Analysing fundamental properties of marker-based vision system designs

Andrew C. Rice [*], Robert K. Harle, Alastair R. Beresford

Computer Laboratory, University of Cambridge
15 JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom

Abstract

This paper investigates fundamental properties of Marker-based Vision (MBV) systems. We present a theoretical analysis of the performance of basic tag designs which is extended through simulation to investigate the effects of different processing algorithms. Real-world data are processed and related to the simulated results. Image processing is performed using Cantag, an open source software toolkit for building Marker-based Vision (MBV) systems that can identify and accurately locate printed markers in three dimensions. Cantag supports multiple fiducial shapes, payload types, data sizes and image processing algorithms in one framework. This paper explores the design space of tags within the Cantag system, and describes the design parameters and performance characteristics which an application writer can use to select the best tag system for any given scenario.

Key words: Computer Vision, Fiducial Tag Design

1 Introduction

Developers of pervasive computing systems have long recognised the utility of determining location information about system components, users and other entities in the operating environment. Machine-based vision systems are becoming an increasingly popular way of collecting these data. Some vision systems locate objects by processing images of the natural environment. However, many vision systems are designed to recognise *fiducial* marker tags rather than operating upon unconstrained images. This approach provides improved performance in terms of runtime

[*] Corresponding Author. Tel: +44 1223 767024 Fax: +44 1223 767009
Email addresses: andrew.rice@cl.cam.ac.uk (Andrew C. Rice),
robert.harle@cl.cam.ac.uk (Robert K. Harle),
alastair.beresford@cl.cam.ac.uk (Alastair R. Beresford).

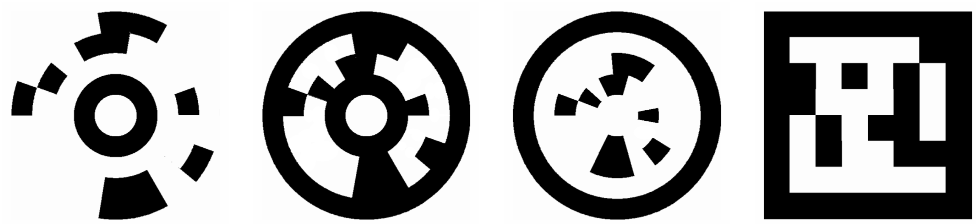
costs and increased reliability in object identification and localisation at the cost of attaching specially designed tags to every object to be tracked. Fiducial markers can be thought of as advanced bar-codes (often printed using commodity printing hardware) with the potential not only to label an object but to position it accurately. The field of Augmented Reality (AR) has been the traditional development domain for such Marker-Based Vision (MBV) systems (Billinghurst and Kato, 1999), (Rekimoto, 1998), (Rekimoto and Ayatsuka, 2000), where they are favoured for their dependence on commodity hardware (decreasing deployment costs) and for their high degree of precision and accuracy across six degrees of freedom (ideal for image-object registration). Most AR applications focus on *video overlay* where three-dimensional models are rendered into the video stream viewed by the user.

As pervasive computing systems emerge, MBV systems also offer the potential to create large scale, ubiquitous tracking environments with a multitude of novel applications. Different applications demand different properties from an MBV system. A mobile user, for example, may wish to trade-off accuracy in favour of extended battery life, whilst another may only be interested in identifying objects in the image without the need to locate them.

This paper makes use of *Cantag* (Rice et al., 2006), an open-source software toolkit suitable for designing and deploying an MBV system as part of a pervasive computing application. Cantag differs from previous MBV systems, which we compare in Section 7, in a number of important ways. In particular, it allows an application writer to:

- select the most appropriate tag design from a wide variety of fiducial types, or implement a custom marker;
- choose the most appropriate algorithm for each stage of image processing, given the application requirements;
- characterise the MBV system through simulation before deployment;
- build a custom MBV system executable, optimised for their particular application;
- efficiently track multiple tag types in the same video stream by sharing common processing steps; and
- deploy their system using normal or high frame-rate Firewire or Video4Linux cameras.

The remainder of this paper describes how pervasive computing researchers can use the Cantag system to design, build and integrate an MBV system with their application. Section 2 provides an overview of the Cantag system and describes how various system components can be composed with C++ templates to provide an optimised executable. In Section 3 we present a theoretical analysis of tag performance to provide high-level insights into design options. Section 4 extends this analysis with simulated results from the OpenGL test-harness in Cantag allowing the evaluation of the performance trade-offs available to application writers when



(a) CircleInner (b) CircleSplit (c) CircleOuter (d) Square

Figure 1. Four example tag types in the Cantag system.

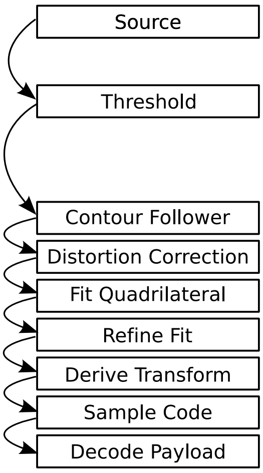
using different algorithms and tag designs. Section 5 checks predicted results from our analysis using the Cantag system with real world images. Section 6 reviews the salient points of our analysis on tag performance and describes how a pervasive computing researcher can make the most effective use of the wide variety of tag designs available in Cantag for their application domain. Section 7 contrasts the Cantag system with related work and Section 8 describes the lessons learned and reviews the key tag features an application designer should consider when deploying an MBV system.

2 Cantag

Cantag is an open-source computer vision framework written in C++. It makes extensive use of the *template* programming metaphor, enabling the compiler to generate an optimised executable for any particular set of tag design and algorithm options. This is important since the use of templates allows us to deliver a flexible tag framework, whilst still providing real-time processing of image data, even for high frame-rate cameras. Since Cantag is written in C++, it can easily be integrated with existing C or C++ code.

Cantag currently only processes 1-bit images, since this methodology is most applicable to resource constrained platforms; however we have developed Cantag with a view to extending support to greyscale and colour processing in the future. Even when constrained to processing 1-bit images, pervasive computing applications have a surprisingly large variety of needs from an MBV system. Cantag allows system designers to choose algorithms with the desired execution costs or accuracies and to customise tag designs to provide the best trade-off between data capacity and reliability.

