Cantag: an open source software toolkit for designing and deploying marker-based vision systems

Conference Paper · April 2006

DOI: 10.1109/PERCOM.2006.13 · Source: IEEE Xplore

Andrew Rice
University of Cambridge

Alastair R Beresford
University of Cambridge

Cantag: an open source software toolkit for designing and deploying marker-based vision systems

Andrew C. Rice
Computer Laboratory
University of Cambridge, UK

Alastair R. Beresford
Computer Laboratory
University of Cambridge, UK

Robert K. Harle
Computer Laboratory
University of Cambridge, UK

Abstract

This paper presents Cantag, an open source software toolkit for building Marker-based Vision (MBV) systems that can identify and accurately locate printed markers in three dimensions. The extensibility of the system makes it ideal for dynamic location and pose determination in pervasive computing systems. Unlike prior MBV systems, Cantag supports multiple fiducial shapes, payload types, data sizes and image processing algorithms in one framework. It allows the application writer to generate a custom tag design and associated optimised executable for any given application. The system includes a test harness which can be used to quantify, compare and contrast the performance of different designs. 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. It presents quantitative analysis of different markers and processing algorithms, which are compared fairly for the first time.

1. Introduction

Developers of pervasive computing systems have long recognised the utility of discovering 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 costs and increased reliability in object identification and localisation at the cost of requiring that specially designed tags are attached 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[1, 11, 12], 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 introduces *Cantag*, 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 design to use 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 describes in detail the types of accuracy and functionality that different pervasive computing applications require, and motivates why different tag designs are suitable for different applications. Section 3 provides an overview of the Cantag system and describes how various system components can be composed with C++ templates to provide an optimised executable. Section 4 summarises some results generated using an OpenGL-based test harness to evaluate the performance trade-offs available to application writers when using different algorithms and tag designs. Section 5 analyses the results of using the Cantag system with real world images. Section 7 contrasts the Cantag system with related work and Section 8 describes the lessons learned and reviews the salient design features an application designer should consider when deploying an MBV system.

2. Design principles

There are several fundamental design decisions that a designer of an MBV system must make:

- *Source image type*: whether the system operates on colour, greyscale or 1-bit (binary) images;
- *Fundamental tag shape*: the shape must provide projective invariants which can be utilised to recover the transformation information necessary for interacting with the tag;
- *Marker coding scheme*: Each marker incorporates a unique element to facilitate identification. This may be template-based (i.e. a unique shape that is recognised through pattern matching) or symbolic (e.g. incorporating a binary code that is read through image sampling). Symbolic codes allow for quantifiable analysis and arbitrary coding schemes.

Systems utilising colour or greyscale information should be able to provide higher accuracy and be read at greater distances than systems using only 1-bit images. However, the price paid for this is increased execution cost when processing the image. We therefore focus initially on processing 1-bit images in order to improve applicability to wearable platforms in particular. There are many pervasive computing applications with different needs from an MBV system. We have identified a number of high-level properties that express these requirements and the inherent trade-offs.

Execution cost is a key issue for many pervasive computing applications. Thus, it is important to investigate the trade-off between computational cost and algorithm accuracy and robustness. For example, projects such as Handheld Augmented Reality[16] perform video overlay on a handheld PDA and thus may be prepared to accept reduced accuracy in order to decrease power consumption or achieve real-time performance.

Payload type can be either symbolic or template-based. Template codes provide visual feedback to the user regarding the functionality provided by a particular tag. This can be useful if tags are used for applications which use the physical juxtaposition of objects to trigger an action; this form of interaction is particularly common in augmented reality applications. Symbolic codes have a number of advantages over template-based systems since they define a quantifiable address space. In particular, varying the size of the data-cells and the use of error-correcting or error-detecting codes provides a tag designer with trade-offs between resilience and data rate[13].

Data capacity of a tag measures the amount of data that a given design can encode; different applications place different requirements upon this capacity. For example, The MagicBook[2] application overlays active content onto the pages of a book and therefore we would hope for a large number of recognisable tags—or have a very short book! However, increasing the number of recognisable tags necessarily reduces the code distance between each tag and so applications requiring a small number of tags might choose low capacity tags in return for increased robustness.

