Generation of fiducial marker dictionaries using Mixed Integer Linear Programming

Article *in* Pattern Recognition · October 2015

DOI: 10.1016/j.patcog.2015.09.023

4 authors, including:

Rafael Muñoz-Salinas
University of Cordoba (Spain)
89 PUBLICATIONS **2,712** CITATIONS

SEE PROFILE

Francisco J. Madrid-Cuevas
University of Cordoba (Spain)
46 PUBLICATIONS **1,999** CITATIONS

SEE PROFILE

Rafael Medina-Carnicer
University of Cordoba (Spain)
89 PUBLICATIONS **1,674** CITATIONS

SEE PROFILE

Generation of fiducial marker dictionaries using mixed integer linear programming

S. Garrido-Jurado[1], R. Muñoz-Salinas*, F.J Madrid-Cuevas, R. Medina-Carnicer

Computing and Numerical Analysis Department, Córdoba University, Spain

Abstract

Square-based fiducial markers are one of the most popular approaches for camera pose estimation due to its fast detection and robustness. In order to maximize their error correction capabilities, it is required to use an inner binary codification with a large inter-marker distance. This paper proposes two Mixed Integer Linear Programming (MILP) approaches to generate configurable square-based fiducial marker dictionaries maximizing their inter-marker distance. The first approach guarantees the optimal solution, however, it can only be applied to relatively small dictionaries and number of bits since the computing times are too long for many situations. The second approach is an alternative formulation to obtain suboptimal dictionaries within restricted time, achieving results that still surpass significantly the current state of the art methods.

Keywords: fiducial markers, MILP, mixed integer linear programming, augmented reality, computer vision.

1. Introduction

Camera pose estimation is a common problem in numerous computer vision applications such as robot navigation [1, 2] or augmented reality [3, 4, 5], which is usually based on obtaining correspondences between environment and image points. While the use of natural features, such as key points or textures [6, 7, 8, 9], is a very popular strategy which does not require altering the environment, the use of fiducial markers is still of great importance since it provides point correspondences more robustly, efficiently and precisely.

In particular, square-based fiducial markers are the most popular in the field of augmented reality [4, 10, 11] since a single marker provides the four points required to estimate the camera pose (given that it is properly calibrated). In general, squared-based markers use an inner binary code for identification, error detection and correction.

The detection process of this type of markers can be split in two main steps. The first step is the candidate search, which consists in finding square shapes in the image that look like markers. The second step is the identification stage, where the inner codification of the candidates is analyzed in order to determine whether they really are markers, and if they belong to the considered set of valid ones, also known as dictionary.

A key aspect of such dictionaries is the inter-marker distance [10], which is the minimum Hamming distance between the binary codes of the markers, considering the four possible rotations. This distance defines the maximum number of bits that can be corrected without producing an inter-marker confusion error, i.e. a marker being erroneously identified as a different one. As a consequence, the inter-marker distance is directly related to the error correction capabilities of a dictionary. The larger the inter-marker, the lower the false negative and inter-marker confusion rates, and therefore, the higher the robustness of the process.

For instance, Figure 1 shows an example of inter-marker confusion error and the importance of large inter-marker distances. The two first markers have a short distance of only 1 bit while the third marker has larger distances of at least 5 bits to the rest of markers. As it can be seen in Figures 1d,e, a single erroneous bit is enough to cause a wrong identification of the second marker. On the other hand, the third marker is correctly identified despite having a higher number of errors.

Most related works propose their own predefined dictionary of markers with a fixed number of markers and bits, *and a constant inter-marker distance*. However, using a predefined dictionary for every application is not the optimal approach. Instead, if the number of required markers and their size is known, it is preferable to create a custom dictionary that maximizes *the inter-marker distance and, consequently,* the error detection and correction capabilities. *Although this is the tendency of the latest proposals [12, 13], they rely on heuristic approaches, none of them being optimal.*

This paper presents two novel dictionary generation methods based on the Mixed Integer Linear Programming (MILP) paradigm. The first MILP model proposed guarantees the optimal inter-marker distance for a specific number

*Corresponding author

Email addresses: i52gajus@uco.es (S. Garrido-Jurado), rmsalinas@uco.es (R. Muñoz-Salinas), fjmadrid@uco.es (F.J Madrid-Cuevas), rmedina@uco.es (R. Medina-Carnicer)

[1]Computing and Numerical Analysis Department, Edificio Einstein. Campus de Rabanales, Córdoba University, 14071, Córdoba, Spain, Tlfn:(+34)957212255

