A LINE DETECTION BASED FIDUCIAL MARKER DETECTION ALGORITHM

Burak BENLİGİRAY

MS Dissertation

Graduate School of Sciences
Electrical and Electronics
Engineering Program

May, 2014

ANADOLU ÜNİVERSİTESİ

JÜRİ VE ENSTİTÜ ONAYI

Burak Benligiray'ın **A Line Detection Based Fiducial Marker Detection Algorithm** başlıklı **Elektrik-Elektronik Mühendisliği** Anabilim Dalı **Elektronik** Bilim Dalındaki Yüksek Lisans tezi 21.05.2014 tarihinde aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim - Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı -Soyadı	**İmza**
Üye (Tez Danışmanı) :	**Doç. Dr. Cüneyt AKINLAR**	.....
Üye :	**Prof. Dr. Ömer Nezih GEREK**	.....
Üye :	**Yard. Doç. Dr. Hatice ÇINAR AKAKIN**	.....

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... tarih ve ..... sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ABSTRACT

MS Dissertation

A LINE DETECTION BASED FIDUCIAL MARKER DETECTION ALGORITHM

Burak BENLİGİRAY

Anadolu University
Graduate School of Sciences
Electrical and Electronics Engineering Program

Supervisor: Assoc. Prof. Dr. Cüneyt AKINLAR
2014, 64 pages

Fiducial markers are artificial reference objects placed on the scene to create correspondances between the real world and computers' interpretations of the real world. Computer vision applications can utilize a prori knowledge about the model of the fiducial marker to estimate the camera's pose relative to the fiducial marker. Camera pose is used in many applications such as augmented reality, human-computer interaction, industrial applications, navigation and so forth. The quality of the pose estimation offered by the fiducial marker implementation is critical in how accurate these applications perform.

The objective of this dissertation is to design a novel fiducial marker system. Firstly, existing fiducial marker systems are investigated to explore proposed solutions to commonly encountered problems. Following that, major performance criteria for fiducial marker systems are determined. Design decisions that should be taken to improve each of these criteria, and tradeoffs these decisions introduce are investigated. A new fiducial marker system, EDMarkers, is proposed according to the collected knowledge about viable design decisions. Finally, the proposed marker system is evaluated using the aforementioned performance criteria.

Keywords: Fiducial marker; Augmented reality; Lexicode; Pose estimation; Quad detection; Computer vision.

ÖZET

Yüksek Lisans Tezi

DOĞRU PARÇASI TESPİTİ TEMELLİ BİR İŞARETÇİ TESPİTİ ALGORİTMASI

Burak BENLİGİRAY

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Cüneyt AKINLAR
2014, 64 sayfa

Referans işaretçiler, bilgisayarların gerçek dünya algılarıyla gerçek dünya arasında ilişkiler kurabilmeleri için suni olarak üretilip sahneye yerleştirilen objelerdir.Bilgisayarlı görü uygulamaları, işaretçinin modeli hakkında önceden sahip olduğu verileri kullanarak kameranın referans işaretçiye olan bağıl pozunu hesaplayabilir. Kamera pozu bilgisi artırılmış gerçeklik, insan-bilgisayar etkileşimi, endüstriyel uygulamalar, yol bulma gibi pek çok uygulamada kullanılır. Referans işaretçinin sunduğu poz bilgisinin kalitesi, bu uygulamaların ne kadar hassasiyetle çalışacağını belirler.

Bu tezin amacı yeni bir referans işaretçi sistemi dizayn etmektir. Öncelikle konuyla ilgili sıkça karşılaşılan problemlere uygulanan çözümleri keşfetmek için mevcut referans işaretçi sistemleri incelenmiştir. Bunu takiben referans işaretçi sistemlerinin temel performans kriterleri belirlenmiştir. Bu kriterlere bağlı olarak sistemin geliştirilebilmesi için alınması gereken dizayn kararları ve bu kararların diğer kriterleri nasıl etkilediği araştırılmıştır. Dizayn kararları ile ilgili derlenen bilgiler toplanarak EDMarkers adında yeni bir referans işaretçi sistemi sunulmuştur. Son olarak sunulan yeni işaretçi sistemi önceden belirtilen performans kriterlerine göre değerlendirilmiştir.

Anahtar Kelimeler: Referans işaretçi; Artırılmış gerçeklik; Kod sözlüğü; Poz tahmini; Dörtgen tespiti; Bilgisayarlı Görü.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ID	Identification
AR	Augmented Reality
CRC	Cyclic Redundancy Check
Quad	Quadrilateral
HCI	Human-Computer Interaction
GNU	GNU's Not Unix
GPL	General Public License
FEC	Forward Error Correction
Lexicode	Lexicographic code
FPS	Frame Per Second
ν	The length of the encoded bit sequence
λ	Number of markers in the library
η	Minimum Hamming distance between valid bit sequences
ϵ	Maximum number of bits to be corrected
γ	Average number of candidate markers in each frame
β	Probability of a single bit of the marker flipping

1. INTRODUCTION

In this chapter, motivation to design and use a fiducial marker system will be discussed. Afterwards, the scope and objective of the study will be explained.

1.1 Motivation to Use Fiducial Markers

The aim of computer vision is to develop methods that comprehend and interpret projective images and determine suitable outputs for the related applications. A 3D description of the associated environment or objects may be needed to achieve this aim. Accordingly, obtaining 3D pose information from 2D projections of the scene is one of the main research topics of computer vision [3, 4].

Pose information refers to the retrieval of spatial orientation and location information of an object or coordinate system with respect to another object or coordinate system. If the camera that captures the projective image is assumed to be the first object, pose information of any other object in the scene may be determined relative to the camera. While this second object may be in the scene naturally, it may also be specifically designed, fabricated and placed to be used as reference. If there are multiple reference objects with the same 3D shape, there should be additional differentiating factors among them. For natural reference objects, these differences will be incidental, while for artificial objects, these differences will be designed and applied deliberately.

To obtain the pose information of a natural object whose 3D model is already known, correspondences between the projective image and the original object is found using robust feature detectors such as SIFT [5], GLOH [6] or SURF [7]. Ideally, a large number of correspondences should be detected between the image and the original object. Subsequently, a robust camera pose estimation algorithm such as Least Median of Squares [8] or RANSAC [9] is used. Vuforia [10], Junaio [11] and other commercial systems produce acceptable results using such methods. See Figure 1.1 for a markerless AR application. Not needing an artificial reference object in the scene is the main advantage of using natural objects as reference objects. On the other

Figure 1.1: A depiction of markerless AR [1]. Lines represent correspondances between features detected by FAST [2].

hand, using artificial reference objects has its own merits.

Artificial reference objects (from now on referred to as "fiducial markers") are specifically designed to be used in computer vision applications for which pose estimation is required. Typically, these markers will have adequately discernable geometrical features to estimate the camera pose and an ID code embedded into them. Fiducial markers are commonly planar and rectangular, similar to ARTag [12] and ARToolkit [13]. These fiducial markers produce four corresponding points between the marker and the image, which essentially belong to the corners of the rectangular marker. There are many methods such as Abidi and Chandra's [14] and Quan and Lan's [15] to estimate the camera pose using four coplanar points. Characteristically, correspondence points for fiducial markers will be lower in number and more reliable compared to feature descriptors found on natural objects in the scene. Moreover, these correspondences may be located in subpixel accuracy. Consequently, using fiducial markers is the preferred method for applications in which higher accuracy and lower processing time are desired [16–18]. See Figure 1.2 for an AR conferencing application using the rectangular fiducial marker, ARToolkit.

Choosing the optimal method is an imperative part of engineering design process. In the case of pose estimation, the first choice to be made is whether to use fiducial markers or not. As expressed earlier, one method

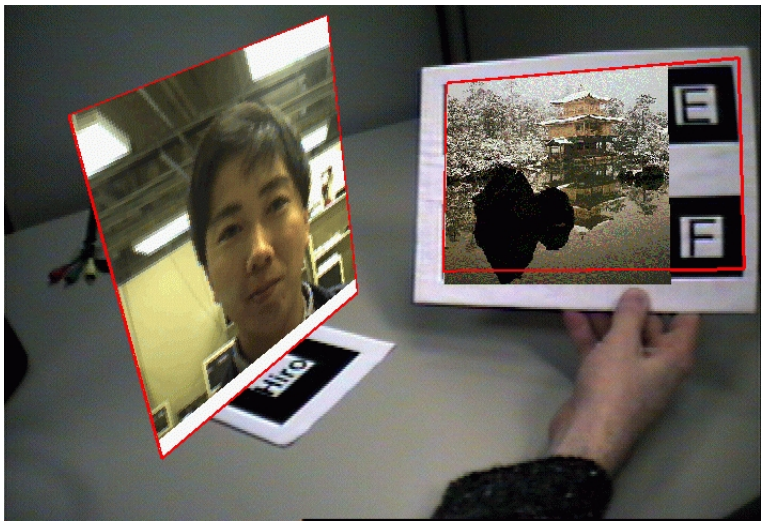


Figure 1.2: AR conferencing application using ARToolkit.

is not ultimately superior over the other. The developer should choose the method dependent on the requirements of the application. In this study, focus will be on fiducial marker implementations, thus markerless implementations will not be discussed any further.

1.2 Aim and Scope of the Study

Fiducial marker system design process can be divided into two parts, mainly marker design and marker detection algorithm implementation. The marker design should complement the implementation of the detection algorithm, and the detection algorithm should make good use of the marker design features. As a result of such duality, marker design and detection algorithm implementation are usually developed in tandem. There are examples to system-wide updates (i.e. updates to both the marker design and the detection algorithm), such as the updates from ARToolkit [13] to ARToolkitPlus [19] and ARToolkit DCT [20], and from Fourier tags [21] to Fourier Tag [22]. However, it is seen that the bulk of the research done on fiducial markers propose an original marker design and detection algorithm, while implicitly deriving features from former studies. Seeing merit in this approach, this study will also be on designing a whole fiducial marker system.

Currently, there are a number of fiducial marker systems, which will be looked into in detail in the next section. This may cause one to question the reasoning of proposing yet another fiducial marker system. The reasoning

comes from the characteristics of fiducial marker research discussed above. Marker systems are designed with the current requirements in mind, but they are generally not updated frequently. Regular updates require small and modular changes, yet the current perspective only allows large overhauls, which are only as feasible as designing a new marker system. This situation requires new marker systems to be developed as new requirements for fiducial marker systems emerge over time.

The aim of this study is to design a new fiducial marker and detection algorithm considering the current requirements and available methods. The exact requirements and methods will be discussed in detail in the following sections.

2. RELATED WORK

Since this study is about the design of a fiducial marker system, the literature review will be focused on existing fiducial marker systems. There are two features looked for in a marker system. The first feature requires the system to gather some kind of pose information using only one fiducial marker on the scene. We will see that while some markers can be used in numbers to achieve better results, all marker systems allow camera pose to be estimated using one fiducial marker. Secondly, the markers should be embeddable with information and the detection algorithms should be able to extract this information. This information will most generally contain the ID code of the marker to differentiate markers from each other. Due to these constraints, non-data bearing markers such as checkerboard pattern used for camera calibration and strictly data bearing markers such as the regular 1D barcode pattern will be omitted.

Majority of the published work about fiducial markers will be discussed below, and numerous others can be added to the list. The reason for the vast number of marker system implementations can be attributed to three factors:

1. Majority of the competitive marker systems are not open source and free to use commercially. Many people who intend to develop an application that uses fiducial markers develop a fiducial marker system first.

2. Target device hardware capabilities and platforms differ. Marker systems generally strive for real time capability, i.e. running in 40ms (although most implementations, like AR, have additional serious computing to be done for each frame, e.g. rendering graphics). While we may assume hardware capabilities change for the better over time, this is not always the case. Main market of AR and some other marker system applications is considered to be for smartphones. This leads to the need for lightweight marker systems implemented for portable devices as well.

3. The last and the most driving factor for designing more innovative marker systems is the perception of requirements. While there are

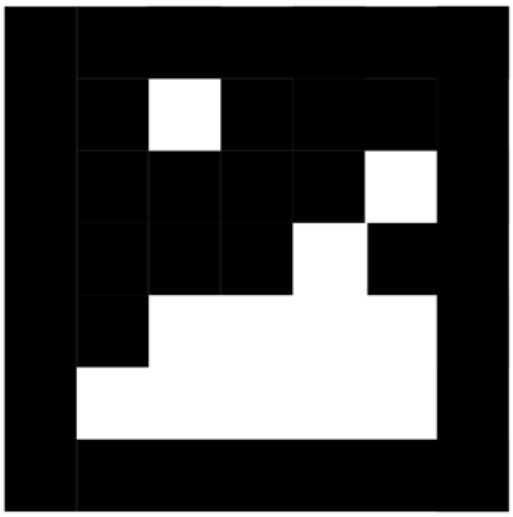


Figure 2.1: Marker design of Matrix.

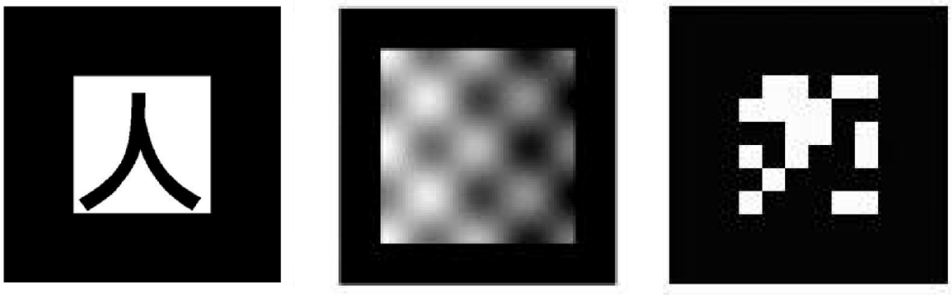
aspects of marker systems that is seen positive by the whole community (e.g. greater marker library size and lower error rate), these aspects are often at odds with each other. The decisions done for these tradeoffs for existing marker systems can be seen as suboptimal, and this leads to the design of a new marker system.

Marker systems have common characteristics that allow for categorizing, such as their border shapes and encoding styles. However, they will be investigated chronologically, as the progression of design decisions through time will give a better perspective on emerging trends in fiducial marker system design.

2.1 Matrix

Rekimoto proposes the first example of the square 2D matrix type fiducial marker (see Figure 2.1) [23]. The marker is encoded with 25 bits. 16 bits of these are payload, while the remaining 9 bits are used for CRC to check errors.

Detection starts with adaptive thresholding, arguing that it gives acceptable results since markers are bi-tonal and thus have high contrast. Detection of the candidate marker from the binarized image is not thoroughly explained, but rather referred to as "doing a heuristic check". Then, 2D to 2D homography is calculated to find the locations of the individual code



(a) ARToolkit. **(b)** ARToolkit DCT. **(c)** ARToolkitPlus.