3D accuracy describes how accurately the system can transform between the object co-ordinate frame (in 3D with its z-axis perpendicular to the tag) relative to the 3D camera co-ordinate frame. This information is required for spatial reasoning: calculating the relative relationships between objects relative to one another. Techniques for recovering full 3D transformation information from circular and square tags require expensive computation.

Projective accuracy describes how accurately the system can project 3D object points into the image. This is the typical mode of operation for visual overlay whereby 3D models are rendered over the scene.

2D accuracy refers to how accurately the system can predict points in the image that lie on the same plane as the tag itself. This information is used for decoding the tag but might also be used for simple labelling of items in the scene.

Different applications require different payload types, data capacities, transformation capabilities and execution cost. The Cantag system has been built to allow the application writer to design or select, build, test and deploy the most suitable tag design for the job.

3. 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.

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 using the *rotational invariance* abstraction[13]. Figure 1 shows four example tags.

Once the user has selected the basic tag design, 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 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.

3.1 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. Tag recognisers on a mobile phone might often be able to assume that the required positioning of the phone close to the tag will ensure that the image is evenly illuminated.

The Adaptive threshold algorithm utilises a moving average across the image to choose the threshold value[17]. Systems recognising images with varying light conditions will need to accept the higher computational cost required to maintain a moving average.

3.2 CircleTag

The perspective transformation of a circle is an ellipse[5] 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[8]. 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.

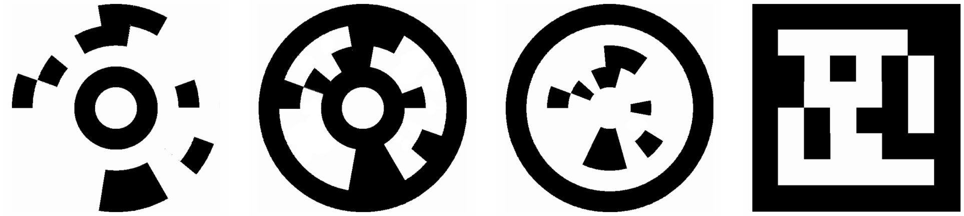
Once the ellipse has been fitted the perspective transformation may then be be derived. The 3D transform algorithm implements an adaption of the Forsyth's ellipse back projection algorithm[7] to recover a general 3D transformation from object co-ordinates to camera co-ordinates. This algorithm is computationally complex but produces accurate 3D information for the tag's position and pose.

Alternatively, the Linear transform algorithm simply scales the located ellipse linearly 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.

3.3 SquareTag

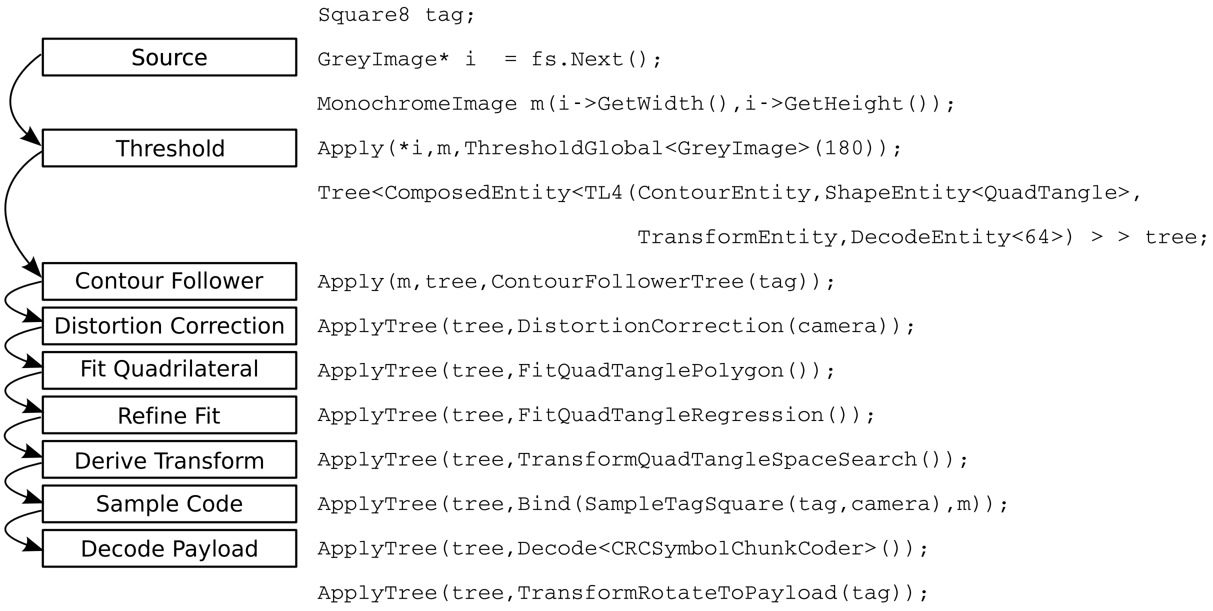
Recognising SquareTags follows a similar process to the circular tags. Perspective projection of a square results in a general quadrilateral and hence the contour follower

