National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

*ARTag Revision 1. A Fiducial Marker System Using Digital Techniques **

Fiala, M.
November 2004

* published as NRC/ERB-1117. November 24, 2004. 46 Pages. NRC 47419.

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

ERB-1117

NRC-CNRC

ARTag Revision 1. A Fiducial Marker System Using Digital Techniques

Fiala, M.
November 2004

Canadä

NRC 47419

NRC 47419/ERB-1117
Printed November 2004

ARTag Revision 1, a Fiducial Marker System Using Digital Techniques

Mark Fiala
Computational Video Group
Institute for Information Technology
National Research Council Canada

Abstract

ARTag is a 2D marker and computer vision system for *Augmented Reality*, a *Fiducial marker system*, that was introduced in a prior NRC publication [6]. *Augmented Reality* (AR) is an emerging display paradigm, an enabling technology of AR is vision based pose tracking. Pose can be found accurately and with low cost using a camera as the only special hardware. *Fiducial marker systems* consist of patterns that are mounted in the environment and automatically detected in digital images using an accompanying detection algorithm. They are useful for AR, robot navigation, and general applications where the relative pose between a camera and object is required. *ARTag* is a marker system that uses digital coding theory to get a very low false positive and inter-marker confusion rate with a small required marker size, employing an edge linking method to give robust lighting and occlusion immunity. *ARTag* markers are bi-tonal planar patterns that consist of a square outline with a digital 36-bit word encoded in the interior. The digital word contains a unique ID number protected from false detection with the digital code techniques of checksums and forward error correction (FEC) providing very low and numerically quantifiable error rates. ARTag's performance is theoretically or experimentally examined for nine characteristics important to AR; *false positive* and *false negative detection* rates, *inter-marker confusion* probabilities, immunity to lighting, immunity to occlusion, minimal marker size, vertex jitter, marker library size, and speed performance. This publication further characterizes ARTag and provides more detailed information and experimental results useful for those interested in utilizing ARTag, and those interested in fiducial marker systems themselves.

Contents

1 Introduction **4**

2 Planar Marker Systems **6**
2.1 Functionality of ARToolkit 9

3 ARTag **11**
3.1 Quad Detection 11
3.2 Digital Processing 15

4 False Positive Marker Detection **17**
4.1 Uniqueness of Markers: Reducing Inter-Marker Confusion 20
4.2 Designing the ARTag Library 23
4.3 Adressing Reflections 24
4.4 Recommended Subset of the ARTag Library 24
4.5 Minimum Pattern Size and Processing Time 30

5 Processing Time **36**

6 Jitter **39**

7 Conclusions **42**

1 Introduction

ARTag is a combination of special 2D patterns and computer vision that was introduced in August 2004. A preliminary report was issued titled *ARTag, An Improved Marker System Based on ARToolkit* [6] and since then ARTag has undergone some improvements. A version, *ARTag_rev1* was released for demo evaluation which has some differences from the system described in the first publication. The edge-linking first stage of ARTag was not discussed and its ramifications, the library is reported as 2002 markers for clarity instead of 2046 markers with 44 non-recommended codes, further experiments were performed to update the speed performance information, and more extensive experiments were done to characterize the minimal marker size and vertex jitter.

Designing markers to add to the environment for robust detection in camera and video imagery is a computer vision application useful to situations where a camera-object pose is desired such as *Augmented Reality* (AR), position tracking, photo-modeling and robot navigation. 2D planar patterns can be added to the environment and recognized in the camera images. A 2D planar marker system consists of both a set of planar patterns and the associated computer vision algorithms to recognize them in an image. *ARToolkit* [7, 9] is a popular such system which contains a 2D planar fiducial marker system, it is used in many Augmented Reality applications and is used frequently in this paper for a basis to evaluate ARTag.

Metrics describing performance of fiducial marker systems are; 1) the *false positive* rate, 2) the *inter-marker confusion* rate, and 3) the *false negative* rate. The false positive rate is the rate of falsely reporting the presence of a marker when none is present. The inter-marker confusion rate is the rate of when a marker is detected, but the wrong id was given, *i.e.* one marker was mistaken for another. Finally, and possibly the least serious, is the false negative rate, where a marker is present in an image but not reported. The false positive and false negative rates are at odds with one another, and represent a trade-off between missing a marker and seeing a non-existent one. Another metric is 4) the *minimal marker size* which is the size in pixels required for reliable detection. The smaller the marker needs to be in the image, the larger the usable range from the camera the marker system can be used at.

ARTag is a bi-tonal system containing 2002 planar markers, each consisting of a square border and an interior region filled with a 6x6 grid of black or white cells. 1001 of ARTag markers have a black square border on a white background, and vice versa for the other 1001. Fig. 2 shows some example ARTag markers. The associated algorithm for detection first locates quadrilaterals which may be perspective views of the marker border, then the interior is sampled into 36 binary '1' or '0' symbols. Further processing is in the digital domain providing a non-linear response giving very low *false positive* and *inter-marker confusion* rates. With ARTag, the probability of

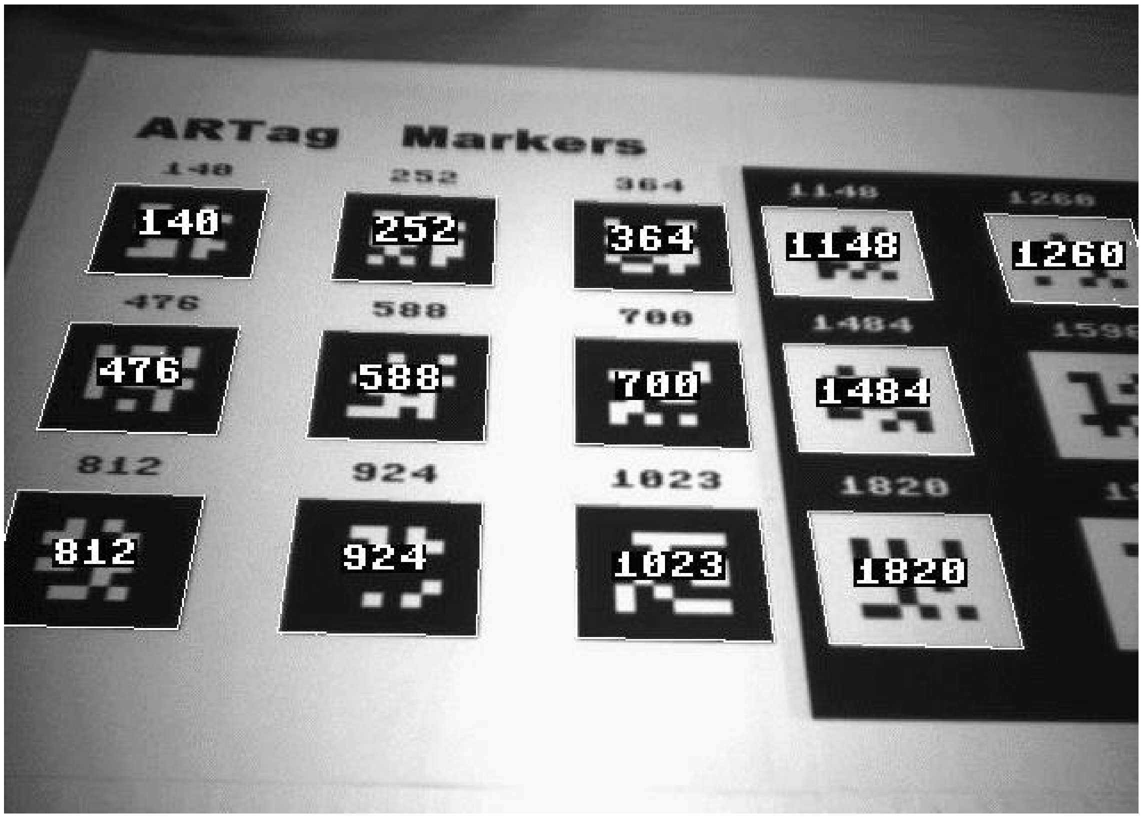


Figure 1: *Markers detected in an image. Overlaid white border and ID number show automatic detection.*

falsely identifying one marker for another, or a piece of the background as a marker, is a probability of < %0.0078. Fig. 1 shows ARTag markers being detected in an image.

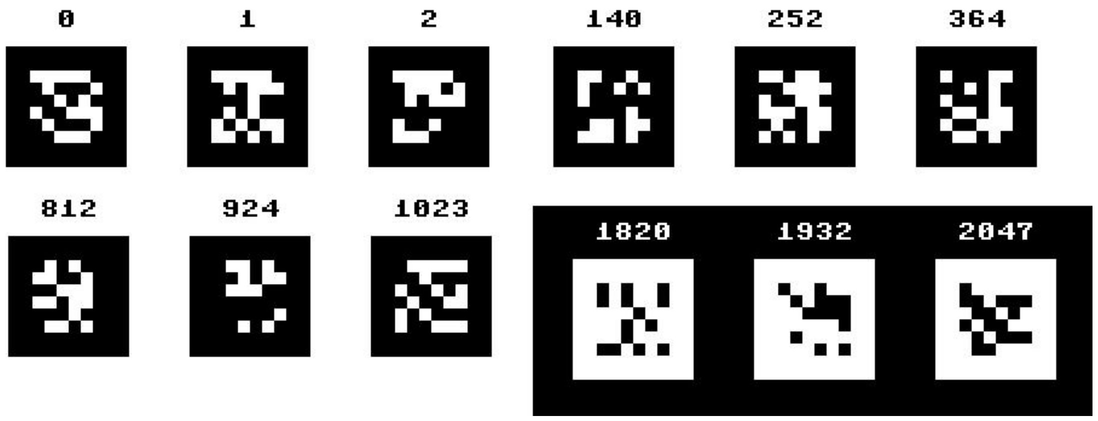


Figure 2: *ARTag markers. ARTag markers are bi-tonal planar marker patterns consisting of a square border and a 6x6 interior grid of cells representing logic '1' or '0'. 12 out of the library of 2002 markers are shown.*

2 Planar Marker Systems

Many of the practical machine vision systems used in industry use two dimensional patterns to carry information in a manner similar to the ubiquitous bar code seen on consumer products. The purpose is to carry information, not to localize as is needed for applications such as augmented reality. *Cybercode* [15] is one early example of a 2D grid of black and white squares used to communicate digital information. The US Postal Service uses the *Maxicode* marker to convey shipping information (Fig.3). *DataMatrix* and *QR (Quick Response)* are two other examples designed to contain information are used in industrial settings for part labelling (also shown in Fig.3). The above three all use or have provision for error correction methods to recover the data when some of the bits are incorrectly read. ECC200 [1] is a standard for *DataMatrix* 2D patterns and uses Reed Solomon error correction, which can recover from situations where part of the information read from the pattern is corrupted. DataMatrix and QR are used for *Direct Part Marking* (DPM) to identify and convey information along an assembly line.

DataMatrix, Maxicode and QR all have a common thread of encoding data using binary values for reflectance, the pattern is typically bitonal reducing the decision made per pixel to a threshold decision. This reduces the lighting and camera sensitivity requirement and removes need for linearization of the signal (*i.e.* no attempts are made to identify shades of grey). Another component is that of redundant information allowing for error detection and correction to increase the overall success rate. Error detection and correction is something not seen as much in computer vision as in other fields such as telecommunications, and is a well understood class of methods to statistically improve the data integrity rate to a very reliable level.

In general, DataMatrix, Maxicode and QR are useful for encoding information, but are not as useful for fiducial marker systems for two reasons. Firstly they are not intended for, and won't function well, in situations with large field of views and the perspective distortion that introduces. And when detected do not provide enough image points for 3D pose calculation. DataMatrix can only adjust for affine warping by using the 'L' shaped locator, *i.e.* with 3 points instead of the 4 required to correct for perspective distortion. Secondly they typically require a large area in the image limiting the range at which the markers can be used.

Locating and identifying simple planar patterns is also used by several photogrammetry, position tracking, and augmented reality systems where less information is carried in the marker, typically only enough to identify it from others. In applications such as photogrammetry, the size of the marker is an issue. In general, the less dense the information is in the marker, the less the minimum pixel requirement is and as a result, the greater the range of distance that the marker can be from the camera.

Small markers can be made by encoding a ring of segments around a circular dot [10]. Several commercially available circular fiducial marker systems exist using a single broken annular ring around a circular dot such as Photomodeler's "Coded Marker Module" [2]. Naimark and Foxlin [13] describe a system extending the number of rings. The number of possible markers is limited, the camera resolution will limit the number of segments that the annular ring can be broken into. The false positive and inter-marker confusion rates are not reported for these systems but due to the small number of digital bits that can be encoded (15 bits for [13]), they will suffer from high false detection and inter-marker confusion rates or small library sizes (if some bits are used for redundancy) or most likely both. For AR and other applications where not just the location but the pose must be determined, these markers are not as useful due to the difficulty in determining the plane they lie upon. Determining the camera pose involves simultaneous detection of several markers.

The distinguishing feature of systems designed for augmented reality such as ARToolkit, ARSTudio [12] and Rekimoto's *Matrix* markers. is that in many cases only one marker is usually visible. Therefore the fiducial marker must have some distinct points, at least four, so as to extract orientation with perspective distortion. ARToolkit and Matrix, for example, use the quadrilateral outline to accurately locate

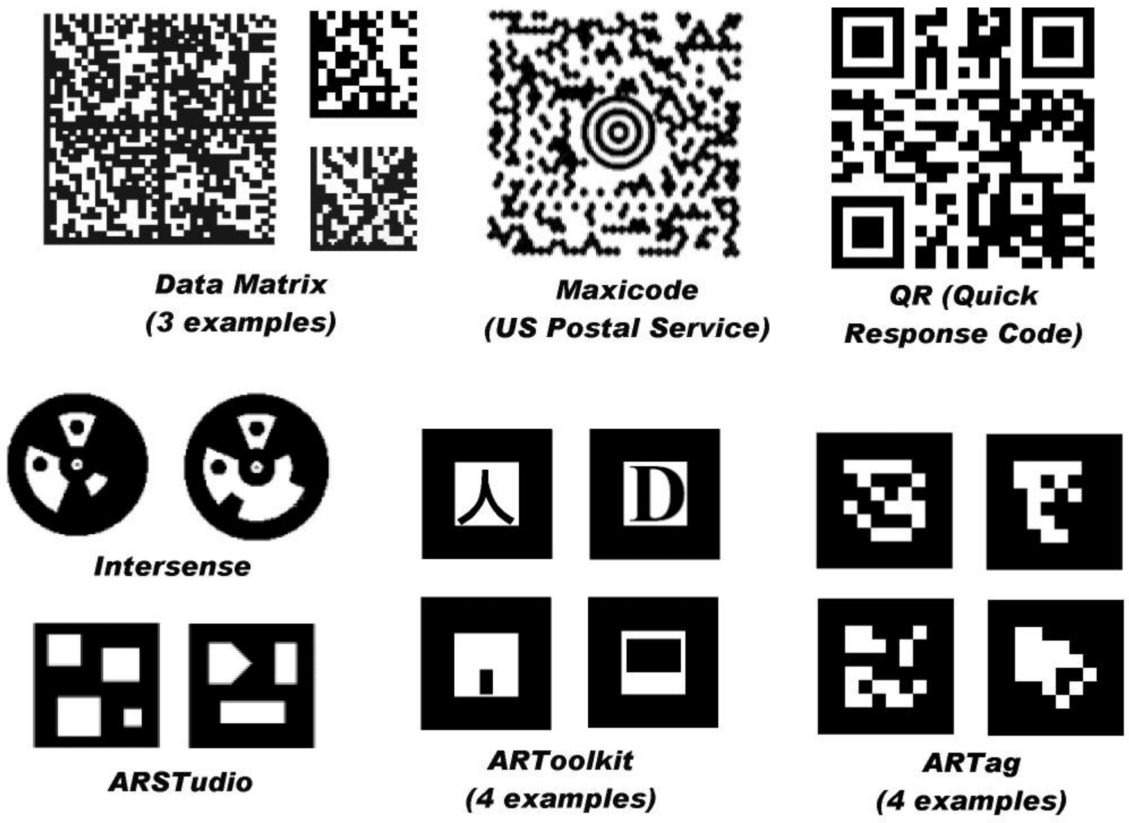


Figure 3: *Several planar pattern marker systems. DataMatrix, Maxicode and QR are industrial systems used for carrying data. The circular Intersense markers are used in position tracking. ARStudio and ARToolkit are patterns designed specifically for augmented reality applications. ARTag is the new marker system introduced in this paper.*

the four corner points to a sub-pixel accuracy. These four points are on the furthest extent of the pattern to get as much orientation accuracy as possible.