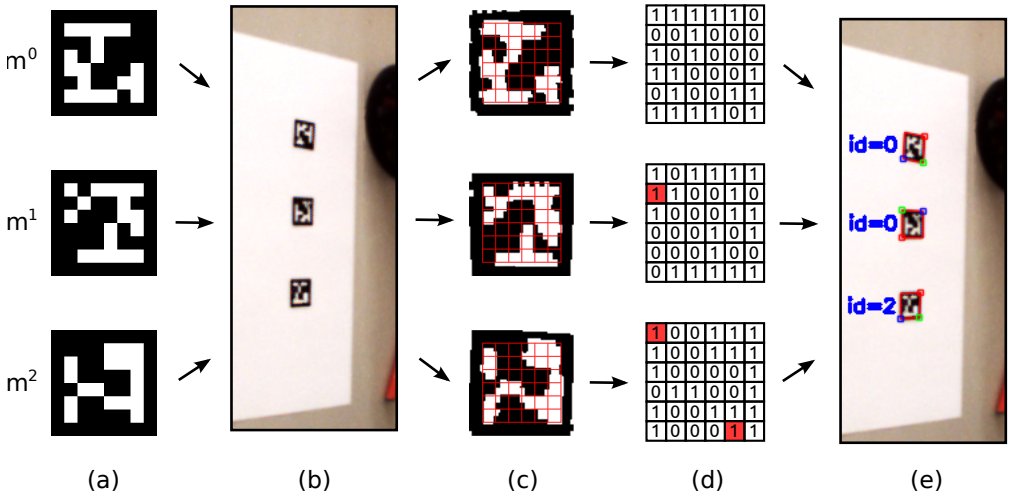


Figure 1: *Example of inter-marker confusion error and how it is avoided with large inter-marker distances. (a) Original marker images. The marker distances are $D(m^0, m^1) = 1$, $D(m^0, m^2) = 5$ and $D(m^1, m^2) = 6$. (b) Real image with the markers placed in the environment. (c) Marker images after removing the perspective. (d) Inner bits extracted from images in c. Erroneous bits are highlighted in red. (e) Final identifiers assigned to each marker. It can be observed that m^1 has been confused with m^0. On the other hand, m^2, despite a higher number of errors, can be correctly identify since the distance to the rest of markers is higher. (Best seen in color).*

of markers and bits. *It is the first approach in the literature, up to our knowledge, that assures optimal results in terms of inter-marker distance.* However, since the convergence time of this model is too long for many applications, we also propose an alternative MILP formulation that converges faster, and, although it does not guarantee optimality, its results surpass the current state of the art significantly. *The two proposed approaches presented in this paper constitute relevant improvements to the error correction capabilities of fiducial marker systems.*

The rest of the paper is structured as follows. Section 2 reviews related work. Section 3 presents a mathematical formalization of the problem. Section 4 details the proposed MILP models to our problem. Finally, Section 5 presents the experimentation carried out, and Section 6 draws some conclusions.

2. Related work

Fiducial markers are synthetic elements placed in the working area either to facilitate the camera pose estimation task or for labeling purposes. They are specially designed to be easily detected even at low resolutions and most of the applications require not one but many different markers (a dictionary). Thus, the ability of identifying them uniquely is an important feature. Several fiducial marker systems have been proposed in the literature as shown in Figure 2.

The simplest proposals are those based on fiducial points. These markers constitute a single scene point and are usually based on leds, retroreflective spheres or planar dots [14, 15]. Their identification is typically based on the relative positions of different points, which can be a limiting and complex process.

A straight evolution of the point markers are the circular markers [16, 17] (Fig. 2a). These markers are similar to the previous ones except for the fact that they include information (as circular sectors or concentric rings) to facilitate the identification process. Their main drawback is that they provide only one correspondence point per marker.

Other types of fiducial markers are based on blob detection. For instance, Cybercode [18] and VisualCode [19] (Fig. 2b,c) are based on the same technology than QR and Maxicode [20] codes, but providing several correspondence points. Other popular fiducial markers based on blob detection are the ReacTIVision amoeba markers [21], that are designed using genetic algorithms (Fig. 2d).

One of the most popular families of fiducial markers are the square-based ones. They contain a black border to ease their detection and employ their inner region for identification purposes. Their main benefit is that each marker provides four prominent points (i.e. its four corners) which can be easily detected and employed as correspondence points, thus allowing camera pose estimation using a single marker. In this category, one of the most popular systems is ARToolKit [4], an open source project which has been extensively used in the last decade, especially in the academic community. ARToolKit markers include a pattern in their inner region for identification which can be customized by the user (Fig. 2e). Despite its popularity, it presents some drawbacks. Firstly, it uses a template matching strategy to identify the markers, which produces a high false positive rate [22]. Secondly, the square detection is based on global thresholding which makes it high sensitive to the lighting conditions.