Figure 2.2: Variations of ARToolkit.

positions on the quad. Since the image is already binarized, code is read directly and error check is done to see if the candidate marker is true or false.

There are no processing time rates given for Matrix. However, both the AR applications in the original paper and the realtime HCI implementation in Koike's work [24] suggest that Matrix achieved acceptable realtime performance with its time's hardware. While the detection algorithm is quite outdated, the marker design is still used without major changes in both published work and no-name systems.

2.2 ARToolkit

A year after Matrix's proposal, Kato and Billinghurst briefly mention ARToolkit in their study about a video conferencing system [13]. ARToolkit's marker also has a square border, yet the encoding method is entirely different. A monoscale pattern is embedded into the marker, and that pattern is associated with an ID code at the decoder-end. This approach adds implicit, yet unmeasurable error detection and correction functionality. It also allows the user to easily distinguish between different markers simply by looking at their patterns (See Figure 2.2).

ARToolkit's detection algorithm starts with thresholding the image. Then, lines are fitted to the contours of the connected regions. The regions that are enclosed by four lines are treated as marker candidates. This causes any kind of occlusion to cause a false negative detection. Since four corners of the marker is found, 2D to 2D homography matrix can be calculated. Using

the homography matrix, pattern in the candidate marker is inversely transformed to obtain a square pattern. This pattern is compared by template matching to the patterns in the database.

As the first fiducial marker library to be distributed under GNU GPL, ARToolkit filled a niche and is appreciated widely in the AR community. The fact that each marker is encoded with a pattern that is also meaningful to humans made it a useful tool for HCI applications.

Possibly due to its popularity, ARToolkit has seen several attempts of improvement. Owen et al. propose a non-arbitrary method of choosing the patterns for markers that use pattern matching (See Figure 2.2) [20], but it comes with the cost of losing the only uncontested advantage of pattern matching over 2D matrix encoding, better human interaction. Another attempt to improve ARToolkit is ARToolkitPlus [19]. It includes changing ARToolkit's sole characteristic feature, pattern coding, to 2D matrix coding (repeats the marker ID four times, then applies an XOR mask), non-global thresholding and vignetting compensation (See Figure 2.2). The major selling point of ARToolkitPlus is that it is implemented in order to be used with portable devices. With the removal of pattern matching, ARToolkitPlus becomes a lightweight marker system.

Fiala raises some questions about the detection algorithm of ARToolkit [17]. Firstly, he argues that neither global nor adaptive thresholding of the image is the ideal solution. Fiala's ARTag [12] and most of its successors use an edge detection based approach. The second downside Fiala mentions is that pattern matching works well only under controlled conditions. He defends Matrix-style 2D matrix encoding, and even improves it by adding FEC. However, Fiala does not compare ARToolkit and ARTag directly, but uses ARToolkitPlus for any comparisons. This proves his point regarding the use of thresholding compared to edge detection. Since comparing pattern matching to binary coding fairly would have been impossible, as the number and shapes of the patterns in the database would be decisive, marker ID decoding parts of ARTag and ARToolkit are not compared. Instead, Fiala claims predictable and quantifiable error rates are preferable for a marker system, hence binary coding of ARTag is superior to ARToolkit's pattern matching.

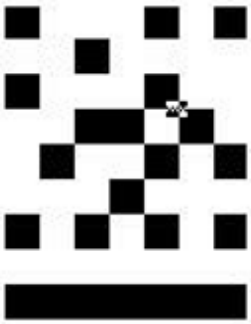


Figure 2.3: Marker design of CyberCode.

2.3 CyberCode

CyberCode is essentially Matrix with a different marker design [25]. Instead of the square border, there is a guide bar at the side of the marker to determine orientation and a square at each corner to estimate homography (See Figure 2.3).

2.4 TRIP

TRIP utilizes a concentric circular fiducial marker design (see Figure 2.4) [26]. All markers have a dot and a ring in the middle for detection, and two outer rings for binary encoding. Outer two rings are divided into 16 aligned slices. Only one of the 16 aligned slice pairs are both black, and this pair indicate the starting point of the encoding. To avoid using black pairs in the rest of the encoding area, encoding is done in ternary, wasting nearly quarter of the bandwidth. 2 slices are used for parity checking, and 4 slices are used to indicate the size of the marker. Every other marker design assumes the size of the marker is known a priori. Remaining 9 slices are used to encode $3^9 - 1$ different ID codes.

TRIP's detection starts with a pseudo-edge detection similar to its predecessors. Ellipses are detected using these edges, and concentric ellipses constitute candidate TRIPtags. The projective transformation of a circle is an ellipse. Using four corners of a square enables one to estimate the camera pose, while using a circle whose transformation is an ellipse results in two possible solutions. In this case, a corner of the black slice pair on the encoding rings are used to verify the correct solution.

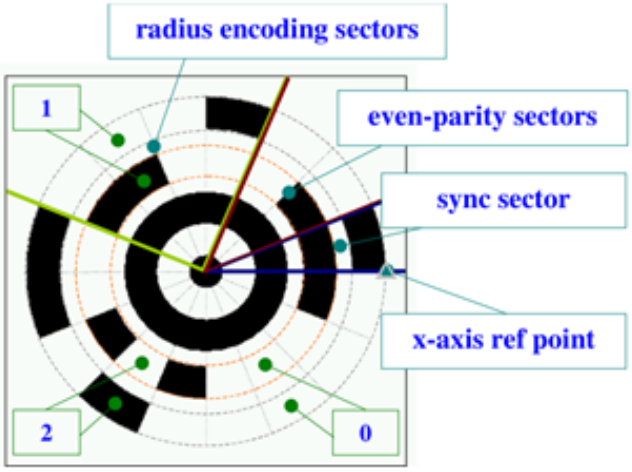


Figure 2.4: Marker design of TRIP.

2.5 Intersense Marker

Intersense Marker is another example to concentric fiducial markers (see Figure 2.5) [27]. The marker has innermost and outermost border rings, and these two rings are connected to create one blob. There are two dots between the rings to estimate decoding grid, and the rest of the space is used for data bits. There are 15 data bits. Error checking and correction are not implemented.

There are lots of heuristics involved in the implementation of the algorithm, as it was designed with a particular embedded platform in mind. Considering current hardware capabilities, much of the detection algorithm may be regarded as irrelevant.

2.6 ARTag

It was mentioned that fiducial marker system design derives mostly from existing marker systems. However, detecting the true requirements and making intelligent decisions as to which parts to derive is essential. ARTag is described by Fiala to contain successful elements of Datamatrix, a strictly data bearing 2D barcode system [28], and ARToolkit [13]. One may also describe it as Matrix [23] with added error correction. However it may be described, ARTag is an assemblage of appropriately selected components.

The marker design is similar to of Matrix's, with the difference that it

Figure 2.5: Marker design of Intersense.

contains 6x6 bits instead of 5x5 bits (see Figure 2.6). 10 bits of these belong to the payload, resulting in ~ 2^{10} IDs. Fiala also proposes to use markers with reverse colors to add another bit, doubling the ID capacity [29]. 16 bits are used for CRC, which is long enough to remove most false positives. The rest of the bits are used for FEC, which can correct up to 2 bits. While this will improve resistance to incorrect thresholding of codes, serious occlusions still causes markers not to be recognized. Fiala chooses the FEC polynomial that will have the largest Hamming distance between the codes and their four rotations. Also, the codes are put in a priority order to maximize the Hamming distances between the ones first to be used.

ARTag's quad detection is edge detection based. Fiala demonstrates benefits of edge detection over thresholding for marker detection in detail [17]. After edge detection, edge pixels are linked into line segments and quads are found as combinations of these line segments. This approach provides some occlusion resistance, yet its extent is not specified, and is probably dependent on heuristics used. For the case in which edges cannot be detected at full scale, ARTag also detects quads at smaller scales. The decoding thresholds of the embedded code is determined using the values of black and white borders of the marker to be decoded. The CRC calculation is done regularly, while FEC is done with a look up operation.

Fiala investigates jitter characteristics of ARTag and ARToolkit without doing stabilization using temporal data [29]. He explains that ARToolkit's stability under default parameters is caused by using hysteresis.

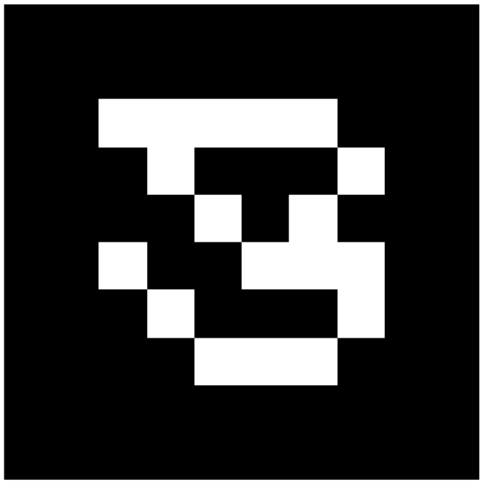


Figure 2.6: Marker design of ARTag.

However, even when using raw data, ARToolkit seems to have less jitter than ARTag under suboptimal conditions such as low light and de-focused camera. This is explained as the advantage of using thresholding based quad detection instead of edge based quad detection. Nevertheless, advantages of edge detection based quad detection is said to outweigh the advantages of thresholding based quad detection.

There is also a camera calibration implementation using ARTag [30]. Later research on this area proposes self-identifying patterns specifically designed for camera calibration [31]. In the latter study, only objection to using fiducial markers for calibration is lower accuracy and less dense markers. The first problem can be solved by using a fiducial marker that has at least one very accurately detected point, and the second problem can be solved by using more frames, possibly extracted from a video stream.

2.7 Cantag

Cantag is an open source toolkit for designing marker systems [32]. Rice et al. suggest that the application developer should decide on the design aspects of the marker system to be used. Accordingly, marker shape, thresholding type and payload size are customizable (see Figure 2.7). The implementations of system modules are not especially noteworthy.

The fact that an OpenGL-based test harness is being used to compare different Cantag design combinations is notable. However, noise and camera

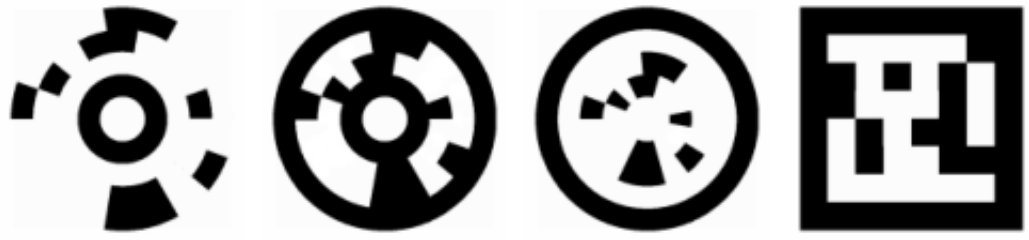


Figure 2.7: Marker design of Cantag.

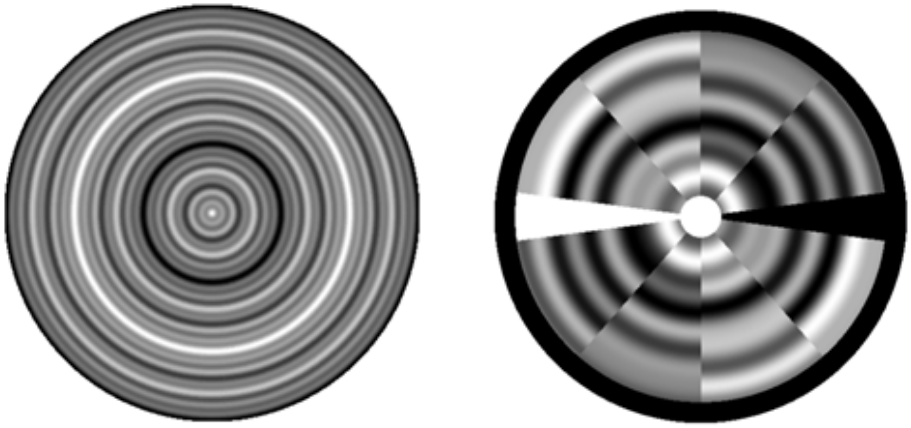
distortion are not simulated, thus these fabricated test images are ideal except the digitization of the image.

2.8 Fourier Tags

Majority of the fiducial markers are bi-tonal, i.e. black and white. Fiala describes the biggest contribution of ARToolkit as avoiding the grayscale non-linearity caused by the printer and the camera [12]. Fiala also advocates a completely digital coding scheme. On the other hand, the non-binary nature of the camera and noise allows for alternatives. Sattar et al. proposes such an approach for a specific application [21]. Detection results of regular fiducial markers are binary. Either the marker is detected and correctly decoded or it is not detected at all. Fourier Tags allow for graceful degradation of the code, meaning that less significant bits of the code degrade first, while more significant bits remain readable under worsened conditions. Fourier Tags detection algorithm does not take in to account that the projective transformation of a circle will be an ellipse, and the authors do not seem to be concerned about pose estimation. While this opposes our fiducial marker definition, Fourier Tags system is still mentioned because of its uncommon coding scheme. Later on, Xu and Dudek address the lack of pose estimation and error checking with an update [22]. See Figure 2.8 for both versions.

2.9 RUNE-Tag

The markers seen before all have a common design feature, there is a distinctive border outside, and the encoding is done inside. Various methods are used to detect these borders, and these detections constitute marker candidates. Inside of this marker is decoded to find the ID of the marker, and



(a) Fourier Tags [21].

(b) Fourier Tag [22].

Figure 2.8: Variations of Fourier Tags.

the border is used again to estimate the camera pose if the candidate is a real marker. The advantage of this approach is that it significantly reduces candidate markers to be decoded, and allows for using various heuristics to eliminate candidates even before decoding. On the other hand, if the marker border is occluded to the point that it is no longer distinctive, same heuristics used to eliminate false candidates also eliminate the occluded true candidate.

RUNE-Tag does not have a border in the traditional sense [33]. The coding dots are arranged in a circular shape, which is adequately unnatural enough not to spontaneously occur on the scene (see Figure 2.9). It also does not have an orientation indicating part similar to most circular marker systems employ, and this adds to its occlusion resistance. Due to its circular nature, it boasts a better pose estimation accuracy compared to ARTag and ARToolkit. As with all fiducial marker systems, there are tradeoffs. The coding dots seem to be prone to false negative detections at lower resolutions or greater distances, which is mentioned by the authors as well.