Matrix is a digital marker system briefly outlined in 1998 in [14] that uses digital encoding, a checksum, and a quadrilateral outline. Unfortunately there is not much information on this system and Rekimoto appears to have left the quadrilateral border out in further work building something similar to DataMatrix.

Zhang [17] and Claus [5] perform a survey of several fiducial marker systems including ARToolkit with respect to processing time, identification, image position accuracy with respect to viewing angle and distance. Matrix is mentioned but was not available for his analysis. ARToolkit [7] is a marker system devised by Dr. Hirokazu Kato of Osaka University, Japan, and supported by the University of Washington [1]. ARToolkit is popular because it is simple, relatively robust, and freely available. ISMAR [3] is an augmented reality conference where many applications use ARToolkit.

ARToolkit markers consist of a square black border with a variety of different patterns in the interior. The quadrilateral black outline is used to calculate a homography to define a sampling grid inside the pattern which is sampled to provide a 256 (or 1024) element feature vector which is compared by correlation to a library of known markers. ARToolkit outputs a so called *confidence factor* which is the result of a normalized vector dot product between the sampled 16x16 (or 32x32) vector and the stored prototypes. Presence of a marker is simply determined by a threshold value on this confidence value.

ARToolkit is useful for many applications, but has a few drawbacks. The use of correlation to verify and identify markers causes high false positive and inter-marker confusion rates. The user typically has to capture prototypes of each marker as seen with the camera and lighting of their application, plus adjust the greyscale threshold in a trade off between these two rates and the false negative detection rate. Only markers whose borders can be defined by this fixed threshold can be found. Also, the uniqueness of the markers deteriorates as the library size increases. The processing time also rises as that the normalized center region of each quadrilateral must be correlated with all marker prototypes in the library. To address the four possible rotations and possible differences in lighting, twelve prototypes are stored for each marker. Owen [4] proposes an ARToolkit library based on spatial frequency components to extend the library but ARToolkit's inter-marker confusion rate still restricts the library size.

2.1 Functionality of ARToolkit

ARToolkit is a widely used fiducial marker system for AR, it warrants a deeper description both to illustrate some of the computer vision issues and to serve as a

[1]http://www.hitl.washington.edu/artoolkit/

comparison to the proposed new ARTag. In many ways, ARToolkit represents the state of the art for AR marker systems, at least for AR marker systems available to most researchers and developers. ARToolkit [7] is widely used [3] by AR and Human Computer Interaction (HCI) systems due to it's available source code and ubiquity of use.

ARToolkit markers (Fig. 3 middled bottom) are planar and can easily be printed out and mounted to a flat card or wall. The markers consist of a square black border enclosing a pattern that is compared to several stored patterns. The marker recognition occurs in two stages; firstly recognizing quadrilateral boundary contours, and secondly by correlating this interior pattern with the known patterns.

ARToolkit first finds black quadrilateral borders by finding connected groups of pixels below a set level, the contours of these groups are found and those contours with four straight sides are identified as potential markers. This greyscale thresholding method is also used in *Matrix* [14]. These first few stages are shown in the middle three images in Fig. 4.

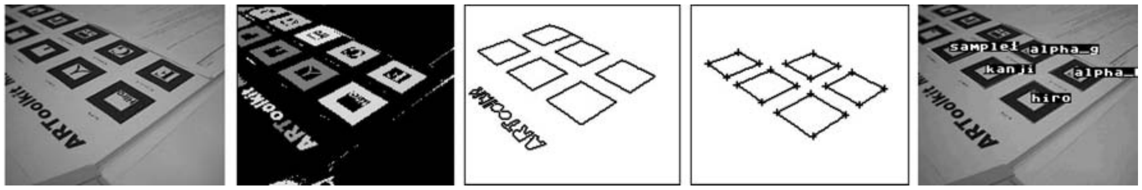


Figure 4: *The first few stages of ARToolkit marker extraction. ARToolkit takes an input image and a grey level threshold parameter and reports the location of markers detected in the image. Left to right; 1) the original image, 2) connected components of pixels with a grey level less than the threshold parameter, 3) external border contours extracted from the connected regions, 4) quadrilateral contours identified and their corners located, and 5) extracted markers labelled and overlaid over the input image.*

The corners of the quadrilateral contours are used to remove the perspective distortion and bring the internal pattern to a canonical front view. The four corners are used to define a homography which is used to sample a N x N grid of greyscale values inside (typically 16 x 16 in the original version although some users modify this to 32 x 32). This region is correlated with several reference grids loaded in from pattern files associated with each marker id. ARToolkit outputs a so called *confidence factor* which is the result of a normalized vector dot product between the sampled 16x16 (or 32x32) vector and the stored prototypes. Presence of a marker is simply determined by a threshold value on this confidence factor.

To use a marker in ARToolkit, one needs both a printable image of the marker to mount on a flat panel, and a corresponding pattern file. The pattern file contains 12 reference grids, which are three versions of each of four possible rotation positions.

The three versions are intended to be taken at different lighting and distance conditions to span the range of possible appearances of the marker when seen by the camera. Instead of rotating the sampled N x N grid, there is instead four pre-rotated versions available in the pattern file. Fig. 5 below shows three markers and what the data inside their corresponding pattern files look like.

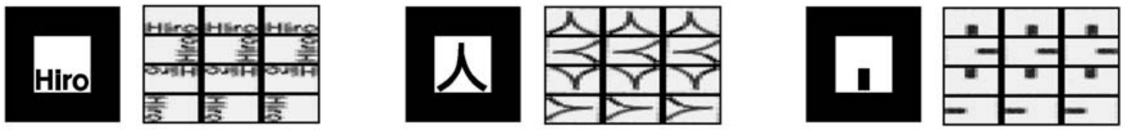


Figure 5: *ARToolkit markers and their corresponding pattern files. (Left to right); 1) the Hiro marker image, 2) the pattern file* patt.hiro*, 3) the Kanji marker image, 4) the pattern file* patt.kanji*, 5) the Sample1 marker image, and 6) the pattern file* patt.sample1.

3 ARTag

ARTag is a planar pattern marker system that has 2002 markers consisting of a square border like ARToolkit, but the detection of the border is edge-based and processing of the internal pattern is performed with a digital approach instead of correlation. The square border used in both systems allows them to be used for AR since the four prominent corner points allow the full extraction of the 6 degree of freedom (DOF) of the relative marker to camera pose (assuming the camera focal length is known).

As with ARToolkit, ARTag locates potential marker projections in an image by first finding the four sided border contour, and using the corner points to define a homography to create a sampling grid to analyze the interior pattern to determine if it's a marker pattern and to identify it if so. The interior is sampled to determine a digital '0' or '1' for each grid point and subsequent processing is in the digital domain.

ARTag finds quadrilateral contours with an edge based method, and does not need a greyscale threshold as does ARToolkit.

Several ARTag markers are shown in Fig. 6. The whole marker is 10 x 10 units, with a border of thickness 2 units leaving 36 cells in the interior to carry information. Each cell is only black or white and carries one bit of digital data. Thus a 36-bit word can be extracted from a camera image of the marker once the boundary is determined.

3.1 Quad Detection

Quadrilateral contours are located in the image which may belong to the outside border of a marker. They are found in ARTag with an edge based method, edge pixels

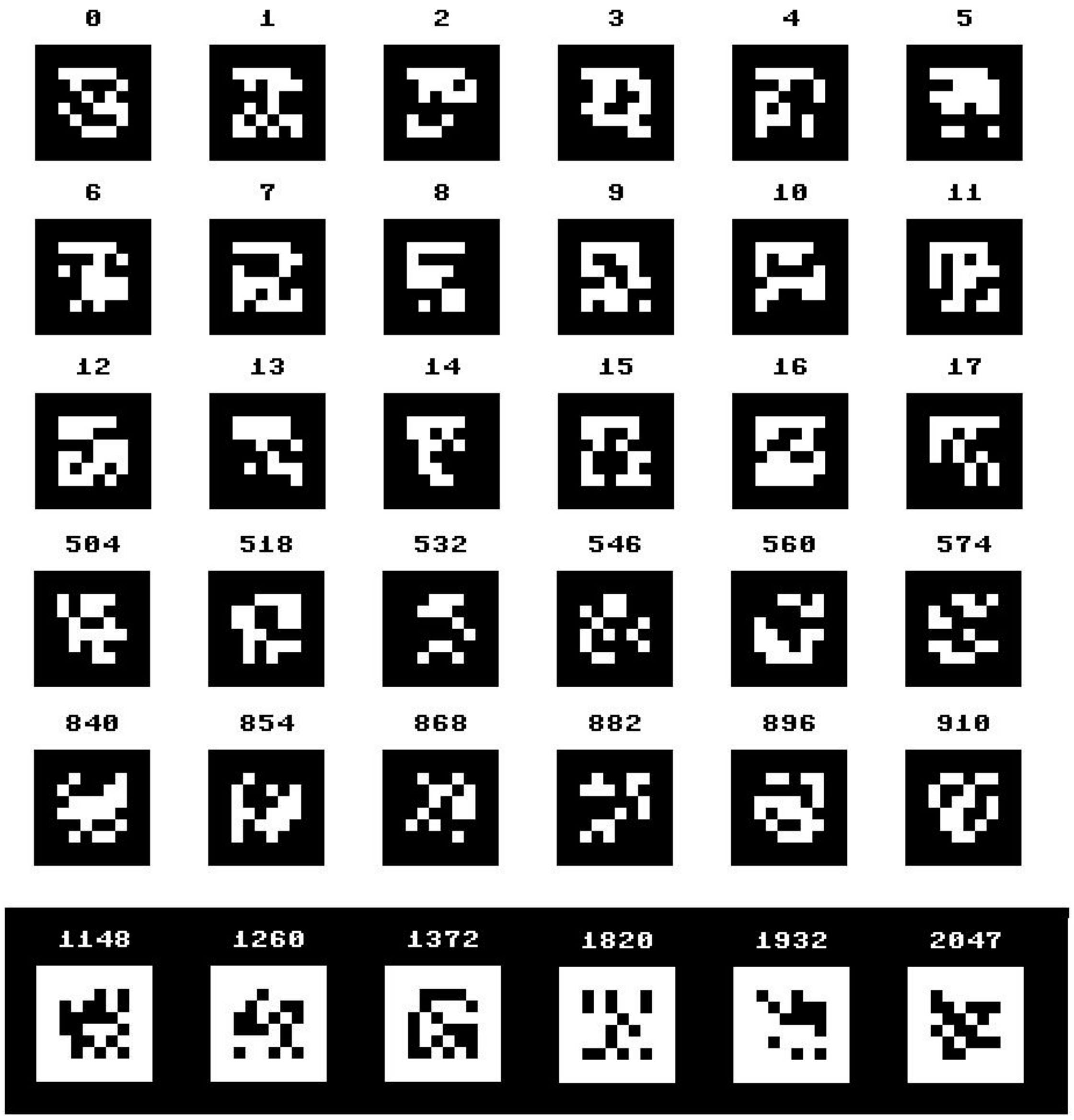


Figure 6: *ARTag markers. 36 out of the range of 2002 markers in the ARTag marker library. A 10-bit code is contained in the interior pattern, protected by checksums and forward error correction (FEC) to greatly reduce false positives and inter-marker confusion. Unlike ARToolkit which only uses a black border on a white background, ARTag uses both polarities of border and background to extend the library size. The inner codes are repeated for markers 1024-2047 where the border is white on black.*

are thresholded and linked into segments, which are in turn grouped into "quads". The four corners of the quad boundary are used to create a homography mapping to sample the marker interior. Fig. 7 shows an image, its extracted line segments, quads, and quads in which ARTag marker codes were found in the interior.

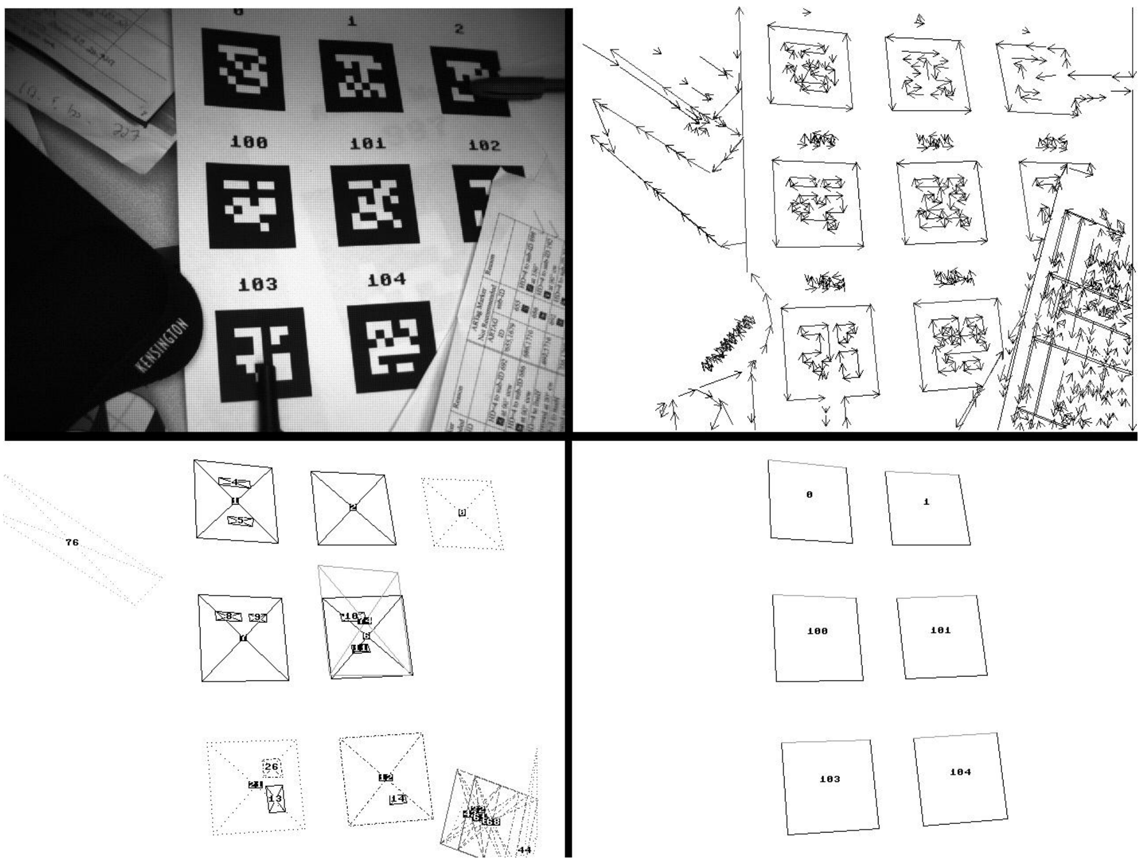


Figure 7: *Quad Extraction: finding ARTag markers. (Upper left) original image, (Upper Right) straight line segments found in image, (Lower left) line segments grouped into quadrilaterals, (Lower right) ARTag markers found from quadrilaterals whose interiors contained a valid ARTag code. Note that Marker #2 (upper right of camera image) did have its quadrilateral border located, but failed the interior validation due to the pen occluding too many of the data bits.*

The edge based approach gives some performance improvement over the greyscale region thresholding approach of ARToolkit. In ARToolkit, groups of connected pixels below a specified threshold are found, and those possessing a quadrilateral boundary are used as potential markers. A single threshold value will not work for markers under different illumination and image capture conditions, even within the same image. Quite often users of ARToolkit have modified it to perform local thresholding or have called the marker detect function multiple times with different thresholds for a single image. Having a spatial derivative of greyscale intensity threshold instead of a simple greyscale intensity threshold allows markers to be found under less controlled lighting conditions. Indeed, the 'white' level of one marker edge may be even be darker than the 'black' level of the other side and the marker still be detected.

Another advantage to ARTag's edge based approach is the ability to still detect marker outlines in the presence of occlusion. In Fig. 7, two of the markers (ID's 103, 104) have a side broken or a corner missing but are still detected by heuristics of line segments that almost meet. Fig. 7 (lower right) shows the occluded quads drawn with dotted lines indicating that they were not obtained from the perfect grouping of four segments.

ARTag has a greater immunity to lighting variation than ARToolkit, an example is shown in Fig. 8 where all markers on a panel of varying illumination are detected by ARTag whereas only one section is visible at a time with ARToolkit.

