaken for a VPS target. Moreover, despite having been designed for rectangular fiducials, the RAG-processing scheme is sufficiently general that it works with other layouts too. For example, Figure 6 demonstrates that hand-drawn fiducials are perfectly acceptable. As long as connectivity is retained, a region can be reduced to a single pixel and the processing will succeed. In terms of speed, VPS achieves roughly 20 frames/sec on a 1.8 GHz Athlon-based PC running Linux, including all video capture, processing and display.

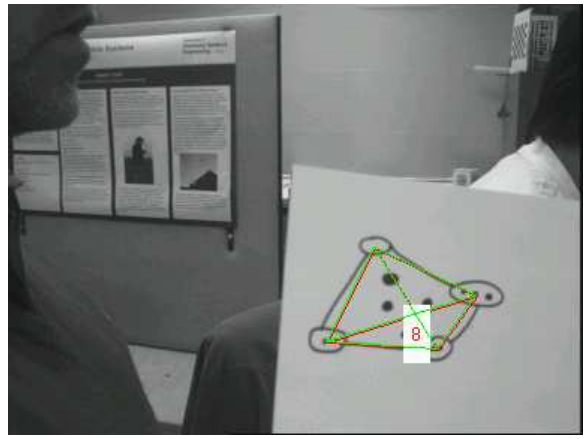


Figure 6: *Use of a hand-drawn fiducial*

5. Pose estimation from fiducial locations

The general approach to the problem of estimating the camera pose from known objects has received a great deal of attention in recent years. Using homogeneous coordinates, one can obtain an expression for the location in the image (u,v) of a 3D position (X,Y,Z). With a little care, this can be separated into two components: the first describes the camera itself (the so-called *camera calibration matrix*, **C**), while the second, **T**, describes the orientation of the camera relative to the origin (*i.e.*, its *pose*):

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{CT} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Given a number of image locations that correspond to known 3D points, one is able to build up a set of simultaneous equations whose solution determines **C**, a process known as *camera calibration*. Subsequently, one can use knowledge of **C**, the known 3D points and their image locations to calculate **T**, the camera pose, again by solving a set of simultaneous equations. The majority of workers do this using non-linear techniques such as Levenberg-Marquardt, though it has been pointed out that a linear solution should be adequate if effective calibration is performed.[2]

Calibration is typically performed using a special rig (Figure 7) of known size, shape and configuration. This is invariably non-coplanar to fulfill the "effective calibration" constraint. The normal way of using VPS is to calibrate the camera prior to use; this is a requirement if one is using coplanar

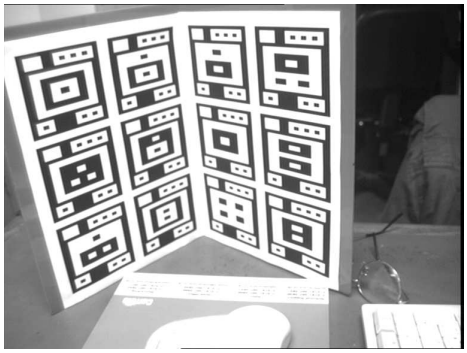


Figure 7: *12-target calibration rig*

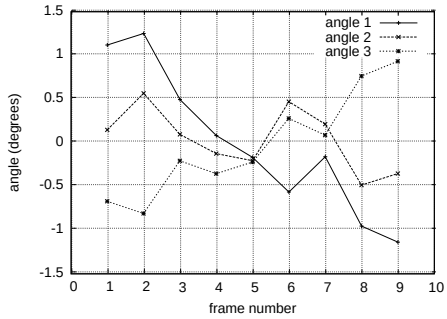


Figure 8: *Deviations in angle during camera translation*

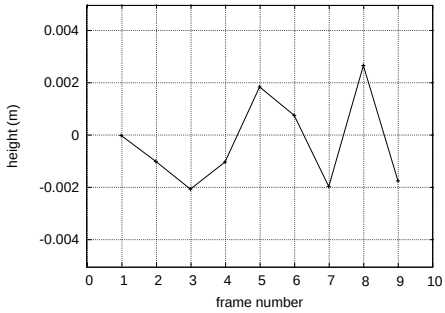


Figure 9: *Deviations in height during camera translation*

targets, as one does when attaching them to, say, the ceiling of a room. However, with non-coplanar targets, one can calibrate the system as it is used. This latter approach is used in the accuracy measurements for the non-coplanar calibration rig presented in the following section.