2.10 AprilTag

AprilTag is an open source marker system that shares the rectangular 2D barcode type design (see Figure 2.10) [34]. However, both the detection algorithm and the coding scheme improve on its predecessors. Firstly, a customized line detection algorithm is used to find the borders. These borders

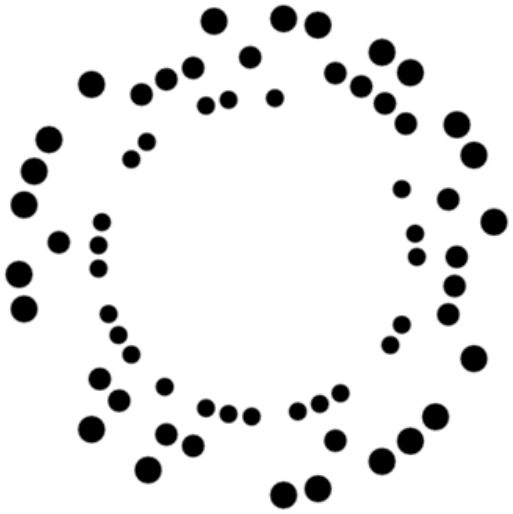


Figure 2.9: Marker design of RUNE-Tag.

are grouped in combinations of four if they fulfill a distance-based heuristic. Decoding the bits on the image is done using an intensity model. Rather than using CRC and FEC similar to ARTag, AprilTag uses lexicodes. Lexicodes are greedily generated error detection and correction codes. Rather than choosing between FEC and CRC polynomials for best Hamming distances, AprilTag's codes are generated to keep a minimum Hamming distance while detecting and correcting errors of a specified number of bits. These codes are pre-generated and distributed by Olson for 6×6, 5×5 and 4×4 marker sizes with specified minimum Hamming distances.

2.11 Random Dot Markers

Random Dot Markers use geometric feature based keypoint matching to interpret randomly scattered dots [35]. It can be compared to naming a constellation based on relative positions of a group of stars. The design features bring about the same advantages mentioned for RUNE-Tag. However, error

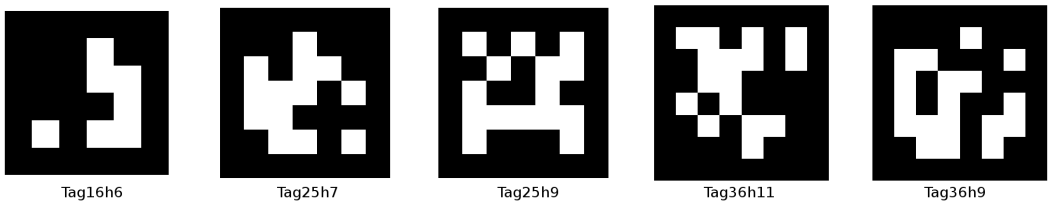


Figure 2.10: Marker design of AprilTag. TagXhY notation indicates the $n \times n$ = X bits on the marker and Y Hamming distance between bit sequences.

detecting and correcting capabilities are not quantified, and running times seem to be too high to be used in real time, especially when multiple markers are present.

Figure 2.11: Marker design of Random Dot Markers.

3. PERFORMANCE CRITERIA

The desired properties of a marker system is often defined vaguely and subjectively. Fiala makes an attempt to list all qualitative and quantitative aspects of planar printed fiducial markers [18]. This list has a lot of overlap in of itself, yet still has some minor aspects left out. To illustrate the factors considered when designing our marker system, it will be appropriate to present our own list of performance criteria. While this list is derived from Fiala's, it is adjusted to reduce overlaps, complete missing parts and have equally important criteria for the sake of a better commentary.

1. False positive rate
   Markers should not be detected when they are not present.
2. False negative rate
   Markers should be detected when they are present.
3. Marker library size
   There should be a large number of marker IDs to use.
4. Intermarker confusion rate
   Markers should not be misidentified as other markers.
5. Feature localization and pose estimation quality
   Marker features should be localized accurately.
6. Running time
   Detection algorithm should run as fast as possible.

3.1 False Positive Rate

False positive is the case in which the marker detection algorithm detects a marker which is not actually present (see Figure 3.1). For this to happen, the false marker must pass all steps of the marker detection algorithm successfully, i.e. it must be detected as a marker candidate and decoding algorithm must verify it as a marker. Consequently, any design decision that improves the elimination of false candidates in any of these steps will reduce false positive rate. Eliminating false candidates as early as possible is preferable, as

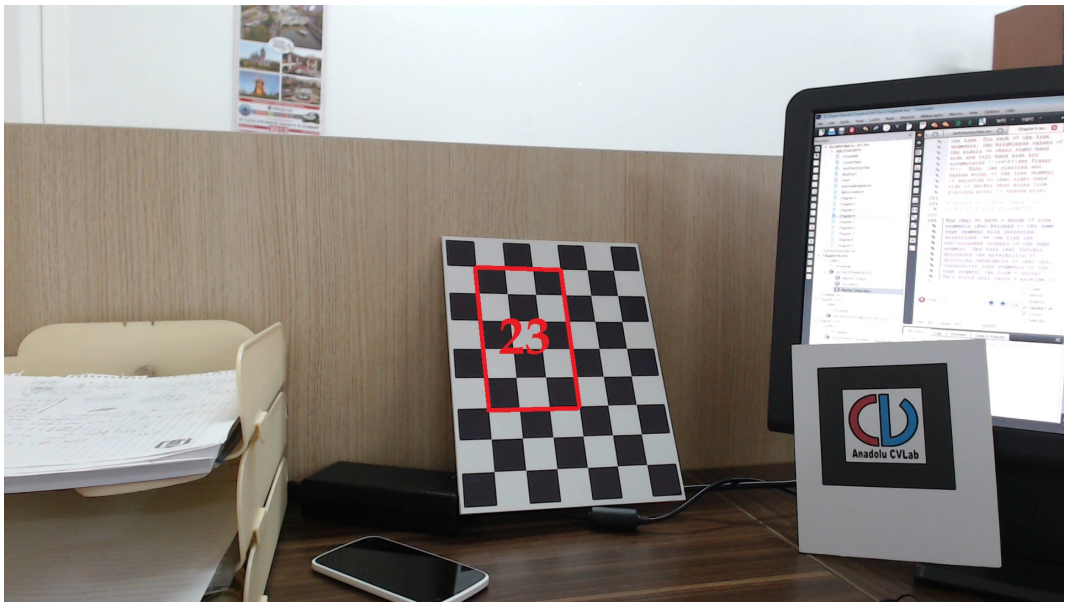


Figure 3.1: False positive example. Marker #23 is detected on the checkerboard pattern. This image is for illustration purpose only, it is not a result of the EDMarkers detection algorithm.

lesser number of candidates to be processed in later steps will result in a shorter running time.

The candidate detection algorithm can be made stricter to detect only the candidates that are very likely to be markers. For example, an occlusion-resistant candidate detection algorithm will come up with more candidate markers compared to its non-resistant counterpart. Having a lesser number of candidates will reduce false positive rate and improve running time, yet it will increase false negative rate. In this case, partially occluded markers that could have been decoded if they were detected as candidates will be missed. Hence, the marker candidate detection algorithm can be designed to accommodate this tradeoff.

As mentioned above, marker candidates are decoded to check if they are true. For digitally encoded markers, this is done by adding some redundancy while encoding the markers and checking if the sequence is still largely intact while decoding. Larger number of redundancy bits for error checking will result in better success in eliminating false candidates. However, doing so will require reducing the number of bits used for payload and error detection, which will reduce marker library size and increase false negative rate respectively.

Measures that can be taken to decrease false positive rate are summa-

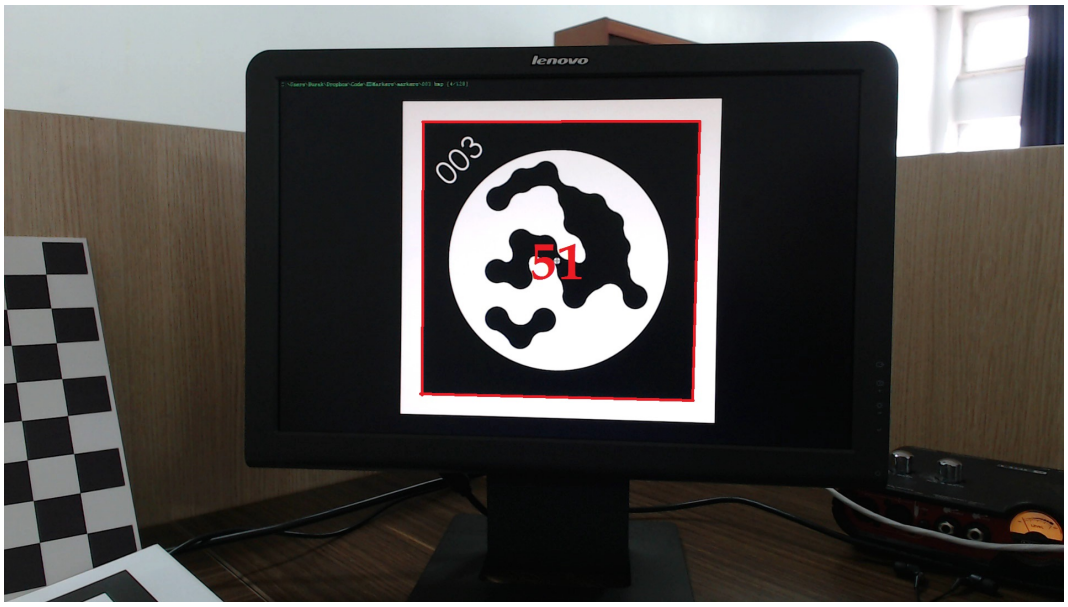


Figure 3.2: Intermarker confusion example. Marker #3 falsely identified as marker #51. This image is for illustration purpose only, it is not a result of the EDMarkers detection algorithm.

rized as follows:

- Have a stricter candidate detection algorithm that will result in a lesser number of candidates. This will improve running time, but increase false negative rate.
- Allocate more bits for error checking that will result in better elimination of false candidates. This will reduce marker library size and/or increase false negative rate.

3.2 Intermarker Confusion Rate

Each marker ID has a bit sequence assigned to them. When bit sequences of two different IDs are very similar to each other, they can be mistaken for each other when some of the bits are flipped (see Figure 3.2). To avoid this, bit sequences assigned to IDs should be as different from each other as possible. The similarity between the bit sequences can be conveniently measured in Hamming distances. There is no optimal method to maximize Hamming distances of a number of bit sequences in the binary space. Hence, maximizing the Hamming distances is an unsolved problem, and it is considered to be out of the scope of fiducial marker design. However, when using a specific

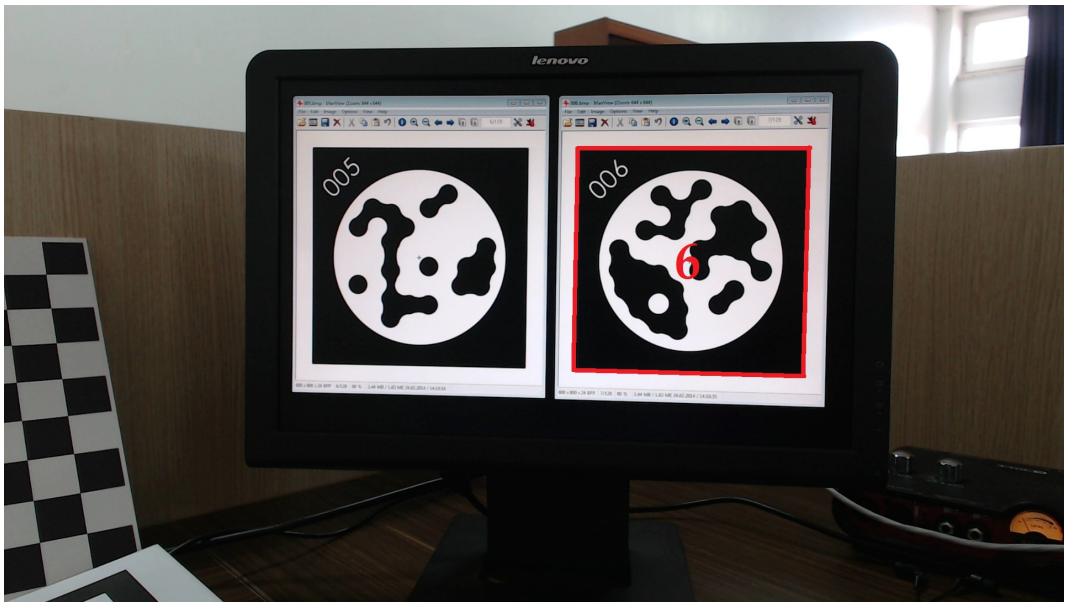


Figure 3.3: False negative example. Marker #5 is not detected even though it is present. This image is for illustration purpose only, it is not a result of the EDMarkers detection algorithm.

method for distributing the bit sequences for maximum Hamming distances, decreasing the number of bit sequences will result in higher Hamming distances.

Measures that can be taken to decrease intermarker confusion rate are summarized as follows:

- Decrease the number of IDs so that their associated bit sequences can be spread to have larger Hamming distances. This will reduce marker library size.

3.3 False Negative Rate

False negative is the case in which the marker detection algorithm does not detect a marker which is actually present (see Figure 3.3). For this to happen, the true marker must fail to pass any of the steps of the marker detection algorithm successfully. Design decisions that will cause the candidate detection algorithm to fail to produce a candidate from a real marker or decoding algorithm to fail to verify that the produced candidate is true will increase false negative rate.

Candidate detection can be made more resistant to false negatives either at the detection algorithm-end or marker design-end. If the rules and

heuristics used for candidate detection are made less strict, the number of falsely eliminated candidates will decrease. However, this will also decrease the elimination rate of false candidates, which will in turn increase false positive rate and running time.

Rather than changing the candidate detection method, the marker design can be changed for candidate detection to be easier. Candidate detection is done using distinctive geometric features of the marker, usually a square or a circular border. These features should be simple, distinguishable and large in scale. Simple features will not lose their characteristics even when they are small. On the other hand, intricate shapes will deteriorate due to digitization and noise as they become smaller in the image. The second requirement, distinguishability, can be interpreted roughly as the thickness of the feature. A thick quad will be easily detected while a thin quad will be harder to detect at smaller sizes. The large size requirement is generally ensured by placing the geometric feature as the outermost border of the marker. Outer border is the largest part of the marker, and accordingly, it is the last one to degenerate due to large distance or low resolution. Another additional benefit of robust geometric features is that they can be localized more accurately with identical feature detection methods.

As the geometric features become larger and thicker, area for encoding becomes smaller. If the number of bits encoded remains constant, bits are coded smaller on the marker. This will cause errors in reading the code, which will increase false negative rate and intermarker confusion rate. If the number of bits are decreased to fit in the smaller encoding area, there will be less number of bits for payload, error checking and error correction, which will in turn reduce marker library size, increase false positive rate and increase false negative rate respectively. It is notable that a modification done to decrease false negative rate may indirectly increase it.