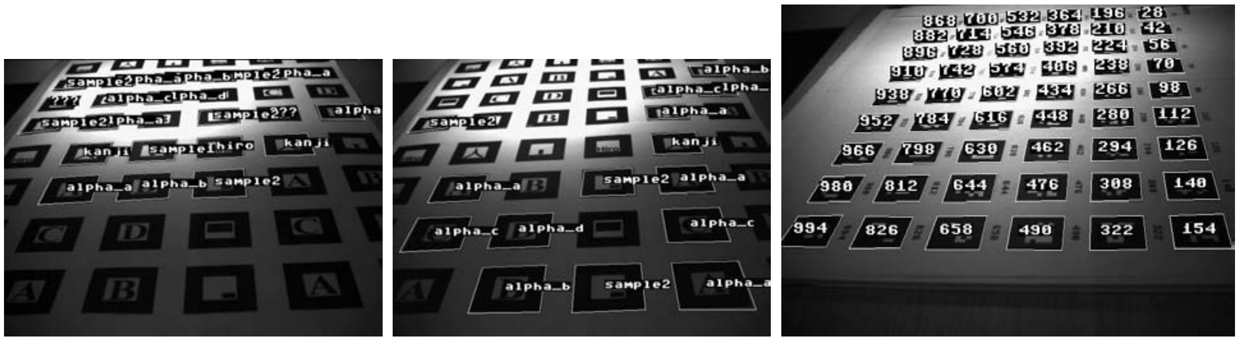


Figure 8: *Marker detection with challenging lighting. ARToolkit performance (Left) and (Middle) demonstrating how a threshold based method cannot recognize all markers simultaneously. (Right) shows how ARTag recognizing all markers. The recognition of a marker is indicated by the overlaid box and number.*

ARTag also can detect markers in the presence of occlusion, a side can be broken and several corners can be missing by occlusions by objects or the image boundary, whereas any slight disturbance in the boundary of an ARToolkit marker results in a failed detection (Fig. **??**.

To further reduce the false negative rate, ARTag searches for quads at three scales. The line segment extraction and grouping into quads is done on the original image size, on a resampled version of half the width and height, and on that of a quarter size. This allows the detection of borders which may be blurry and not have an spatial derivative above the threshold. [2]

[2]To speed performance on lower processing power systems, the quad detection can be turned off for the full resolution speeding up ARTag by about 3-4 times.

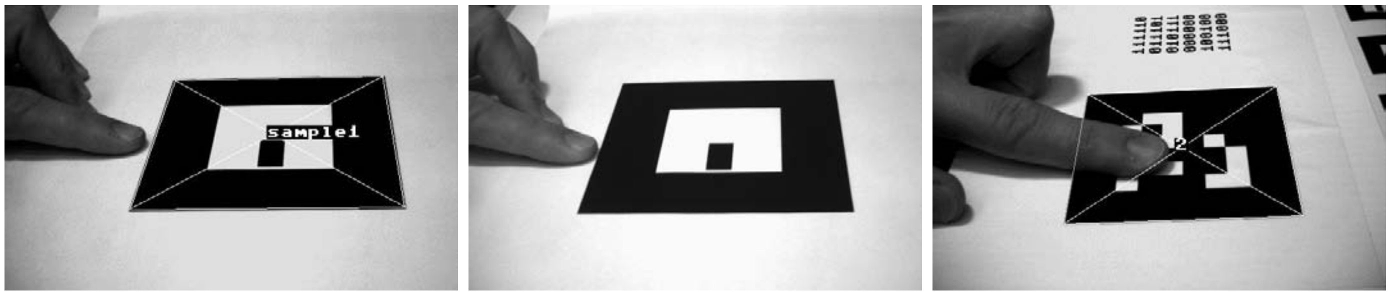


Figure 9: *Marker detection under occlusion. ARToolkit performance (Left) and (Middle) shows how recognition is lost if even a small piece of the border is occluded. (Right) shows how ARTag markers can be recognized even despite large occlusions.*

3.2 Digital Processing

Once quadrilateral border contours have been located, the internal region is sampled with a 6 x 6 grid and assigned digital symbols '0' or '1'. The threshold applied is derived from the intensities found around the quad border. All subsequent processing to verify and identify the marker is performed digitally. Four 36-bit binary sequences are obtained from the 2D 6 x 6 digital symbol array, one for each of the four possible rotation positions. Only one of the four sequences may end up being validated in the decoding process. The 36-bit binary sequence encoded in the marker encapsulates a 10-bit ID using digital methods. The extra 26 bits provide redundancy to reduce the chances of false detection and identification, and to provide uniqueness over the four possible rotations. The *Cyclical Redundancy Check* (CRC) and *forward error correction* are digital methods used to identify if the 36-bit code is part of the ARTag marker set, and to extract its ID.

These methods use a digital algebra called *GF-2* or *Modulo-2* mathematics and involves concepts of addition using logical XOR, convolution and deconvolution operators, and *generating polynomials* which are prime numbers in this base 2 number system. It is beyond the scope of this paper to explain other than to describe that in practice one manipulates short binary symbol sequences with various operators, the most important one to ARTag marker decoding is the deconvolution/division operator. The reader is directed to [16], digital communications and storage texts, and standards documents for more information. In two stages of decoding ARTag markers, digital codes are divided by specially chosen binary polynomials where the dividend and remainder are both used.

The system can be abstractly described as a communication system, where a 10-bit ID is attempted to be sent through a medium of image capture to be received by the ARTag vision software. The creation of a marker pattern from an ID is the encoding phase, and the recognition of an ID from the extracted 36-bit code is the decoding phase.

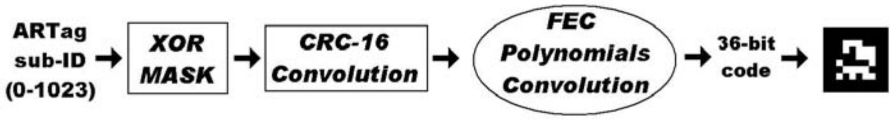


Figure 10: *Digital encoding process : creating ARTag markers. A sub-ID number (the lower 10 bits of the ARTag ID) is converted to a 10-bit binary sequence which goes through several stages to produce a 36-bit binary sequence which is encoded in the marker as white and black cells.*

There are three main stages for encoding when creating the 2D pattern to mount in the environment, with their operations performed in reverse when finding ARTag markers. The digital encoding operations are shown in Fig. 10 below. An ARTag marker image is created by filling the marker according to the 36-bit sequence created by this encoding from an input ID number 0-2047 (excluding 682 and 1706). The ID range 0-2047 is spanned by an 11-bit binary number, the MSB of which decides if the border will be black on white or vice versa. The remaining lower 10 bits are called the *sub-ID* herein.

The printing of the marker pattern, the reflectance of the marker, lighting, other objects in the scene, light capture and digital image formation by the camera including noise, and perspective pose of the marker all constitute the "communications medium". After a 6 x 6 grid of binary symbols is extracted from the image, the digital decoding steps outlined in Fig. 11 are performed and the verdict of ARTag marker presence or not is decided, and the ID reported if present.

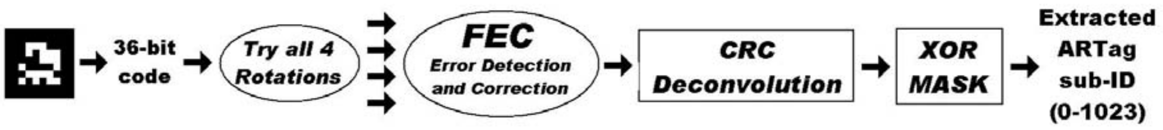

36-bit code
Try all 4 Rotations
FEC
Error Detection and Correction
CRC Deconvolution
XOR MASK
Extracted ARTag sub-ID (0-1023)

Figure 11: *Digital decoding process : confirming and identifying ARTag ID's in the binary pattern from the interior of an ARTag marker. The binary pattern from the marker as seen in the camera image is converted into four possible 36-bit codes for each possible rotation. Each code passes through the FEC stage which can detect and correct some bit errors, the result is then analyzed by a CRC checksum-like procedure to verify if it belongs in the ARTag marker set. If it is, a payload 10-bit binary number (sub-ID) is extracted and combined with the border polarity and reported as a located ARTag marker.*

The XOR operation is used to scramble the codes a bit since it's expected that users would use the lower numbers 0,1,2,... etc and to make the ID of 0 usable. The CRC-16 polynomial (also known as CRC-CCITT in the fields of data storage and communication) is $x^{16} + x^{12} + x^5 + 1$ and is applied by convolving the XOR'd ID with the binary string *10001000000100001*. The deconvolution in the decoder is similar to a division operation yielding a dividend and remainder. The remainder must be 0 otherwise the code is considered to not be from an ARTag marker, only $\frac{1}{2^{16}}$ of the possible binary sequences pass this test protecting against random codes found from quadrilateral objects in the camera view that are not ARTag markers.

The forward error correction (FEC) decoding block is the most sophisticated digital processing component of the ARTag system and allows several erroneous bits in the input 36-bit code to be detected and repaired. This increases the false positive rate by a bit but improves the false negative rate by recognizing codes that are close to a correct code, that are likely an ARTag code with a sampling error due to sources such as an imperfect threshold, misalignment of the detected quadrilateral border, specular reflections inside the pattern, partial occlusion, and general image noise.

Two ARTag ID numbers, 682 and 1706, are absent from the ARTag marker library reducing the library size to 2002. The reason for this is that the sub-ID 682 is a degenerate case that leads to a 36-bit code containing all 0's. This would translate to a fully black or white interior which will be frequently falsely detected in the environment. The library is further reduced to 2002 markers by the removal of 44 ID's as shown in Section 4.2 to improve the inter-marker confusion rate.

4 False Positive Marker Detection

One failure mode of a marker detection system is when a marker is erroneously reported when it does not exist, *i.e.* the system reports that a marker is present when it is not. This is a problem with ARToolkit (Fig. 12), reducing false positives is the motivation for much of the effort spent on implementing ARToolkit. Improving performance usually means creating the ARToolkit pattern files for a specific application, using the same camera and lighting as used in the application.

ARTag processes the internal pattern differently, it is sampled and processed as a digital code and has a much lower false positive rate which can be mathematically described. $1001 \cdot 4 = 4004$ of those digital codes map to a correct ARTag marker viewed from one of four orientations. There are 36 points sampled within the pattern, giving $2^{36} = 68.7$ billion digital codes from an arbitrary randomly filled quadrilateral. The forward error correction accepts %35.6 of them (24.5 billion) as "correctible". A correctible code is one that either contains 0-N error bits from a correct ARTag code (N=number of bits the FEC can correct, =2 in the first version of ARTag released), or one that contains more error bits but fools the FEC into thinking it was corrected.

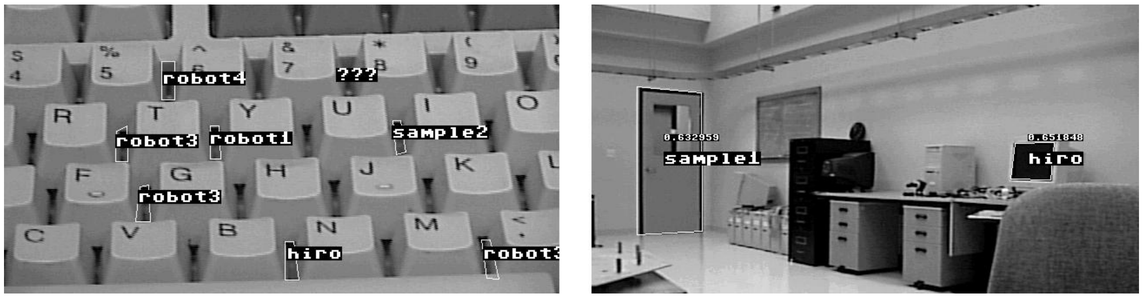


Figure 12: *Examples of ARToolkit false positives. Detected markers are overlaid over the image. Black quadrilateral regions, such as between the keyboard keys, or the computer monitor or doorway have interior pixels that correlate to one of the marker definitions with a c.f. value above the threshold resulting in a false positive detection.*

With N=2, each valid ARTag 36-bit code would be mapped from $1 + 36 + 36^2 = 1333$ 36-bit codes, yielding $4004 \cdot 1333 = 5.34$ million 36-bit pattern interiors which would cause ARTag to report the presence of a marker. So the probability of a false positive detection from a random 36-bit number is $\frac{5.34 \cdot 10^6}{68.7 \cdot 10^9} = 7.8 \cdot 10^{-5} = \%0.0078$, or about one in 12,800. Therefore a non-marker quadrilateral has a $\%0.0078$ chance of producing a false positive marker event assuming equal likelihood of pattern interiors (which is not usually the case giving an even more rare probability). Most natural (non-marker) quadrilaterals in the environment are solid black, or contain much less entropy than the ARTag pattern interiors.

A comparison experiment was performed with several image sequences which do not contain any ARToolkit or ARTag markers to collect statistics on false positives. Both the ARToolkit code and the ARTag library were compiled into one program so that they both saw the exact same image frames. Table 1 shows the results for marker detection with video from several cameras which were moved around our lab in which no ARToolkit or ARTag markers were present, thus any marker detection is a false positive. A movie was also used as an image sequence in order to get more random and varied imagery than video footage from our lab.

Looking at the results in Table 1, one could suggest simply raising the threshold c.f. factor to 0.90 and thus avoid all false positives for ARToolkit. However, with the same conditions, the c.f. had to be lowered to 0.75 to obtain a practically usable false negative rate.

No ARTag false positives were seen in any of the experiments, or noticed when using the system, however the probability is still non-zero, about $\frac{1}{12600}$ as mentioned above. However, this is the probability after a quadrilateral has been found assuming equal likelihoods of all 36-bit codes. Whereas most quadrilateral shapes in the environment will likely not have a full frequency content as the ARTag patterns do (the

Imagery		ARToolkit				ARTag
Seq-uence	Num. Frames	c.f.= 0.50	c.f.= 0.70	c.f.= 0.80	c.f.= 0.90	
A	2723	175	15	9	3	0
B	3514	235	9	3	3	0
C	3318	401	70	0	0	0
D	1450	135	88	70	27	0
E	1318	68	14	3	0	0
F	1318	68	14	3	0	0
G	2893	410	3	0	0	0
H	215625	7917	408	112	0	0

Table 1: Comparison between false positive detection rates between ARToolkit and ARTag in several image sequences. Different cameras were moved around a room devoid of either markers so that all detections would be false positives. A single common set of ARToolkit pattern files were used. Note that no false positive ARTag markers are declared in any of the frames. A-G are tests with various cameras aimed around a lab scene, H is the movie "Bladerunner" (Ridley Scott 1982).

pseudorandom nature of the convolution codes used give a wide use of the spatial frequency domain). In fact, most of the time a quadrilateral shape from the environment is mostly all dark or all white. Therefore the probability of false positive detection is less than %0.0078 explaining why it is not unexpected to see zero false positives even in an experiment as large as conducted in Table 1.

It is useful to compare ARTag to Rekimoto's *Matrix* [14]. There was mention of a checksum, and uses a 5x5 grid, giving 25 data bits of which he claims there was a library size of 2^{16} different markers which would indicate a 7-bit checksum (25 bits minus 2 to account for 4 rotation positions and 16 for the ID equals 7) yielding a false positive rate of $\frac{1}{128}$ = %0.78 which is 100 times more likely than ARTag's false positive rate. Also, Matrix appears to have no error correcting capability and so will have a higher false negative rate (i.e. a lower detection rate). Matrix does not seem to address the concept of inter-marker confusion, without such an analysis it is not unlikely that some markers may be too similar to others in the library. It is perhaps a loss to the AR community that Matrix was not developed further, and used instead of the pattern correlation in ARToolkit. People were not using Matrix for AR projects, and was the existence of the system itself was only discovered after the development of ARTag. Matrix would likely perform better than ARToolkit for the false positive and inter-marker confusion rates.

4.1 Uniqueness of Markers: Reducing Inter-Marker Confusion

As well as a low false detection rate, a good marker system should have a low rate of confusion between markers. The user should have a high confidence that one marker won't be mistaken for another. In the real world one cannot have 100% but with proper marker system design the rate that this occurs can be minimized, ideally to very low levels that don't appear in practice. Owens *et al.* [4] explores the similarity between ARToolkit markers with the *Mean Squared Error* (MSE) approach and proposes a library of markers based upon spatial frequencies in an attempt to address this problem.

ARTag does not use image correlation as does ARToolkit, instead the internal pattern is sampled into a digital code and processed from there as digital symbols. The inter-marker confusion rate can be analyzed by considering the probability of mistaking one code of digital symbols for another. A measure of how easily two binary codes can be confused with each other is to calculate the *Hamming distance*[8], which is simply the sum of the differences between two digital sequences. For example, if the sequences *01001* and *00011* are lined up, one can count two bits which are different thus they are said to have a Hamming distance of 2 from one another. The probability of an inter-marker confusion event can be calculated using knowledge of the Hamming distances within a marker set.

