NRC·CNRC

NRC Publications Archive
Archives des publications du CNRC

ARTag, an improved marker system based on artoolkit

Fiala, Mark

For the publisher's version, please access the DOI link below./ Pour consulter la version de l'éditeur, utilisez le lien DOI ci-dessous.

Publisher's version / Version de l'éditeur:
https://doi.org/10.4224/5763247

Report (National Research Council Canada. Radio and Electrical Engineering Division : ERB); no. ERB-1111, 2004

NRC Publications Archive Record / Notice des Archives des publications du CNRC :
https://nrc-publications.canada.ca/eng/view/object/?id=b080c6d3-c00c-4527-863b-ee575af5045a
https://publications-cnrc.canada.ca/fra/voir/objet/?id=b080c6d3-c00c-4527-863b-ee575af5045a

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at
https://nrc-publications.canada.ca/eng/copyright
READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site
https://publications-cnrc.canada.ca/fra/droits
LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

Questions? Contact the NRC Publications Archive team at PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca. If you wish to email the authors directly, please see the first page of the publication for their contact information.

Vous avez des questions? Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez pas à les repérer, communiquez avec nous à PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca.

National Research Council Canada | Conseil national de recherches Canada

Canada

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

*ARTag, An Improved Marker System Based on ARToolkit **

Fiala, M.
July 2004

* published as NRC/ERB-111. July 28, 2004. 36 Pages. NRC 47166.

Copyright 2004 by
National Research Council of Canada

Permission is granted to quote short excerpts and to reproduce figures and tables from this report, provided that the source of such material is fully acknowledged.

Canada

National Research Council Canada
Conseil national de recherches Canada

ERB-1111

Institute for Information Technology
Institut de technologie de l'information

NRC-CNRC

ARTag, An Improved Marker System Based on ARToolkit

Fiala, M.
July 2004

Copyright 2004 by
National Research Council of Canada

Permission is granted to quote short excerpts and to reproduce figures and tables from this report, provided that the source of such material is fully acknowledged.

Canada

NRC-47166

NRC 47166/ERB-1111
Printed July 2004

ARTag, An Improved Marker System Based on ARToolkit

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

Abstract

ARToolkit is a very successful and robust marker system used for Augmented Reality (AR), its robust performance has spawned many applications in AR and computer vision. ARToolkit consists of several 2D planar fiducial marker patterns and software that recognizes and identifies these markers in images. It functions well in finding markers, however its performance with respect to false positive detections and inter-marker confusion could use improvement. Quite often markers are confused for one another or falsely detected in the background. ARToolkit markers consist of a square black border enclosing a pattern that is compared to several stored patterns by correlation. This paper proposes a method to robustify this marker system by replacing this correlation step with a digital symbol method. The interior greyscale pattern is replaced by a digital pattern of 36 bits which contains a unique ID number protected from false detection with the digital code techniques of checksums and forward error correction (FEC). This proposed new system, *ARTag* has a very low and numerically quantifiable error rate, has a reduced processing time, and can encode up to 2046 different unique ID's with no need to store patterns. Experimental results are shown validating this method.

Contents

1 Introduction **4**

2 Planar Marker Systems **5**

3 ARToolkit **8**

4 ARTag **11**

5 False Positive Marker Detection **13**

6 Uniqueness of Markers: Reducing Inter-Marker Confusion **18**
6.1 Designing the ARTag Libary 21
6.2 Adressing Reflections 22
6.3 Recommended Subset of the ARTag Libary 26

7 ARTag and ARToolkit Minimum Pattern Size and the False Negative Detection Rate **28**

8 Processing Time **29**

9 ARTag Usage **30**

10 Conclusions **30**

1 Introduction

Designing markers to add to the environment for robust detection in camera and video imagery is a computer vision application useful to *Augmented Reality*, position tracking, photo-modeling and robot navigation. *Augmented Reality* (AR) or *Mixed Reality* consists of rendering virtual objects in real imagery, and is typically intended for applications using a head mounted display (HMD). The camera image for a monocular system, or the images from two cameras in a binocular system, are overlaid with rendered virtual objects and then presented to the eyepiece displays inside the HMD. If the pose of a camera on an HMD is correctly determined, the virtual objects can be rendered onto the camera image(s) with correct location and perspective to provide the illusion of being present.

The HMD or camera's pose can be determined with magnetic field, radio, active LED or laser beacons; or most simply and inexpensively, with passive marker patterns and computer vision. The future promises passive camera localization systems using only the natural environment, but for now the most robust and available systems use patterns or objects added to the environment for the vision system to detect in order to localize the camera.

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. Some sytems allow the full six degrees of freedom of pose as required for augmented reality systems. 3D objects can be rendered using the position of the recognized pattern in the image (if some intrinsic camera parameters are known). Pose information can be extracted from the *homography* [15] that describes the pattern to image plane mapping.

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 tradeoff between missing a marker and seeing a non-existent one. With ARToolkit, this is directly manipulated by the user by setting a threshold confidence parameter. With ARTag, this tradeoff was done in the algorithm design when the the number of digital bits of checksum was balanced with the number of bit errors the error correction would be set to correct.

Several 2D pattern systems are available (Section 2), of which *ARToolkit* [13, 16] applications constitute a large share of AR systems (ISMAR [8] is an augmented reality conference with many ARToolkit application papers). Zhang [21] performs a

survey of several fiducial marker systems including ARToolkit with respect to processsing time, identification, image position accuracy with respect to viewing angle and distance. ARToolkit is popular becaue it is simple, robust, and freely available. It is a successful and popular system consisting of a square black marker on a white background. If a marker is successfully located, the four corners are used to compute the camera pose for rendering virtual objects to augment natural imagery.

ARToolkit 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 contours that could be the markers' outside boundaries, and secondly by correlating this interior pattern with the known patterns. This paper proposes a method to robustify this marker system by replacing this second step with a digital symbol method. In this proposed method, this interior pattern is replaced with a 6x6 grid of square regions of solid black or white colouring. This encodes a 36-bit binary code which is a encoded form of a smaller 10 bit ID number with checksums and error correction redundancy. This system is called *ARTag*.

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 create a sampling grid to extract an appearance vector. ARToolkit (usually) uses a 16x16 grid of interior points, sampled with 256 grey levels. ARTag only samples a 6x6 grid in the interior, and thresholds to extract a 36-bit code of single '0' or '1' symbols. This digital code is analyzed digitally instead of comparing greyscale image patches by correlation. The different approach is conceptually similar to the difference between digital radio communications such as modern cell phones, and analog radio. A non-linear decision is made early in the processing path and the analog signal quantized to one of several discrete levels with subsequent processing performed on digital symbols.

As with ARToolkit, all four possible rotation configurations are tried. Forward error correction (FEC) is used to robustify the system and can recover the 10 bit ID code with several of the interior region bits occluded or erroneously sampled.

There are three main advantages to this system; an almost zero false detection rate, a simpler system requiring no stored patterns, and a smaller marker size. The first is the principle motivator for ARTag, it reduces the probability of falsely identifying one marker for another, or a piece of the background as a marker, to a probability of $< \%0.0079$.