Our system currently implements two fundamental tag types: the CircleTag describes tags based around a circular bullseye; and the SquareTag describes tags based around a square border. The CircleTag can be further configured to control the relative proportions of the tag that are occupied by the bullseye and data rings, giving rise to four tag shapes which may contain either template or symbolic payloads. The symbolic payload can be configured to store an arbitrary amount of data



```
Square8 tag;
GreyImage* i  = fs.Next();
MonochromeImage m(i->GetWidth(),i->GetHeight());
Apply(*i,m,ThresholdGlobal<GreyImage>(180));
Tree<ComposedEntity<TL4(ContourEntity,ShapeEntity<QuadTangle>,
                        TransformEntity,DecodeEntity<64>) > > tree;
Apply(m,tree,ContourFollowerTree(tag));
ApplyTree(tree,DistortionCorrection(camera));
ApplyTree(tree,FitQuadTanglePolygon());
ApplyTree(tree,FitQuadTangleRegression());
ApplyTree(tree,TransformQuadTangleSpaceSearch());
ApplyTree(tree,Bind(SampleTagSquare(tag,camera),m));
ApplyTree(tree,Decode<CRCSymbolChunkCoder>());
ApplyTree(tree,TransformRotateToPayload(tag));
```

Figure 2. A sample Cantag pipeline with example code.

using the *rotational invariance* abstraction (Rice et al., 2004). Figure 1 shows four example tags. We will see later in the paper that there are various performance trade-offs associated with the choice of fiducial. Therefore an application designer should choose the fundamental tag design with care.

Once the basic tag design has been selected, the Cantag system then allows a number of different algorithm choices for the image processing. The programming abstraction used by Cantag models a tag processing pipeline by a sequence of *algorithms* (C++ function objects) operating on *entities*. Examples of entities include contours, ellipses, quadrilaterals and payload data.

The system is extended by adding additional algorithms which explicitly indicate the types of entity used as argument and result types. For example, a simple processing pipeline could use seven processing stages to build a fully-functional MBV system: image capture, thresholding, building a tree of contours, correcting camera lens distortion, testing and fitting tag shapes to contours in the image, calculating each camera-to-tag transform, and finally decoding the tags in the scene. This process is shown visually in Figure 2—in this example the application designer would write approximately one line of C++ code for each stage in this pipeline.

The Cantag framework also allows the construction of more complex processing pipelines. For example, we can build pipelines which process multiple tag types within the same scene (and share common processing steps), or dynamically change processing algorithms depending on the current needs of the application. The remainder of this section briefly describes the various algorithms currently available within Cantag and summarises their performance.

2.1 Thresholding

The thresholding algorithms are used to convert an input image to a 1-bit image. The Global threshold algorithm takes a fixed threshold value. Every pixel in the image is converted to black or white depending on whether its intensity is greater or less than the threshold. This algorithm has a very low cost per pixel and is suitable for images where the lighting intensity is uniform across all areas of interest in the image. For example, tags captured on a mobile phone camera will often be taken at a relatively short range and therefore the tag is likely to have an even illumination.

The Adaptive threshold algorithm utilises a moving average across the image to choose the threshold value (Wellner, 1993). Systems recognising images with varying light conditions will need to accept the higher computational cost required to perform an adaptive threshold.

2.2 CircleTag

The perspective transformation of a circle is an ellipse (Eves, 1972) which contains (almost) enough information to deduce the projective mapping (known as back-projection) between the real-world position of the tag and the resulting image. Therefore the first stage of recognising a CircleTag is that of fitting ellipses to contours in the image. The Least squares algorithm performs a least-squares ellipse fit to the contour points (Halíř and Flusser, 1998). This algorithm requires numerous non-trivial floating point algorithms [1] . However, the quality of the position and pose information produced by the system is directly dependent upon the quality of the ellipse fit and so systems requiring accurate positioning information might consider this a necessary expense.

The Simple fit algorithm provides a low cost alternative to least-squares fitting of an ellipse. This algorithm calculates the central point of the ellipse as centre of gravity of the contour and then finds the major and minor axes as the longest and shortest distances from the centre. Low power or high-speed applications may be prepared to accept the reduced accuracy in return for a simple, fast algorithm.

Once the ellipse has been fitted the perspective transformation may then be be derived. The 3D transform algorithm implements an adaption of Forsyth’s ellipse back projection algorithm (Forsyth et al., 1991) to recover a general 3D transformation from object co-ordinates to camera co-ordinates. This algorithm is computationally complex but produces accurate 3D information for the tag’s position and pose.

Alternatively, the Linear transform algorithm simply scales the located ellipse lin-

[1] In particular, it is necessary to solve a 3x3 eigensystem for a non-symmetric matrix.

early within the image. This requires little computational overhead but provides a transform which is only valid when projected into the image—and so overlay of 3D models and 3D position information are unavailable. This transform also makes assumptions about the perspective transform which are invalid under large perspective distortion.

2.3 SquareTag

Recognising SquareTags follows a similar process to the circular tags. Perspective projection of a square results in a general quadrilateral and hence the contour follower must identify the four corners of the quadrilateral. The Corner fitting algorithm slides a window around the contour and returns all points with discrete curvature above a chosen threshold. This algorithm is fast, efficient and easy to implement but is susceptible to noise on the contour. Its resilience can be increased by simplifying the polygon of points using a convex hull algorithm and identifying corners based on maximal local curvature: this has been implemented within the Convex hull simplification algorithm. The Polygon simplification algorithm (Douglas and Peucker, 1973) repeatedly hypothesises polygon approximations to the contour and adds additional vertexes in order to reduce the contour's deviation from the polygon. It has a high cost but is better able to withstand contour noise.

A further option is to apply the Linear regression algorithm to fit each set of points corresponding to a side of the quadrilateral to best estimate the infinite line passing through the set. The four intersections of the infinite lines represent the best estimate of the true corner points. This algorithm ignores the samples near the corners and bases the corner determination on the more reliable body of points between them. Note that regression needs the contour points to be segmented into the four edges of the quadrilateral, implicitly requiring an estimate of the corners. Such estimates can be derived using any of the aforementioned algorithms: an advantage of regression is that the corner estimate need not be highly accurate, merely sufficient to partition the dataset.