6. The accuracy of VPS

Determining the accuracy of a system such as VPS is actually rather difficult. There are many external factors that affect accuracy — including the positioning of targets in the environment, the calibration of the camera, its lens distortions, and video capture irreguarities — as well as those due to the actual processing. The results described here are indicative of the performance that can be expected in practice.

A good-quality, analogue, monochrome camera was used, with a fixed focal length lens. Images were captured at 360×288 pixels both to speed processing and to remove effects due to the two fields in a PAL frame. Two particular configurations have been examined:

A 3D calibration rig. This employs two sets of 6 fiducials, each printed on a single piece of A4 paper, attached to a rigid frame with an accurate 90° angle between them (Figure 7). Hence, the target positions are known accurately, the configuration is small in physical extent and non-coplanar.

Targets attached to the ceiling. By comparison, the target positions are not known accurately (though locations are nominally 60 cm apart), the space is large in physical extent and targets are coplanar (Figure 1).

6.1. Using the calibration rig

Using the calibration rig with all 12 targets in view, 99 consecutive frames were captured and processed. The standard deviations obtained were $(0.00026, 0.00046, 0.00059)$ m for the position and $(0.021, 0.025, 0.017)$ degrees for the orientation. These low values indicate that the system is stable, and show the limiting accuracy that can be achieved due to noise *etc.*.

The camera was pointed at the calibration rig and moved away in a straight line for a distance of 0.8 m, measurements being taken at 0.1 m intervals. Deviations in the measured angles and heights are presented in Figure 8 and 9 respectively.