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 ubiquitious bar code seen on consumer products. The purpose is to carry information, not to localize as is

needed for augmented reality. The US Postal Service uses the *Maxicode* marker to convey shipping information (Fig.1). *Data matrix* and *QR (Quick Response)* are two other examples designed to contain information are used in industrial settings for part labelling (also shown in Fig.1). 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 [4] is a standard for *Data matrix* 2D patterns and uses Reed Solomon error correction, which can recover from situations where part of the information read from the pattern is corrupted. Data Matrix and QR are used for *Direct Part Marking* (DPM) to identify and convey information along an assembly line (see [1] for the automotive industry).

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.

The steps in the three commercial systems shown at the top of Fig. 1 are; 1) *Identification* of the marker, 2) *Alignment* of sampling axis with the marker pattern, 3) *Digitization* of the image intensity to a binary logic '0' or '1', 4) *Error Detection and Correction*, and finally 5) *Decoding* of the corrected bit pattern into usable information. The first step, identification, uses some unique identifier to find the marker in the image, for example the bull's eye concentric rings in the Maxicode system. The second stage, alignment, attempts to line up the pattern for digitization. This second stage is analogous to a linear bar code scanner that uses calibration stripes at the beginning, end and sometimes middle to use to linearly interpolate positions within to make binary sampling decisions. Finding the peaks in spatial frequency plots (usually FFT's) is used in both linear and two dimensional patterns to identify the spatial frequency and phase of the dominant repetitive intervals. The third stage of digitization is a threshold procedure, either absolute or local, to abstract to a field of 0's and 1's. 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.

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.

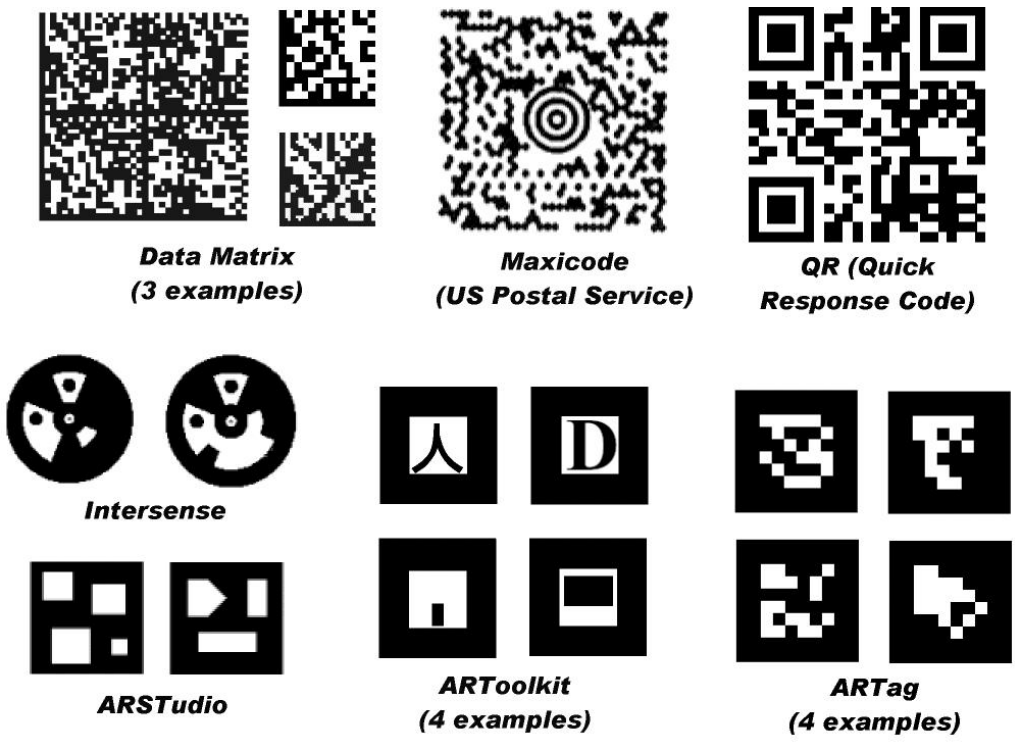


Figure 1: *Several planar pattern marker systems. Data Matrix, Maxicode and QR are industrial systems used for carrying data. The circular Intersense markers are used in position tracking. ARStudio and ARToolkit are patterns designed specifically for augmented reality applications. ARTag is the new marker system introduced in this paper.*

Small markers can be made by encoding a ring of segments around a circular dot. 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" [7], or positioning products from Aicon[2], Capture3D [3], and others. These systems are used for photogrammetry applications such as measuring the "as built" measurements of industrial plants or the deformation of automobile parts after crash tests. Several use special photoreflective material in the markers which reflect light from a light source mounted on the camera [5]. The number of possible markers is limited, the camera resolution will limit the number of segments that the annular ring can be broken into. Typically there is no more than 8 segments that are present and absent, limiting to a handful the size of the marker library. Any error checking redundancy will reduce this set further. Knyaz and Sibiryakov [17] divide the space between a central circle and a ring into 10 sites for solid dot to sit, since each dot requires an empty space beside this is equivalent to dividing an annular ring into 20 segments. However, after addressing rotation and reflection, only 76 ID's are possible. The Intersense markers [19, 6] in Fig.1 extend this concept to several radii. However, all these circular fiducial markers must be seen in sets (more than one at a time) to allow pose calculation and hence the total pixel size required starts to grow.

The distinguising feature of systems designed for augmented reality such as AR-Toolkit and ARSTudio [18] is that only one marker is usually visible so the fiducial marker must have some distinct points, at least four, so as to extract orientation with perspective distortion. ARToolkit, for example, uses the quadrilateral outline to accurately locate 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.

The popular ARToolkit, and the new ARTag system introduced in this paper, were designed for augmented reality applications [9], but are can also be used for landmarks in robot control [12] and mobile robot navigation [11]. The issues of reliability and a minimal false positive and inter-marker confusion rate addressed in this paper apply equally to these domains also.

3 ARToolkit

ARToolkit [13] is very popular by designers and users of augmented reality and Human Computer Interaction (HCI) systems due to it's available source code and ubiquity of use.

ARToolkit is a marker system devised by Dr. Hirokazu Kato of Osaka University, Japan, and supported by the University of Washington [1]. The markers are planar and can easily be printed out and mounted to a flat card or wall, several are shown in

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

Fig. 2. ARToolkit markers consist of a square black border with a variety of different patterns in the interior. The quadrilateral black outline is analogous to the unique feature for step 1: *Identification* in the above listed commercial systems. The corners are used for *Alignment*, they then define a sampling grid inside the pattern which is sampled to provide a 256 element feature vector which is compared (in four possible orientations) to a library of known markers.