After a candidate is extracted from a marker in the image, its embedded code is read. In the case that some bits on the code is flipped during the reading, error correction is employed to compensate. Assigning more number of redundant bits for error correction to reduce false negative rate will result in a lesser number of bits for payload and error checking, which will in turn reduce the marker library size and increase false positive rate.

Measures that can be taken to decrease false negative rate are summarized as follows:

- Have a less strict candidate detection algorithm that will result in more number of candidates. This will hinder performance and increase false positive rate.

- Adjust marker design to have robust candidate generating features. Doing so will also improve localization accuracy. Encode codes in a smaller area with the same bit sequence length. This will increase false negative rate and intermarker confusion rate because of the errors during reading the code.

- Adjust marker design to have robust candidate generating features. Use a shorter bit sequence in the remaining area. This will increase false positive rate, increase false negative rate and reduce marker library size.

- Use more redundant bits for error correction. Assuming the number of bits are constant, number of bits allocated for error detection and payload will decrease, which will in turn increase false positive rate and decrease marker library size.

3.4 Feature Localization and Pose Estimation Quality

During candidate marker detection, markers' manifestations on the image are localized (see Figure 3.4). While the quality of this localization largely depends on the detection algorithm, the shape to be recovered is also important. A square's projective transformation is a quad. Each line that represents a vertex of this quad is detected using its respective contributing pixels. This means that the detection of each vertex is only affected by one fourth of the quad pixels. On the other hand, when detecting the ellipse, which is the projective transformation of the circle, all pixels contribute to the fitting of the ellipse. Consequently, we would expect the ellipse shape to be localized more correctly compared to the quad shape, when they are similarly sized. However, square shaped borders are generally preferred due to ease of pose estimation implementation. Thicker borders allows for better localization, yet this will waste area that could have been used for encoding.

ARTag [12] and ARToolkit [13] practically have the same marker design considering candidate detection, yet their raw vertex jitter characteristics differ noticeably. Feature extraction methods used to form candidates matter

Figure 3.4: Feature localization error example. While marker is detected, its contour localization is unsuccessful. This image is for illustration purpose only, it is not a result of the EDMarkers detection algorithm.

for stability of the detection. While thresholding is worse than edge detection in suboptimal conditions, it may outperform it stability-wise under optimal conditions.

Four coplanar points which none three are collinear, e.g. the corners of the quad, can be used to estimate the camera pose completely. On the other hand, an ellipse gives two possible solutions, among which the right one can be chosen using an additional known point. Nearly all marker systems investigated in this study are capable of estimating the camera pose, as being able to do so was included in our definition of fiducial markers. All marker systems inherently have the ability to improve pose estimation accuracy by averaging the individually computed poses if there are multiple coplanar markers. Such cooperation does not require additional marker design consideration to implement. However, deliberate decisions can be made to widen the options for cooperation. For instance, it is mentioned that when using coplanar points to estimate pose, at least 4 points are needed. Markers that use this method have four points localized in similar accuracy. For these types of markers, using multiple markers to estimate pose is an afterthought, and thus, does not improve the outcome very effectively. On the other hand, if a point on the marker could be localized very accurately, this point may be used to estimate pose using at least four markers.

Temporal filtering is usually employed to reduce jittering, yet this will hinder the responsiveness of the system. If stability is more important than responsiveness for a particular application, temporal filtering can always be implemented independently from the marker system.

The projective transformation of a square is a quad, and the projective transformation of a circle is an ellipse. However, camera distortion causes additional changes to these shapes. Most marker detection algorithms assume that images are undistorted, and the candidates have the mentioned ideal shapes. Doing the contrary and allowing for further distorted shapes to be detected as markers will result in a loss of accuracy when undistorted images are used. Due to this impracticability, immunity to photometric calibration is often not accommodated for, and is usually a quality of the marker system by chance.

Measures that can be taken to increase feature localization and pose estimation accuracy are summarized as follows:

- Prefer a unitary geometric feature such as a circle for estimating pose, rather than a sectional shape (a shape composed of independent vertices) such as a quad. Allocate more of the marker area to these features for them to be more robust.
- Use the feature extraction method that localizes features best.
- Design a usage scheme with multiple markers that will improve pose estimation when used cooperatively.
- Employ temporal filtering to reduce jitter. This is an application-specific measure that can be taken when worse responsiveness is not an issue.
- Undistort the image as a primary separate step and have a stricter candidate detection algorithm that does not accommodate for camera distortion. This will reduce the number of candidates, decrease false positive rate, but the camera needs to be calibrated beforehand.

3.5 Marker Library Size

It is very difficult for humans to interpret digitally encoded IDs and associate them with individual markers, especially when there are more than a few

markers. On the other hand, digital decoding's only alternative, template matching has unpredictable and suboptimal false detection and intermarker confusion rates.

The payload of the bit sequence coded on the marker is the marker ID. The length of the payload determines the marker library size. Therefore, to have a larger marker library size, one should increase the number of bits dedicated to payload. As one can deduct, marker library size is at odds with low false positive rate and low false negative rate.

Markers in the literature tend to be designed to work individually. They receive some benefits to pose estimation when they are used in cooperation, but no work has been done to implement a marker system that assist each other in the encoding phase when used together. Applying a priori knowledge about markers' relative position to increase marker library size or reduce false detection rates is certainly possible. For example, if the library size is N for a single marker, and there are M combined markers, the resulting combination may represent N^M different ID codes. As discussed, these examples of cooperation do not require additional marker design consideration to implement. However, deliberate decisions can be made to widen the options for cooperation.

Measures that can be taken to increase marker library size are summarized as follows:

- Employ digital coding. Doing so provides more predictable performance metrics, yet using patterns for encoding present a more user-friendly interaction experience.
- Increase the payload of the bit sequence. Doing so will reduce the redundant bits used for error checking and correction, which will in turn increase false positive and false negative rates.
- Design a usage scheme with multiple markers that will increase marker library size when used cooperatively.

3.6 Running Time

The easiest characteristic of a detection algorithm to quantify is its speed performance under recommended conditions. This is both related to the implementation of the algorithm, and the design decisions done considering the

robustness of the algorithm. Consequently, it is not reasonable to compare two candidate detection algorithms' speeds if one is more resistant to occlusions than the other. A less robust marker detection algorithm will typically run faster than a more robust one.

We have mentioned the usage of multiple markers cooperatively. For multiple markers to be used in cooperation, candidate detection algorithm should not suffer speed or accuracy-wise when multiple markers are on the scene. Again, some robust algorithm implementations suffer greatly speed-wise when multiple markers are on the scene.

Encoding of IDs and creation of marker libraries are done offline and not considered to be a performance factor. However, decoding of the candidate markers are done in real-time, so this has to be achieved efficiently. Since bit sequences are fairly short, digital encoding is done quickly, especially with the usage of techniques like look-up tables. On the other hand, template matching methods such as ARToolkit's [13] may encumber the system related to its library size. Whichever method is used, a larger library size will result in worse performance, as more processing and memory resources will be used.

Measures that can be taken improve speed performance are summarized as follows:

- Implement a less robust marker candidate detection algorithm that will only detect candidates that are obviously visible (e.g. no missing vertices or corners). This will result in higher false negative rate.
- Employ digital coding and/or use a small library size.

3.7 Tradeoffs

There are two main resources that are shared to improve performance factors. The first is the marker area. The marker area is used for two purposes, the geometric feature used in candidate detection and encoding. Larger and thicker geometric features result in better localization and provide robustness for candidate detection against noise or large distances, which will improve false negative rate. On the other hand, a marker with a larger encoding area with the same number of bits will still be readable from afar. Larger encoding area can also be used for a larger number of bits, which can be used for error checking, error correction or larger marker library size. The second shared

resource is the processing power. The manner of allocating this resource to either candidate detection, verification or decoding again affects performance metrics.

Using limited resources is not the only reason that creates the need to make decisions for which performance metric to favor. There are also the cases in which one cannot be certain if the object in the image is a true marker. For example, if a candidate marker's code is read to be 3 bits different than a marker bit sequence, it is not possible to be certain if it is a true marker. If a lower false negative rate is desired, it should be accepted as a true marker. However, this will also cause other possibly false candidates to be verified as true markers, which will result in a higher false positive rate.

Tradeoffs between the performance metrics are roughly illustrated in Table 3.1. It can be seen that improving marker library size hurts all other metrics. Therefore, it is recommended to use the smallest library size possible for a given application. False positive rate, false negative rate and marker library size form a strong triple constraint, meaning that one cannot be improved without hurting any of the other two. This is simply due to the fact that they share another implicit resource, number of bits of the code. There are no other obvious relationships between the metrics. One should simply adjust their design based on their desired performance from their marker.

	False positive rate	False negative rate	Marker library size	Intermarker confusion rate	Feature localization and pose estimation quality	Running time
False positive rate		-	-		-	+
False negative rate	-		-	-	+	-
Marker library size	-	-		-	-	-
Intermarker confusion rate		-	-		-	
Feature localization and pose estimation quality	-	+	-	-		
Running time	+	-	-			

Table 3.1: Performance criteria tradeoffs.

4. EDMARKERS

4.1 Marker Design

Markers generally consist of two sections (see Figure 4.1). The first section is the geometric feature that is used for candidate detection. The second section is the encoding area in which the marker ID is encoded. The distribution of marker area to these two sections affects the performance of the marker system. Moreover, both the candidate generating feature and encoding may be implemented differently to improve marker system performance under desired conditions.

The most common geometric features that are used are square or circle shaped borders. Matrix [23], ARToolkit [13], ARTag [12] and numerous other marker implementations use a thick square border, while TRIP[26], Intersense [27] and other similar implementations use circular outer border as the candidate generating geometric feature.

The advantages of using a square border are as follows:

- Candidate detection can be done by line detection followed by combining the lines to construct candidates.
- Four corners of the quad on the image can be used for pose estimation. An additional point is needed for pose estimation from ellipse to validate the solution.

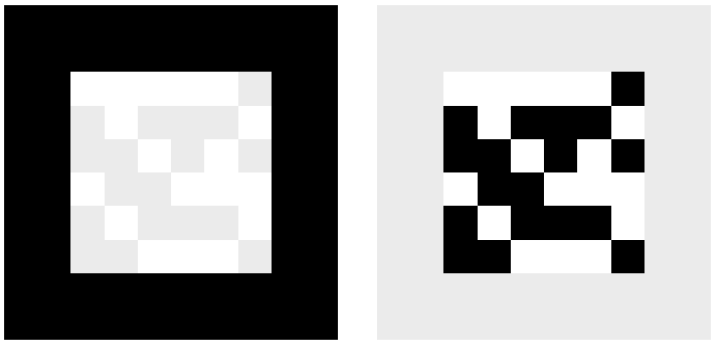


Figure 4.1: Two parts of a marker, geometric feature used for candidate detection and encoding.

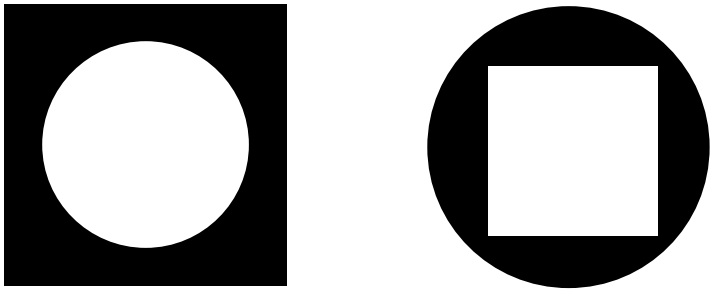


Figure 4.2: Markers borders that include a square and a circle.

- Digital encoding is done as a square grid. While its efficiency is debatable, it is simple to implement and easy to scale.
- It is straightforward to tile multiple markers for multi-marker applications (e.g. camera calibration). The markers are tiled as a square grid.

The advantage of using a circular border is as follows:

- All pixels of the ellipse on the image are used in ellipse detection. This will result in better localization and pose estimation. On the other hand, each vertex of the quad is only voted on with quarter of the pixels on the quad. Similarly, each corner of the quad is only voted on with half of the pixels on the quad.

All previous marker systems use only one of these options. They either have a border that is square inside and outside, or they have a border that is circle inside and outside. However, selecting either strictly is not necessary. The border may be square outside and a circle inside, or vice versa (see Figure 4.2). Normally, we would want the feature used for pose estimation to be outside, so that it is larger and more accurately localizable.

We have mentioned that using the circle border for pose estimation results in two solutions. An additional point should be known to validate one of the solutions. A robust marker design does not depend on localization of such critical points, meaning that the fact that any occlusion to that point will result in a failure in pose estimation makes this option undesirable. Instead, EDMarkers uses the square border for single marker pose estimation, similar to other square bordered markers. The circle border on the marker is

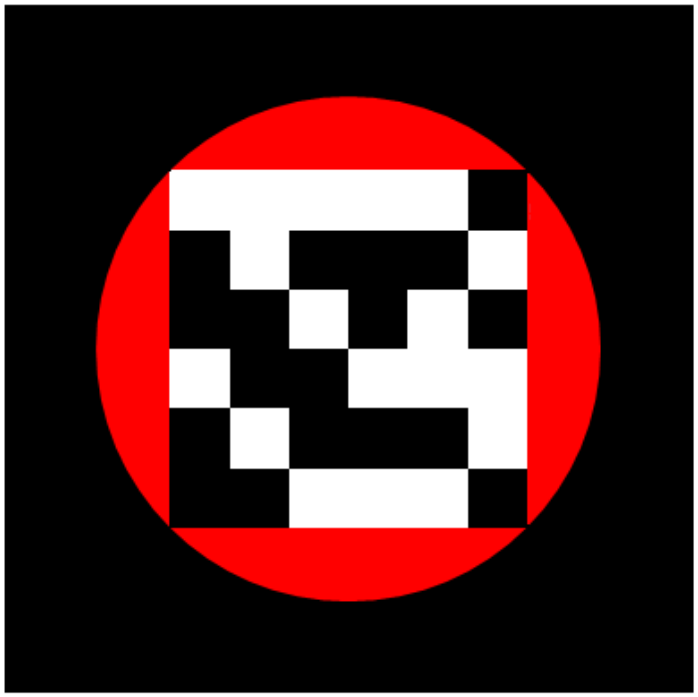


Figure 4.3: Red area is wasted when square grid tiling is used.

used for localizing a single point, namely its center, in precise accuracy. As a result, we have four quad corners that are reasonably well localized, and an ellipse center that is comparably more accurately localized. While localizing this point is less significant when estimating pose with a single marker, these ellipse centers can be used to estimate pose when multiple markers are used. Since these points are localized better, the pose estimation derived from this information will be superior to one that uses the quad points of the markers or individual pose estimations of the markers.