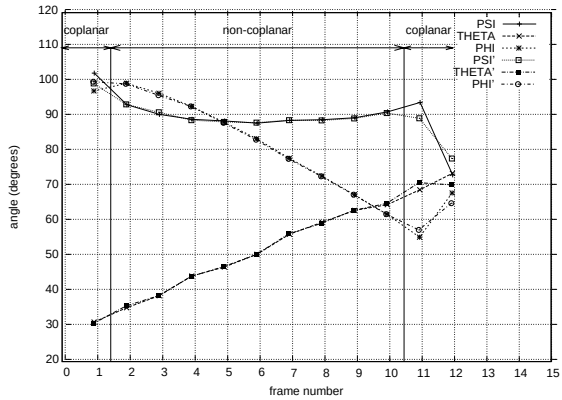


Figure 10: *Measured angles during camera rotation*

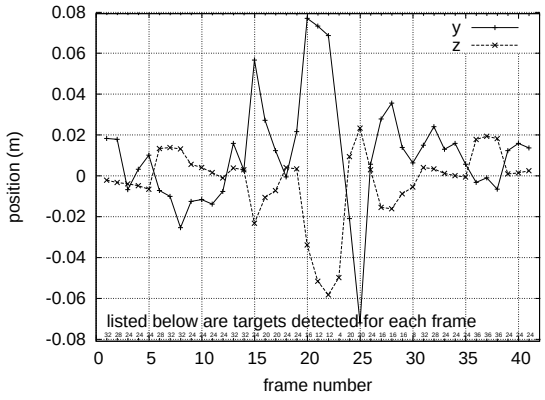


Figure 11: *Deviations in y and z while translating in x*

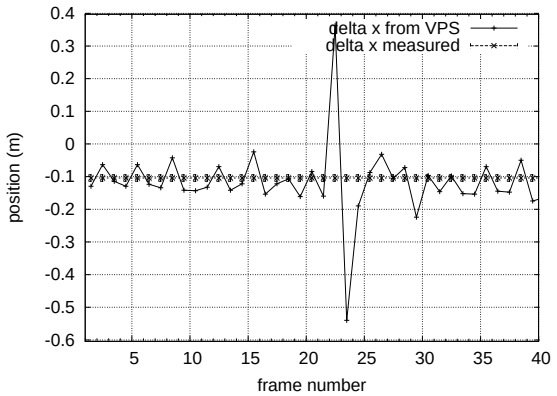


Figure 12: *Positional differences while translating in x*

To determine the rotational accuracy, the camera was fixed to a turntable and pointed at the calibration target. Great care was taken to ensure that the principal axis was normal to the turntable and aligned with its centre. The turntable was then rotated 55° clockwise with measurements taken every 5°. The camera was then rotated back, with measurements again being taken every 5°. Figure 10 shows that the forward and backward passes (primed and unprimed angles respectively) produce similar results. The variation is approximately linear in the non-coplanar region, though less so when coplanarity arises.

6.2. Using ceiling-mounted targets

The camera was pointed upwards to view the ceiling-mounted targets and moved in a straight line parallel to the x-axis over a distance of 4 m in 0.1 m steps. The deviations in y and z are plotted in Figure 11 and the differences between successive x positions in Figure 12. The errors in this case have increased significantly beyond those obtained using the calibration rig. Moreover, they appear to be systematic rather

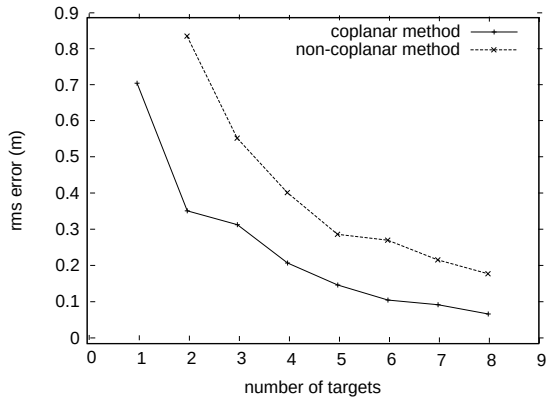


Figure 13: *Variation of accuracy with number of targets*

than random: fitting straight lines to each of x, y and z yielded a correlation coefficient of -0.94 between the x and y residuals. It is also noteworthy that outliers appear when only a single target (*i.e.*, 4 points) is available and the problem becomes under-determined. Nevertheless, the accuracy is quite adequate for the applications that have been developed using VPS to date (Section 7).

Finally, it is valuable to know how accuracy suffers as the number of targets in view changes. The number of targets was varied through software simulation, as this solves the problem of gathering images of different numbers of targets that are somehow "equivalent." A single image of the 12 targets of the calibration rig was used. A set of 11 targets can be formed in 12 ways by removing a single target from it; similarly, a set of 10 targets can be formed in 11×12 ways. To avoid a combinatorial explosion of cases, an upper limit of 100 combinations was employed and 100 random ways of choosing a particular number of targets were automatically selected. The baseline position to calculate errors is that derived from all 12 targets. With a single target in view, accuracy is around 20 cm while, with 11 targets in view, accuracy is around 1 cm. The positional error decreases with the number of targets but there are diminishing returns after, say, 5 targets. After this stage, the biggest source of errors is elsewhere.

In a similar way, an image of the ceiling was used with 8 identifiable targets, and again the number of targets was varied by software simulation. The variation of the RMS error with the number of targets is shown in Figure 13 for both the coplanar and non-coplanar cases. The accuracy is somewhat better in the coplanar case than the non-coplanar one. This may seem surprising — conventional wisdom states the converse — but merely reflects the fact that the calibration data are more accurate in the coplanar case, as discussed in Section 5.