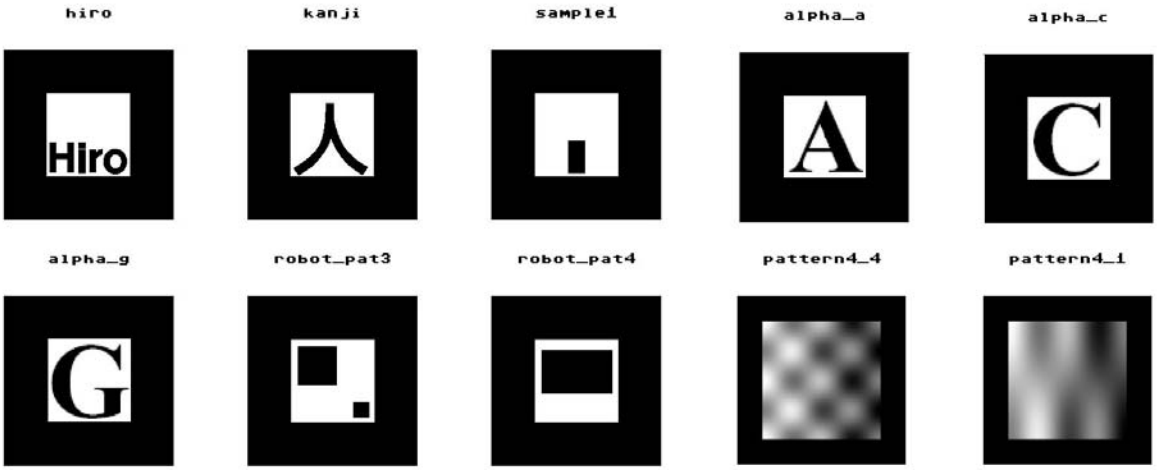


Figure 2: *ARToolkit markers. Top row and 'G' marker on second row are part of the set of original markers used. 2nd and 3rd from the left on the second row are markers made in our lab to reduce inter-marker confusion. Rightmost two markers on the second row are the DCT based markers from Owen* et al.*[10] also designed to address the misidentification problem with ARToolkit.*

ARToolkit's robust operation is due in part to its utilization of the large contrast between white paper and black ink, the difference can allow the use of a single greyscale threshold to distinguish between them. This threshold is the sole parameter given to ARToolkit's *arDetectMarker()* function and is used for the first stage where the image is binarized. ARToolkit first finds black quadrilateral borders by finding connected groups of pixels below this level, the contours of these groups are found and those contours with four straight sides are identified as potential markers. These first few stages are shown in the middle three images in Fig. 3.

The corners of the quadrilateral contours are used to remove the perspective distortion and bring the internal pattern to a canonical front view. In practice 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

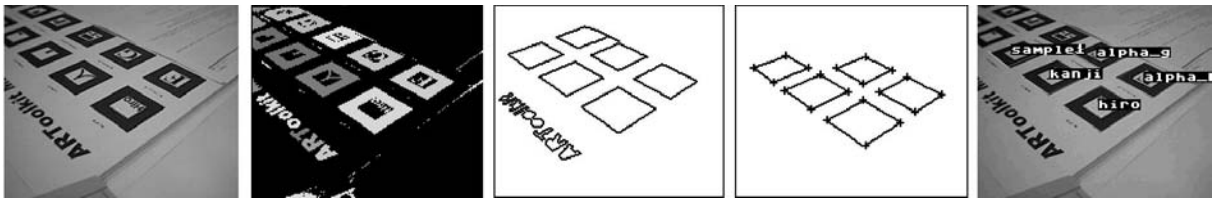


Figure 3: *The first few stages of ARToolkit marker extraction. ARToolkit takes an input image and a grey level threshold parameter and reports the location of markers detected in the image. Left to right; 1) the original image, 2) connected components of pixels with a grey level less than the threshold 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.*

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. Similar marker patterns, especially the Kanji and letter 'A' patterns in the original set provided with the software, are frequently mistaken for one another.

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. 4 below shows three markers and what the data inside their corresponding pattern files look like.

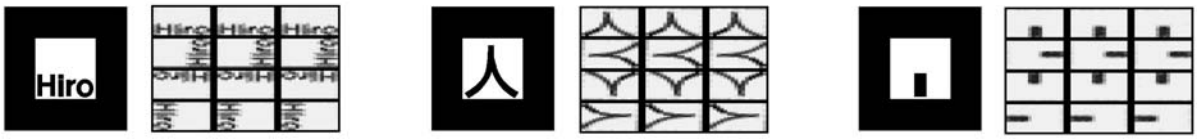


Figure 4: *ARToolkit markers and their corresponding pattern files. (Left to right); 1) the Hiro marker image, 2) the pattern file* patt.hiro*, 3) the Kanji marker image, 4) the pattern file* patt.kanji*, 5) the Sample1 marker image, and 6) the pattern file* patt.sample1*.*

Some of the original *ARToolkit* markers are shown in Fig.1. Owen [10] demonstrates that these original patterns inside the black border are not necessarily optimal, and proposes others based on spatial frequency components.

4 ARTag

ARTag is a planar pattern marker system developed at the National Research Council of Canada that has 2046 markers, with improved performance to ARToolkit in identification and verification, and no need of pattern files.

ARTag is a new system designed to address some of the shortcomings of ARToolkit. The main problem with ARToolkit is that it falsely detects markers where there are none, and frequently confuses them. The paradigm of using a square border with an interior pattern is maintained, but the processing of the internal pattern is replaced with a digital approach.

The feature of ARToolkit that, in the eyes of this author, contributes most to its functionality is the use of only black and white for the border, using only two extremes of reflectance in a marker allows many issues of image capture and greyscale non-linearity to be avoided. This binary decision is extended in ARTag from just defining the border to defining the inner pattern as well.

The ARToolkit square border is useful for AR since it has four prominent corner. Four 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). ARTag extends this concept by allowing both polarities of borders; a black border on a white background and an white border on a black background, whereas ARToolkit uses only the former. This allows a doubling of the space of possible ID's with no extra marker size or complexity.

ARTag was designed to contain the successful elements of ARToolkit and Datamatrix, and take the best of both and make a minimal but robust system for AR. Datamatrix has a robust data communication capability but requires that the pattern be mostly parallel to the image plane (viewed straight on) and 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. Datamatrix is not suitable to be viewed with any non-negligible perspective distortion. Datamatrix is also designed to carry more information than just an ID and so it is typically larger and requires more image pixels, reducing the range of distances from the camera that it can be used at.

Several ARTag markers are shown in Fig. 7. The main characteristics are a square border of either polarity (white on black or black on white) and a 6 x 6 square grid dividing up the interior. 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. Detecting this boundary is performed in the same way as ARToolkit (at least in the first version of ARTag complete at the time of this writing) by thresholding a greyscale version of the image, performing connectivity of pixels above and below

this threshold, and finding those connected objects with a quadrilateral boundary (as shown in Fig. 3).

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' using the same threshold used to find the 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 [20], 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.

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. 5 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. 6 are performed and the verdict of ARTag marker presence or not is decided, and the ID reported if present.

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 *1000100000100001*. 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 and general image noise.

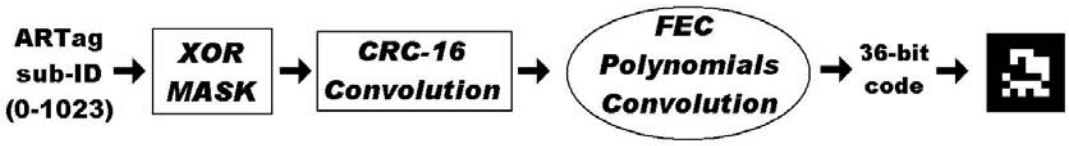


Figure 5: *Digital encoding process : creating ARTag markers. A sub-ID number (the lower 10 bits of the ARTag ID) is converted to a 10-bit binary sequence which goes through several stages to produce a 36-bit binary sequence which is encoded in the marker as white and black cells.*

Two ARTag ID numbers, 682 and 1706, are absent from the ARTag marker library reducing the library size to 2046. 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 interior which will be frequently falsely detected in the environment.