The preference of having either the square or the circle outside is done based on the above discussion. Since pose estimation from a single marker is more common, square should be outside. This general rectangular shape also makes printing and tiling the marker easier. When we have the circler border inside, we can no longer use grid encoding without causing considerable waste of marker area (see Figure 4.3). We have mentioned that non-digital coding, i.e. template matching, is an application specific method. Since EDMarkers is designed for general use, it uses binary coding.

Marker systems in the literature focus on the coding schemes, yet graphical implementation on the marker is often overlooked. The most common

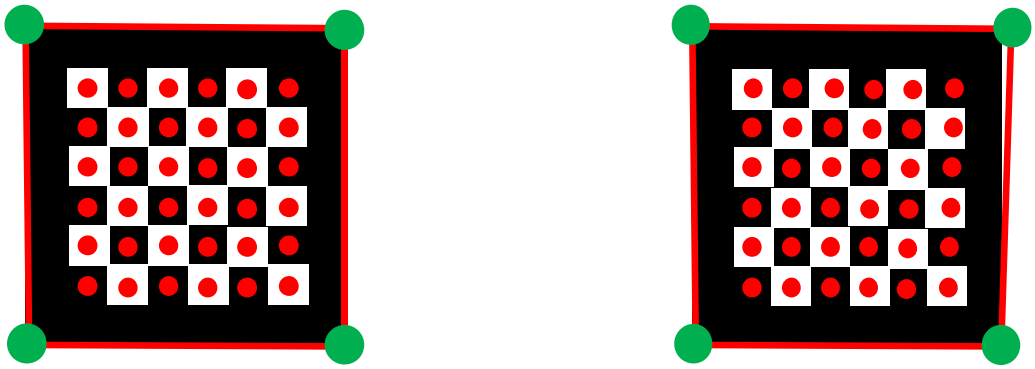


Figure 4.4: Effect of localization error to code reading. The marker on the right-hand side is localized incorrectly, hence the corresponding code locations are localized incorrectly as well.

implementation is grid coding with squares. As implemented in AprilTag [34], this grid can easily be scaled as 4×4, 5×5 etc. We can see that the aspect ratios of codes remain same with scaling. The fact there are no markers with grid coding that uses rectangular codes show that there is some awareness about the importance of the shape of the codes. However, coding with squares is not adequately justified in any of the marker systems that employ it. Ideally, marker detection algorithm will localize the center of the code square and read the code. Due to errors in marker candidate localization, the center of the code square may not be found precisely. In this case, detection algorithm will read a few pixels off the target (see Figure 4.4). Each code bit is represented as large as possible to accommodate for this error. However, there is no reason for one to assume that this error will be biased in any direction. If we want to accommodate for errors in all directions equally, we have to enlarge the code as a circle. An example of such an implementation is SCR (see Figure 4.5) [17]. Intersense [27] and TRIP [26] code bits with curved rectangles, which are even further from ideal. While square codes are closer to ideal, they cannot be considered ideal.

Let us calculate the best case scenario for area usage efficiency with square grid coding. Assume that the coding area is $2nr \times 2nr$, and each code circle has the radius r. Code circles are tiled on a square grid. There are a total of n^2 code circles in the encoding area. Encoding area can be calculated as:

$$2nr \times 2nr = 4n^2r^2 \tag{4.1}$$

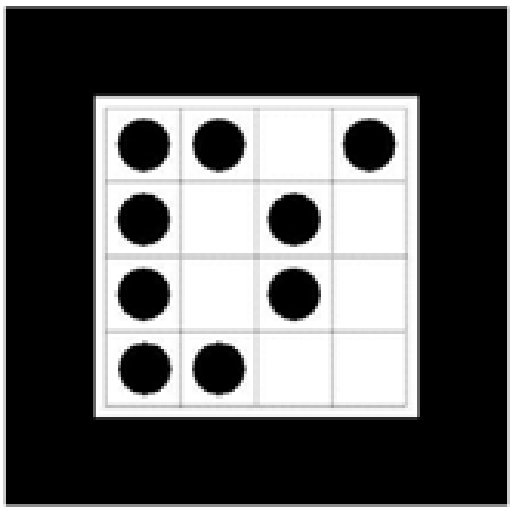


Figure 4.5: Marker design of Siemens SCR.

Each code circle has the area πr^2. Total area of code circles are:

$$n \times n \times \pi r^2 = \pi n^2 r^2 \tag{4.2}$$

The efficiency of grid tiling can be calculated as:

$$\frac{\pi n^2 r^2}{4n^2 r^2} \times 100\% = \frac{\pi}{4} \times 100\% \approx 78.54\% \tag{4.3}$$

Although square tiling seems to use all of the space in the encoding area, if we acknowledge that effective coding shape is a circle, it only uses 78.54%. Before trying to improve this tiling, let us explore other constraints for tiling that may be hidden at first glance. We have briefly mentioned one of the constraints earlier, which is scalability. Square tiling can be easily scaled by keeping the number of code circles at rows and columns the same (i.e. in $n \times n$ formation). Tiling efficiency for square grid tiling is independent from scale. More complex tiling schemes may be more difficult to scale up or down and they may lose efficiency. While the ease of scalability seems to be a good argument for using square grid, scaled versions of markers are rarely used in applications. One may choose to decrease the number of code circles for them to be visible from further distances. However, instead of decreasing the number of bits, some of the bits allocated to payload may be transferred to error correction, which will also result in better decoding from further distances.

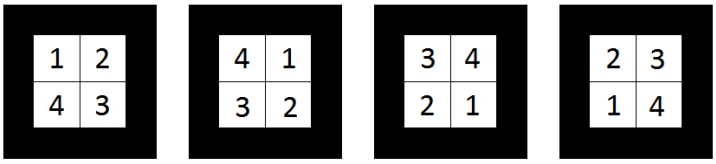


Figure 4.6: Reading a marker four-ways. Bit sequences associated with marker IDs are determined such that only one of these readings can be associated with an existing bit sequence.

The square grid has an inherent symmetry that is actually critical for decoding. While some markers such as CyberCode [25] has a rotation indicating feature, relying on such features for rotation detection is generally dismissed on the basis that markers should be homogeneously significant. The fact that the total occlusion of the rotation indicating feature will render the marker not decodable makes the usage of such features unacceptable. For purely circular markers, the only way of not using any rotation indicating features is encoding radially (i.e. encoding in concentric full circles), similar to the first version of Fourier Tags. However, the solution is easier for square markers. Since we detect the four corners of the quad, only problem is to find which corner of the quad corresponds to which corner of the original marker. There is only one solution to this problem in the literature, which is reading the code four times assuming that all four rotations are plausible (see Figure 4.6). It is guaranteed by digital encoding methods that only one of these codes corresponds to a real marker identifying bit sequence, hence the respective rotation is the correct one.

Guaranteeing that only one of four codes read from different rotations implicitly requires for the marker to be read predictably from all four directions. Normally, one would write the code reading algorithm such that center of the code circles are read when there is no rotation. However, if the marker lacks a certain kind of symmetry, the algorithm will not be able to read the centers of the code circles for all four rotations. 4-fold rotational symmetry is required for markers with square borders to be read reliably in all four directions.

In Figure 4.7, there is a marker design that does not satisfy the aforementioned condition. Although all four quadrants look similar, it does not possess rotational symmetry. When the marker is not rotated, codes are read from the correct locations. However, when the marker is rotated, the

Figure 4.7: Code tiling scheme that does not have the required rotational symmetry.

readings are taken from points between code circles, which will yield unpredictable results. We can argue that the rotated reading is not likely to produce a bit sequence that will pass error checking. However, this risk is difficult to quantify reliably, while the objective of digital encoding is to be able to precisely quantify failure frequency. Moreover, we do not need to take this unpredictable risk, when we can simply design a marker that has 4-fold rotational symmetry.

We have mentioned that square grid tiling is not as efficient as it seems to be. Instead of using square grid tiling, code circles can be packed into the square with 4-fold symmetry as efficiently as possible. While it is possible to pack equal circles in a square more densely than square grid tiling can, these packing schemes do not possess 4-fold symmetry [36]. However, this is irrelevant to us, as we have decided to use a border that is square outside and circle inside. Therefore, our problem is packing equal circles in a circle. Number of bits desired to be encoded is between 20 and 50. The actual number of bits will be decided based on packing efficiency.

During the early steps of the design, the bit sequences were decided to be of 19 bits length, as 19 unit circles can be packed in a circle with 80.32% density. This is theoretically proven to be the densest possible packing scheme for 19 circles [37]. Figure 4.8 illustrates this coding scheme. This coding scheme proved to be unsatisfactory in two ways. As one can easily notice, it has only 2-fold rotational symmetry. This causes the problem discussed above. Furthermore, the number of bits is simply too low for having a large marker library size with good error checking and correcting capabilities.

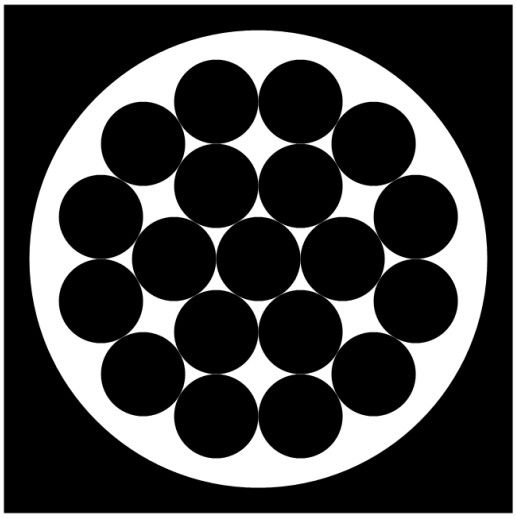


Figure 4.8: 2-fold symmetric 19 bit coding scheme. There is a thin blank ring between the circular border and code circles to obtain an unbroken edge segment from the circular border.

The problem with using recreational mathematics studies for finding a suitable packing scheme is that we have an arbitrary constraint, namely the need for 4-fold symmetry, and the literature is generally interested in finding the theoretical best packing with no constraints [38]. Thus, we have to produce our own packing composition considering our constraints. Since we are looking for 4-fold symmetry, we do not need to pack circles in a circle, instead we can pack circles in a quadrant of a circle. A custom and highly randomized simulated annealing algorithm was used to find a sufficiently optimal packing of the circles. The algorithm is basically as follows:

Algorithm 1: Simulated annealing algorithm

Data: n is the number of circles to be packed.
k is the number of randomized movement iterations before the algorithm gives up.

Randomly place n small circles in a circle quadrant.
while *Circles overlap* **do**
  Randomize the locations of the circles.
end
$i \leftarrow 0$
while $i < k$ **do**
  while *Circles do not overlap* **do**
    Enlarge all circles.
  end
  while *Circles overlap* **do**
    Move circles in a random direction while protecting 4-fold symmetry.
    if *Circles overlap* **then**
      Undo the last movement.
    end
    $i \leftarrow i + 1$
  end
  $i \leftarrow 0$
end
Undo the last enlargement.

Among the results, 48 code circles were found to be packed with 74.96% density. While this is a 3.58% decrease in density compared to square grid tiling, it is excused for the fact that there is now a circle border that can be used to localize the center of the marker very accurately. The new packing scheme is seen in Figure 4.9. This packing scheme can be scaled by using the results of the simulated annealing algorithm for different number of circles. However, we did not feel the need to implement this for the reasons discussed in the marker detection algorithm section.

When there are neighboring code squares, there is no blank space left between them. While the positive effect of this is hard to quantify, we can assume that this provides robustness against noise and blooming. The complex nature of our packing scheme requires a universal method for filling such gaps. After encoding the marker with simple code circles, additional smaller circles are placed between close code circles to guide and accelerate the next step. Then, the circles are morphologically dilated and eroded repeatedly to achieve a smoother look. The code is then re-encoded to make

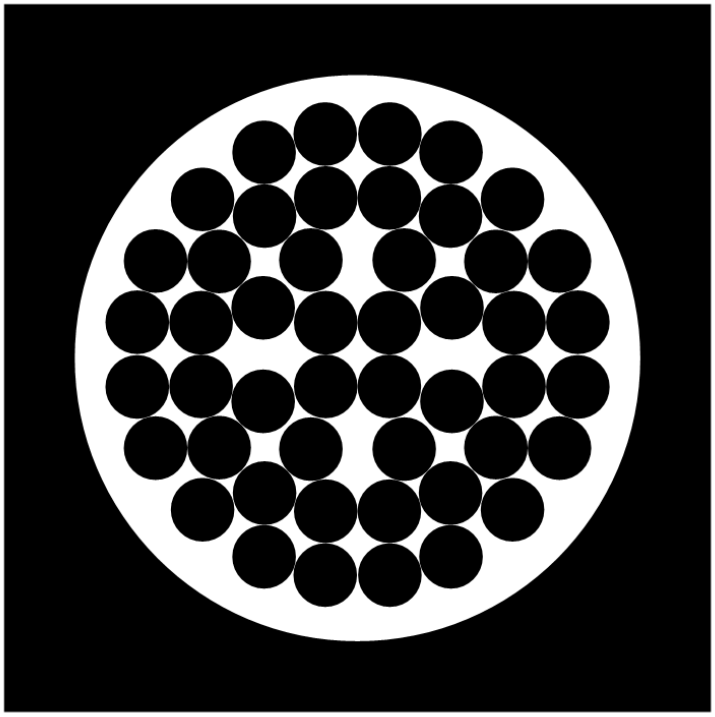


Figure 4.9: 4-fold symmetric 48 bit coding scheme.

sure the morphological operations did not affect the actual code circles (see Figure 4.10). Since marker generation is done one-time and markers are pre-generated markers and are planned to be distributed to the users, the excessive running time of morphological operations are overlooked. Moreover, morphological operations can be entirely skipped as they are not critical for this step.

4.2 Encoding

The graphical representation of the bit sequence on the marker is discussed in the previous section. The only relevant part for encoding is that the bit sequence must be 48 bits long and when read in four different rotations, only one of them should represent a real marker code.

There are several features that could be implemented while designing the encoding of the marker. If all of these features are omitted and payload is directly encoded to the marker, such as in Intersense marker, the marker library will be of its maximum size. One can argue that implementing an extensive encoding scheme is simply out of the scope of marker system design.