The Projective transform algorithm may then be applied to the recognised quadrilateral to recover the 3D projection. This algorithm returns a 3D transformation suitable for 3D model overlay but is susceptible to noise in the image, making 3D position information unreliable. The algorithm solves a set of linear equations for the four point correspondence between the corners of the tag in object co-ordinates and in image co-ordinates. These constraints are not sufficient to preclude independent scaling of the vertical and horizontal object axes. However, any warping is exactly cancelled out when projecting from the surface of the tag in object co-ordinates into image co-ordinates. Furthermore, the error in resulting re-projected projection co-ordinates (as used in visual overlay) is often sub-pixel and so this algorithm is a good choice for systems that do not require 3D position or pose

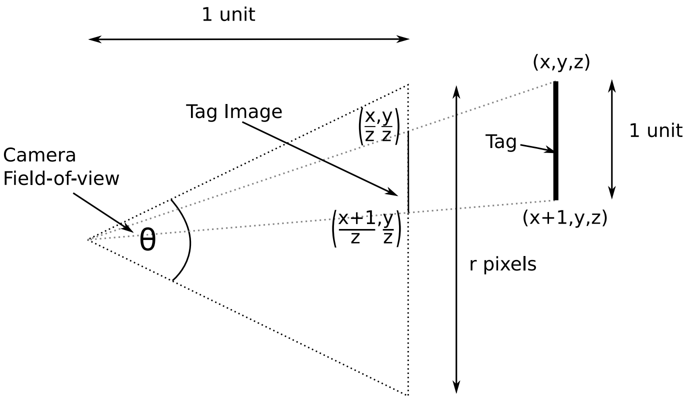

1 unit
(x,y,z)
Tag Image
$\left(\frac{x}{z},\frac{y}{z}\right)$
Tag
1 unit
Camera
Field-of-view
θ
$\left(\frac{x+1}{z},\frac{y}{z}\right)$
(x+1,y,z)
r pixels

Figure 3. The size of the tag in the camera image is inversely proportional to the distance from the camera.

information.

Better 3D transform results are possible using a non-linear transform algorithm since this can be used to incorporate (the inherently non-linear) constraints relating to a square into a four-point correspondence problem. This algorithm requires multiple iterations to find a non-linear solution and is therefore computationally expensive to execute.

We systematically name tags according to the algorithms selected for their decoding based on the concatenation of tag name (as shown in Figure 1), shape fitting algorithm, back-projection algorithm, and payload size. For example, a CircleInner tag, using the Least squares shape fitting algorithm, followed by the 3D transform algorithm, with a payload of 36 bits is named CircleInnerLS3D-36.

3 Modelling Tag Performance

The Cantag system permits an examination of the fundamental limits of tag readability. In this Section we use a series of mathematical models to assess the performance of specific tag designs. We begin by providing some insight into how the performance of a system will change as the payload size of the chosen tag is altered.

We expect the performance of the system to decrease as the tracked tag's distance from the camera increases. However, a more useful metric to consider is the tag's size (in pixels) in the camera image. This metric is inversely proportional to the distance from the camera. The constant of proportionality encodes the camera resolution and field-of-view. This is shown in Figure 3. The distance between the two projected points is $(x+1)/z - x/z = 1/z$ in units of tag size. The total width (also in units of tag size) of the camera image is $2\tan(\theta/2)$. The total width occupies r pixels and so the total width of the imaged tag in pixels is $r\tan(\theta/2)/2$. This defi-

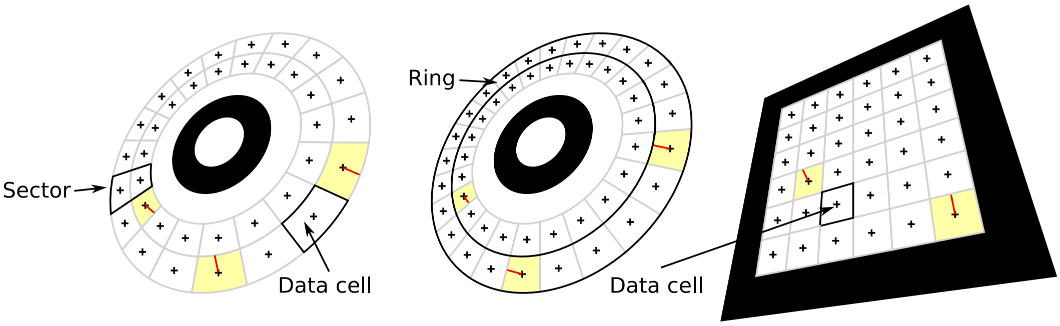


Figure 4. Example minimum sample distances for circular and square tags.

nition of tag size should be interpreted as width (in pixels) that the tag image would occupy if the tag were in its current position without any rotation of the normal vector.

A tag design which incorporates a symbolic payload contains a series of data cells positioned at precise locations around the fiducial. A sample point for each data cell is located in the centre of the cell in tag coordinate space. In normal operation Cantag uses the transformation from camera coordinates to tag coordinates (deduced from analysis of the image of the fiducial marker) to estimate the position of the sample point for each data cell in the image plane of the camera. The data held in the thresholded image at each projected sample point can then be used to read the symbolic payload of the tag.

For a given tag at a specific location and pose, we define the *minimum sample distance* as the minimum distance between the projected sample point for any data cell and the edges of that cell in the image plane of the camera. Figure 4 shows a number of candidate minimum sample distances—the shortest candidate distance for all data cells corresponds to the minimum sample distance. This minimum sample distance gives a measure of how hard the tag is to read at this pose and location—the smaller the value the less margin for error in estimating the position of the sample point.

If the minimum distance of a particular data cell is less than one pixel then, even if an algorithm can deduce the precise pose of the tag, the sample point may still read the pixel value of an adjacent cell. Therefore there is a fundamental lower-bound on the minimum sample distance of 1 pixel if we want to reliably read the payload of a symbolic tag. This situation is analogous to the Nyquist-Shannon sampling theorem which states that a discrete representation of an analogue signal is only possible if the highest frequency component of the analogue signal is less than half the sampling rate. Therefore, data cells may occur no more frequently than once every two pixels in the image plane of the camera. This corresponds to a minimum distance of one pixel from the centre of the cell to the edge. If the minimum distance is any less than this then the tag data cells will suffer aliasing and accurately measuring the payload will become impossible.