5 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 sytem reports that a marker is present when it is not. With ARToolkit and ARTag this could happen when quadrilateral

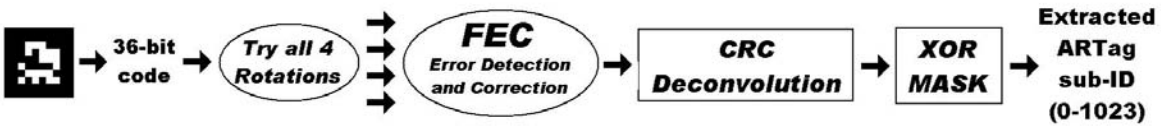


Figure 6: *Digital decoding process : confirming and identifying ARTag ID's in the binary pattern from the interior of an ARTag marker. The binary pattern from the marker as seen in the camera image is converted into four possible 36-bit codes for each possible rotation. Each code passes through the 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.*

shapes are found in the environment, and the interior pattern passes the verification/identification test as shown in Fig. 8. This is a problem with ARToolkit, 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. Using the *arSavePatt()* function to save the interior pattern as seen by that camera in the environment of the application is recommended, this will give the highest confidence factor (c.f.) when the marker is seen. The main purpose of this is to allow the threshold for c.f. to be set as high as possible for the purpose of reducing false positives. However, this only alleviates the problem and reduces the flexibility, the application can not be easily moved to another camera and environment without reducing the c.f. and getting more false positives as a result.

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. $1023 \cdot 4 = 4092$ 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. With N=2, each valid ARTag 36-bit code would be mapped from $1 + 36 + 36^2 = 1333$ 36-bit codes, yielding $4092 \cdot 1333 = 5.45$ 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 $\frac{5.45 \cdot 10^6}{68.7 \cdot 10^9} = 7.9 \cdot 10^{-5} = \%0.0079$, or about one in 12,600. Therefore a non-marker quadrilateral has a %0.0079 chance of producing a

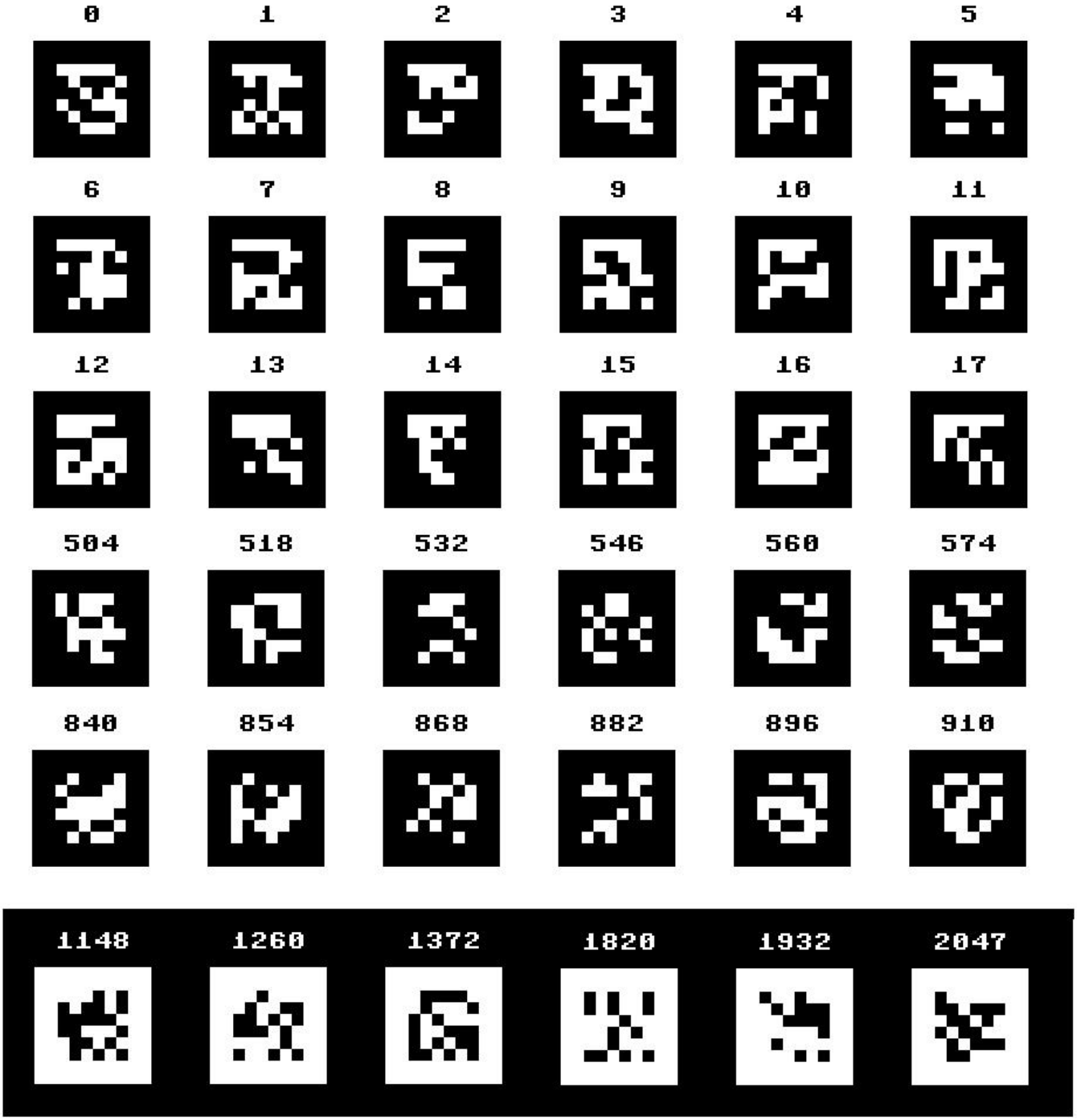


Figure 7: *ARTag markers. 36 out of the range of 2046 markers in the ARTag marker library. A 10-bit code is contained in the interior pattern, protected by checksums and forward error correction (FEC) to greatly reduce false positives and inter-marker confusion. Unlike ARToolkit which only uses a black 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.*

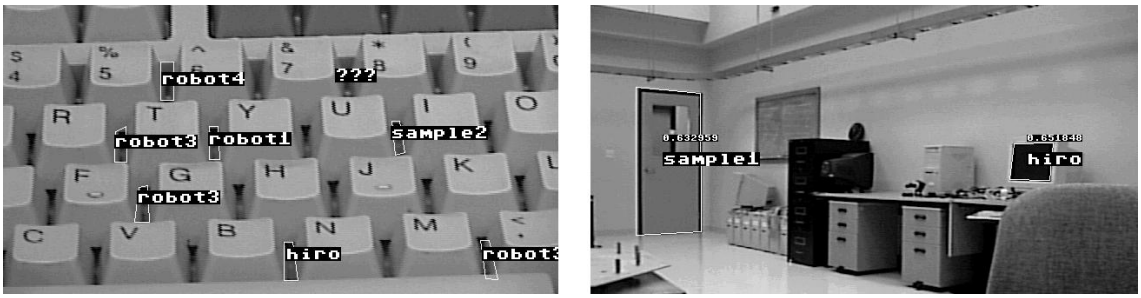


Figure 8: *Examples of ARToolkit false positives. Detected markers are overlaid over the image. Black quadrilateral regions, such as between the keyboard keys, or the computer monitor or doorway have interior pixels that correlate to one of the marker definitions with a c.f. value above the threshold resulting in a false positive detection.*

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. Three different USB webcams and a Sharp VL-AH150U NTSC camcorder were used, and the camcorder was digitized with two different frame grabbers; an ATI All-in-Wonder card and a USBLive USB peripheral. The movie "Bladerunner" (Ridley Scott 1982) was also used as an image sequence in order to get more random and varied imagery than video footage from our lab. The c.f. value of the false positive markers detected by ARToolkit were used to classify them.