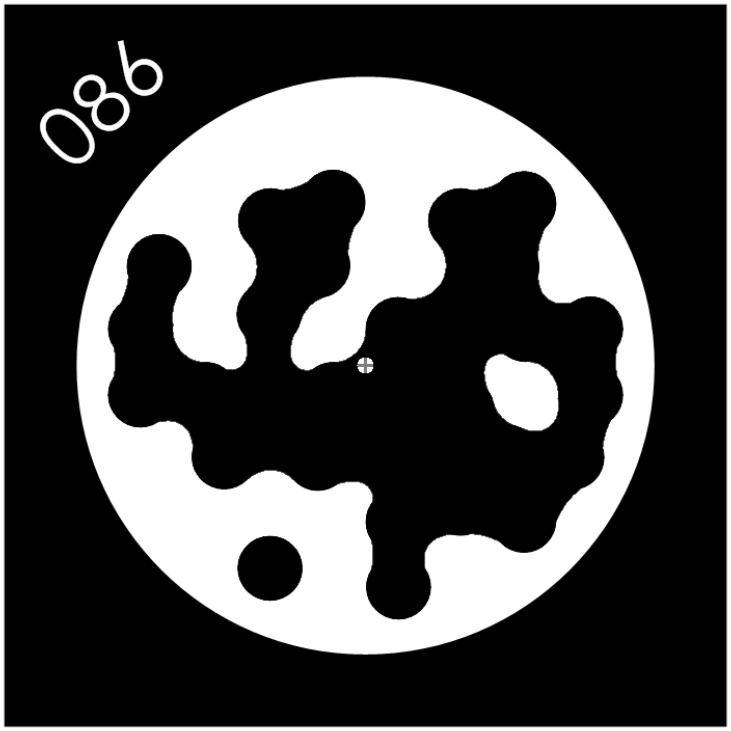


Figure 4.10: The marker associated with ID number 86. Code circles are morphologically merged with each other to reduce blooming effect. The middle of the marker has a white spot to check if it is being localized correctly. ID number of the marker is printed to the top left corner for ease of use.

Any bit sequence can be written to or read from the marker, and as long as the marker system designer presents the method for doing so, it is of user's responsibility to implement the encoding and decoding scheme. However, the encoding system cannot be so simply abstracted, as we present an arbitrary constraint for decoding, based on our marker design, namely the need for the codes to not be meaningful when read in false rotations. Therefore, it is appropriate to implement a complete encoding scheme.

The encoding scheme should have two capabilities, namely error checking and error correction. Error checking is implemented with CRC for Matrix [23], and many other marker systems to follow. Error correction is encountered rarely, as a large number of redundancy bits are required for a meaningful amount of errors to be corrected. ARTag [12] uses a method that uses a generator polynomial, yet the details are scarce. On the other hand, AprilTag [34] uses lexicodes as the bit sequences that represent marker IDs. Additionally, all bit sequences and their rotations should have maximum Hamming distance from each other to minimize intermarker confusion rate. While different generator polynomials are tried for ARTag to maximize Hamming distance, optimally generated lexicodes will have larger Hamming distances.

When generating lexicodes, we have two variables. The first is the number of bits in the bit sequence, and the second is the desired minimum Hamming distance. In our case, not only the bit sequences, but also their four rotations should be a minimum Hamming distance away from each other. In addition, we would want the bit sequences to be a minimum Hamming distance away from all 1's and all 0's for obvious reasons. Normally, lexicode generation algorithm iteratively places vectors in lexicographical order. The result is a list of bit sequences that are at least a minimum Hamming distance away from each other. While not using the same method, we can achieve a similar result using a different method. This time, instead of starting the desired minimum Hamming distance, the number lexicodes to be generated is chosen. The bit sequences are randomized and scattered in the binary space at the start. Then, very similar to the previously mentioned simulated annealing algorithm, the bit sequences are altered randomly. If the result improves the minimum Hamming distance, it is kept. While this algorithm converges to a solution when the number of bit sequences is chosen appropriately, it does not find the optimal solution. The idea was that since this task is not time-critical, we could run this algorithm for a very long time

to find a solution that is very close to optimal. However, no improvements after very long running times indicate that the solution is driven to a local maxima. The algorithm is also restarted to achieve the ideal randomized starting conditions for the algorithm to solve the problem to no avail. One curious phenomena was that also keeping the changes that improves overall Hamming distance (that is, the sum of all Hamming distances between bit sequences and their rotations) converged the solution to an even worse local maxima.

The algorithm explained above distributes 128 48-bit long sequences with at least 13 Hamming distance. Note that minimum Hamming distance applies to all bit sequences, their rotations and all 0 and all 1 codes. Each of these sequences are associated with a marker ID. Then, applying error correction is rather simple. When a bit sequence is read from a marker candidate, it is compared to all of the bit sequences in the list. If its Hamming distance to one of them is smaller than the number of errors to be corrected, that candidate is associated with the respective marker ID. The running time of the implementation may be improved by placing the bit sequences on a binary tree, yet this would only be required for much larger library sizes.

4.3 Marker Detection Algorithm

Marker detection is done in two steps. In the first step, candidate generating geometric features on the marker, which is generally its border, is localized on the image. This marker candidate is then decoded to verify if it is a true marker. If the marker detection algorithm fails to produce a marker candidate from a marker on the image, that marker will not be able to be detected. Therefore, candidate detection is a crucial step in marker detection.

Assuming that there is no information about the size of the marker, it can be said that any convex quad on the image may be the projective transform of a true marker. The vertices will be curves rather than lines if the radial distortion of the camera is not compensated for. However, we have mentioned that rather than using an unconstrained candidate detection algorithm, it is better to use a strict candidate detection algorithm with undistorted images for localization accuracy. Hence, the input images are assumed to be undistorted. The four vertices of the quad are expected to appear as straight line segments.

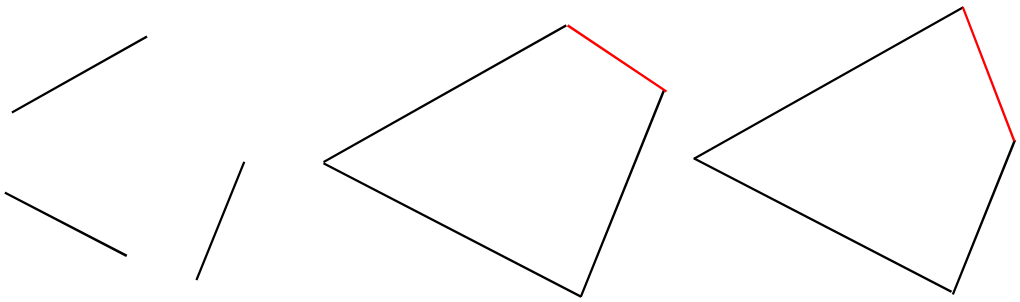


Figure 4.11: When there are less than 4 line segments, shape of the quad is ambigious. The shape of the resulting quad can only be determined using the fourth vertex shown in red.

Candidate detection algorithms in the literature either starts with thresholding or edge detection. It is shown that edge detection is more reliable compared to thresholding and blob detection under suboptimal lighting conditions [17]. EDMarkers uses EDPF [39] as the edge detection algorithm, which is basically Edge Drawing [40] run with extreme parameters. After EDPF, EDLines [41] is used to detect the line segments on the edge segments. Four lines that are extracted from each vertex respectively are needed to characterize a quad. It is theoretically impossible to identify a quad with lesser number of line segments (see Figure 4.11). Assuming there are N line segments found on the image, 4-combinations of these line segments will identify all possible quads on the image. While this is the most occlusion resistant method that uses line segments, it also returns N^4 quads. As the resolution and line generating objects on the scene increase, number of candidates to be verified increases greatly. This is also reported by Olson for AprilTag [34]. The first implementation of EDMarkers used this same method. While this method is resistant to occlusions, using it without any additional heuristics increase running time beyond practicality. Heuristics such as grouping line segments based on the distances between improve running time a bit, while decreasing occlusion resistance. Nevertheless, this method loses feasibility even with the heuristics when there are 20+ markers on the scene.

Quad detection is improved by using Edge Drawing's and EDLines' characteristics as leverage. Edge Drawing finds the edges on the image as chain segments. Then, EDLines walks through these segments to find the line segments. Consequently, each line segment belongs to an edge segment. While we are traversing along the edges of the marker, as long as it is not occluded, the lines that represent the vertices of the quad will belong to the

Figure 4.12: A marker whose all corners are occluded. Since the occlusions will break the edge segments, extracted line segments will belong to different edge segments.

same edge segment. By deliberately sacrificing some occlusion resistance, we can dramatically reduce the possible combinations of line segments, which will improve running time.

Using all 4-combinations of the line segments will yield us all possible quads on the image. Even the quads whose vertices were detected in different edge segments will be detected. This would be the case in which all four corners of the marker is occluded (see Figure 4.12). It can be argued that this will rarely be the case in practice. Generally, there is a single object that occludes the marker. It can occlude a vertex or two consecutive vertices along with the corner. For that object to occlude non-consecutive vertices, it must pass along the encoding area. As error correction works up to a few bits, such occlusion of the encoding area will result in a false negative detection anyway. This concludes that such resistance to occlusion will hardly improve detection performance, yet it will increase the running time because of the large number of detected candidate markers.

We have mentioned that at least one line segment from each vertex is needed to identify a marker. Our constraint is that each of these line segments must come from the same edge segment consecutively. While this seems strict at first, it actually provides all the occlusion resistance that is actually useful. Assume that two vertices and the corner they form are

Figure 4.13: A marker whose one corner is occluded. Since unoccluded vertices and corners are on a single edge segment, this marker will be detected.

occluded. The remaining three corners will have four line segments, each belonging to the same edge segment. The line segments belonging to the corners that neighbor the occluded corner can be extended to repair the occluded corner (see Figure 4.13).

The candidate detection algorithm treats each edge segment separately. It simply ignores edge segments that have less than four line segments on them, as at least four line segments are needed to form a quad. If there are at least four line segments on an edge segment, the directions of the lines are found. Edges appear on the locations where there is a gradient transition from dark to light. Hence, we can say that one side of an edge segment is darker, while the other side is darker. Since EDLines detect line segments on the edge segments, line segments also have a light side and a dark side. The quads we are looking for has a black interior and a white exterior. While EDLines does not provide the information about which side is darker, we can extract this information from the image and represent it as the direction of the line. For each of the line segments, the brightness values of the pixels on their right-hand side and left-hand side are accumulated (see Figure 4.14). Then, the starting and ending point of the line segment is adjusted so that right-hand side is darker when going from starting point to ending point.

Now that we have a group of line segments that belongs to the same

Figure 4.14: Brightness values of each side of the line segments are accumulated. Starting and end points of the line segments are arranged so that right-hand side of the line segment has lower brightness values. The sampling points are actually greater in number and closer to the line segments.

edge segment with corrected directions, we can find the non-occluded corners on the edge segment. One rule that further decreases the possibility of detecting candidates is that only consecutive line segments on the edge segment can form a corner. This would only cause a problem if the vertices are so radially distorted that multiple line segments are extracted from a single vertex. Since we assume that the input image is undistorted, we can assume that this will not happen, and thus improve false positive rate and running time.

Each corner is represented by two consecutive line segments and their intersection. Since the detected quad will be convex, the inside of the corner must be darker, while the reflex angle side must be lighter. This can be checked algebraically so:

$$(line1_{end} - line1_{start}) \times (line2_{end} - line2_{start}) <= 0 \tag{4.4}$$

If the above equation is true, inside of the corner formed by $line1$ and

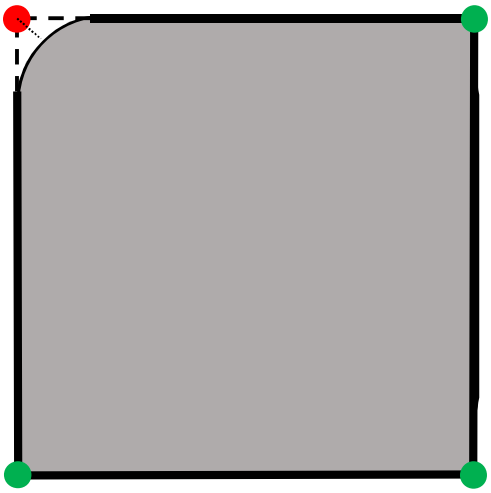


Figure 4.15: Elimination of corners based on proximity to edge segment. The red corner is found by extending belonging vertices, yet it is not validated due to its distance to the actual edge segment.

*line*2 is darker.

After verifying that inside of the corner would be dark if the line segments intersected, the intersection is found. If the intersection is too distant from any of the line segments, this corner is discarded. Note that we are looking for the non-occluded corners here. For such corners, the line segments are a few pixels distant from the intersection at most. The default threshold used for this is 5 pixels.

Sometimes circular edge segments, such as ones extracted from the coding circles, yield line segments. If these line segments are allowed to produce corners, this will unnecessarily burden the candidate detection implementation. To eliminate such cases, the detected corner is required to be on the edge segment that its line segments are extracted from (see Figure 4.15). Due to the smoothing step of Edge Drawing, the corners on the edge segments are somewhat rounded. Therefore, we threshold the distance to the edge segment, rather than looking for an exact overlap. Since precision is not critical at this step, Manhattan distance is used instead of Euclidian distance. When a corner is detected, the edge segment it belongs to is parsed to find if there is a pixel that is closer to the corner than the distance threshold. Since the worst case requires us to go through every element of the edge segment, using a lightweight distance function is worthwhile. If the corner belongs to a quad, the corner will be near to the edge segment, while if the corner belongs to a curved shape, the corner will be further from the edge segment.

We have mentioned that any pair of line segments can form a corner, yet we form corners using line segments consecutively found on a single edge segment to decrease the number of corners. Similarly, any pair of oppositely positioned corners is sufficient to identify a quad. Instead, we will use consecutive thirds of corners found on a single edge segment. When parsing through the corners of an edge segment, N^{th} and $N+2^{nd}$ corners are checked to see if they are opposite to each other. Only these two corners are sufficient to identify the quad. However, $N+1^{st}$ and $N+3^{rd}$ corners are also checked to see if they are the remaining corners of the quad. If they satisfy the requirements, they are used as the corners of the quads. Otherwise, missing corners are artificially generated using the opposite corners. This parsing scheme reduces the complexity from $O(n^2)$ to $O(n)$. If we had checked the 2-combinations of all corner pairs on the edge segment, there would be a much higher number of candidates. This optimization causes very rarely encountered miss cases. The first is the case in which radial distortion causes the line segment of a vertex to break into two, which result in an additional corner in the middle of the vertex. An additional corner has the risk of breaking the regular consecutive form of the corners, which is assumed by our optimization. However, we have mentioned that we assume the image is undistorted, thus this is not actually a problem. The second case is that we cannot detect candidates when two of the opposite corners are occluded, but the other two is detected on the same edge segment. When there is an occlusion on a corner, the occluding object generally breaks the edge segment. For this case to be encountered, both occluding objects on the opposite corners must not break the edge segment belonging to the marker. This case is even more unlikely, so it is feasible to forfeit this case to decrease the number of candidate quads.