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



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

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



```
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.

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[4] 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 on the contour. 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 do *not* require that the tag remain a square and so noise in the image produces a warping of the vertical and horizontal object axes. However, the 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 3D projection co-ordinates is often sub-pixel and so this algorithm is a good choice for systems that do not require 3D position or pose 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 the four-point correspondence problem. This algorithm requires multiple iterations to find a non-linear solution and is therefore computationally expensive to execute and time consuming to implement.

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, an 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.

4. Evaluating 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 to understand 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 source of error is derived from the pixelation of the image. Hence this harness can be used to place a quantitative *upper bound* on the capabilities of a specific tag. 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 remainder of this section summarises the results from profiling the performance of the various algorithm choices before we relate these findings to the real-world behaviour of the system.

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. In the following graphs all distances are measured in unitless dimensions of *tag widths*. The reader may prefer to interpret this as: if the tag is 1m across then all distances are in metres.

The *minimum sample distance* is defined, for a given tag location and pose, as minimum distance between the projected sample point for any cell and its edges. This minimum sample distance gives a measure of how hard the tag is to sample at this position—the smaller the value the less margin for error in picking the sample point. For a particular orientation of the tag this value varies linearly with the size of the tag in the image. The size of the tag is inversly proportional to the distance from the camera projection plane (i.e. along the z axis) along a particular ray. A linear interpolation was applied to these values to find the distance from the camera where the minimum sample distance is one pixel. Figure 3 shows the distance from the camera such that the minimum sample distance is one pixel for a Square-36 and CircleInner-36 tag. Both halves of