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, then the problem of false negatives, declaring there is no marker present when there really is, starts to be a problem. The c.f. values depend on the pattern files and the camera and environment used. So to put Table 1's data in perspective, the confidence values seen when the markers were present with the same conditions (except the movie of course) were recorded. The c.f. values were recorded for four markers as a function of the marker width in pixels. The markers were moved forward and backward from the cameras and the confidence value plotted with respect to the pixel distance between two corners of the marker. Four different markers were used and their curves overlaid

Imagery		ARToolkit							ARTag
Camera/Sequence	Frames/ Dura-tion	c.f.= 0.50	c.f.= 0.60	c.f.= 0.70	c.f.= 0.75	c.f.= 0.80	c.f.= 0.85	c.f.= 0.90	
NTSC, ATI	2723 2 mins	604	175	34	15	9	3	0	0
NTSC, ATI	3514 2 mins	799	235	21	9	3	3	0	0
NTSC, USBLive	3318 2 mins	786	401	120	70	0	0	0	0
Intel CS120	1450 2 mins	261	135	94	88	70	27	27	0
Intel PC Pro	1318 2 mins	395	68	19	14	3	0	0	0
Intel PC Pro	1318 2 mins	395	68	19	14	3	0	0	0
Telemax WC50	2893 2 mins	727	410	10	3	0	0	0	0
Movie (Bladerunner)	215625 120 mins	7917	2418	408	180	112	0	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. The detection rates for the ARToolkit markers are given for several confidence factor (c.f.) threshold values; the lower the c.f. threshold, the higher the number of false positives. Note that no false positive ARTag markers are declared in any of the frames.

(Fig. 9. Looking at these graphs we see that the markers need to be at least 40 pixels wide to have a chance of detecting them often if the confidence value threshold is set as high as 0.75. Setting the threshold to 0.90 would yield only occasional detection which would likely not be useful.

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 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 envi-

ronment is mostly all dark or all white. Therefore the probability of false positive detection is less than %0.0079 explaining why it is not unexpected to see zero false positives even in an experiment as large as conducted in Table 1.

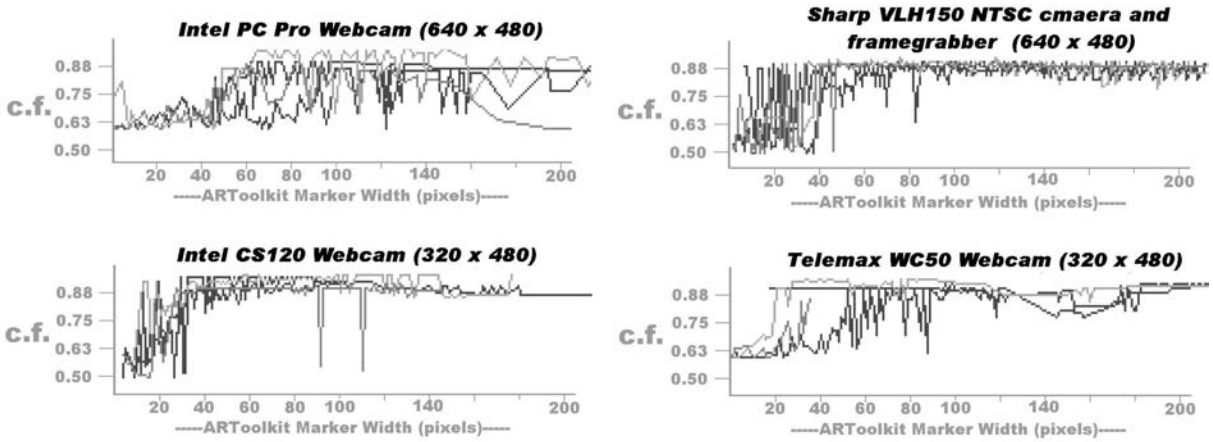


Figure 9: *ARToolkit Confidence value (c.f.) plotted as a function of marker width in pixels for four cameras used in Table 1. The 'Hiro', 'Kanji', 'Sample1' and 'Robot4' markers from Fig. 2 were used, and their results overlaid.*

The conditions of these two experiments were a bit harsh, in that the pattern files were not customized and the video is of random, unstructured scenes. The pattern files were the ones provided (in the original ARToolkit release) and not customized for that camera and lighting. Many ARToolkit applications have a constrained environment where the camera is fixed and mostly only markers move into the field of view. The procedure of creating pattern files, not having any other black quadrilaterals appear often, and adjusting the c.f. threshold to balance false positives and negatives will likely work in constrained applications. However, it is inconvenient to do this preparation and limits the general use such as augmented reality for the mass consumer market. The low false positive rate, and removal of the need to capture pattern files are advantages of ARTag over ARToolkit. Also, the improved false positive performance allows ARTag to be used for other applications such as robot navigation with ARTag markers as landmarks.

6 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. For example, if an ARTag marker #208 is reported

by ARTag as marker #146 due to some error, the usefulness and robustness of the system would be called into question. 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.

In this failure mode, the marker has passed the unique feature test, which is having a quadrilateral shape for ARToolkit and ARTag. The second stage of reading the unique inner code can fail and the wrong id can be reported. Owens *et al.* [10] explores the simularity between ARToolkit markers with the *Mean Squared Error* (MSE) approach. An intrepretation of the MSE approach is that of considering the image patches inside as 256 (for the original 16 x 16 ARtoolkit sampling) element vectors and finding the euclidean distance between them. They report the MSE between the "Kanji" and letter "A" in the originally released ARToolkit marker set as 0.498 and an MSE of 0.820 between the letters 'C' and 'G' (these markers are in Fig. 2). These are in the same units as ARToolkit's confidence value and mean that even with the *c.f.* threshold set to 0.80, those markers can still be often confused. They proposed a new set of markers which aim to reduce the MSE to nil by using the orthogonality between different spatial frequencies. Users of ARToolkit inevitably end up trying different marker patterns to reduce confusion, such as the "robot3" and "robot4" markers created in our lab to try achieve less confusion for an application of ARToolkit for robot control [12]. However, the problem is only reduced, and the set of possible markers must be kept small to allow uniqueness.

This new proposed ARTag system does not use image correlation, 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*[14], 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.

6.1 Designing the ARTag Libary

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. 5,6 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 [20]. 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 6.2) when non-recommended sub-ID's were chosen (Section7).

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. 10. 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.11.