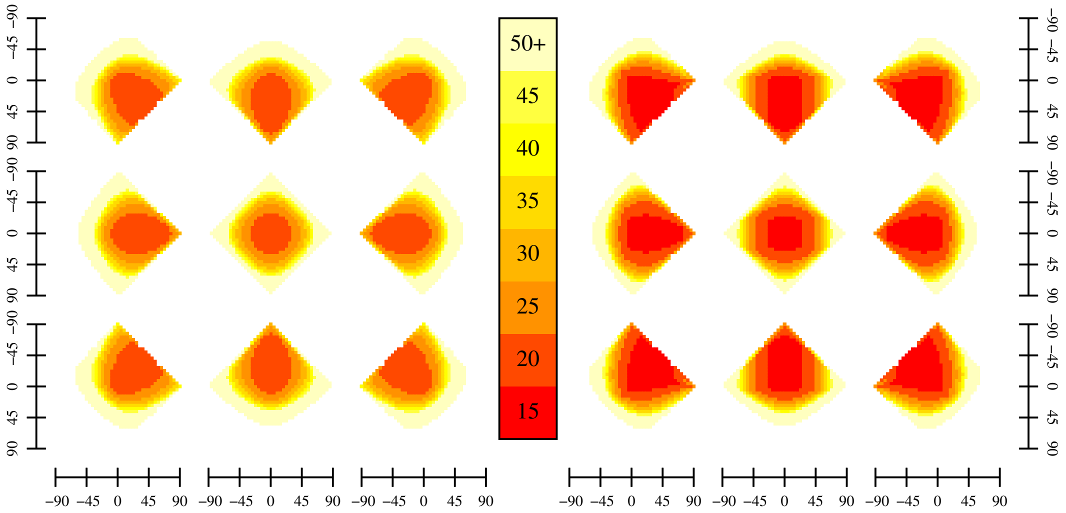


Figure 5. Minimum tag size (pixels) such that the minimum sample distance is one pixel for CircleInner-36 (left) and Square-36 (right) tags.

For a particular tag pose, the minimum sample distance varies linearly with the size of the tag in the image. The size of the tag in the image is also inversely proportional to its distance from the camera projection plane (i.e. along the z axis) along a particular ray. We use the term *ray* to refer to a straight line drawn from the camera origin to some infinite point. Therefore linear interpolation can be used to find the distance from the camera when the minimum sample distance is one pixel. This tag size represents the fundamental maximum distance at which a tag of a particular shape and pose can be read. This result does not guarantee the tag can be read at this distance by any particular implementation; rather, it provides a fundamental upper bound on the possible read distance of a tag.

Figure 5 shows the tag size in pixels such that the minimum sample distance is one pixel for a Square-36 and CircleInner-36 tag. Both halves of the figure contains nine sub-plots corresponding to one of nine equally sized regions in the image. For example, the top-left sub-plot (on both sides of the figure) corresponds to a ray that goes through a point in the top left corner of the image. The axes of each sub-plot represent the x and y components of the tag's normal vector. For example, the centre of each sub-plot corresponds to a fully facing tag and the bottom-right of each sub-plot corresponds to a tag facing down and to the right. The value at each point on a sub-plot shows the tag size in the image such that the minimum sample distance is one pixel. The white regions around the edges of each sub-plot indicate orientations where it is not possible to make the minimum distance one pixel no matter how close the tag is brought towards the camera. As expected, a tag positioned at the top of the image (above the camera) is more easily read when facing downwards in the image rather than upwards–this explains the truncation of the circular plot pattern for the sub-plots corresponding to the edges of the image. We also note that the square tag achieves longer read distances than the circular tag for small angles of inclination. However, under more extreme angles of inclination the performance of the two designs converge.

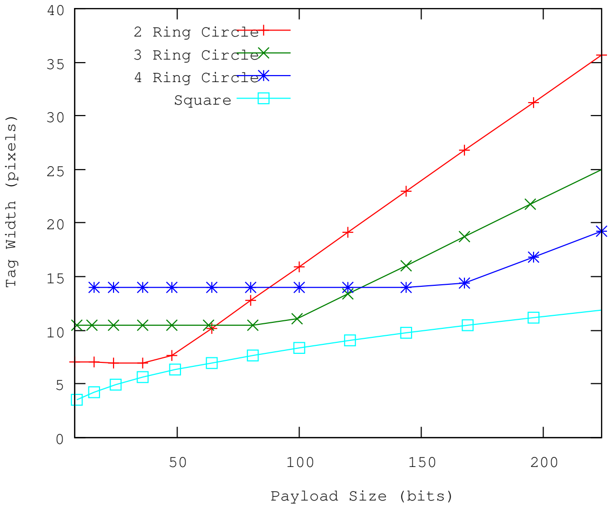
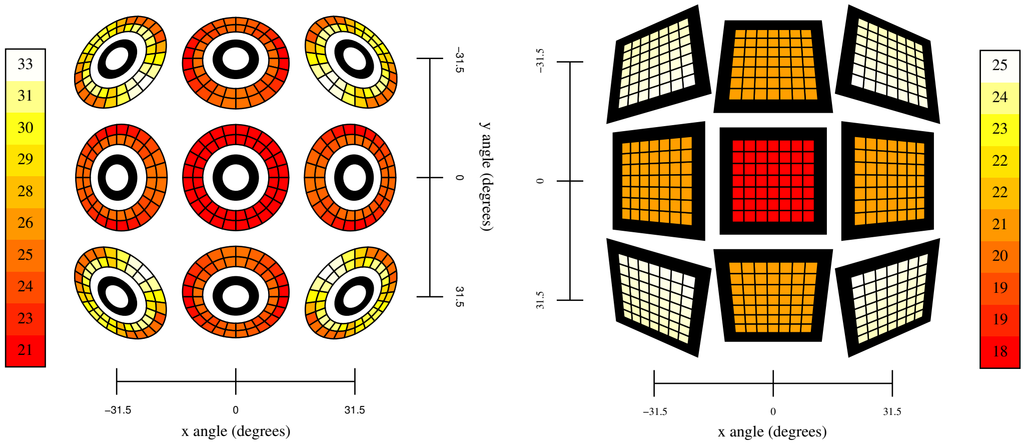


Figure 6. Minimum tag size for varying payload size.

The effect of the shape of the data cells is evident in the way that the performance of the tags drops off as the tag inclination is increased. The high degree of rotational symmetry possessed by the Circular tag means that when the tag is in the centre of the image the degradation in performance is only dependent upon the angle between the normal vector and the camera vector. The square tag is more directionally sensitive, the square edges of this shape are due to the fact that tilting the tag in the x direction will reduce all the cells in the far edge row in size. Subsequently tilting in the y direction will not reduce the minimum distance of these cells until the tilt exceeds that applied in in the x direction. Rotation of the square tag around its normal vector causes the same rotation in the direction of the square edges seen in the figure because cells' favoured direction of tilt is moved round.