Each ARTag marker pattern can be turned into another if the right '1' and '0' symbols are changed. Ideally the Hamming distance should be as high as possible between all possible markers, taking rotation and optionally mirroring into consideration.

The probability that a digital code can be mistaken for another in a set of codes is given by a summation of the probabilities that the given marker can be falsely recognized as each of the other codes in the set. If there are 36 bits in a code set, such as our system, and the Hamming distance between code A and code B is 10, then the probability of A been seen as B (or vice versa) is the probability of exactly the correct 10 bits been flipped which is $p^{10}(1-p)^{26}$ where p is the probability of a bit been flipped. In our case, each marker contains a sub-ID code encoded in one of four orientations, so we need to add together all four probabilities. If A can be turned into B with a Hamming distance (H.D.) of H.D.=10 with no rotation, of H.D.=12 at a rotation of 90°, H.D.=16 at 180°, and H.D.=24 at 270°, then the total probability is: $p^{10}(1-p)^{26}+p^{12}(1-p)^{24}+p^{16}(1-p)^{20}+p^{24}(1-p)^{12}$. If we also consider mirroring, the case that B is seen in a mirror, then we need to add four more terms.

If we want to find the probability that A can be mistaken for any other code in the set, we add these four (for rotation only) or eight (rotation and mirroring) probabilities for each marker. We can add the probabilities together by grouping together all the cases of equal Hamming distance. Thus we can express the frequency of each Hamming distance by making a function, or histogram, of Hamming distances $HD(n)$ where HD is the number of cases where a Hamming Distance of n occurs. Thus we can use Eqn.2 where the bit error rate p is decoupled from the inter-marker Hamming distances. Since p depends on many factors in the system, the Hamming distance histogram can be calculated for the marker set itself, independent of the system.

To calculate the final probability $P(\neq A)$ that a marker A can be mistaken for another (in the set used in a system), the bit error rate p in Eqn.1. $p(n)$ and $q(n)$ are the probability of n bits being falsely and correctly detected, respectively. If the bit errors are uncorrelated and independent events, then the probability of n bits toggling falsely is $p(n) = p^n$ and not toggling is $q(n) = (1-p)^n$ allowing us to rewrite the probability as Eqn. 2.

The probability of inter-marker confusion is now divided into two parts; the system dependent probabilities (noise, etc), and the component due to the distinctiveness of the markers themselves represented by the Hamming distance histogram. In this way, we can optimize, *i.e.* reduce, the probability of inter-marker confusion for any system by optimizing this histogram $HD(n)$. We seek to reduce the frequency of those of low values of n. The more that the histogram can be pushed out to the right (if plotted as in this paper), and the lower $HD(n)$ is for the first few (low n value) non-zero values of $HD()$, the more immune to inter-marker confusion a marker set will be.

$$P(\neq A) = \sum_{n=1}^{36} HD(n) \cdot p(n) q^{(36-n)} \tag{1}$$

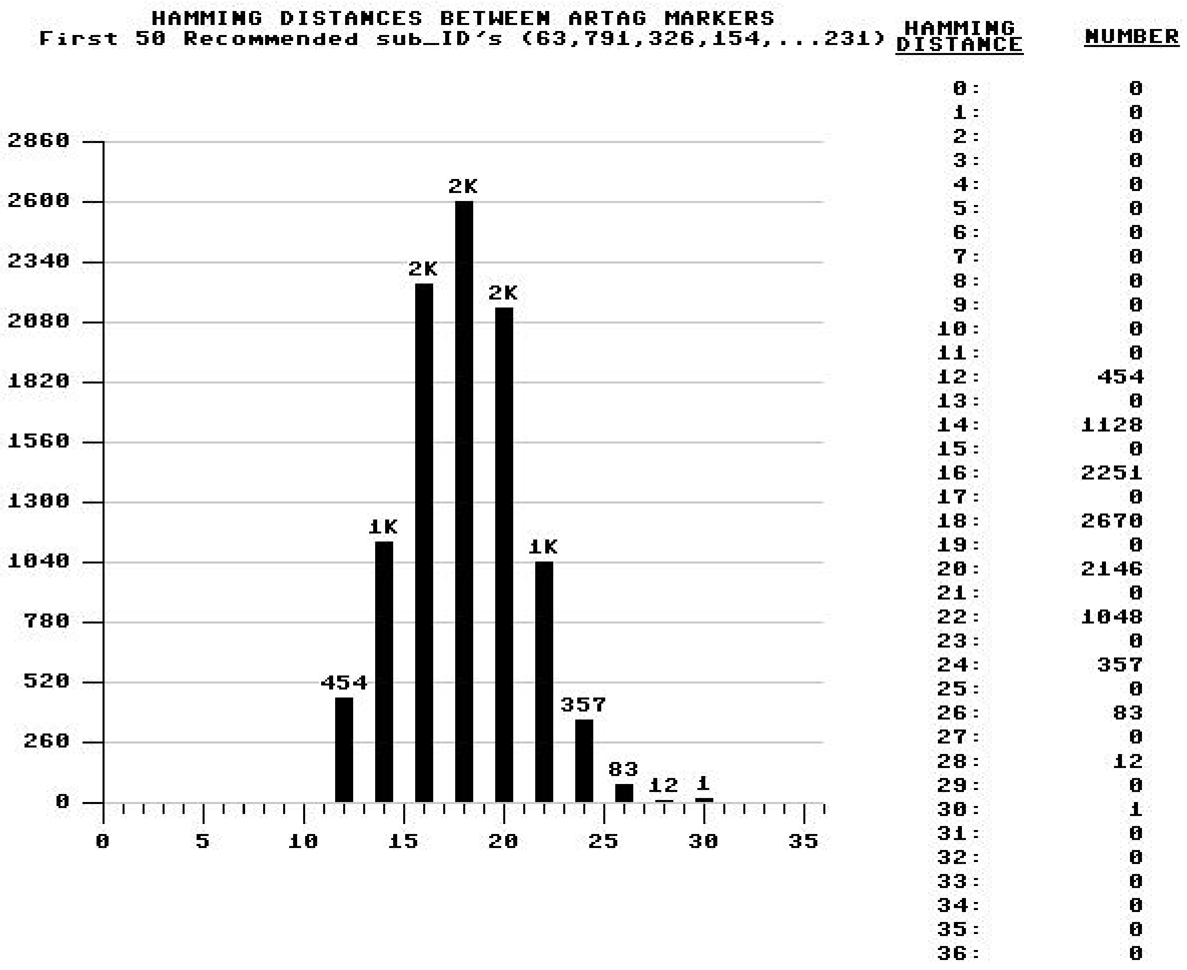
$$P(\neq A) = \sum_{n=1}^{36} HD(n) \cdot p^n (1-p)^{36-n} \tag{2}$$

This Hamming distance histogram will be different for each candidate symbol A. If the histograms are added together for all markers in the set, Eqns.1 or 2 can be used to find the probability that any marker in the set will be mistaken for any other marker in the set, giving a single probability of inter-marker confusion.

When applying Eqn.1, the first few non-zero entries (for low n) will dominate the calculated probability. This defines a measure of the total marker set to optimize: measure the Hamming distance between every possible marker and all others in all four possible rotation positions, aggregate this into a histogram of Hamming distances, and attempt to reduce the minimum non-zero value and area under the first part of the histogram.

This was done considering rotations and reflections between all possible combinations of markers to select the most optimal FEC polynomials and to select a subset of 2002 ID's from the possible range of 2002 [6]. For example sub-ID's 75 and 692 are not recommended due to having a Hamming distance of only 4 when 75 is compared to 692 rotated 90° ccw. This reduces the ARTag library to 1001 sub-ID's, which translates to 2002 possible markers using both polarities of borders.

For most applications, all 2002 markers will not be necessary and hence a priority order list was made. A user takes the top k markers in order from the top of this list to obtain a low inter-marker confusion rate. In this way an application can be made very immune to inter-marker confusion (Fig. 14).



HAMMING DISTANCE	NUMBER
0:	0
1:	0
2:	0
3:	0
4:	0
5:	0
6:	0
7:	0
8:	0
9:	0
10:	0
11:	0
12:	454
13:	0
14:	1128
15:	0
16:	2251
17:	0
18:	2670
19:	0
20:	2146
21:	0
22:	1048
23:	0
24:	357
25:	0
26:	83
27:	0
28:	12
29:	0
30:	1
31:	0
32:	0
33:	0
34:	0
35:	0
36:	0

Figure 13: *Cumulative histogram of Hamming distances between markers considering rotation and reflection. Histogram considering the first 50 from the recommended list.*

4.2 Designing the ARTag Library

When applying Eqn.1, the first few non-zero entries (for low n) will dominate the calculated probability. This defines a measure of the total marker set to optimize: measure the Hamming distance between every possible marker and all others in all four possible rotation positions, aggregate this into a histogram of Hamming distances, and attempt to reduce the minimum non-zero value and area under the first part of the histogram.

This measure was taken into account when designing the system. Using the system shown in Figs. 10,11 we see the parameters are the 10-bit XOR mask, the checksum convolution code and the convolution codes used for the FEC. These could be variables in designing the marker system. The XOR mask does not affect the Hamming distances due to the linearity of the GF-2 operators used (it's effect can be moved to the output as XOR'ing with a fixed 36-bit XOR mask), the CRC-16 convolution polynomial was kept constant due to its successful use in communications and data storage [16]. This leaves the choice of convolution polynomials chosen for use in the forward error correction as variables to change to obtain an optimal histogram of Hamming distances measured for the full marker set. All possible combination were tried and the FEC polynomials that produced the best hamming histogram considering only rotation were chosen. The negative phenomenon of mirroring was considered less likely to occur and so not considered in the Hamming distance calculations at this stage to relax the constraints. Mirroring was addressed (Section 4.3) when non-recommended sub-ID's were chosen (Section4.5).

Since the same internal patterns are used for ARTag ID's $0-1023$ and $1024-1047$, the following histograms in this paper are only calculated within the sub-ID set 0-1023 since the markers border of white/black or black/white separates them into two sets which are very unlikey to be confused with each other.

The best Hamming distance histogram was chosen both for the minimum Hamming distance, and the area of the histogram in the first few entries at and after this minimum distance. This histogram is shown graphically and as a table in Fig. 14. The minimum Hamming distance is 4 symbols, there is six combinations that are that close. The peak of the histogram is at a Hamming distance of 18, due to the total code size being 36 bits and the use of convolution polynomials that have a pseudorandom nature. The histograms produced by several other choices of convolution polynomials are shown in Fig.15.

Looking at Hamming distance histograms of the chosen system in Fig. 14 and the best four second place contender systems in Fig.15, we see they all start with values at $n = 4$, however the chosen system has six elements at this distance as opposed to only one in the other four contenders. This would seem contrary to the stated goal of decreasing the first few histogram bin entries, however, not only the first bin entry was taken into consideration. These four other choices were not chosen due to the higher value at $n = 10$, the first few combinations at $n = 4$ can be removed by removing from a recommended marker list the markers that cause them. The five choices all had about the same value at $n = 6$ and $n = 8$, but the chosen system had the least number of marker pairs at a Hamming distance of $n = 10$. It was felt this would have a greater effect on a probability calculation and overall system robustness than the $n = 4$ markers. The markers who contributed to the $n = 4$ bins could be simply removed from use, since taking a few out from the total library size of over 2000 markers would not affect system usefulness. These markers were not declared illegal, as is sub-ID #682 (markers #682,1706), but rather were put on both the not-recommended list (see Table 3) and moved down on the order of recommendation list (Table 2).

4.3 Adressing Reflections

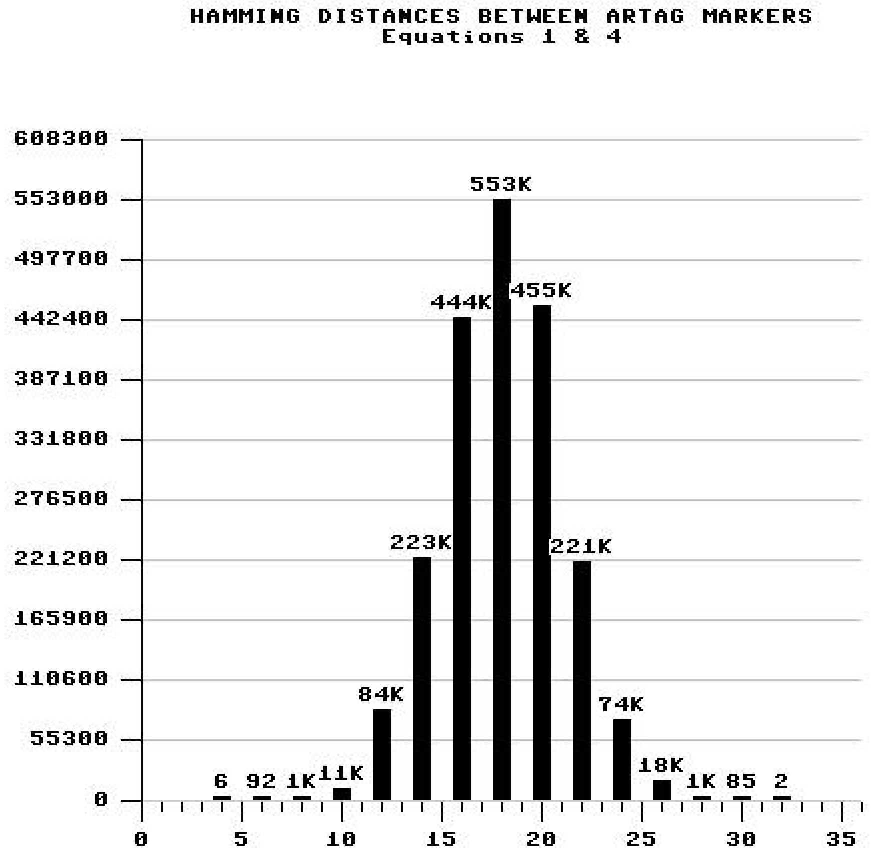
A marker system should not confuse markers if they are seen in reflections, one marker sub-ID should not be mistaken for another if seen in a mirror. Therefore the Hamming distance histogram should be calculated for the case of the 6x6 codes being mirrored. It is assumed that in most cases a system would not want to detect markers unless they are seen un-mirrored, and so the Hamming distance between a marker and its reflection should be high also.

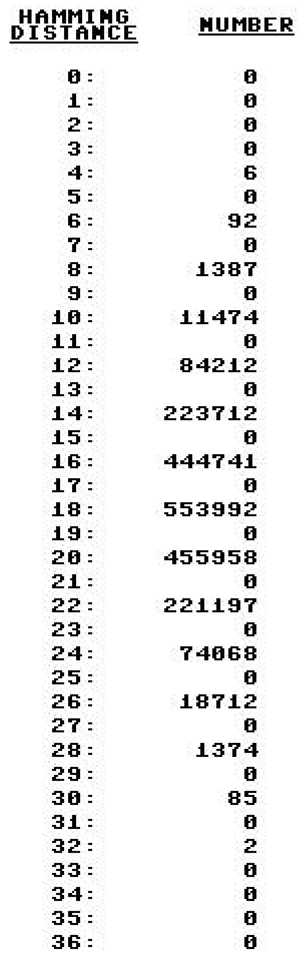
The Hamming distances of mirrored markers were not added to the histograms in Section 4.2 since they are expected to occur less often, and bringing them into the initial design analysis would add a constraint that would result in a system with a reduced inter-marker confusion immunity for the most common event of markers seen directly without a mirroring surface in the marker-camera optical path. However, their effect is analyzed and used in creating some recommendations in choosing markers.

The Hamming distance histogram is shown in Fig.16. The entry at $n = 2$ is for sub-ID #270 which is close to a mirrored image of itself by only two bits different. It thus appears on the not-recommended list (Table 3), and appears last in the recommended order sequence in Table 2. Likewise, 9 other sub-ID's were placed on the not-recommended list and placed low in the recommended order sequence to allow an ARTag user to avoid having these few bad combinations raise the inter-marker confusion probability rate.