Looking at Hamming distance histograms of the chosen system in Fig. 10 and the best four second place contender systems in Fig.11, we see they all start with values

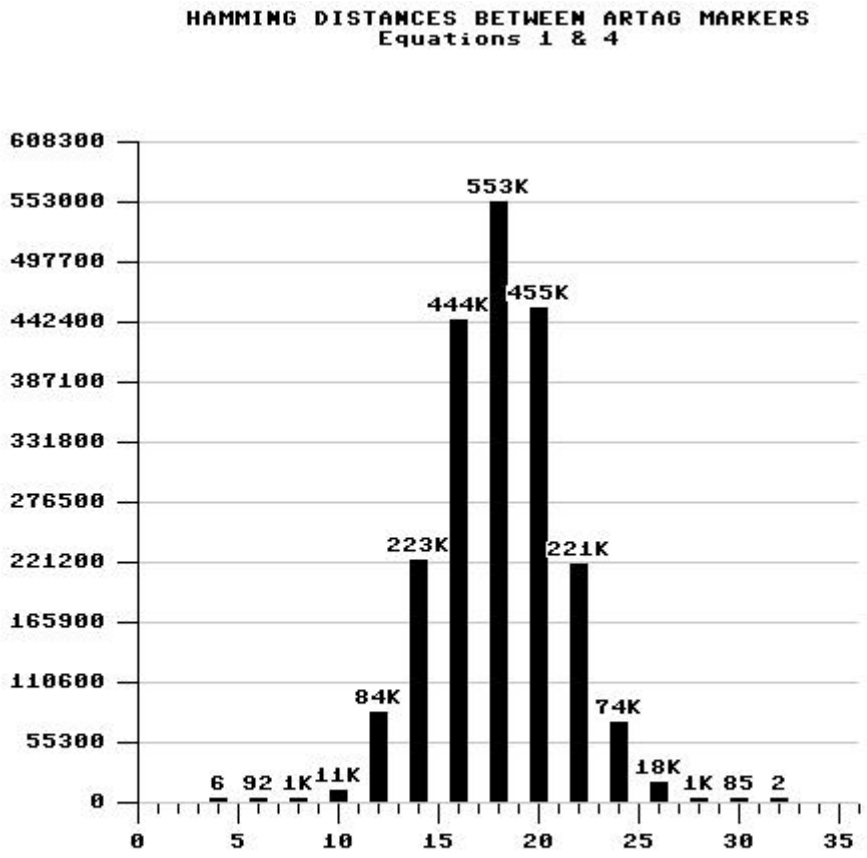
at $n = 4$, however the chosen system has six elements at this distance as opposed to only one in the other four contenders. This would seem contrary to the stated goal of decreasing the first few histogram bin entries, however, not only the first bin entry was taken into consideration. These four other 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).

6.2 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 6.1 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.12. 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.



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 10: *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.*

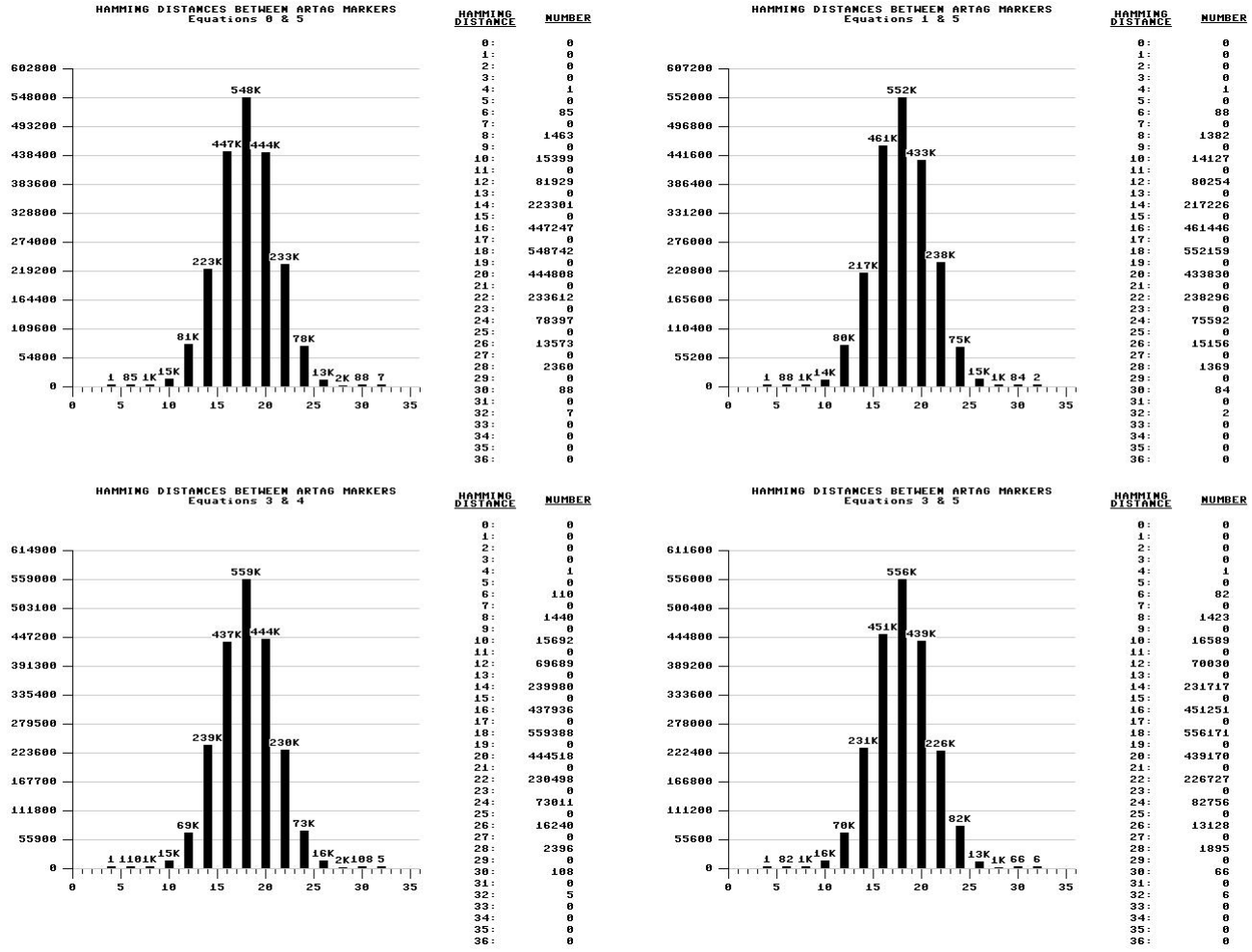
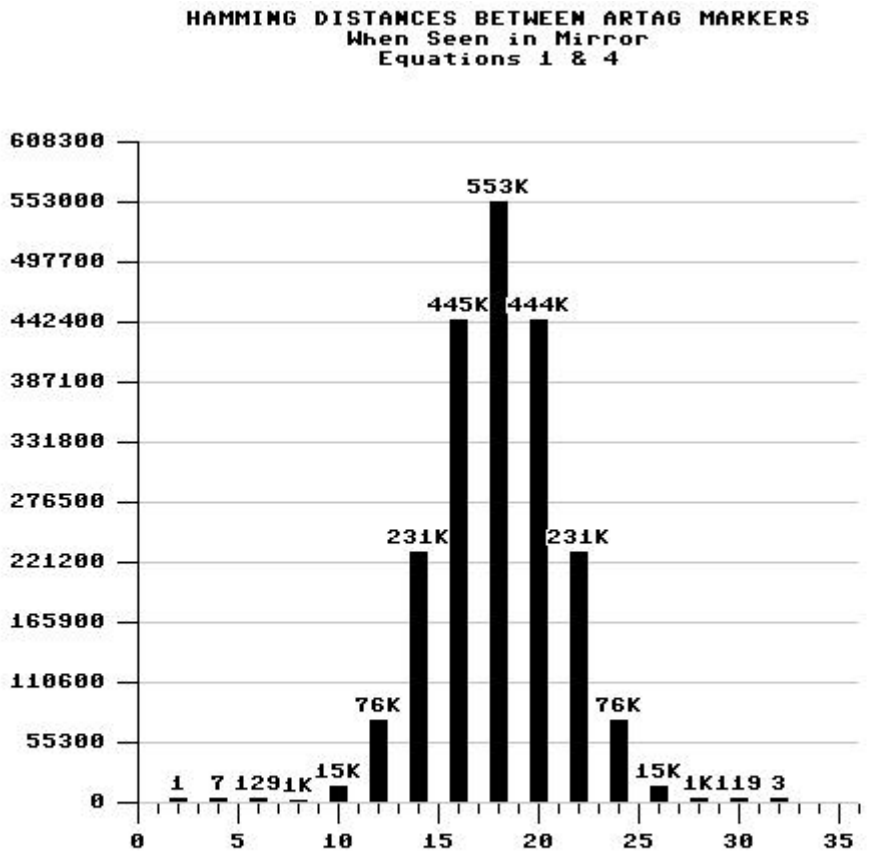