Figure 6 shows the effect of increasing payload size on the minimum sample distance. The tag size in the image such that the minimum sample distance is one pixel was found for increasing payload sizes. We expect that the square tag will experience a decrease in read performance in proportion to the square of the payload size. This is because going from an $n \times n$ tag to an $(n + 1) \times (n + 1)$ tag adds one data cell along the edge causing a linear decrease in the minimum sample distance for a quadratic increase in payload size. The curved line on the graph (which has the same shape as $y = \sqrt{x}$) for the square tag is due to this effect. We also see that the circular tags show no loss in performance when increasing payload size by adding to a small number of sectors—this is because the distance between the data rings is less than the distance between sectors (consequently the distance between the data rings rather than the distance between the sectors limits the tag performance). This graph also shows the benefit of adding additional data rings to the tag once the payload size increases. For example, a four ring tag with 37 sectors (148 bits) has a better minimum distance value than a smaller capacity tag with 3 rings and 49 sectors (147 bits).

Figure 5 indicates the likelihood of a single bit error for a particular position and



(a) CircleInner

(b) Square

Figure 7. Systematic data cell errors due to tag geometry.

pose of the tag. If the tag is utilising an error correcting code then a designer might expect better performance because a certain number of bit errors can be tolerated before the tag becomes unreadable. Figure 7 shows data captured from a tag located at the centre of the camera image with a range of normal vectors. For each data cell on the tag, the tag size, such that the minimum distance is one pixel, is shown. Data cells with a value indicating small tag size are more robust (i.e. can be read at a greater distance from the camera) than data cells requiring a larger tag size. For the square tag design all the data cells produce errors at approximately the same distance from the camera. This suggests that there is little value in using an error correcting code to recover from errors due to the tag geometry—although errors from other sources such as image noise may still be worth correcting.

The circular tag shown in Figure 7(a) shows a more significant variation in data cell distances. Tags with an extreme inclination in one axis (e.g. large tilt in the x-axis direction and no tilt in the y-axis direction) show minimal change for the data cells close to the axis of rotation and a drop in performance for cells perpendicular to the axis of rotation. This is because the minimum distance for this tag design (2 rings with 18 sectors) is radial (between rings) rather than tangential (between sectors). This makes the tag more amenable to the addition of more sectors than to the addition of more rings (Figure 6). Thus, when the tag is rotated, those cells near to the axis of rotation are compressed in the tangential direction. This does not affect the minimum distance. However, cells perpendicular to the axis of rotation are compressed radially. This does reduce the minimum distance. As previously mentioned the optimal tradeoff for a circular tag is to balance the width of the rings with the size of the sectors. The results in Figure 7 further suggest that designers unable to exactly match these parameters should err on the side of decreasing the sector size rather than the ring width.

4 Simulating Tag Designs

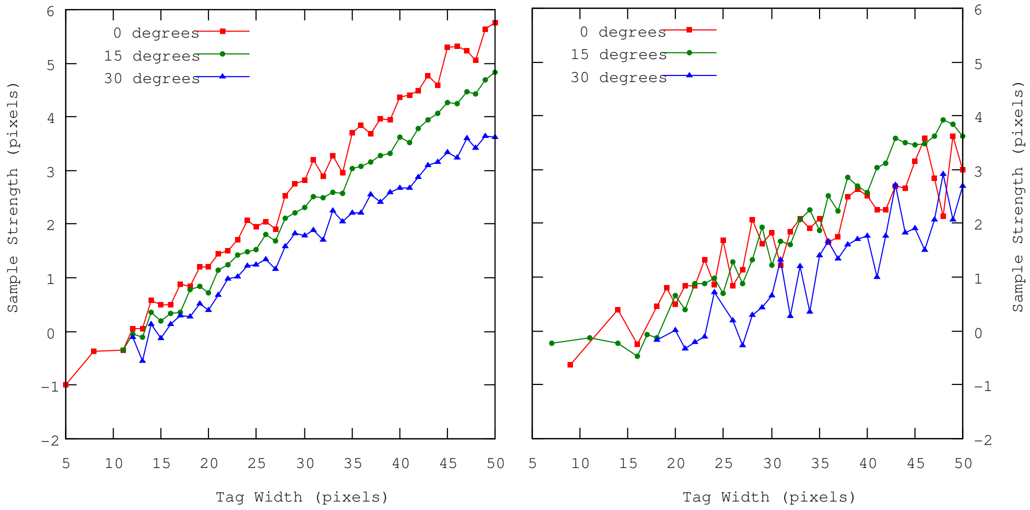
The Cantag system incorporates an image source for processing artificial images produced by OpenGL. Tags may be rendered with arbitrary positions and poses, processed by the system, and the resulting data compared against the ground-truth input data. This mechanism provides a vital means to ensure that the algorithms offered by the system are correctly implemented. However, it also provides a means of understanding the relative performance of different tags and algorithms since it allows huge numbers of images containing a variety of tag orientations to be systematically simulated.

The images produced by the test harness can be considered ideal: there is no camera distortion, lighting artefacts, or measurement error: the only sources of error are derived from the pixelation of the image and any algorithmic approximations used in the processing pipeline. Hence this harness can be used to place a quantitative *upper bound* on the capabilities of a specific tag using a particular set of processing steps. Thus, in addition to providing a means for comparing two possible configurations of the Cantag system, we can also answer questions as to whether some performance needs are actually possible with current algorithms.

The minimum sample distance described in the previous Section measured how amenable a particular position and pose is to data decoding. However, there is also the issue of how accurately the sample points are estimated from the image of the tag. To investigate these effects we compute the *maximum sample error* by measuring the distance between the estimated sample point and the actual sample point for each data cell on the tag. A simple check of the simulated data shows that if the maximum sample error is less than the minimum sample distance then we experience no data errors when reading the tag. We refer to the difference between the maximum sample error and minimum sample distance as the *sample strength*. If the sample strength is positive then we have successfully read the tag because the maximum error is less than the minimum error tolerance.