4.4 Recommended Subset of the ARTag Library

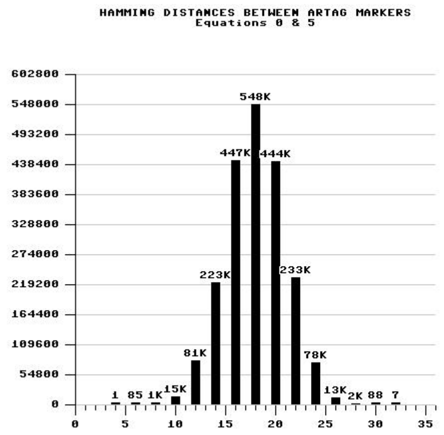
The range of ARTag ID's is 0-2047 with 2 forbidden values; 682 and 1706, leaving 2046 values allowed for use. A further 44 ID's are not recommended to decrease



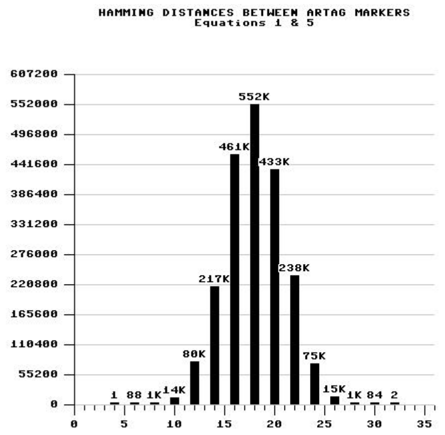


HAMMING DISTANCE	NUMBER
0:	0
1:	0
2:	0
3:	0
4:	6
5:	0
6:	92
7:	0
8:	1387
9:	0
10:	11474
11:	0
12:	84212
13:	0
14:	223712
15:	0
16:	444741
17:	0
18:	553992
19:	0
20:	455958
21:	0
22:	221197
23:	0
24:	74068
25:	0
26:	18712
27:	0
28:	1374
29:	0
30:	85
31:	0
32:	2
33:	0
34:	0
35:	0
36:	0

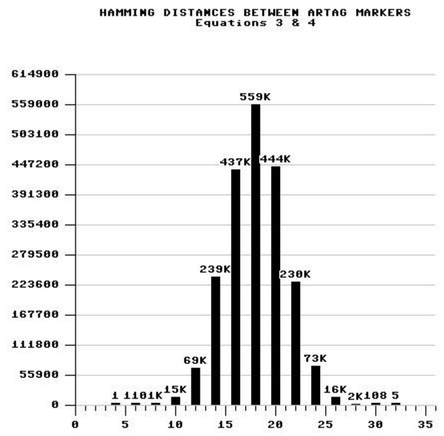
Figure 14: *Hamming distances between ARTag markers. Shown as a histogram with the number of marker pair combinations displayed as a function of the Hamming distance. For example, six combinations of two ARTag markers (out of the set of 1023) can be changed to be identical to the other (at a specific orientation) with 4 bit changes. Likewise there are 1387 pairs of patterns that differ by 10 bit changes.*



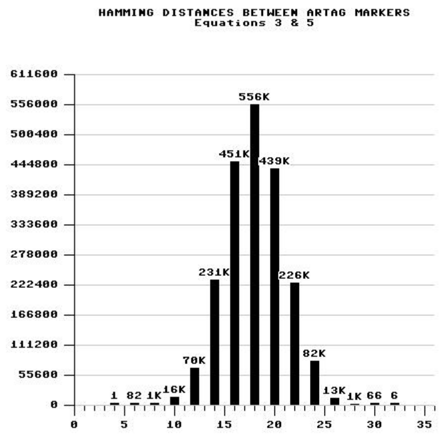
HAMMING DISTANCE	NUMBER
0:	0
1:	0
2:	0
3:	0
4:	1
5:	0
6:	85
7:	0
8:	1463
9:	0
10:	15399
11:	0
12:	81929
13:	0
14:	223301
15:	0
16:	447247
17:	0
18:	548742
19:	0
20:	444808
21:	0
22:	233612
23:	0
24:	78397
25:	0
26:	13573
27:	0
28:	2360
29:	0
30:	88
31:	0
32:	7
33:	0
34:	0
35:	0
36:	0



HAMMING DISTANCE	NUMBER
0:	0
1:	0
2:	0
3:	0
4:	1
5:	0
6:	88
7:	0
8:	1382
9:	0
10:	14127
11:	0
12:	80254
13:	0
14:	217226
15:	0
16:	461446
17:	0
18:	552159
19:	0
20:	433830
21:	0
22:	238296
23:	0
24:	75592
25:	0
26:	15156
27:	0
28:	1369
29:	0
30:	84
31:	0
32:	2
33:	0
34:	0
35:	0
36:	0

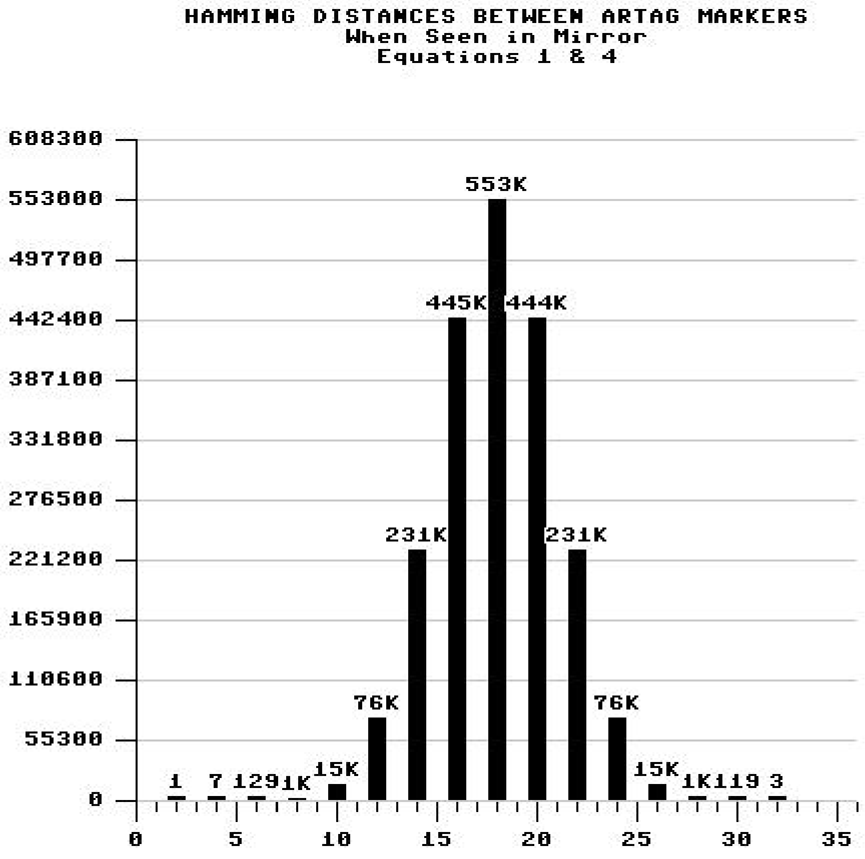


HAMMING DISTANCE	NUMBER
0:	0
1:	0
2:	0
3:	0
4:	1
5:	0
6:	110
7:	0
8:	1440
9:	0
10:	15692
11:	0
12:	69689
13:	0
14:	239980
15:	0
16:	437936
17:	0
18:	559388
19:	0
20:	444518
21:	0
22:	230498
23:	0
24:	73011
25:	0
26:	16240
27:	0
28:	2396
29:	0
30:	108
31:	0
32:	5
33:	0
34:	0
35:	0
36:	0



HAMMING DISTANCE	NUMBER
0:	0
1:	0
2:	0
3:	0
4:	1
5:	0
6:	82
7:	0
8:	1423
9:	0
10:	16589
11:	0
12:	70030
13:	0
14:	231717
15:	0
16:	451251
17:	0
18:	556171
19:	0
20:	439170
21:	0
22:	226727
23:	0
24:	82756
25:	0
26:	13128
27:	0
28:	1895
29:	0
30:	66
31:	0
32:	6
33:	0
34:	0
35:	0
36:	0

Figure 15: *Hamming distances between ARTag markers with different FEC polynomials. The FEC polynomial set chosen in Fig. 14 was selected for ARTag due to the minimum area below a Hamming distance of 11 bits, even though the selected set has 6 entries in the lowest Hamming distance=4 bin. There are six possible polynomials available, giving 15 possible sets for use when correcting up to two bit errors in a marker. These four histograms plus Fig. 14 constitute 5 out of these possible 15 sets.*



HAMMING DISTANCE	NUMBER
0:	0
1:	0
2:	1
3:	0
4:	7
5:	0
6:	129
7:	0
8:	1942
9:	0
10:	15541
11:	0
12:	76422
13:	0
14:	231621
15:	0
16:	445623
17:	0
18:	553635
19:	0
20:	444559
21:	0
22:	231651
23:	0
24:	76528
25:	0
26:	15495
27:	0
28:	1828
29:	0
30:	119
31:	0
32:	3
33:	0
34:	0
35:	0
36:	0

Figure 16: *Hamming distances between ARTag markers in ARTag system when markers are viewed in a mirror. If a reflection of an ARTag marker is seen, the 6x6 binary pattern extracted from the pattern will be flipped. None of the markers will be confused with each other, or with itself (*i.e. *won't be detected in the mirror) with a Hamming distance less than 4. Of the*

the inter-marker confusion rate. 22 sub-ID numbers were identified from the above Hamming distance experiments, which translates into 44 ID's (due to the white/black and black/white border polarity). 12 sub-ID's are not recommended because they map to another with only a Hamming distance of 4. Another 10 are chosen from the mirror image Hamming distance histogram, these resemble too closely themselves or another marker when seen in a mirror. A good marker system should only recognize a marker when seen directly, not in a reflection. For example, sub-ID 270 has a Hammming Distance of only two to a reflection of itself.

The most recommended markers follow in Table 2.

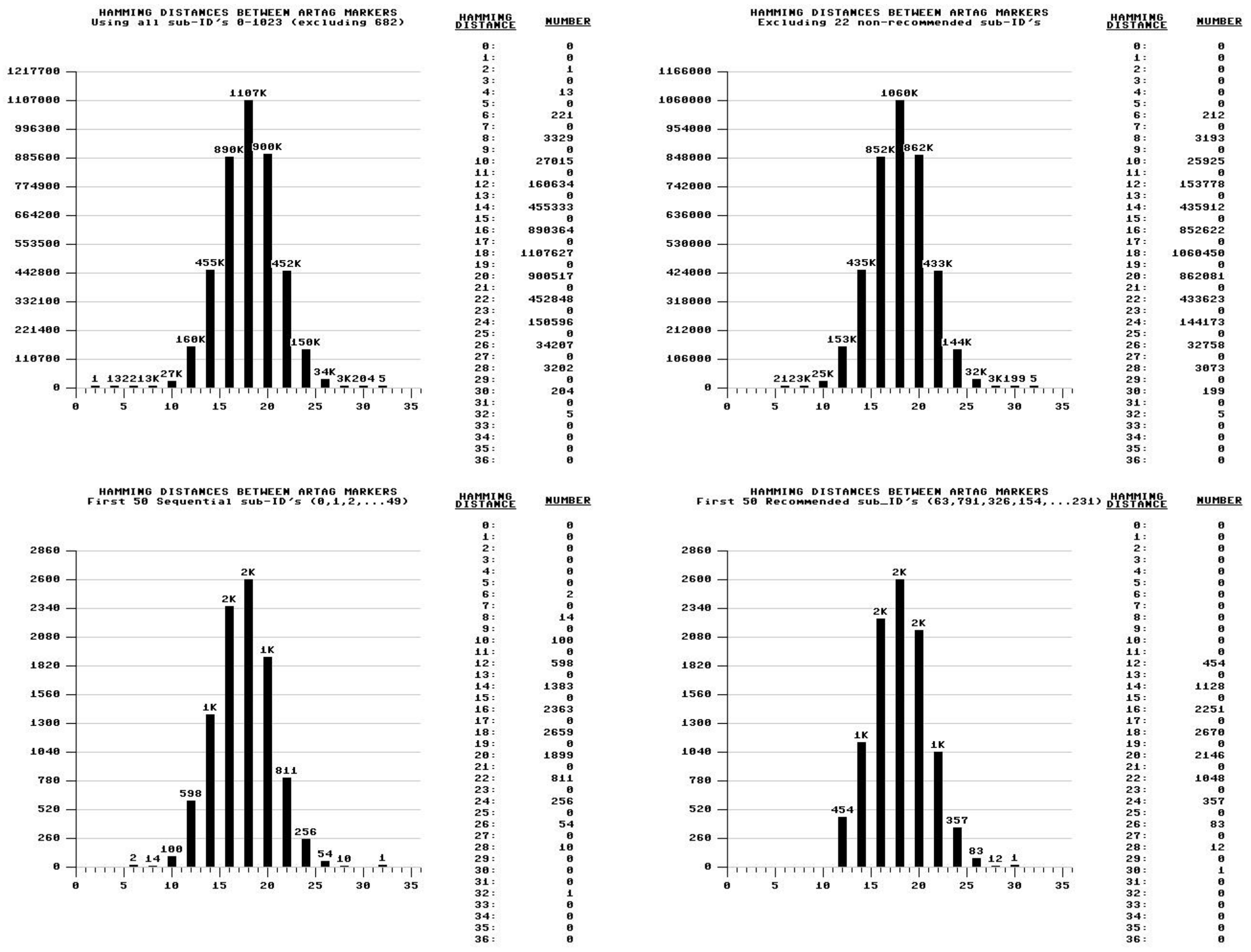
The least recommended codes, chosen for their Hamming distances to themselves or other markers taking rotation and mirroring into account are listed below in Table 3 along with their reasons. If possible, it is advised to not use these.

In summary of the inter-marker confusion rate analysis; it was performed to make design decisions within ARTag, and was used to provide a way to the users to choose a marker set to minimize this condition.

The probability of this undesirable event occurring was shown to be divisible into a system dependent, and code set dependent component. The latter is represented with a distribution of *Hamming distances* $HD(n)$ between marker sub-ID's, which are an aggregate sum of the number of bit changes n it takes to confuse one marker with another. Formulae (Eqns.1,2) for calculating the inter-marker confusion rate using this histogram were presented.

Different histograms were shown, depending on whether they considered the phenomenon of markers been seen in a reflection or not, and were made for different marker sets. All of the histograms contained the inter-marker Hamming distances for all four possible relative rotations between each marker pair considered. The set of all 1023 sub-ID's were considered without mirroring to select which FEC convolution polynomials would be used in ARTag. A histogram involving mirroring was calculated for all 1023 sub-ID's and the most problematic markers were put on a *not-recommended* list (Table 3) in [6]. With ARTag Rev1, these were simply declared as illegal, as is 682 and 1706. Thus the library size is reported herein as 2002 markers as opposed to 2046 as in [6].

For most applications, all 2002 markers will not be necessary and hence a priority order list was made (available in the downloadable ARTag library, the first 72 elements of which are in Table 2). A user takes the top k markers in order from the top of this list to obtain a low inter-marker confusion rate. One way to do this is to call the ARTag library function *artag_get_id()* function sequentially. This can be done by replacing ARToolkit's *init_artoolkit()* pattern loading function if one is modifying an existing ARToolkit application. An example of the benefit of taking the sub-ID's from this list instead of choosing them arbitrarily was shown in Fig.17. Here the Hamming distance histogram was calculated for a marker set containing the first 50



HAMMING DISTANCE	NUMBER	NUMBER	NUMBER	NUMBER
0:	0	0	0	0
1:	0	0	0	0
2:	1	0	0	0
3:	0	0	0	0
4:	13	0	0	0
5:	0	0	0	0
6:	221	212	2	0
7:	0	0	0	0
8:	3329	3193	14	0
9:	0	0	0	0
10:	27015	25925	100	0
11:	0	0	0	0
12:	160634	153778	598	454
13:	0	0	0	0
14:	455333	435912	1383	1128
15:	0	0	0	0
16:	890364	852622	2363	2251
17:	0	0	0	0
18:	1107627	1060450	2659	2670
19:	0	0	0	0
20:	900517	862081	1899	2146
21:	0	0	0	0
22:	452848	433623	811	1048
23:	0	0	0	0
24:	150596	144173	256	357
25:	0	0	0	0
26:	34207	32758	54	83
27:	0	0	0	0
28:	3202	3073	10	12
29:	0	0	0	0
30:	204	199	0	1
31:	0	0	0	0
32:	5	5	1	0
33:	0	0	0	0
34:	0	0	0	0
35:	0	0	0	0
36:	0	0	0	0

Figure 17: *Minimum Hamming distances: A histogram of the* minimum *Hamming distance between all markers in a set, considering rotation and mirroring. (Upper Left) Histogram if all legal 1023 sub-ID's are used. (Upper Right) Histogram if 1002 recommended sub-ID's (all 0-1023 except 682 and the 22 sub-ID's listed in Table 3 (Lower Left) Histogram using set of first sequential 50 sub-ID's. (Lower Right) Histogram using smaller set of first 50 most recommended sub-ID's from Table 2, the benefit of using the recommended list is demonstrated by the absence of histogram entries before $n = 12$.*

sequential sub-ID's (#0-49) and compared to a marker set containing the first 50 sub-ID's from Table 2. In this way an application can be made very immune to inter-marker confusion.