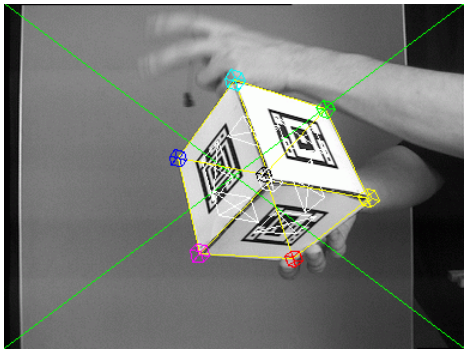


Figure 14: *Use of VPS in a "video joystick"*

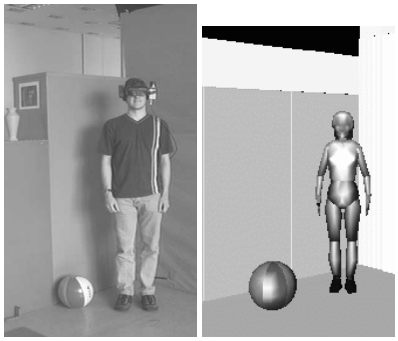


Figure 15: *Use of VPS to position an avatar in a virtual environment*

7. Example applications

As implemented, VPS is essentially a library, written in C and usable from C++, which takes in an image and yields estimates of the position and orientation. The library also provides an interface to the video capture sub-system on Linux. Location information can be fed directly into the industry-standard OpenGL library. Ancillary information derivable includes the positioning of fiducials in the scene and the camera parameters.

To date, VPS has been applied to two specific applications. The first of these is a "video joystick" in which the user rotates a cube with VPS fiducials attached to each face; see Figure 14. The motion of the cube is used to control the motion of a 3D object within the computer, providing a more natural 3D manipulation tool than, say, a mouse. In Figure 14, the object being rendered is simply a wireframe model of a cube, which has been co-located with the real cube and rendered using OpenGL.

The second application is to control the motion of an avatar in a virtual environment (Figure 15). VPS fiducials are attached to the ceiling and an upward-pointing camera attached to a person allows the wearer's position to be determined continuously. The position and orientation are fed to a shared virtual environment system[6] and used to control an avatar.

8. Conclusions and further work

This paper has provided an overview of VPS, a system for performing position and orientation determination using fiducials in real time on commodity hardware. Our experiments have shown it to be reasonably accurate and exceptionally robust, and the paper has outlined some of the applications to which it has been applied.

VPS is available for download from

`http://vase.essex.ac.uk/software/vps/`

and is free for research use. The distribution comprises the library, an example application, documentation, and ≈ 300 fiducials in PostScript form.

There are two specific areas to be addressed in the future. The processing needs to take into account the radial distortion that is always present in low-end lenses such as those found on webcams. It should also be able to achieve improved accuracy by performing a global optimisation of target location data over the entire space.

Acknowledgements

The authors would like to thank the EPSRC for funding DJJ through a studentship. They would also like to thank Ipsita Sinha for her efforts in the calibration work.

References

1. Pagionitis Ritsos and Adrian F. Clark. Mobile augmented reality archæological reconstruction: Ergonomic and accuracy considerations. In *Proceedings of the UNESCO Conference on Virtual World Heritage*, October 2002. 1

2. Olivier Faugeras. *Three-Dimensional Computer Vision, A Geometric Viewpoint*. The MIT Press, 1993. ISBN 0-262-06158-9. 1, 5

3. G. Thomas, J. Jin, T. Niblett, and C. Urquhart. A versatile camera position measurement system for virtual reality in TV production. In *Proceedings of IBC'97*, pages 284–289, September 1997. 1

4. M. Billinghurst and H. Kato. Collaborative mixed reality. In *Proceedings of International Symposium on Mixed Reality (ISMR'99)*, pages 261–284, 1999. 2

5. Ramesh Jain, Rangachar Kasturi, and Brian G. Schunk. *Machine Vision*. McGraw-Hill, 1995. 2, 5

6. J. A. Carson and A. F. Clark. Multicast shared virtual worlds using VRML97. In *Proceedings of the Fourth International Conference on the Virtual Reality Modelling Language and Web 3D Technologies*. IEEE Press, February 1999. 8