Figure 8 shows the effect of using the less complex Simple fit ellipse fitting algorithm for various angles of inclination for the tag and distance from the camera (shown here as the size of the tag in the image). The large variations in sample strength shown by the Simple fit algorithm confirm that it is more susceptible to noise in the shape contour. We also see that the algorithm performs badly for tags which fully face the camera since the contour is circular in shape, making the identification of the longest and shortest axes an ill-posed problem. The Least squares algorithm shows a much less noisy trace suggesting it is better at withstanding contour noise caused by pixel truncation in the image.

Measurement of the sample strength is problematic in real-world images. However, the sample strength is affected by the accuracy of the transform used to recognise



(a) Least squares algorithm

(b) Simple fit algorithm

Figure 8. The sample strength of the ellipse fitting algorithms for the CircleInner tag.

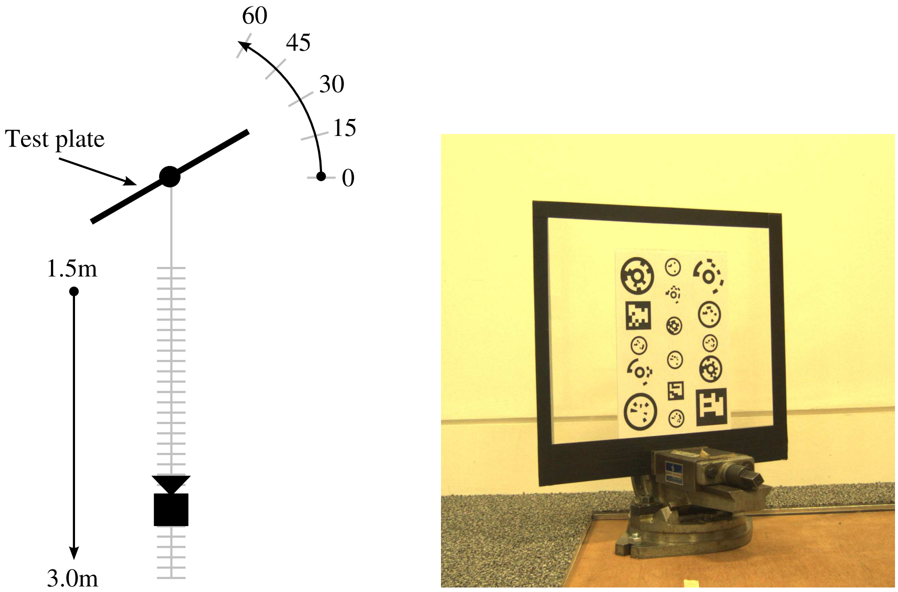
the tag and so we expect that a tag reading at a position with a large sample strength will generate more accurate location information than a position with small sample strength.

5 Real World Results

In order to validate the predicted trends from the test harness data we produced a plate containing a number of different tags of different sizes. We photographed the plate at distances between 1.5 and 4.5 metres from the camera with intervals of 10cm and at inclinations of 0, 30 and 45 degrees to the camera. We then mapped the distance measurements on to tag size (in pixels) in the image (the camera's vertical field of view is approximately 40 degrees). Figure 9 shows the experimental setup and an example captured image.

We have asserted that the tag's pixel size in the image is inversely proportional to its distance from the camera and actual size. This is validated in the data whereby results from different sized versions of the same tag design produce similar results when they appear with the same pixel size in the image. In the following graphs all distances are measured in unit-less dimensions of *tag widths*. The reader may prefer to interpret this as follows: if the tag is 1m across then all distances are in metres.

Figure 10 shows both real-world and simulated 3D location error for a number of different tags and processing combinations. The location error values shown in each subgraph have been clamped at 5 tag units so that the trends in the data remain visible despite the noise in the results. We notice that as predicted by the sample



(a) Experimental Setup (b) Example Image

Figure 9. Experimental Setup

strength measure the Simple fit algorithm is more susceptible to image noise than the Least squares algorithm particularly when the target tag is fully facing the camera. This is also evident in the simulated real-world location error.

We also see that the Linear regression algorithm performs more reliably than the simple Curvature algorithm. These results suggest that the algorithms making use of the entire contour are more robust than the simple algorithms but the effect on the location accuracy is surprisingly small. We note that the Projective transform (not included in the figure) produced errors at least an order of magnitude worse that the Non-linear square transform for the smaller tags.

It is important to note that the actual errors reported by Cantag (of the order of 5–10cm) are not significantly bigger than the possible measurement error in our experiment and so further work is needed with more precise equipment to be sure of the absolute performance of the system. We limit ourselves here to examining trends and relative performance of different tags and algorithms. The real-world accuracy of the circular tag designs follows a similar shape curve to the results predicted by simulation. CircleSplit and CircleOuter tags produce similar accuracy results because they have the same radius for the outer edge of the target bullseye. The results from the square tags contain much more error in the real-world results than predicted in simulation. This is because there are numerous other factors affecting system operation which are not accomodated in the simulation. Examples include incorrect thresholding of the original image due to lighting variation and error in the calibration of the camera equipment. It seems that the square tag design is much more succeptible to these unmodelled effects than the circular design.

The thresholding step at the beginning of the vision pipeline is particularly prob-