Previous step results in convex quads whose all four corners are localized. Now we have to verify if these candidates belong to true markers. We have mentioned that if we have no information about the size of the marker, all convex quads may belong to true markers. If a marker has the size of a football pitch and it is seen from a close range, its projective transform on the image will be very different than a square. However, we can assume that markers typically come in smaller sizes. If the marker is smaller, perspective distortion will be less effective as long as marker is relatively far away from the camera.

The square border transforms into a quad via projective transform.

Projective transform can be divided into two aspects, affine transform and perspective transform. For our case, we can say that any parallelogram may be an affine transform of a marker. If two lines are parallel originally, they will still be parallel after any affine transform. All other aspects, such as rotation or scale may change. Perspective transform (as we define) causes the parallel lines not to be parallel anymore.

In Euclidian geometry, parallel lines are assumed not to intersect. This can also be explained such that parallel lines intersect at infinity. This is the case for the original square border's opposite vertices. When a projective transform is applied to the square, opposite vertices are no longer parallel. This means that while they intersected at infinity, now they intersect at a finite point. We can say that projective transformation (and specifically perspective transformation) can move points from infinity to a finite location. Let us define this relative directional change of vertices as perspective distortion.

When there is absolutely no perspective distortion, opposite vertices of the quad are parallel. In this case, the intersections of opposite vertices are at infinity. If perspective distortion exists, the opposite vertices are no longer parallel. The non-parallel vertices now intersect at finite points. While these intersection points were on the line at infinity before the transformation, they are now on a finite line. We can say that line at infinity is at an infinite distance before the transformation, while it is at a finite distance after the transformation. The line at infinity is moved to a closer location. Moreover, the distance it is moved to is actually relevant. Projection of the line at infinity will be closer as perspective distortion is more dominant [42].

Perspective distortion causes some ambiguity regarding distance and sizes of objects. If there are two objects that we know to be of the same distance from the camera, we can easily tell which is bigger. Similarly, if we know that two objects are of the same size, we can tell which is further away from the camera. However, if we do not know the sizes and distances of two identically shaped objects, we cannot make any estimates with monocular vision. In our case, we know the original shape of the marker, which is a square. By observing the perspective distortion on the shape, we can gather information about the relative distance of the points on the marker. Projection of the line at infinity comes closer as projective distortion increases. We can use the Euclidian distance from a point on the marker to the projection

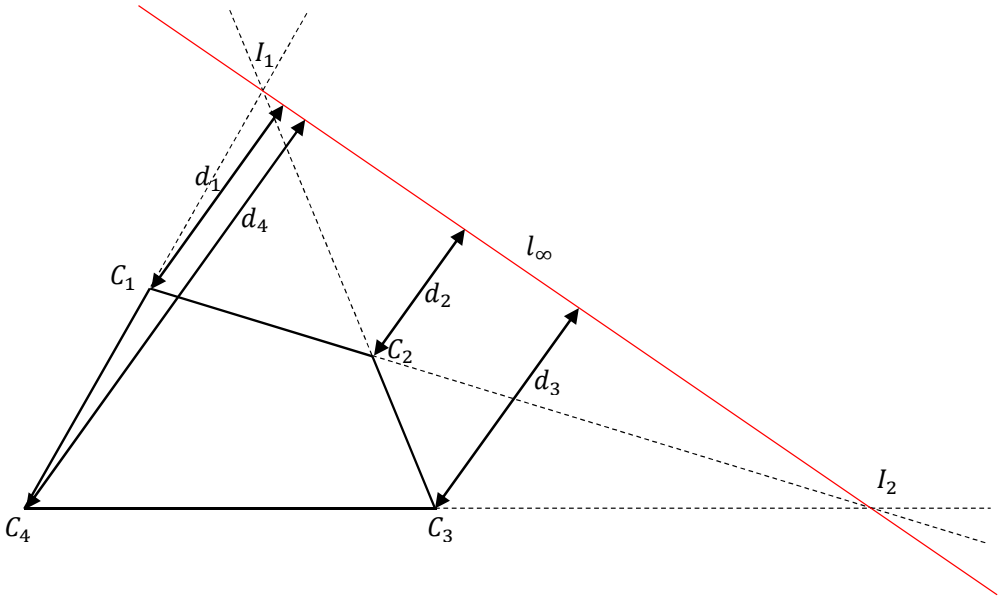


Figure 4.16: α_i is the relative distance from a point on the marker to center of the camera. d_i is the distance from a point on marker to the projection of the line at infinity.
$\alpha_i = k{d_i}^{-1}$ where k is a constant.

of the line at infinity as the metric of perspective distortion. We know that perspective difference occurs because of the distance difference between two points. We can say that distances of the points to the camera are proportional to perspective distortion, hence the distance to the projection of the line at infinity (see Figure 4.16).

Using the information above, we can find the relative distances of points on the marker to the camera. This information will be useful in eliminating oddly-shaped quads. Assume one side of our marker is 10cm. If we assume that the marker will not be used closer than 20cm to the camera, we can calculate the worst case scenario as one corner of the marker is 20 cm away from the camera, while the opposite corner is $20 + 10\sqrt{2}$ cm away. Hence the maximum relative distance ratio is 1.707. If the maximum ratio is calculated larger than that for a quad, it can be eliminated safely.

After all the verification steps, candidates will be decoded. All marker systems estimate the homography to find the code locations on the image. We have a much faster and deterministic method for this. We know that we can find the projection of the line at infinity. Using this line at infinity, perspective distortion of the quad can be removed [3]. Once this method is

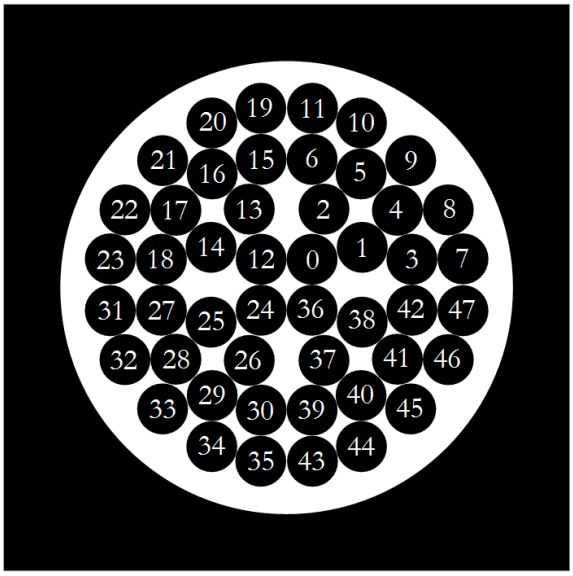


Figure 4.17: Allocation of bit significance. Instead of reading for all 4 rotations, code is read once and shifted for 12 bits to obtain bit sequences for other rotations.

applied, the quad turns into a parallelogram. Using ratios, we can easily find the point on the parallelogram that corresponds to the code location on the square. Then, we can project this point back on the quad to find the code location on the quad. Hence, all code locations on the quad can be found without using any iterative error minimization algorithm.

To read the binary code on the marker, first the borders of the marker is sampled to find the extreme brightness levels for black and white. Simply, the average of these two values are used to threshold the code readings. 48 bits are read from the marker. Due to the bit formation on the marker (see Figure 4.17), we only need to read once. To get the readings from the other 3 directions, we can just shift the bit sequence for multiples of 12, thanks to 4-fold symmetry.

The bit sequences that are associated with marker IDs are kept in a list. When the code is read, it is compared to each one of them regarding their Hamming distances. If one of them has less Hamming distance to the code read than number of error bits to be corrected, the candidate is verified as a true marker. When this happens, marker search on the current edge segment is discontinued. In other words, we assume that there can be only one real marker on an edge segment. If none of the bit sequences are close to the code read from the candidate, the candidate is discarded and search

on the edge segment continues.

After the bit sequence is decoded and verified to be true, the ellipse that is the transform of the inner circular border is detected using ellipse fit [43]. Only the center of this ellipse is kept to be used for multi-marker applications. However, the pose of the single marker can be further refined using the whole ellipse.

5. RESULTS

Performance criteria were discussed in detail in Chapter 3. Let us review what they were:

1. False positive rate
2. False negative rate
3. Marker library size
4. Intermarker confusion rate
5. Feature localization and pose estimation quality
6. Running time

We have mentioned that these performance metrics vary notably based on arbitrarily chosen system characteristics. For the sake of being able to quantify these to a degree, we will also make such decisions. While we tried to make decisions to achieve the best performance for a general-use marker system, it is a better strategy to customize these characteristics for specific applications. For example, one would rather not apply any error correction if false positive errors are fatal for the related application.

The changeable characteristics of EDMarkers used in experiments are as follows:

- The length of the bit sequence that will be encoded on the markers will be denoted by ν.
- The size of the marker library is denoted by λ.
- Minimum Hamming distance between the bit sequences that are associated with the marker IDs will be denoted by η.
- Maximum number of bits to be corrected will be denoted by ϵ.
- Average number of marker candidates for each frame is γ.

Since digital coding was used, some probabilities regarding the performance criteria can be theoretically calculated. Experiments will be conducted as well to see how well these theoretical values reflect reality.

Average number of marker candidates are calculated using a 1 minute-long video shot indoors. Only the candidates that were eliminated by the decoding step are considered. The rest of the argumentation will be conducted based on the following marker system characteristics:

ν	48
λ	128
η	13
ϵ	6
γ	2.32

5.1 False Positive Rate

False positive errors happen when a false candidate marker is falsely verified to be a true marker. A false candidate can have one of the 2^ν possible bit sequences. There are λ true bit sequences that are associated with marker IDs. Since we do error correction up to ϵ bits, each of these bit sequences occupy following amount of space in the binary space:

$$\textit{Area of a Single Marker ID} = \sum_{k=0}^{\epsilon} \binom{\nu}{k} \tag{5.1}$$

The ratio of valid bit sequences to the total space is as follows:

$$\frac{\textit{Occupied Area}}{\textit{Total Area}} = \frac{\lambda \sum_{k=0}^{\epsilon} \binom{\nu}{k}}{2^\nu} \tag{5.2}$$

If we calculate this according to our chosen characteristics, the result is 6.4560×10^{-6}. This is the probability of a random bit sequence to be matched with a marker ID. γ number of false candidates appear every frame. For a frame to not have any false positives, all candidates should be eliminated. The probability can be calculated as so:

$$P\ (\textit{No false positives in a frame}) = (1 - \frac{\lambda \sum_{k=0}^{\epsilon} \binom{\nu}{k}}{2^\nu})^\gamma \tag{5.3}$$

For our case, this probability is 0.999985. Consequently, we can say that

the probability of detecting any false positives on a frame is 1.4978×10^{-5}. A video shot in 25 FPS will have 1500 frames in a minute. The probability of detecting any false positives on this video is 2.2216%. Note that this probability changes dramatically with number of bits to be corrected. If we had kept ϵ at 2 as it was for ARTag [12], the probability of detecting any false positives on a 1 minute video would have been 1.86×10^{-4}%.

5.2 False Negative Rate

False negative errors happen when a marker is either not detected as a candidate, or eliminated during decoding. Unfortunately, it is difficult to quantify the success rate of candidate detection regarding false negatives. The only way to test this is setting up difficult cases and see if the marker is detected. However, one cannot objectively quantify the results. See Figure 5.1 for detection examples under difficult cases.

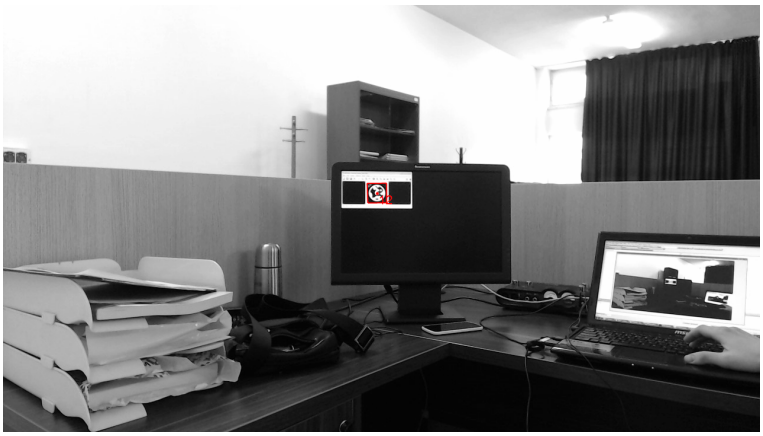
The embedded bit sequence on the marker may be read incorrectly due to noise, lighting conditions, occlusion or poor feature localization. Error correction is used in these cases to identify the marker. The probability of a single bit of the marker flipping varies dramatically depending on conditions. Still, let us denote it as β. For the marker to be eliminated falsely, at least $\epsilon + 1$ bits should be flipped.

$$P\left(\begin{matrix}\textit{A true marker being}\\ \textit{eliminated during decoding}\end{matrix}\right) = 1 - \sum_{k=0}^{\epsilon} P\ (k \textit{ number of bits flip}) \quad (5.4)$$

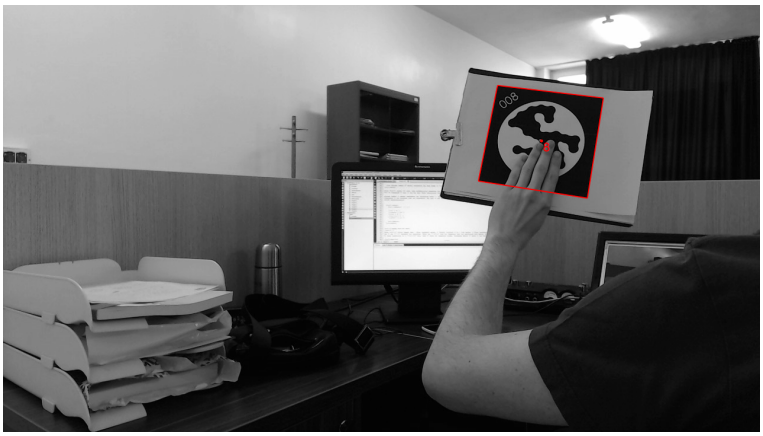
Assuming bit flips are independent events, the probability of a true marker being eliminated during decoding is as follows:

$$P\left(\begin{matrix}\textit{A true marker being}\\ \textit{eliminated during decoding}\end{matrix}\right) = 1 - \sum_{k=0}^{\epsilon} (1-\beta)^{\nu-k}\beta^{k} \quad (5.5)$$

There are mainly two concerns here. The first one is the fact that β is immeasurable. The second is the fact that bit flips are nor independent events, as they are assumed to be above. Hence, it is not possible to theoretically estimate false negative rate. On the other hand, it is undebatable that increasing ϵ will decrease false negative rate.