Figure 11: *Hamming distances between ARTag markers with different FEC polynomials. The FEC polynomial set chosen in Fig. 10 was selected for ARTag due to the minimum area below a Hamming distance of 11 bits, even though the selected set has 6 entries in the lowest Hamming distance=4 bin. There are six possible polynomials available, giving 15 possible sets for use when correcting up to two bit errors in a marker. These four histograms plus Fig. 10 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 12: *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*

6.3 Recommended Subset of the ARTag Libary

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 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 occuring 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 calculed for all 1023 sub-ID's and the most problematic markers were put on a *not-recommended* list (Table 3).

For most applications, all 2046 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-

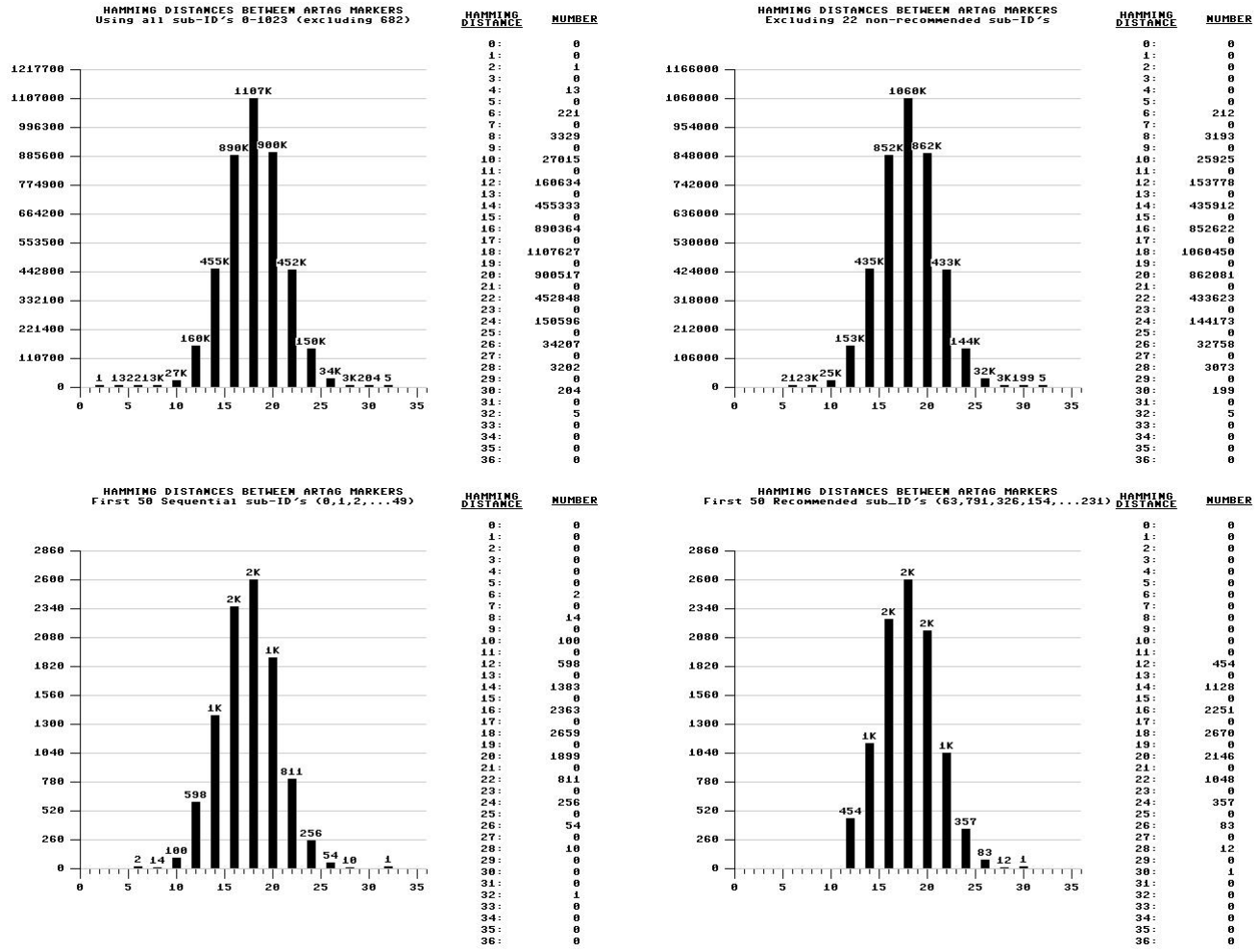


Figure 13: *Minimum Hamming distances: A histogram of the* minimum *Hamming distance between all markers in a set, considering rotation and mirroring. (Upper Left) Histogram if all legal 1023 sub-ID's are used. (Upper Right) Histogram if 1002 recommended sub-ID's (all 0-1023 except 682 and the 22 sub-ID's 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* $n = 12$.

ID's from this list instead of chosing them arbitrarily was shown in Fig.13. Here the Hamming distance histogram was calculated for a marker set containing the first 50 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.

7 ARTag and ARToolkit Minimum Pattern Size and the False Negative Detection Rate

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 requiring as small as possible of a size requirement in pixels.

Zhang [21] reports a minimum width of 14 pixels required for detection, however only one marker was used, and his experiments with multiple markers were performed with the ARToolkit markers being at least 60 pixels wide. This correlates with the our experiment shown in Fig. 9 where three of the cameras used reached a plateau in c.f. value around this width (somewhere 30-50 pixels).