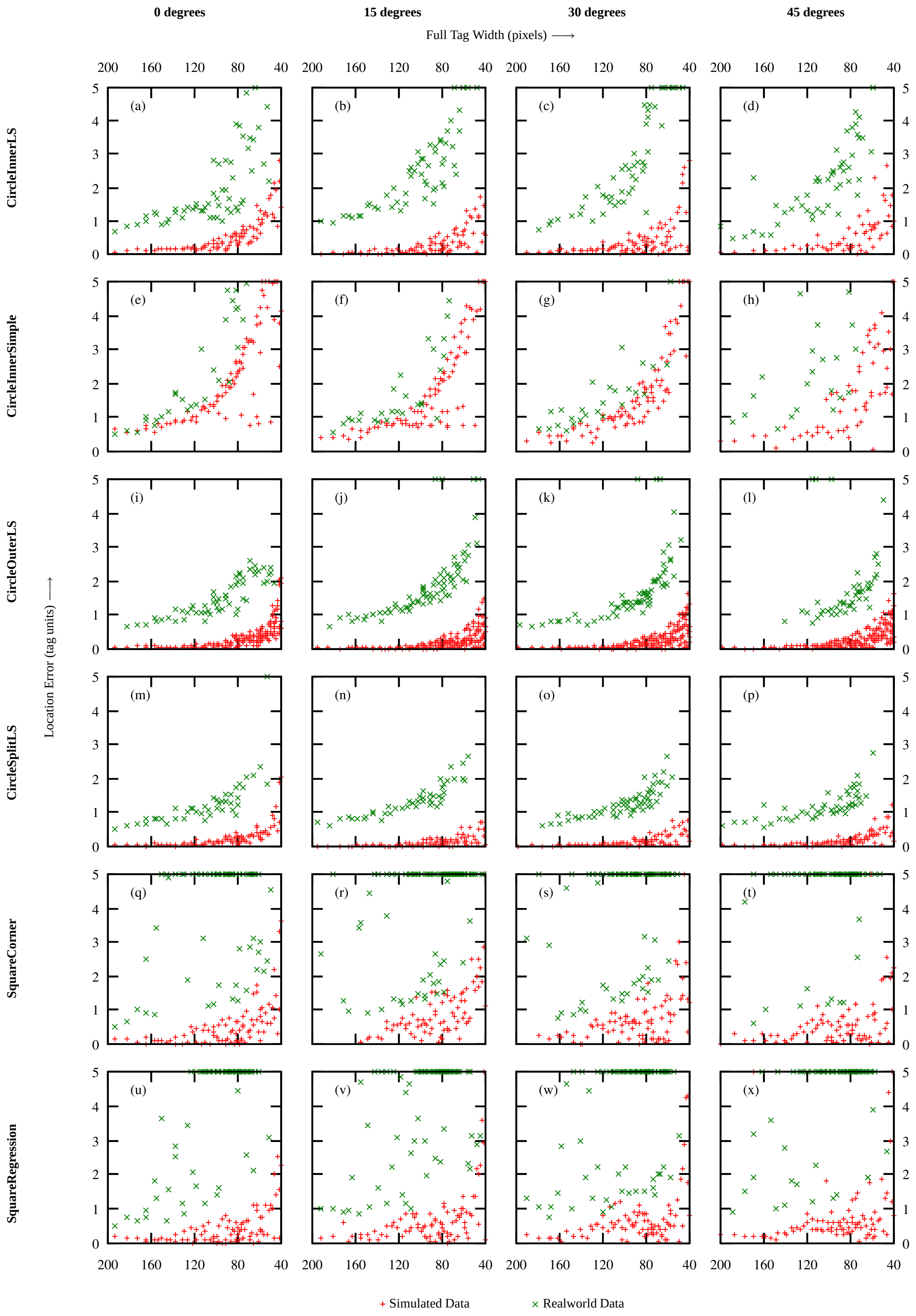


Figure 10. Real-world and Simulated location error for different tags and processing algorithms

lematic in real-world systems because selection of the best technique and thresholds to use varies at run-time. Projects such as ARToolKitPlus ((Wagner et al., 2005)) introduce automatic thresholding which attempts to search for, and to track, the best threshold. We also notice that, in addition to whether the tag is successfully recognised or not, the positioning accuracy of the system is also dependent upon the chosen threshold. We are currently attempting to develop techniques to detect and compensate for this. Further errors in the real-world data due to lens distortion also require additional investigation. Results from photogrammetry suggest that these errors can be corrected to high accuracy (Brown, 1966) although further work is required to identify the trade-offs in the various possible correction algorithms.

A number of trends predicted by simulation are borne out in the real-world data but the effects of image noise amplify any algorithmic instabilities. The test harness results generally suggest that a square-based fiducial marker is superior to a circular design. The square-based markers scale better to large data payloads and the algorithms for detecting and reading them are simpler to implement than for circular tags. The real-world data show that shape fitting algorithms are increasingly robust as more contour points contribute to the fitted shape. For this reason, circular tags provide more robust location information than squares in real-world images.

6 Discussion

The data produced by mathematical modelling, the OpenGL test harness and the real-world results suggest a number of high-level design rules for MBV application developers to bear in mind.

For short-range applications the expected performance of the system is directly determined by the number of pixels occupied by the tag. This is evidenced by the OpenGL test harness which produces the same results for a high resolution camera picturing a distant tag as for a low resolution picture of a nearby tag. We expect atmospheric effects to become significant only over large distances—perhaps affecting applications using high magnification telephoto lenses.

The performance of tags with a cell based payload structure is governed by the minimum distance between the sample point and the edge of the cell. The result of this is that circular tags should balance data-ring radius against sector angle. Analysis of the distribution of data cell errors due to tag geometry suggests that error correcting codes will have little mitigating effect for square tags due to the even drop-out rate across the tag. Circular tags experience errors limited to particular regions of the payload and so might expect an error correcting code to improve read performance.

Circular tags provide more robust location information than square tags especially when using the simpler shape fitting techniques. This can be seen in Figure 10 where the traces for the circular tags show less jitter and noise than those for the square tags. Square tags are, however, capable of carrying larger payloads than circular tags for the same tag dimensions. It is also apparant that decoding of data payload stored on a square tag is successful despite the large location errors sometimes produced.

There are numerous combinations of algorithms and designs producing different behaviour. Selection of these must be done carefully to optimize the trade-off between functionality and performance. For example, use of the expensive Least squares ellipse fitting algorithm provides little advantage over the Simple fit algorithm if the Linear transform algorithm is used later in the pipeline.

7 Related Work

Numerous MBV systems exist, particularly in the field of Augmented Reality. These systems display huge heterogeneity in tag design and implementation. Cantag currently implements the processing pipeline of a number of these systems and we note the additional implementation required for support of the remainder.

Arguably the most popular system for video overlay is ARToolKit (Billinghurst and Kato, 1999). ARToolKit utilises square tags which are detected and read from black and white images. The four corner points in the image serve to compute the projective transform and a template-based scheme is used to recognise specific tags from a database of issued templates within the perspective-corrected image. Owen *et al.* presented a scheme for selecting template images which maximises the distance between tags (before projective distortion effects) (Owen et al., 2002). The addition of a template matching algorithm for decoding the tag data would be sufficient for Cantag to implement the ARToolKit pipeline.

Matrix (Rekimoto, 1998), CyberCode (Rekimoto and Ayatsuka, 2000) and Rohs' mobile phone-based tag reader (Rohs and Gfeller, 2004) also make use of a square-based tag design. However, the use of symbolic codes (as opposed to a template-based system) allows the number of distinct tags to be quantified. ARToolKit and Matrix tags use a solid black border around the entire tag and so shape recognition follows from detecting a quadrilateral in the image. In contrast, the CyberCode and Rohs systems use a combination of marker bars and points detected using region-growing and computing a second-order moment. These algorithms are not currently implemented in Cantag but can be straight-forwardly integrated into the framework and make use of common steps such as recovering the tag position and decoding the binary payload. The ARTag system (Fiala, 2004) also makes use of a square tag design but detection is done based on the results of multi-resolution edge detection rather than image thresholding. Again, extension of Cantag to support this system design requires only the implementation of edge detection and segment linking algorithms because the remainder of the image processing pipeline reuses existing algorithms.