4.5 Minimum Pattern Size and Processing Time

The size of the marker, as seen in the image, along with the image resolution and camera focal length sets the range of minimum and maximum distances that the marker can be seen. Ideally this range should be as large as possible, and thus the marker system should function with as small as possible of a minimum marker size requirement in pixels.

ARTag's pattern is 10x10 units and most of the internal digital bits must be sampled for a pattern to be detected, setting the theoretical lower limit on the ARTag marker size. The first released revision of ARTag (Rev 1) has an internal minimum size of 13x13 pixels for the quad detection for each resolution level. Thus we would expect a minimum width of 13 pixels when using the full resolution mode, and 26 pixels using the half resolution mode, very close to what was found.

ARToolkit's minimum size is less straightforward to calculate, the border needs to be detected and the inside pattern must correlate to a value higher than a user selected *confidence factor* threshold and higher than the correlation with the wrong pattern. ARToolkit samples the interior pattern of quadrilaterals with a default resolution of 16x16 but a library of simple large shapes could be differentiated from one another with smaller than this. Perhaps this explains the recent markers being used by Dr. Billinghurst, a founder of ARToolkit, and colleagues in their recent works where the internal patterns were 3x3 cells of black and white squares [11]. Zhang [17] reports a minimum width of 14 pixels required for detection using ARToolkit, which correlates with our experiments of when the *detection rate* dropped below 75%.

An experiment was performed with ARTag and ARToolkit measuring their *detection rate* as a function of pattern size (in pixels). Detection Rate is the probability of a marker being detected, and is 1 minus the false negative rate.

The minimum size of markers with both systems was determined experimentally with several cameras; principally a greyscale and a colour Dragonfly IEEE-1394 camera from Point Grey Research [3] was used as the high quality example, and the other cameras were lower quality USB web cams and an NTSC camera/frame-grabber combination. The three USB webcams used were an Intel CS120 (320x240), an Intel Pro (640x480) and a Telemax WC-50 (320x240). A Sharp VL-AH150U NTSC camcorder was used in conjunction with an NTSC framegrabber set to 640x480.

The detection rate was measured as a function of marker size by moving the camera close to and away from a panel of 9 markers. Two panels were made up, one

[3] http://www.ptgrey.com/

of ARTag markers and one of ARToolkit and presented separately to each camera. The panels were viewed at different distances with the focus adjusted to get the best image. Ten frames were taken at each distance and the detection rate was determined by dividing by the expected 90 markers expected to be seen.

Figs. 18,19 and show the results with the higher quality Dragonfly cameras. The minimum ARTag marker size is about 13 pixels wide for the greyscale camera when the full resolution mode is on, and 27 or so with the faster half resolution mode. The results are slightly worse (larger) with the colour Dragonfly for the full resolution mode; about 18 pixels wide. The half resolution mode gives approximately the same cut off of 28 pixels which is not unexpected since the blurring function when sampling to get a higher resolution removes a lot of image deterioration due to the colour filter pattern added onto the CCD pixels. The ARTag results in Fig. 18 are quite good both for the reason of agreeing well with the theoretical prediction and that the failure happens abruptly, the region of sporadic detection is over a narrow range of marker widths. In other words, ARTag provides good detection up to its limits, the combination of the variable scale processing, forgiving edge segment heuristics in forming quads, and the error correction allow detection to be 100% (or near to it) with a good focused camera. The ARTag experiments were repeated under lower light conditions but the results were very similar and not included.

For ARToolkit, this experiment was performed using both the pattern files provided in the ARToolkit download package (referred to as un-optimized patterns), and by pattern files made by the *mk_patt* program that uses the markers as seen by that camera (referred to as optimized patterns). The *mk_patt* program is used to capture 12 correlation images for each marker as seen through the camera that will be later used by ARToolkit to correlate with in the detection procedure. Fig. 19 and Fig. 20 show the results for the greyscale and colour models of the Dragonfly camera, with a plot for both the optimized pattern files (sampled using the same camera) and the un-optimized (default pattern files from the ARToolkit download). Three plots are shown for each, for three different c.f. thresholds for ARToolkit. At lower c.f. threshold settings, more markers are detected giving a higher detection rate (lower false negative rate). It should be noted that the detected markers were not checked for false detections and likely false detections and inter-marker confusion events happened in this test.

Comparing these results between ARTag and ARToolkit one can see a more predictable performance with ARTag, full detection occurs until the threshold is reached for the algorithm and there is a sharp drop in detection rate. ARToolkit, on the other hand, has a more chaotic and jittery detection rate which drops off at a faster rate. One can see an improved performance using the pattern files optimized for that camera, since ARToolkit recognizes by correlation, it helps to have views of the pattern as seen through that camera with the same lighting. This reliance on camera and

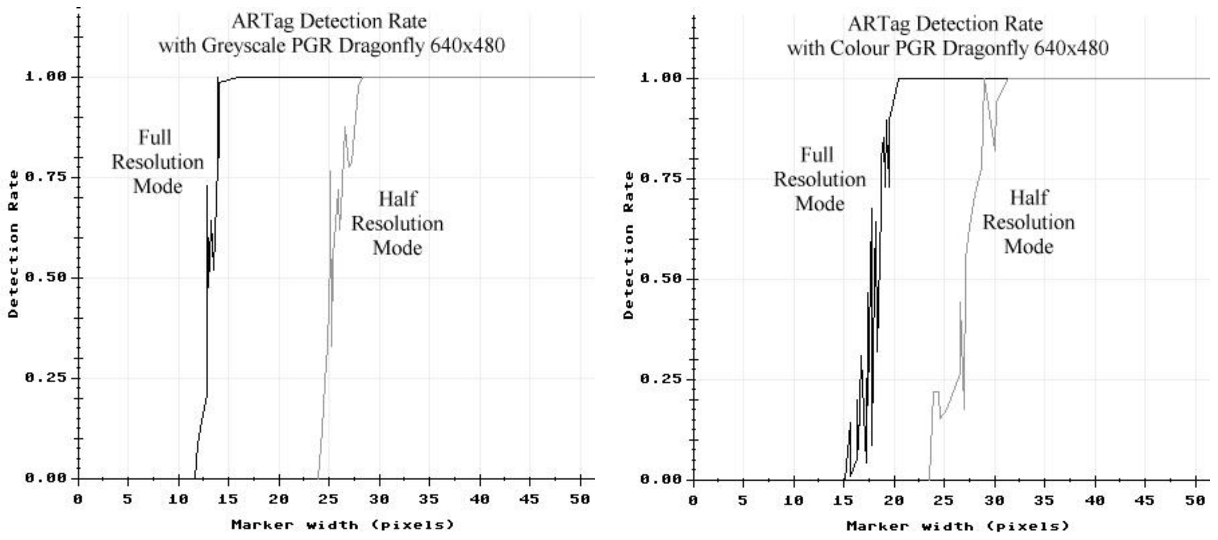


Figure 18: *ARTag detection rate (1-false-negative-rate) with two versions of 640x480 Point Grey Dragonfly digital video cameras. The left shows the results with a greyscale dragonfly (no bayer pattern colour filters) and the right shows the results with the colour version of the same camera. Detection rate is plotted as a function of marker width and is shown for two setting of ARTag; in the slower full resolution detection mode, and in the faster half resolution mode.*

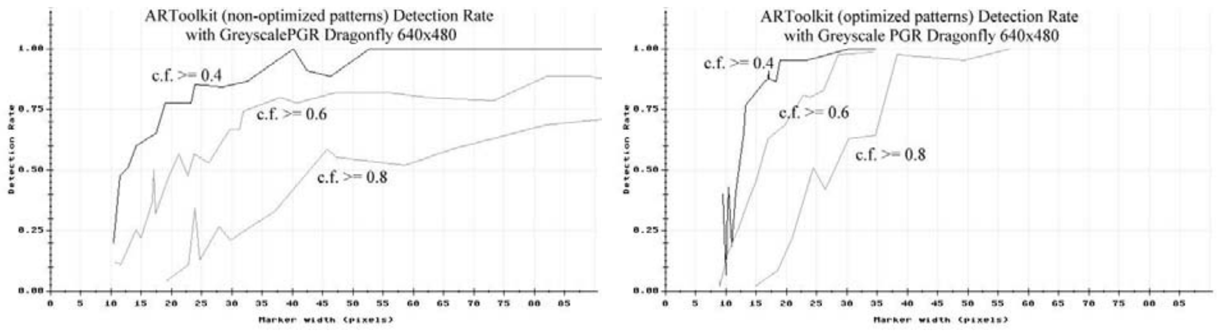
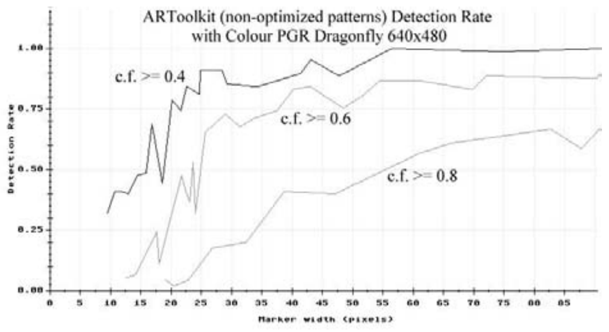


Figure 19: *ARToolkit detection rate (1-false-negative-rate) using the greyscale IEEE-1394 Dragonfly video camera. Using the default (un-optimized) pattern files are shown on the left, and captured with this camera (optimized) pattern files shown on the right. The marker detection rate is shown as a function of the* confidence value *threshold.*



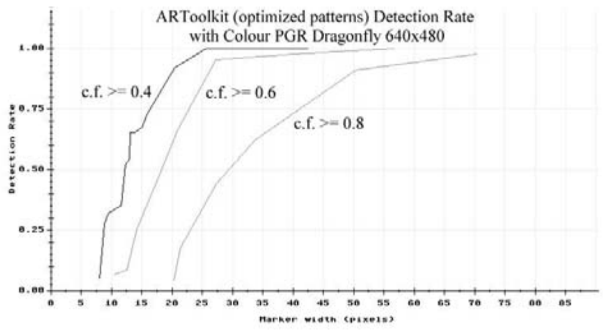


Figure 20: *ARToolkit detection rate (1-false-negative-rate) using the colour IEEE-1394 Dragonfly video camera. Using the default (un-optimized) pattern files are shown on the left, and captured with this camera (optimized) pattern files shown on the right. The marker detection rate is shown as a function of the* confidence value *threshold.*

lighting makes ARToolkit less robust and platform independent than ARTag.

These experiments were repeated with the four other lower quality cameras and their results are shown below.

The full resolution mode detection rate for ARTag fell below 75% at about a width of 17 pixels, or about 28 pixels for the half resolution mode, in the three USB webcams. This is very similar to the Colour Dragonfly, perhaps because of both being colour imagers. Both the Colour Dragonfly and USB webcams use a Bayer (or similar) pattern of filters over the pixels on the imaging device.

The performance with the NTSC camcorder is interesting in that the full resolution mode does not work any better than the half, which could be due to horizontal resampling in the frame grabbing process. However this author believes the greatest cause is is due to the edge enhancement performed in the camera to provide a sharper looking image. This light and dark fringe (Fig. 25) is caused by some sort of differentiation edge enhancement effect not found in any of the other cameras. This causes a type of inter-marker confusion, here markers 13 and 1037 are reported for the same marker (Fig. 25 right), the white-to-bright fringe is solid enough to be detected as a quadrilateral contour just outside the proper contour. Since the sub-ID and interior pattern is the same with 1037 and 13, and 1036 and 12 ($13 + 1024 = 1037$ as does $12 + 1024 = 1036$) and the quad edges are close enough to both correctly decode the interior, two marker ID's get reported for the same marker.

An additional experiment was performed to attempt to address the more unstable results with ARToolkit using the four lower quality cameras; the three USB webcams and the NTSC camera/framegrabber. The confidence factor was measured as a function of marker size for four different ARToolkit patterns, this time using the default pattern files provided with the ARToolkit download. Fig. 26 below shows the results,

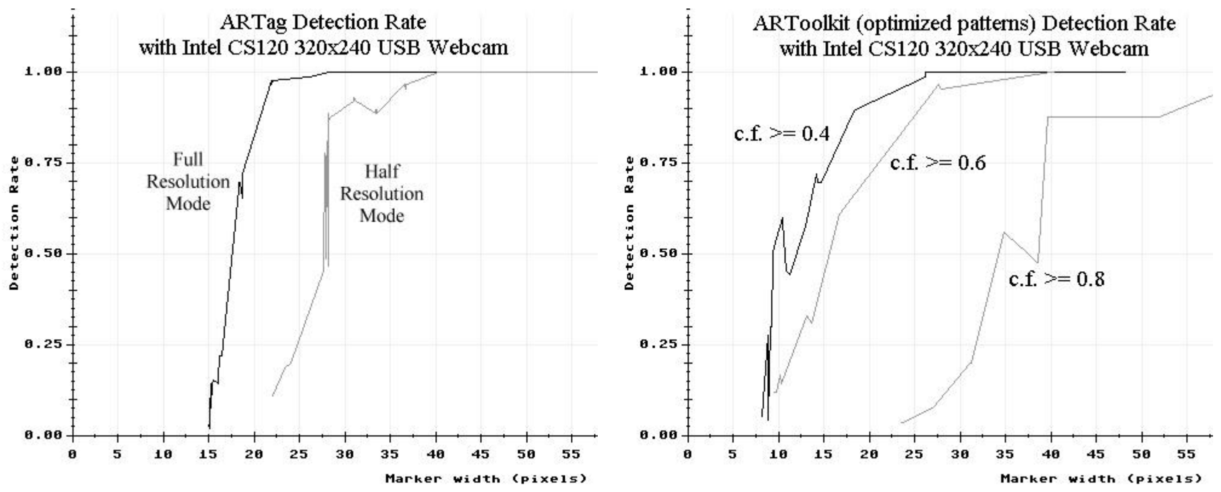


Figure 21: *ARTag and ARToolkit detection rate (1-false-negative-rate) using the Intel CS120 USB Webcam video camera (320x240). Detection rate is plotted as a function of marker width and is shown for two setting of ARTag; in the slower full resolution detection mode, and in the faster half resolution The optimized pattern files were captured for the ARToolkit test, and the marker detection rate is shown as a function of the* confidence value *threshold.*

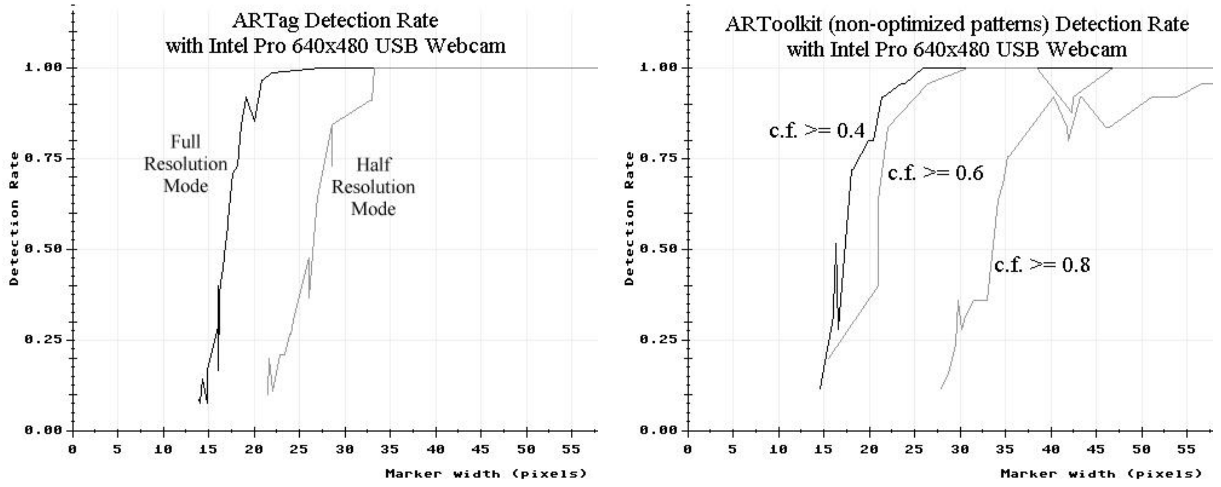


Figure 22: *ARTag and ARToolkit detection rate (1-false-negative-rate) using the Intel Pro USB Webcam video camera (640x480). Detection rate is plotted as a function of marker width and is shown for two setting of ARTag; in the slower full resolution detection mode, and in the faster half resolution The optimized pattern files from the Intel CS120 was used for the ARToolkit test, and the marker detection rate is shown as a function of the* confidence value *threshold.*