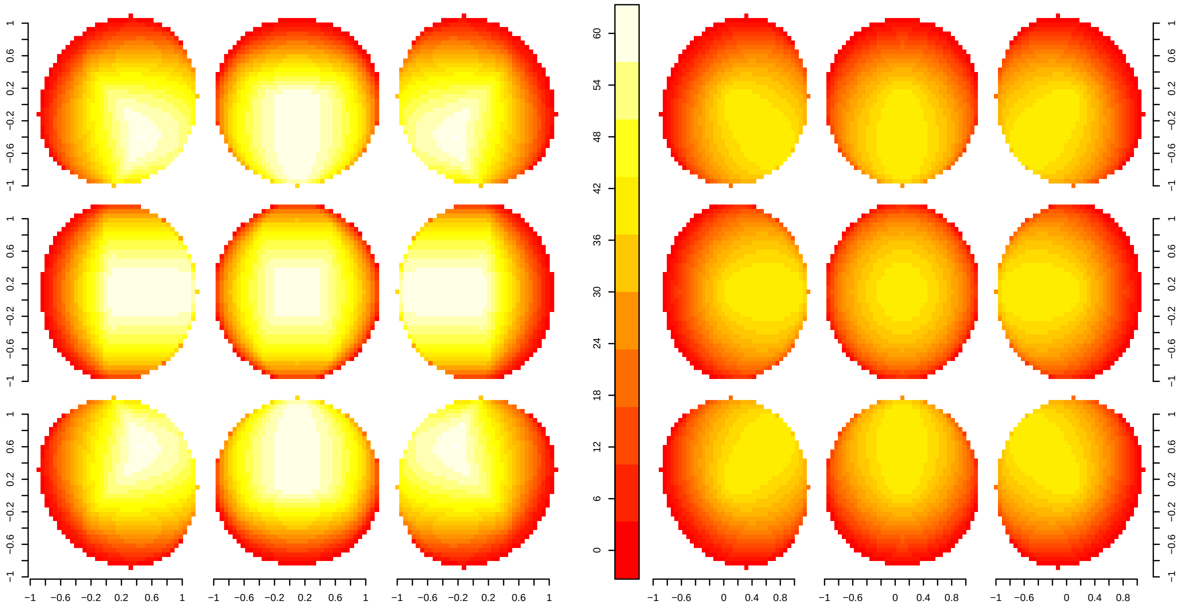


Figure 3. Maximum distances from the camera such that the minimum sample distance is one pixel for Square (left) and Circular (right) tags.

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 distance from the camera such that the minimum sample distance was 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 4 shows the effect of increasing payload size on the minimum sample distance. The tag size in the image such that the miniumum 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 nxn tag to an $(n+1)$x$(n+1)$ 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 and so is the limit on 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).

The minimum sample distance describes 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 5 shows the effect of using the less complex Sim-

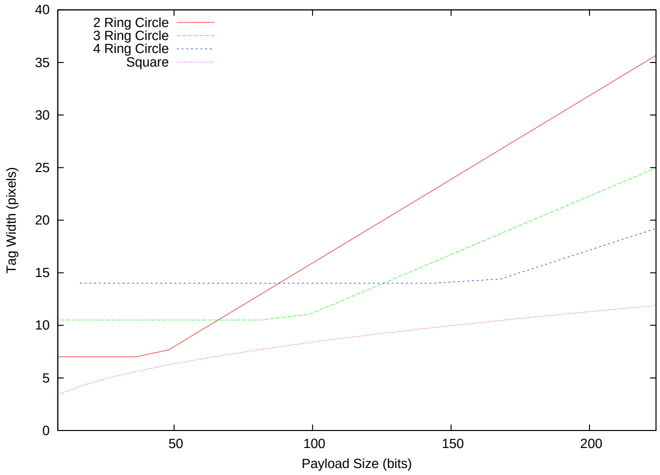


Figure 4. Minimum tag size for varying payload size.

ple 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 succeptible 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 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).

Our earlier statement concerning the relation between the tag's pixel size in the image and its distance from the camera and actual size 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.

Figure 6 shows 3D location error for the SquareNL and CircleSplit3D tags for selected tag inclinations. We notice that as predicted by the test harness the Simple fit algorithm is more succeptible to image noise that the Least squares algorithm particularly when the target tag is fully facing the camera. 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 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 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. A number of trends predicted in the test harness are borne out in the real-world data but the effects of image noise amplify any algorithmic instabilities. We see that shape fitting algorithms are increasingly robust as more contour points contribute to the fitted shape.

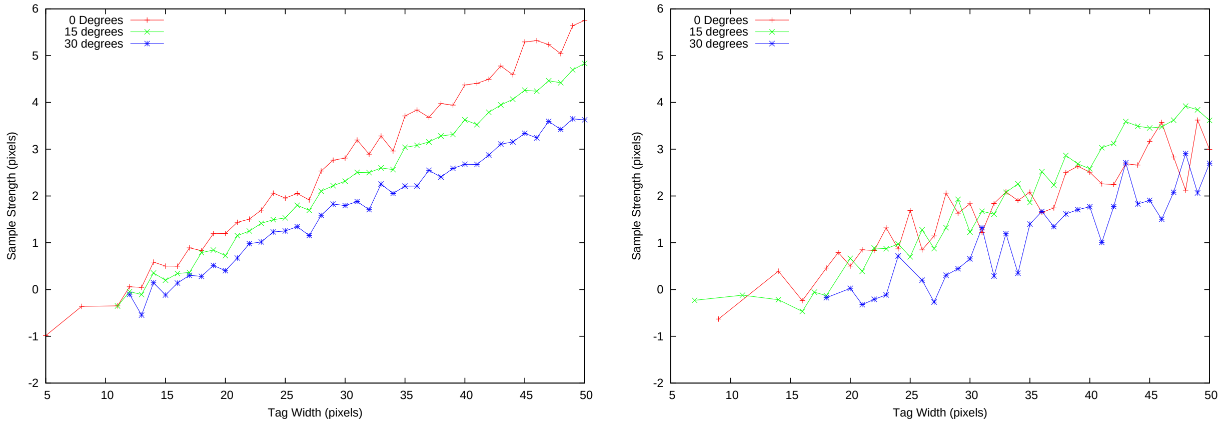
6. Summary