Examples of circular fiducial tags also exist in the literature. The TRIP location system (de Ipiña et al., 2002) uses circular tags with a symbolic code arranged around the outside of a circular bullseye. Naimark and Foxlin's tracker (Naimark and Foxlin, 2002) also utilises a circular tag with additional asymmetric eyelets to orient the tag. The Free-D camera tracking system (Thomas et al., 1997) uses circular tags to determine the position of a mobile camera within a TV studio. Bundle adjustment is used to derive an estimate of the camera position from the angulation measurements to a set of sighted tags whose identifiers are encoded using up to nine concentric circles. Support for this tag design in Cantag requires the implementation of a radial sampling algorithm to read the tag and a bundle adjustment algorithm to estimate position over the set of sighted tags.

The need for trade-offs in the design of marker tracking systems is evident in projects such as Handheld Augmented Reality (Wagner et al., 2005) which perform video overlay on a handheld PDA and might be prepared to accept reduced accuracy algorithms in order to decrease power consumption or achieve real-time performance. The MagicBook (Billinghurst et al., 2001) application overlays active content onto the pages of a book and therefore we would hope for a large number of recognisable tags at the cost of reducing the code distance between each tag.

8 Conclusion

This paper has presented a comparative analysis of the expected performance of many different fiducial tag designs. We have identified fundamental limits to the decoding of imaged tags and used this analysis to quantify the fundamental differences between square and circular tag designs. We have demonstrated how the position and pose of the tracked tag can cause systematic errors in tag decoding.

We have demonstrated how the Cantag system can be used to select the most appropriate tag design for a given application. Important results have been derived through simulation using the OpenGL test harness to compare the performance of different tag designs. For example, the choice of fiducial shape provides a performance trade-off for a tag designer: square tags carry a larger symbolic data payload than a circular tag of the same size, whereas circular tags offer better location and pose accuracy. The test harness can also be used by tag designers to determine whether their particular application idea will function at all, or whether their design is overly optimistic.

The design space for fiducial marker tags is large and currently poorly understood.

Previous investigations into the performance of tag tracking systems have compared implementations rather than fundamental properties. The Cantag framework enables the direct comparison of different tag designs and algorithm choices, providing benefit to fiducial tag designers and application developers alike: new designs may be systematically profiled against each other and the most suitable design for a chosen application can be selected and used without requiring in-depth knowledge of system operation.

9 Acknowledgements

Grateful thanks are due to Andy Hopper for his support. Thanks to Alan Mycroft for his insights and suggestions. This work was supported by the EPSRC.

References

Billinghurst, M., Kato, H., 1999. Collaborative mixed reality. In: Proceedings of the First International Symposium on Mixed Reality. pp. 261–284.

Billinghurst, M., Kato, H., Poupyrev, I., 2001. The MagicBook—moving seamlessly between reality and virtuality. IEEE Computer Graphics and Applications 21 (3), 6–8.

Brown, D., 1966. Decentering distortion of lenses. Photogrammetric Engineering and Remote Sensing 32 (3), 444–462.

de Ipiña, D. L., Mendoną, P. R. S., Hopper, A., May 2002. TRIP: a low-cost vision-based location system for ubiquitous computing. Personal and Ubiquitous Computing 6 (3), 206–219.

Douglas, D. H., Peucker, T. K., 1973. Algorithms for the reduction of the number of points required to represent a line or its caricature. The Canadian Cartographer 10 (2), 112–122.

Eves, H., 1972. A Survey of Geometry. Allyn and Bacon Incorporated, Ch. 6, pp. 256–261.

Fiala, M., 2004. ARTag revision 1, a fiducial marker system using digital techniques. Tech. Rep. NRC 47419/ERB-1117, National Research Council Canada.

Forsyth, D., Mundy, J. L., Zisserman, A., Coelho, C., Heller, A., Rothwell, C., Oct. 1991. Invariant descriptors for 3-D object recognition and pose. IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (10), 971–991.

Halíř, R., Flusser, J., 1998. Numerically stable direct least squares fitting of ellipses. In: The Sixth International Conference in Central Europe on Computer Graphics and Visualization.

Naimark, L., Foxlin, E., Sep. 2002. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In: IEEE International Symposium on Mixed and Augmented Reality. pp. 27–36.

Owen, C. B., Xiao, F., Middlin, P., Sep. 2002. What is the best fiducial? In: The First IEEE International Augmented Reality Toolkit Workshop. pp. 98–105.

Rekimoto, J., Jul. 1998. Matrix: A realtime object identification and registration method for augmented reality. In: Proceedings of Asia Pacific Computer Human Interaction. pp. 63–68.

Rekimoto, J., Ayatsuka, Y., 2000. CyberCode: Designing augmented reality environments with visual tags. In: Proceedings of DARE 2000 on Designing augmented reality environments. pp. 1–10.

Rice, A., Cain, C., Fawcett, J., 2004. Dependable coding for fiducial tags. In: Proceedings of the 2nd Ubiquitous Computing Symposium. pp. 155–163.

Rice, A. C., Beresford, A. R., Harle, R. K., 2006. Cantag: an open source software toolkit for designing and deploying marker-based vision systems. In: Fourth Annual IEEE International Conference on Pervasive Computer and Communications (PerCom).

Rohs, M., Gfeller, B., 2004. Using camera-equipped mobile phones for interacting with real-world objects. Advances in Pervasive Computing, Austrian Computer Society (OCG), 265–271.

Thomas, G. A., Jin, J., Urquhart, C., 1997. A versatile camera position measurement system for virtual reality TV production. In: International Broadcasting Convention. pp. 284–289.

Wagner, D., Pintaric, T., Ledermann, F., Schmalstieg, D., 2005. Towards massively multi-user augmented reality on handheld devices. In: Third International Conference on Pervasive Computing.

Wellner, P., 1993. Adaptive thresholding for the DigitalDesk. Tech. Rep. EPC-93-110, EuroPARC.