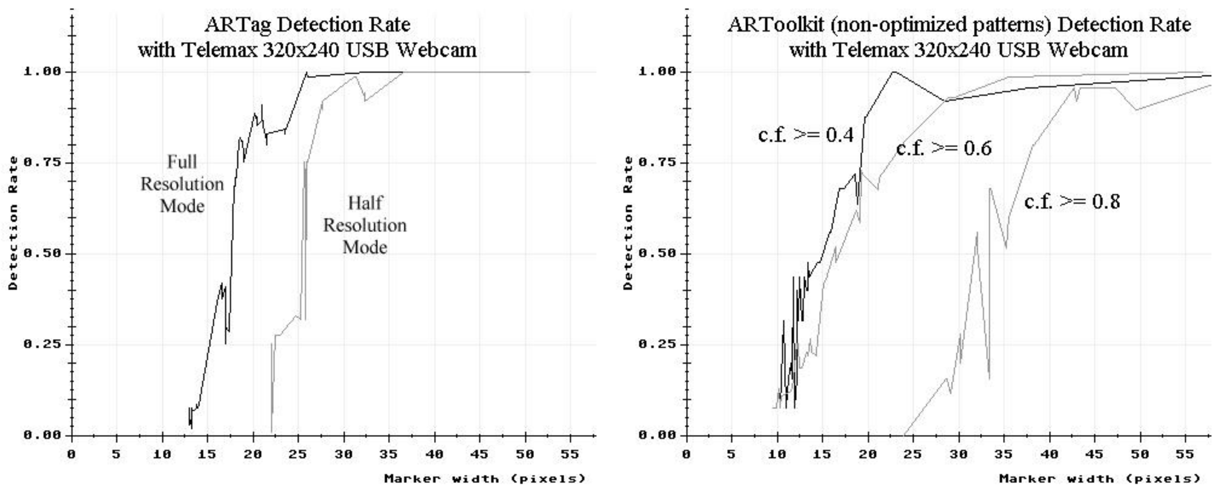


Figure 23: *ARTag and ARToolkit detection rate (1-false-negative-rate) using the Telemax WC-50 USB Webcam video camera (320x240). Detection rate is plotted as a function of marker width and is shown for two setting of ARTag; in the slower full resolution detection mode, and in the faster half resolution The optimized pattern files from the Intel CS120 was used for the ARToolkit test, and the marker detection rate is shown as a function of the* confidence value *threshold.*

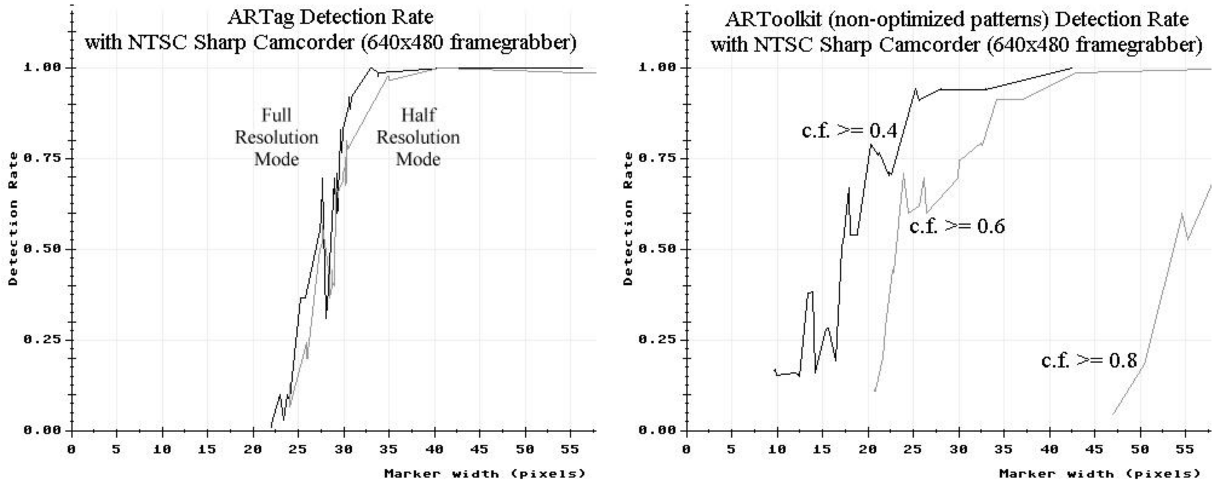


Figure 24: *ARTag and ARToolkit detection rate (1-false-negative-rate) using the Intel CS120 USB Webcam video camera. Detection rate is plotted as a function of marker width and is shown for two setting of ARTag; in the slower full resolution detection mode, and in the faster half resolution The optimized pattern files from the Intel CS120 was used for the ARToolkit test, and the marker detection rate is shown as a function of the* confidence value *threshold.*

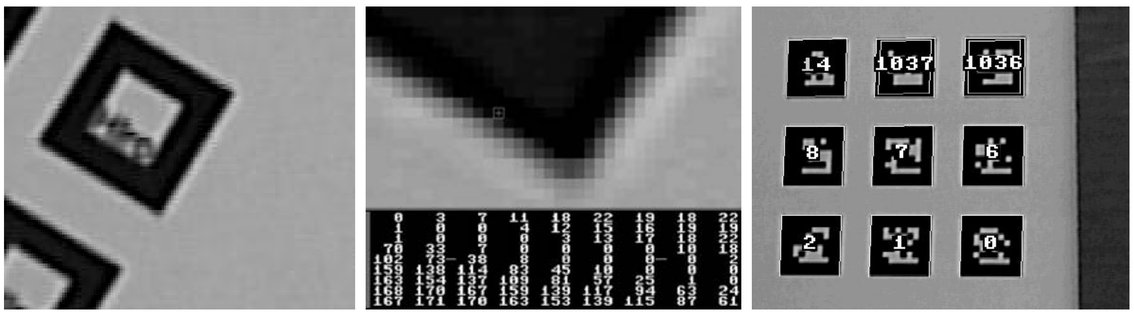


Figure 25: *Edge enhancement artifacts with the NTSC camcorder. Left: an image section of a corner, there is a fringe of brighter pixels on the lighter side of the edge, and a fringe of darker pixels (not visible) on the darker inside edge. Middle: a zoomed in view of the edge with the pixel neighborhood around the cursor, the neighborhood pixel zero or near zero values show the dark fringe. Right: An effect that can occur with this large edge enhancement, both polarities of ARTag markers are found on a single marker. Markers 1037 and 13 are coincident as are 1036 and 12.*

three of the cameras used reached a plateau in c.f. value around a width somewhere after 30-50 pixels. In Zhang's survey [17] experiments with multiple ARToolkit markers were performed with the markers being at least 60 pixels wide. That width is consistent with our findings where the detection rate stayed steady near 100%.

Summarizing the minimum marker size for ARTag and ARToolkit, it was shown that with a good quality camera the theoretical minimum width of 13 pixels can easily be reached if the full resolution mode is set (default), and a width of about 26 pixels is the cut-off for the half resolution mode. With poorer cameras, ARTag patterns can be reliably detected to about a width of 25 pixels in full resolution mode to achieve a near 100% detection rate (0% false negative rate). With ARToolkit the minimum ranges from 30-50 pixels if a confidence factor threshold of 0.80 or more is used. Also it was found that nearly the full range is available with ARTag, as that the detection rate drops abruptly as opposed to the slower decline with ARToolkit.

5 Processing Time

The processing time a fiducial marker system takes is important, ARToolkit's real time performance is one reason for its ubiquity in AR systems. Theoretically ARTag should be faster since it only samples 6x6 points inside each quadrilateral instead of 16x16 or 32x32 as in ARToolkit. Also the first operation performed on the extracted binary data is a deconvolution operation with a small number of memory lookups for the FEC's operation. This is less computation than the 16x16 integer multipli-

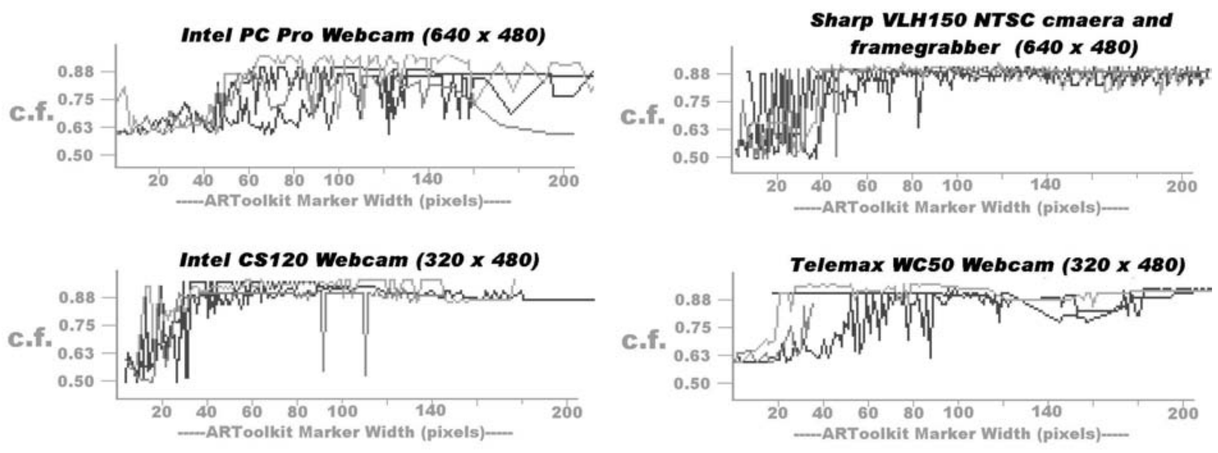
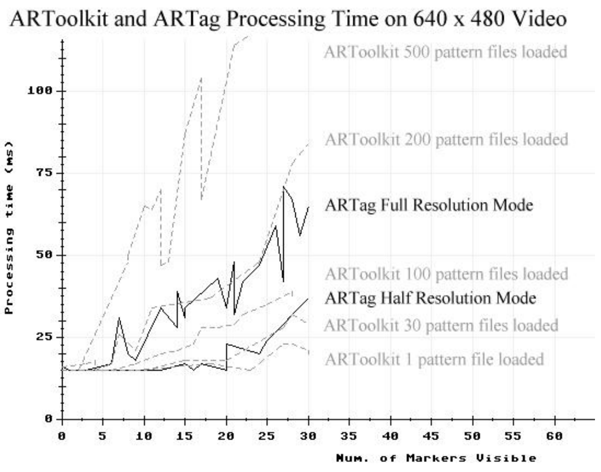


Figure 26: *ARToolkit Confidence value (c.f.) plotted as a function of marker width in pixels for four cameras used in Table 1. The non-optimized pattern files were used (i.e. default patterns from the ARToolkit download) The 'Hiro', 'Kanji', 'Sample1' and 'Robot4' markers from Fig. 3(lower right) were used, and their results overlaid.*

cation with each of the 12 sub-patterns in each loaded pattern file. A 16x16 point (default) implementation of ARToolkit does 3072 multiplications for correlating and another 256 multiplications for normalizing the sampled image patch. If 10 patterns were loaded, then ARToolkit must perform 33,280 integer multiplications for every quadrilateral object found in the image. ARTag only has to perform a 36-bit divide/deconvolution operation and perform two searches among lookup tables of 12 entries long, all together this only has to be done once to evaluate the first step. If this first step of FEC filtering is passed, three more deconvolution and an XOR stage are performed. The processing time per quadrilateral in the image does not rise with the number of markers in the library with ARTag as it does with ARToolkit. ARTag can recognize from the library of 2002 markers as quickly as if only 10 markers are used.

A large part of the processing time is identifying the quadrilaterals, this first half of the marker system is performed differently for ARToolkit and ARTag. ARToolkit's quad detection involves visiting each pixel in the image once for thresholding and then performing connectivity on the binary image, whereas ARTag's quad detection strategy is to find and link edge segments and requires at minimum that two edge masks must be passed over the image. ARToolkit only needs to visit every pixel once in the first pass whereas ARTag needs to visit each pixel several times for the edge operators.

Therefore, we expect a processing time that rises with linearly with both the number of visible quadrilaterals and loaded patterns with ARToolkit, and linearly with the number of edges with ARTag. We also expect ARToolkit to be faster when a small number of patterns are loaded when a small number of markers are viewed.



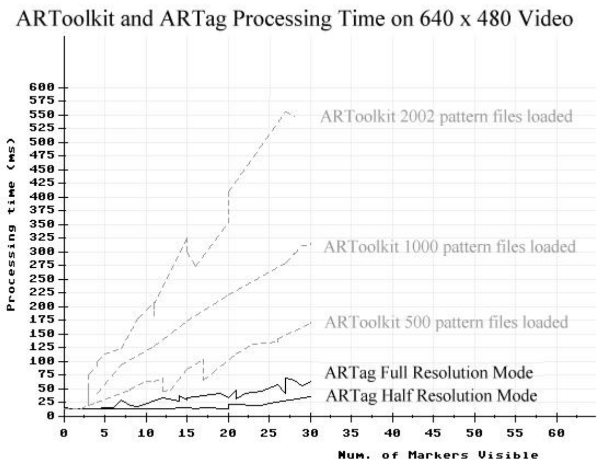


Figure 27: *ARTag and ARToolkit processing time. Measured with a 640x480 image as a function of number of visible markers. ARTag's processing time is shown for both the full and half resolution settings. ARToolkit processing time depends on the number of patterns loaded into memory, the left plot shows when 1, 30, 100, 200 or 500 patterns are loaded, the right plot shows when 500, 1000, or 2002 patterns are loaded.*

The time in milliseconds was measured before and after the marker detection function was called, the results are shown in Fig. 27 for both ARToolkit and ARTag with different settings for each. With ARTag the processing time differs depending on if it is placed in the full (default) or half resolution processing mode. With ARTag the processing time rises with the number of markers visible. For ARToolkit the processing time depends both on the number of markers visible and the number of patterns loaded, as that it has to do correlation operations with each of them. These measurements were taken on a PC running Windows 2000 with a Pentium 4 processor running at 3.0 GHz.

ARToolkit is faster with a small number of markers loaded, it is about 3 times faster than the full resolution mode of ARTag and with 30 markers loaded it is about the same speed as the half resolution mode of ARTag. ARToolkit was only run in its half resolution mode in these experiments, it is the default setting with in the ARToolkit releases and is the mode most likely used in applications. The first few levels of processing in ARToolkit or ARTag only need to access 1/4 the number of pixels if run in a half resolution mode, and so a fair comparison of time would be with the half resolution mode of ARTag.

ARToolkit is faster than the current (Rev 1) release of ARTag if small numbers of markers are loaded. ARTag is faster when the number of patterns loaded is large. In the case where 2002 markers are loaded into ARToolkit, it would take over 2000 ms to process a frame with 100 quadrilateral shapes in it. ARToolkit is not practical to use with large library sizes and so this slow down phenomenon may not be an issue.

6 Jitter

An important issue with augmented reality is the amount that the virtual object shakes with respect to the real world objects. This especially reduces the quality of the user's experience if the camera-pattern pose is fixed and the virtual object still appears to vibrate. The amount by which the vertices reported marker fiducial patterns move due to noise affects this, even a sub-pixel movement on one of the corners is enough to noticeably shift the 3D virtual object. This erroneous motion is worse for coordinates further away from plane of the pattern.

The jitter is a function of the first part of both marker systems, locating quadrilaterals. Both ARTag and ARToolkit find line equations for the four sides and intersect these to find vertices, a process resulting in sub-pixel resolution corners.

ARToolkit employs *hysteresis* to stabilize the 3D object. The ARToolkit marker detection *ARMarkerDetect()* function compares the coordinates of a located marker to the past history and keeps the old vertex values if the marker does not move more than a threshold amount. This helps to hide the jitter for stationary objects, but creates a larger jumping like motion for slow motion where the vertices move less than this threshold distance between frames.