The data produced by the OpenGL test harness and the real-world results suggest a number of high-level rules for designers to bear in mind.

For short-range applications the expected performance of the system is directly determined by the number of pixels visible for 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 sector radius against sector angle.

Circular tags provide more robust location information than square tags especially when using the more simple



(a) Least squares algorithm

(b) Simple fit algorithm

Figure 5. The sample strength of the ellipse fitting algorithms for the CircleInner tag.

shape fitting techniques. This can be seen in Figure 6 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.

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.

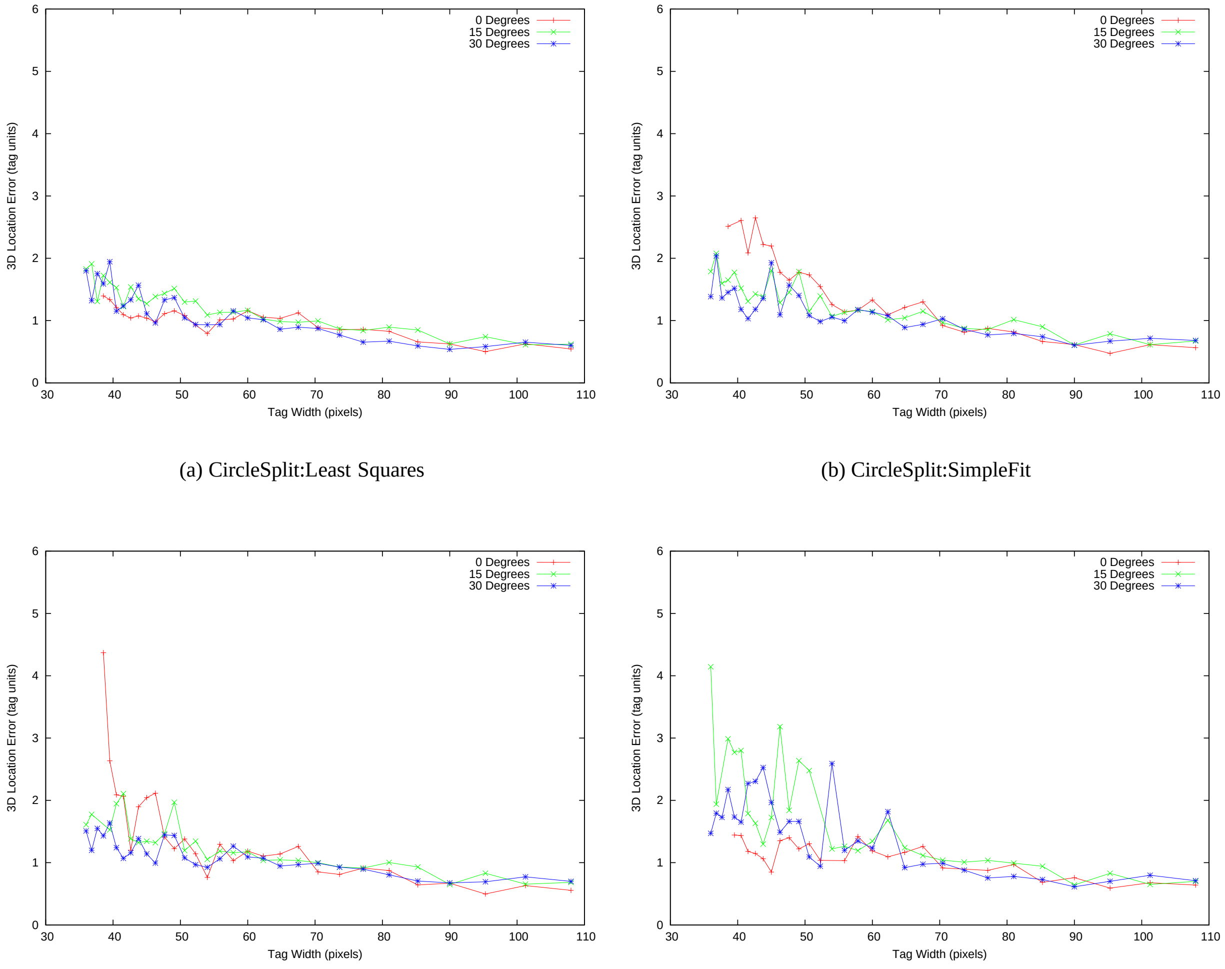
7. Related Work

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[1]. 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)[10]. The addition of a template matching algorithm for decoding the tag data would be sufficient for Cantag to implement the ARToolKit pipeline.

Matrix[11], CyberCode[12] and Rohs' mobile phone-based tag reader[14] 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 straightforwardly integrated into the framework and make use of common steps such as recovering the tag position and decoding the binary payload. The ARTag system[6] 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[3] uses circular tags with a symbolic code arranged around the outside of a circular bullseye. Naimark and Foxlin's tracker[9] also utilises a circular tag with additional asymmetric eyelets to orient the tag. The Free-D camera tracking system[15] 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 al-



(a) CircleSplit:Least Squares

(b) CircleSplit:SimpleFit

(c) Square:Convex Hull with Linear Regression

(d) Square:Curvature

Figure 6. The effect of the shape fitting algorithms on the real-world location error for the Square and CircleSplit tags.

gorithm to read the tag and a bundle adjustment algorithm to estimate position over the set of sighted tags.

8. Conclusion

This paper presented Cantag, an open source toolkit for programmers building pervasive computing applications. The toolkit differs from previous image processing libraries by allowing application writers to customise their tag design based on application needs.

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 by 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, 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.