With ARTag and the cameras used in this paper, one needed about 35-40 pixels wide to get over %95 detection, lower than a %5 false negative detection rate. The markers could occasionally be detected as small as 12 pixels wide. With the marker pattern being 10 units wide, that would set the lower bound and twice that (a width of 20 pixels) should be attainable with good optics and a low noise imager. The cameras used in these experiments were consumer USB cameras and an NTSC camera and so were of fairly low quality and better results, such as the $<$ %5 false negative detection rate observed, is anticipated for high quality cameras. The full experimental results for our experiments of detection rate (1.0 - false negative rate) is shown below in Table 4.

In practice, when designing a system using ARTag or ARToolkit, the border needs to contrast a background. The marker widths mentioned above are only of the border, the white/black border is important for the unique identification and thus the marker panel should have a minimum area of background black or white enlarging the total size by at least a few pixels. For best operation, one should have as much as 50% of the marker width extra of background around the marker when mounting them in the environment.

The false negative rate is a function of the first half of the marker detection system, that of finding the border, whereas the false positive and inter-marker confusion rates are a function of the second half. This second half is addressed in creating this new

system called ARTag, and more improvements improving the false negative rate are a future work.

8 Processing Time

The processing time a fiducial marker system takes is important, ARToolkit's real time performance is one reason for its ubitquity 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 should be less computations than the 16x16 integer multiplication 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 searchs among lookup tables of 12 bytes 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 2046 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 (as described in the last section) performed similarly for both ARToolkit and ARTag and involves visiting each pixel in the image once for thresholding and then performing connectivity. Therefore, we expect a constant time if no markers are visible, and then an increased processing time for visible markers.

The time in milliseconds was measured before and after the marker detection function was called, the times are given below in Table 5 for both ARToolkit and ARTag, for two different cameras and several numbers of visible markers. These measurements were taken on a PC running Windows 2000 with a Pentium 4 processor running at 3.0 GHz. An Intel CS120 USB webcam running with DirectX's DirectShow library and a firewire (IEEE 1394) Dragonfly digital video camera with the provided SDK from Point Grey Research were tested. ARToolkit was loaded with 10 pattern files. The time measurement was not able to be measured with a good granularity when using the IEEE 1394 Dragonfly camera (in increments of 15 ms only) and so only ranges of times are reported.

ARTag is similar to ARToolkit in processing time for the case of there being no

markers visible, but has an improved performance as the number of markers rise, for example ARTag only takes 9ms as opposed to 22ms for ARToolkit when 48 markers are visible in the field of view.

9 ARTag Usage

ARTag is available for download [2], [3], [4] for evaluation and usage in non-commercial systems. A header file *artag_revX.h* and a library file for Windows and Linux programs is provided, along with sample applications.

An existing ARToolkit application can be readily converted with a few simple steps; including the header file, calling the initialization function *init_artag(image_width,image_height)*, and simply replacing calls to ARToolkit's *arDetectMarker()* function call with the function *artagDetectMarker()* which will return detected markers in the same data structure as ARToolkit. New applications can more simply instead call the function *artag_find_marker()* or *artag_find_marker_white_only()*, the latter of which saves processing time by only looking for ARTag markers in the range 0-1023.

The application will work well using sequential sub-ID's of 0,1,2,...etc, but enhanced immunity to inter-marker confusion can be gained by pulling sub-ID's from the recommended list by calling *artag_get_id()* in place of ARToolkit's *init_artoolkit()* pattern loading function. In this way a set of ARToolkit markers with large inter-marker Hamming distances will be chosen.

10 Conclusions

A new marker system, called *ARTag* was created to improve apon the successful *ARToolkit* marker system based on passive vision of 2D planar markers. The quadrilateral border concept of ARToolkit was used along with a digitally encoded, error corrected ID code to replace ARToolkit's pattern recognition and identification step. ARTag has a library of 2046 unique ID markers without the need to load any pattern files. ARTag manages to achieve a low false negative rate comparable to ARToolkit, but has a vastly lower false positive error rate (%0.0079) and very low inter-marker confusion rate.

The ARTag system was designed to lower the probabilities of false positive detection and inter-marker confusion and successfully reduced both by a large amount relative to ARToolkit. The processing time performance is also superior when many markers are visible simultaneously.

[2] /http://www.cv.iit.nrc.ca/research/ar/downloads.html

[3] /http://www.cs.ualberta.ca/ fiala/artag (tilda before fiala)

[4] /http://www.millennium3engineering.com/artag/

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

References

[1] http://www.aiag.org/ (automotive industry action group website).

[2] http://www.aicon.de.

[3] http://www.capture3d.com.

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

[5] http://www.geodetic.com.

[6] http://www.isense.com.

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

[8] *2003 IEEE / ACM International Symposium on Mixed and Augmented Reality (ISMAR 2003), 7-10 October 2003, Tokyo, Japan*. IEEE Computer Society, 2003.

[9] G. Roth C. McDonald and S. Marsh. Red-handed: Collaborative gesture interaction with a projection table. pages 773–778.

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

[11] M. Fiala. Artoolkit applied to panoramic vision for robot navigation. In *Proc. of Vision Interface*, pages 119–127, June 2003.

[12] M. Fiala. Vision guided robots. In *Proc. of CRV'04 (Canadian Confernence on Computer and Robot Vision*, pages 241–246, May 2004.

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

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

[15] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. Cambridge, UK, 2000. Cambridge University Press.

[16] 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.

[17] 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.

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

[19] 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.

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

[21] 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 mimimum 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 recommded 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.

Imagery		ARTag Detection Rate					
Camera Type	Camera Resolution	Width = 15	Width = 20	Width = 25	Width = 30	Width = 35	Width = 40
NTSC, ATI	640 x 480	0.00	0.17	0.33	0.92	1.00	1.00
Intel PC Pro	640 x 480	0.04	0.49	0.70	0.73	0.85	1.00
Intel CS120	320 x 240	0.12	0.68	0.80	0.95	0.98	1.00
NTSC, USBLive	320 x 240	0.11	0.60	0.99	1.00	1.00	1.00
Telemax WC50	320 x 240	0.11	0.37	0.87	0.87	0.99	1.00

Table 4: ARTag Detection rate for several cameras. Rate is given (0.00-1.00) of when an ARTag marker is detected as a function of marker width in pixels. The camera focus and ARTag binary threshold had to be adjusted for each test. The tests consisted of presenting a grid of 9 markers and moving the camera (or zooming the NTSC Camcorder) and recording the number detected over 100 frames. Note: The Sharp NTSC camera was used with two frame-grabbers, it appears that the USBLive frame-grabber has better results. However, it has half the image resolution and so its results should be compared to a marker width twice as wide with the ATI frame-grabber.

		ARToolkit				ARTag			
Camera		Number of Markers Visible				Number of Markers Visible			
Type	Resolution	0	24	32	48	0	24	32	48
Intel CS120	320 x 240	3 ms	17 ms	22 ms	22 ms	2 ms	7 ms	8 ms	9 ms
IEEE 1394 Dragonfly	640 x 480	0-15 ms	0-15 ms	15-31 ms	15-31 ms	0-15 ms	0-15 ms	0-15 ms	0-15 ms

Table 5: ARToolkit and ARTag Processing Times as a function of camera and number of markers present in the image. Tests were performed on a 3.0 GHz Pentium 4 processor PC. For both systems, the processing time per frame rises with the number of markers to process.