ARToolkit has a second marker detection function that does not use hysteresis, called *ARMarkerDetectLite()* which provides access to the jitter performance of the raw marker detection. The Rev1 release of ARTag does not perform any hysteresis on located marker vertices, preferring to leave strategies to disguise marker jitter to the discretion of the system builder applying ARTag. [4]

Vertex jitter was measured with both ARTookit's *ARMarkerDetectLite()* function and ARTag. Table 4 shows some jitter measurements with 6 different cameras; the greyscale and colour IEEE-1394 Dragonfly cameras, the three USB cameras used in Section 4.5, and the Sharp NTSC camera. The lighting was normal, bright indoor lighting and the focus was adjusted to the sharpest position. The jitter is slightly worse in the half resolution mode of ARTag as compared the full resolution mode, and the full resolution mode jitter is about the same as ARToolkit for the 6 cameras.

The main information to be gleaned from Table 4 is that with medium sized

[4]The Rev2 release will, however, have hysteresis in the "drop-in" function for those converting ARToolkit programs or keeping compatibility between the two systems to make it more compatible with what people expect from ARToolkit. The raw marker vertex data will still be available with another function to keep it useful for general purpose applications.

markers under good illumination and focus, both ARTag and ARToolkit have with most cameras a vertex jitter with a standard deviation of 0.05 pixels or less.

The above table gives the average jitter performance, further experiments were performed with the best quality camera, the greyscale IEEE-1394 Dragonfly camera as to the effect of low light and bad focusing on the jitter performance. Fig. 28 shows some jitter results as a function of marker size in pixels, full or half resolution modes, and the focusing and illumination conditions for both marker systems.

The jitter under low light was about twice what it was under good illumination as would be expected due to the increased effect of noise. The Dragonfly uses a CCD imager and so suffers from increased "dark noise" when the gain is high. The electrical noise is roughly constant and when the light level is low, this electrical noise is amplified. The increased noise is clearly visible looking at the video output. With ARTag, the jitter rises with the decreasing marker width more than ARToolkit probably due to ARTag's use of edge detectors, differentiation magnifies random noise.

The most obvious difference between the three rows of Fig. 28 is with the de-focused case, ARToolkit is clearly less sensitive to having the camera out of focus. In these tests the camera was de-focused until just before the detection rate dropped below 100%. Since there is no quantitative measure on the focus, the comparison is perhaps not very informative, since a more robust marker detection system would detect the markers under worse focusing conditions. Regardless, ARToolkit does not suffer as badly from jitter as does ARTag. The reason for this is not clear, with a blurry image the boundary edge is spread into a wider region giving a wider variation in where the edge line is found, this would affect both systems. Small image noise has a larger effect on edge position when the edge region is wide with a shallow intensity gradient, this possibly explains why the half resolution mode of ARTag has lower jitter, noise would be lower in the half scale image after the averaging operation. When a quad is found on several scales in the same location, the quad definition from the higher resolution scale is chosen, potentially causing the full resolution noisier image to be used to define the quad.

The practical outcome of these jitter measurements is a threshold value for motion to be used in with hysteresis for augmented reality to decrease the visible virtual object shaking. This threshold could be set to 0.1 pixels for most cases, or 0.5 pixels to cause the object not to move with a noisy but reasonably focused camera. Hysteresis was not added to ARTag_rev1 because it was thought best to leave it to the specific application. ARTag_rev2 will have it in the *artagDetectMarker()* function to make it more "plug and play" for people used to ARToolkit's hysteresis. In ARTag_rev2, the *artagDetectMarkerLite()* function will be available as the non-hysteresis version. Currently in ARTag_rev1, the *artagDetectMarker()* does not have hysteresis implemented.

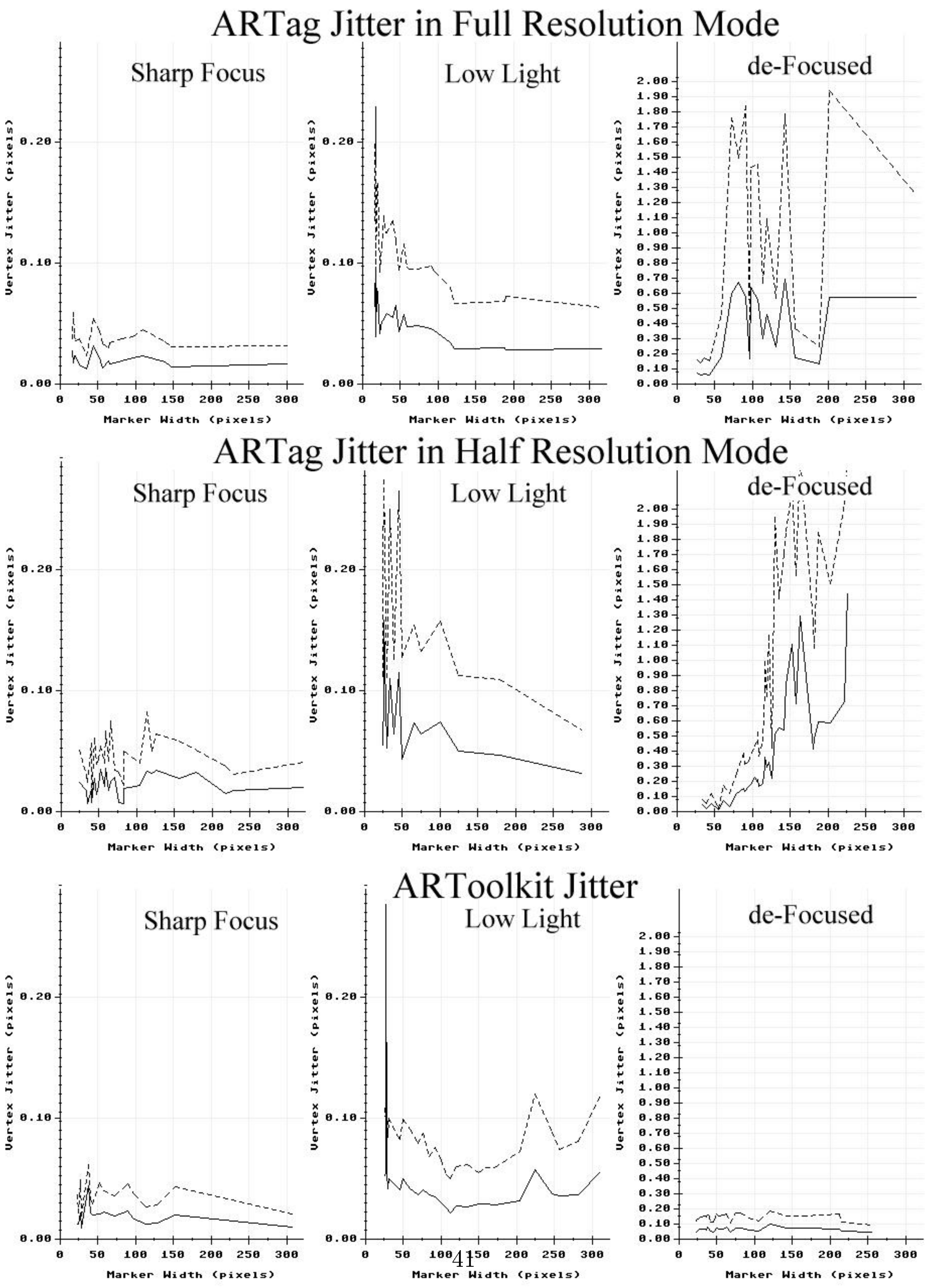


Figure 28: *ARTag jitter as a function of marker size in image. Black lines represent standard deviation of vertex coordinates, dotted lines represent maximal deviations from the mean. The measurements are taken in both the full (top row) and half (bottom row) resolution modes. The measurements were taken with the markers in focus (left) and good light, in focus with low light (middle), and with the markers de-focused (camera focus blurry) (right).*

7 Conclusions

A new marker system, called *ARTag* was created to improve upon the successful *ARToolkit* marker system based on passive vision of 2D planar markers. ARTag uses an edge linking method to detect the quadrilateral borders instead of the contour of thresholded objects in ARToolkit. As a result of the edge based quadrilateral detection, ARTag markers can still be detected in the presence of occlusion and uncontrolled lighting, two shortcomings of ARToolkit. The interior pattern in ARTag is a digitally encoded, error corrected ID code to replace ARToolkit's correlation based pattern recognition and identification step. ARTag has a library of 2002 unique ID markers without the need to load any pattern files. ARTag manages to achieve a lower false negative rate (higher detection rate) than ARToolkit, but has a vastly lower false positive error rate (%0.0078) and very low inter-marker confusion rate.

With experiments using 6 different cameras it was found that at good focus settings, ARTag can detect markers down to about 13 pixels wide in the full resolution mode and 26 pixels wide in the half resolution mode. A more conservative width specification for poor conditions and cameras is 20 and 35 pixels respectively. ARToolkit's detection rate fails more gradually with decreasing marker size and is more difficult to characterize.

ARToolkit has benefits over ARTag in two aspects; reduced vertex jitter under de-focusing and reduced processing time when a small number of patterns are loaded into ARToolkit. Under normal operating conditions, the jitter of vertex position with both ARTag and ARToolkit have a similar standard deviation of about 0.05 pixels or less.

ARTag is available for download for evaluation and free usage in non-commercial systems.

References

[1] http://www.eia.org/ (electronics industry association website).

[2] http://www.photomodeler.com.

[3] *2004 IEEE / ACM International Symposium on Mixed and Augmented Reality (ISMAR 2004), 2-5 November 2004, Arlington, VA, USA*. IEEE Computer Society, 2004.

[4] F. Xiao C. Owen and P. Middlin. What is the best fiducial? In *First IEEE International Augmented Reality Toolkit Workshop (at ISMAR)*, Sep 2002.

[5] D. Claus and Fitzgibbon. Reliable fiducial detection in natural scenes. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*, May 2004.

[6] M. Fiala. Artag, an improved marker system based on artoolkit. In *National Research Council Publication NRC 47166/ERB-1111*, 2004.

[7] I. Poupyrev H. Kato, M. Billinghurst. *ARToolkit User Manual, Version 2.33*. Human Interface Technology Lab, University of Washington, 2000.

[8] R. Hamming. Error detecting and error correcting codes. In *Bell Systems Technology Journal*, volume 29, pages 147–160, Apr 1950.

[9] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proc. the 2nd IEEE and ACM International Workshop on Augmented Reality*, pages 85–94, San Francisco, CA, USA, Oct 1999.

[10] V. Knyaz and A. Sibiryakov. The development of new coded targets for automated point identification and non-contact 3d surface measurements. In *Graphicon*, Moscow, Russia, 1998.

[11] G. Lee, C. Nelles, M. Billinghurst, and G. Kim. Immersive authoring of tangible augmented reality applications. In *ISMAR 2004: IEEE / ACM International Symposium on Mixed and Augmented Reality, Arlington, VA, U.S.A.*, Nov. 2004.

[12] S. Malik, G. Roth, and C. McDonald. Robust 2d tracking for real-time augmented reality. In *Proc. of Vision Interface*, pages 399–406, 2002.

[13] L. Naimark and E. Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *ISMAR 2002: IEEE / ACM International Symposium on Mixed and Augmented Reality, Darmstadt,Germany*, Sept. 2002.

[14] J. Rekimoto. Matrix: A realtime object identification and registration method for augmented reality. In *Proceedings of 3rd Asia Pacific Computer Human Interaction. (APCHI '98)*, pages 63–68, 1998.

[15] 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, Elsinore, Denmark, Oct 2000.

[16] A. S. Tanenbaum. *Computer networks.* Prentice-Hall, Inc, Upper Saddle River, NJ, 1988.

[17] X. Zhang, S. Fronz, and N. Navab. Visual marker detection and decoding in ar sytems: A comparative study. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 97–106, Sep 2002.

#	ARTag sub-ID	H.D.	#	ARTag sub-ID	H.D.	#	ARTag sub-ID	H.D.	#	ARTag sub-ID	H.D.
0	4	-	18	24	12	36	196	12	54	740	12
1	57	16	19	25	12	37	202	12	55	838	12
2	260	16	20	26	12	38	209	12	56	954	12
3	5	14	21	30	12	39	227	12	57	0	10
4	52	14	22	38	12	40	237	12	58	1	10
5	59	14	23	60	12	41	245	12	59	12	10
6	65	14	24	64	12	42	252	12	60	13	10
7	244	14	25	72	12	43	268	12	61	17	10
8	397	14	26	73	12	44	278	12	62	20	10
9	540	14	27	74	12	45	308	12	63	23	10
10	929	14	28	76	12	46	311	12	64	27	10
11	2	12	29	99	12	47	329	12	65	28	10
12	6	12	30	102	12	48	518	12	66	29	10
13	7	12	31	148	12	49	534	12	67	32	10
14	9	12	32	158	12	50	604	12	68	35	10
15	14	12	33	161	12	51	620	12	69	40	10
16	15	12	34	191	12	52	646	12	70	42	10
17	19	12	35	192	12	53	673	12	71	43	10

Table 2: Recommended order of creating a set of ARTag markers for an application. Select the marker sub-ID's in ascending from *Order*=0, creating a marker set in this way will maximize the Hamming distance between any of the markers which minimizes the chance of inter-marker confusion. The third column for each entry lists the Hamming distance (H.D.) for the set of markers from *Order* = 1 up until that entry. For example, using the first 50 markers in this table (*Order* = 0 - 49) will result in a minimum Hamming distance between any two markers in the library of 12. Likewise, taking the first 68 markers (*Order* = 0 - 67) will result in a minimum H.D. of 10. This information is available in the *artag_recommended*[*order*] and *artag_recommended_hd*[*order*] statically defined arrays in the ARTag library.

ARTag Marker Not Recommended		Reason	ARTag Marker Not Recommended		Reason
ARTAG ID	sub-ID		ARTAG ID	sub-ID	
75,1099	75	HD=4 to sub-ID 692 at 90° ccw	655,1679	655	HD=4 to sub-ID 898 at 180°
192,1216	192	HD=4 to sub-ID 686 at 90° ccw	686,1710	686	HD=4 to sub-ID 192 at 90° cw
182,1206	182	HD=4 to itself mirrored at 90° cw	692,1716	692	HD=4 to sub-ID 75 at 90° cw
270,1294	270	HD=2 to itself mirrored at 90° cw	736,1760	736	HD=4 to sub-ID 574 at 180°
377,1401	377	HD=4 to sub-ID 933 at 90° ccw	791,1815	791	HD=4 to itself mirrored at 90° ccw
384,1408	384	HD=4 to itself mirrored at 90° ccw	828,1852	828	HD=4 to sub-ID 609 mirrored at 0°
507,1531	507	HD=4 to sub-ID 966 mirrored at 0°	898,1922	898	HD=4 to sub-ID 655 at 180°
574,1598	574	HD=4 to sub-ID 736 at 180°	927,1951	927	HD=4 to sub-ID 938 at 90° ccw
609,1633	609	HD=4 to sub-ID 828 mirrored at 0°	933,1957	933	HD=4 to sub-ID 377 at 90° cw
643,1667	643	HD=4 to itself mirrored at 180°	938,1962	938	HD=4 to sub-ID 927 at 90° cw
648,1672	648	HD=4 to itself mirrored at 90° ccw	966,1990	966	HD=4 to sub-ID 507 mirrored at 0°

Table 3: ARTag markers not recommended for use due to small Hamming distances (HD); either with the normal (non-mirrored) case, or in the special case of a marker been seen in a mirror. Six of the markers (sub-ID's 182, 270, 384, 643, 648, 791) are close to their own reflections.

Camera	Marker	ARTag		ARToolkit
	Width (pixels)	Half Res. Mode	Full Res. Mode	
Greyscale	55	0.04/0.09	0.03/0.07	0.02/0.04
Dragonfly	90	0.01/0.03	0.02/0.04	0.01/0.02
Colour	55	0.04/0.06	0.02/0.03	0.02/0.04
Dragonfly	90	0.02/0.04	0.02/0.04	0.02/0.03
USB Intel	55	0.05/0.10	0.05/0.10	0.05/0.09
CS120	90	0.06/0.13	0.03/0.07	0.04/0.09
USB Intel	55	0.05/0.09	0.05/0.10	0.05/0.10
Ipro	90	0.05/0.10	0.04/0.08	0.04/0.09
USB	55	0.05/0.08	0.03/0.07	0.05/0.11
Telemax	90	0.05/0.09	0.03/0.06	0.03/0.07
NTSC	55	0.02/0.03	0.08/0.17	0.06/0.12
Camcorder	90	0.05/0.11	0.05/0.10	0.02/0.04

Table 4: Vertex jitter measurements for ARTag and ARToolkit for 6 cameras. Format is std.dev./max, for example with the CS120 at a marker width of 55 pixels, the corners found by ARTag in half resolution mode had a standard deviation of 0.04 pixels and a maximal distance from the mean of 0.09 pixels.