We have discussed how the Cantag system can be extended to support tag recognition based on techniques other than contour following in a thresholded image. In future work we hope to implement these additional tag designs and assist others in doing the same.

9. Acknowledgements

Grateful thanks are due to Andy Hopper for his support. Thanks to Eben Upton for suggesting the *Simple fit* algorithm and to Christopher Cain for his insights and suggestions. This work was supported by the EPSRC.

References

[1] M. Billinghurst and H. Kato. Collaborative mixed reality. In *Proceedings of the First International Symposium on Mixed Reality*, pages 261–284, 1999.

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

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

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

[5] H. Eves. *A Survey of Geometry*, chapter 6, pages 256–261. Allyn and Bacon Incorporated, 1972.

[6] M. Fiala. ARTag revision 1, a fiducial marker system using digital techniques. Technical Report NRC 47419/ERB-1117, National Research Council Canada, 2004.

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

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

[9] L. Naimark and E. Foxlin. 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*, pages 27–36, Sept. 2002.

[10] C. B. Owen, F. Xiao, and P. Middlin. What is the best fiducial? In *The First IEEE International Augmented Reality Toolkit Workshop*, pages 98–105, Sept. 2002.

[11] J. Rekimoto. Matrix: A realtime object identification and registration method for augmented reality. In *Proceedings of Asia Pacific Computer Human Interaction*, pages 63–68, July 1998.

[12] J. Rekimoto and Y. Ayatsuka. CyberCode: Designing augmented reality environments with visual tags. In *Proceedings of DARE 2000 on Designing augmented reality environments*, pages 1–10, 2000.

[13] A. Rice, C. Cain, and J. Fawcett. Dependable coding for fiducial tags. In *Proceedings of the 2nd Ubiquitous Computing Symposium*, pages 155–163, 2004.

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

[15] G. A. Thomas, J. Jin, and C. Urquhart. A versatile camera position measurement system for virtual reality TV production. In *International Broadcasting Convention*, pages 284–289, 1997.

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

[17] P. Wellner. Adaptive thresholding for the DigitalDesk. Technical Report EPC-93-110, EuroPARC, 1993.