(a) The marker has 33 pixel-wide edges. It is detected stably accross all frames.

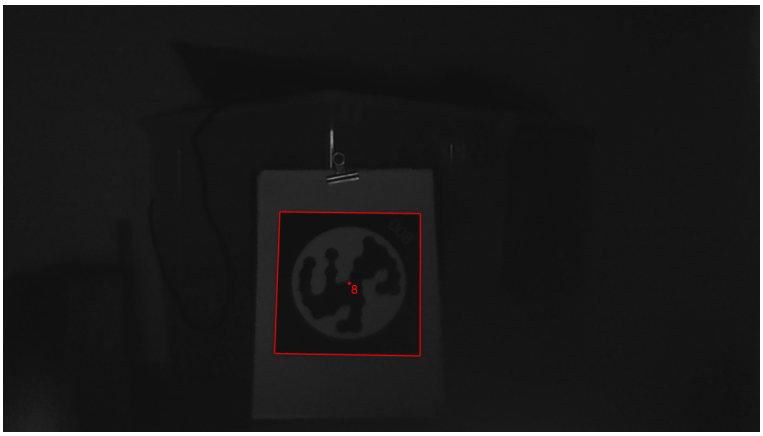


(b) An edge and some code circles of the marker are occluded.

(c) A corner and some code circles of the marker are occluded.

(d) The marker is seen at an extreme angle and there is blooming due to excessive lighting.

(e) The camera is defocused.



(f) The scene is lit poorly.

Figure 5.1: Examples of detection under difficult conditions.

5.3 Marker Library Size

Marker library size is directly controlled by the marker system designer. For our case, λ is 128. One should use the minimum marker library size for the application to improve other performance metrics. 128 is chosen for the sake of discussion, as it should be enough for any medium-scale application.

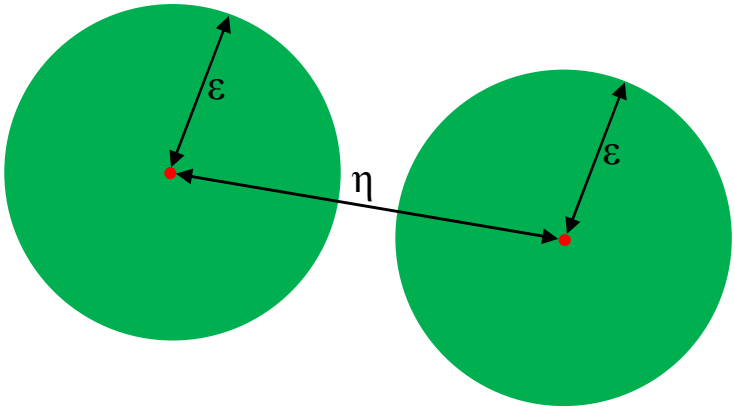
5.4 Intermarker Confusion Rate

ϵ, number of bits to be corrected, was chosen to be 6. Surely, there is an upper limit to the number of bits that can be corrected. See Figure 5.2a for a 2D depiction of the binary space. Considering we want a fully symmetrical coverage of space around the true bit sequence, ϵ should not exceed the half of η. If η is 13, number of bits to be corrected cannot exceed 6.

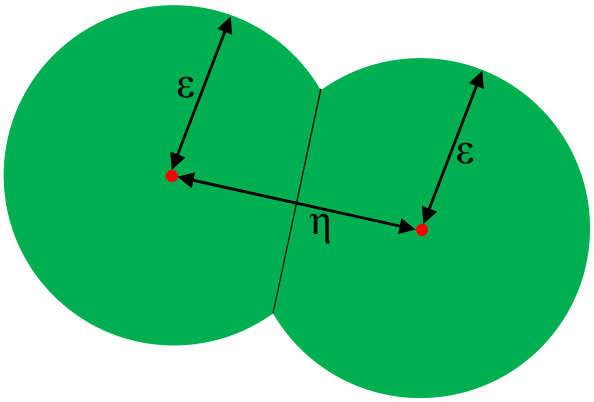
There is a workaround for using ϵ larger than the half of η. When the bit sequence being decoded is in an overlap area, it can be mapped to the marker ID with the smallest Hamming distance(see Figure 5.2b). However, that would mean that ϵ amount of bit flips to the direction of the nearest true bit sequence will result in intermarker confusion, rather than correct detection using error correction.

While Fiala discusses intermarker confusion rate in terms of Hamming distances [18], ϵ is actually just as influential. Suppose η is increased while ϵ is constant. A case that would cause intermarker confusion will now result in a false negative. Now, suppose ϵ is decreased while η is constant. Again, the intermarker confusion case will now transform into a false negative case. Decreasing marker library size and error correction to trade intermarker confusion with false negative is not really feasible. However, maximizing η is still valuable due to the discussion in Figure 5.2a.

Fiala does not quantify the probability of the intermarker confusion rate due to the unpredictability of β. Instead, he investigates cumulative histograms of Hamming distances between valid bit sequences [18]. On the other hand, Olson argues that we should only be interested with the minimum Hamming distance, most because the lexicode generating algorithm only guarantees to improve the worst case [34]. Olson's approach sets a standard for all bit sequences (η), hence allows for a larger ϵ to be used, while Fiala's approach may score better using his histogram metric.



(a) Red points are valid bit sequences associated to marker IDs. Green areas contain the bit sequences that will be associated with the respective red point using error correction. ϵ should be at most the half of η to avoid any overlap. Overlap will mean that some bit sequences are mapped to multiple marker IDs.



(b) The bit sequences in the overlapping area can be divided evenly to the marker IDs. However, in that case, error correction will not be ϵ in all directions.

Figure 5.2: Depictions of bit sequence associations in binary space.

Point	Mean of Jitter	Standard Devation of Jitter
Corner 1	0.0575	0.0442
Corner 2	0.0839	0.0771
Corner 3	0.0165	0.0169
Corner 4	0.0301	0.0347
Center	0.0078	0.0095

Table 5.1: Jitter characteristics of detections in pixels.

The fact that minimum Hamming distance between valid codes are 13 is the best quantification that can be made at this point. Since one cannot estimate β and bit flips are not independent, further effort on this issue is unnecessary.

5.5 Feature Localization and Pose Estimation Quality

Pose estimation is done using the geometric features on the marker. Therefore, accurate and stable feature localization will lead to successful pose estimation. Fiala addresses this metric as vertex jitter [29], yet it seems likely that he is using corner jitter measurements. He does not compare the detections with ground truths, as it is very difficult to produce accurate ground thruths with real images. Instead, he only measures the spatial stability of the localization. While the same test can be repeated, it will not be meaningful to make comparisons as the test images will not be taken under similar conditions.

When using 1280×720 images taken with Logitech c920 webcam, mean and standard deviation of the jitter can be seen in Table 5.1. As was argued earlier, the localization quality of the center point is much better than of corners'. Using the center for pose estimation when multiple markers are present will result in a great improvement in pose estimation quality.

5.6 Running Time

Running time will vary under different circumstances. Higher resolutions will result in higher number of primitive features and consequently higher number of candidates to be tested. Similarly, high number of markers or marker-like objects on the scene will increase the number of candidates and

Resolution	# of Markers on Scene	Running Time (ms)
640×480	0	8.814
	4	10.048
	32	22.832
1280×720	0	26.262
	4	27.253
	32	54.456
1920×1080	0	51.517
	4	54.010
	32	82.328

Table 5.2: Average running time under various conditions.

total running time. See Table 5.2 for average running times with Intel Core i7-3430 QM 2.40 GHz CPU. It is clear that the running time increases linearly with increasing resolution and having large number of markers on the scene does not hinder performance excessively.

6. CONCLUSION

The main objective of this study was to design a novel fiducial marker system to address the current requirements. A general knowledge of marker systems had needed to be established before proceeding. Existing studies are investigated to identify trends and deficiencies regarding the field. Using the gathered information, requirements to be fulfilled are proposed. The effects of different design decisions to these performance metrics are discussed in detail. Since these design decisions affect multiple aspects of marker system performance, tradeoffs that should be regarded are exposed clearly. Logical choices that are application-independent are specified to be used in our system design.

After introducing marker system design aspects, design decisions done for the proposed marker system, EDMarkers, are explained. The tradeoffs caused by these decisions are clearly justified. The implementation details of the marker design and detection algorithm design are reported for future reference. Continuing this, the resulting marker system is evaluated using the performance metrics stated beforehand.

The emerging marker system uses cutting-edge feature detection algorithms which helps with feature localization and running time. The marker candidate algorithm is designed to run fast, yet have the robustness to detect most difficult cases. The coding utilizes lexicodes, which have predictable, customizeable and high error detection and correction capabilities. The users can implement their own lexicode scheme and integrate it with the marker system easily.

In general, EDMarkers turned out to be a successful marker system that can be customized to be used in any kind of application that utilizes fiducial markers. The next objective will be refining the system regarding usability, and releasing it to be used in fiducial marker applications.

REFERENCES

[1] N. Ho, "Markerless augmented reality," 2011.

[2] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking.," in *IEEE International Conference on Computer Vision*, vol. 2, pp. 1508–1511, 2005.

[3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 ed., 2003.

[4] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[5] D. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, pp. 1150–1157 vol.2, 1999.

[6] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 10, pp. 1615–1630, 2005.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, 2008.

[8] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American statistical association*, vol. 79, no. 388, pp. 871–880, 1984.

[9] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[10] I. Qualcomm Connected Experiences, "Vuforia," 2014.

[11] I. Metaio, "Junaio," 2014.

[12] M. Fiala, "ARTag, a fiducial marker system using digital techniques," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 590–596 vol. 2, 2005.

[13] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on*, pp. 85–94, 1999.

[14] M. Abidi and T. Chandra, "A new efficient and direct solution for pose estimation using quadrangular targets: algorithm and evaluation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 17, no. 5, pp. 534–538, 1995.

[15] L. Quan and Z. Lan, "Linear n-point camera pose determination," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 8, pp. 774–780, 1999.

[16] X. Zhang, S. Fronz, and N. Navab, "Visual marker detection and decoding in ar systems: a comparative study," in *Mixed and Augmented Reality, 2002. ISMAR 2002. Proceedings. International Symposium on*,

pp. 97–106, 2002.
[17] M. Fiala, "Comparing ARTag and ARToolkit Plus fiducial marker systems," in *Haptic Audio Visual Environments and their Applications, 2005. IEEE International Workshop on*, pp. 6 pp.–, 2005.
[18] M. Fiala, "Designing highly reliable fiducial markers," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 7, pp. 1317–1324, 2010.
[19] D. Wagner and D. Schmalstieg, "Artoolkitplus for pose tracking on mobile devices," *Proceedings of 12th Computer Vision Winter Workshop (CVWW'07)*, pp. 139–146, 2007.
[20] C. Owen, F. Xiao, and P. Middlin, "What is the best fiducial?," in *Augmented Reality Toolkit, The First IEEE International Workshop*, pp. 8 pp.–, 2002.
[21] J. Sattar, E. Bourque, P. Giguere, and G. Dudek, "Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction," in *Computer and Robot Vision, 2007. CRV '07. Fourth Canadian Conference on*, pp. 165–174, 2007.
[22] A. Xu and G. Dudek, "Fourier tag: A smoothly degradable fiducial marker system with configurable payload capacity," in *Computer and Robot Vision (CRV), 2011 Canadian Conference on*, pp. 40–47, 2011.
[23] J. Rekimoto, "Matrix: a realtime object identification and registration method for augmented reality," in *Computer Human Interaction, 1998. Proceedings. 3rd Asia Pacific*, pp. 63–68, 1998.
[24] H. Koike, Y. Sato, and Y. Kobayashi, "Integrating paper and digital information on enhanceddesk: a method for realtime finger tracking on an augmented desk system," *ACM Transactions on Computer-Human Interaction*, vol. 8, no. 4, pp. 307–322, 2001.
[25] J. Rekimoto and Y. Ayatsuka, "Cybercode: designing augmented reality environments with visual tags," in *Proceedings of DARE 2000 on Designing augmented reality environments*, pp. 1–10, ACM, 2000.
[26] D. López de Ipiña, P. R. S. Mendonça, and A. Hopper, "TRIP: A low-cost vision-based location system for ubiquitous computing," *Personal Ubiquitous Comput.*, vol. 6, no. 3, pp. 206–219, 2002.
[27] L. Naimark and E. Foxlin, "Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker," in *Mixed and Augmented Reality, 2002. ISMAR 2002. Proceedings. International Symposium on*, pp. 27–36, 2002.
[28] "Information technology – Automatic identification and data capture techniques – Data Matrix bar code symbology specification," 2006.
[29] M. Fiala, "ARTag revision1. a fiducial marker system using digital techniques," 2004.
[30] M. Fiala and C. Shu, "Self-identifying patterns for plane-based camera calibration," *Machine Vision and Applications*, vol. 19, no. 4, pp. 209–216, 2008.
[31] B. Atcheson, F. Heide, and W. Heidrich, "CALTag: High precision fiducial markers for camera calibration.," in *VMV*, pp. 41–48, Eurographics

Association, 2010.
[32] A. Rice, A. Beresford, and R. Harle, "Cantag: an open source software toolkit for designing and deploying marker-based vision systems," in *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*, pp. 10–21, March 2006.
[33] F. Bergamasco, A. Albarelli, E. Rodola, and A. Torsello, "RUNE-Tag: A high accuracy fiducial marker with strong occlusion resilience," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 113–120, June 2011.
[34] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3400–3407, 2011.
[35] H. Uchiyama and H. Saito, "Random dot markers," in *Virtual Reality Conference (VR), 2011 IEEE*, pp. 35–38, 2011.
[36] E. Specht, "The best known packings of equal circles in a square," 2014.
[37] F. Fodor, "The densest packing of 19 congruent circles in a circle," *Geometriae Dedicata*, vol. 74, no. 2, pp. 139–145, 1999.
[38] E. Specht, "The best known packings of equal circles in a circle," 2014.
[39] C. Akinlar and C. Topal, "EDPF: A real-time parameter-free edge segment detector with a false detection control," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 26, no. 01, 2012.
[40] C. Topal, C. Akinlar, and Y. Genç, "Edge drawing: A heuristic approach to robust real-time edge detection," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 2424–2427, 2010.
[41] C. Akinlar and C. Topal, "Edlines: Real-time line segment detection by edge drawing (ed)," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pp. 2837–2840, Sept 2011.
[42] B. Benligiray, C. Topal, C. Akinlar, and Y. Genc, "Quantification of projective distortion for fiducial markers," in *Signal Processing and Communications Applications Conference (SIU), 2013 21st*, pp. 1–4, 2013.
[43] A. Fitzgibbon, M. Pilu, and R. Fisher, "Direct least square fitting of ellipses," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 5, pp. 476–480, 1999.