Instead of using template matching, the majority of square-based marker systems employs a binary codification in their inner region [23, 10, 11]. Most approaches use a

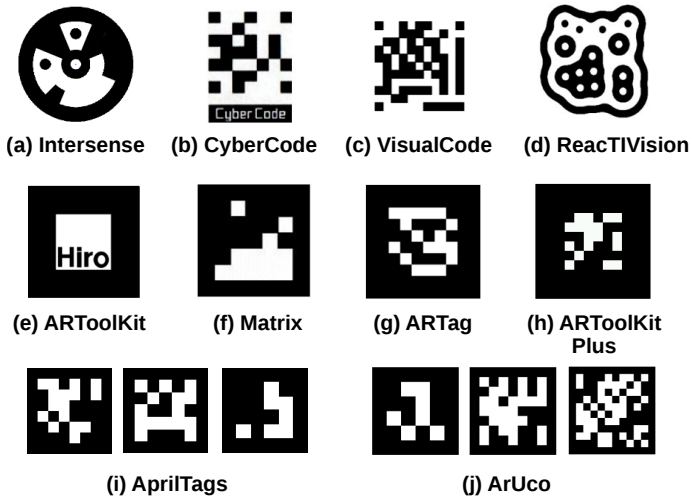


Figure 2: Examples of fiducial markers proposed in previous works.

codification based on classic methods of signal coding, such as CRC codes [24], achieving a more robust identification and facilitating the error detection and correction processes.

Matrix [23] is one of the first and simplest proposals which uses a binary code with redundant bits for error detection (Fig. 2f). ARTag [10] (Fig. 2g) is based on the same idea but it employs a more robust codification. Furthermore, it improves the square detection using an edge-based method instead of the global thresholding of ARToolKit. ARTag provides its marker dictionary in a specific order so that the inter-marker distance is maximized. Its main drawbacks are that the marker size is fixed to 6×6 bits and the error correction process can correct up to one bit, independently of the inter-marker distance of the used marker subset.

ARToolKit Plus [11] (Fig. 2h) improves some of the features of its predecessor ARToolKit. Firstly, it employs a dynamic method to update the global threshold depending on the pixel values in the previous detected markers. Secondly, as in ARTag, it provides a binary codification for marker identification. The first ARToolKit Plus version includes 512 markers whose codification is based on repeating four times a 9-bits identifier, achieving a minimum distance of four bits between any pair of markers. The last known version instead, proposes a dictionary of 4096 markers with 6×6 bits based on a BCH codification [25] that presents a minimum distance of two bits so that error correction is not possible. ARToolKitPlus project was halted and followed by the Studierstube Tracker [26] project which is not publicly available.

The main problem of codification based on classic coding techniques is that they need to deal with the different marker rotations, which affect negatively to the inter-marker distances of the generated dictionaries. Some approaches employ special anchor points, such as QR and Maxicode [20], to remove the rotation ambiguity. However, this complicates the detection process and, more importantly, these anchor points are vulnerable to errors since they are not protected by any coding system.

Most recent approaches rely on heuristics to select a set of markers with large inter-marker distances, considering rotation. However, since the search space is very large even for small dictionaries with a low number of bits, an exhaustive search is unfeasible and optimality can not be guaranteed.

One of the first and simplest proposals is the BinARyID system [27], whose dictionary generation process is based on selecting those markers that accomplish a minimum Hamming distance of one to any of the previous selected markers, so that rotation ambiguities are avoided. The markers are analyzed in ascending order until the desired number of markers is achieved. Its main problem is that it does not allow error detection and correction (since the distance is one) and the generation times can be prohibitive for large dictionaries.

The generation method proposed by the AprilTags library [13] (Fig. 2i) employs a similar approach than BinARyID, but with some significant improvements. Firstly, the minimum Hamming distance can be provided by the user, generating dictionaries with larger inter-marker distances and, hence, allowing error detection and correction. Secondly, instead of analyzing markers one by one in ascending order, larger increments are performed based on an heuristic approach, so that markers with larger inter-marker distances are found faster. Finally, selected markers also need to accomplish a minimum geometric complexity to increment the number of bit transitions. Its main downside is that the generation time is still very large, specially for large marker sizes.

In [12], the ArUco coding system is presented (Fig. 2j). Its generation is based on maximizing both the inter-marker distance and the number of bit transitions. *Contrary to AprilTags, the minimum inter-marker distance does not need to be provided by the user, instead it is automatically derived during the generation process.* However, it has two main downsides. Firstly, it uses a time consuming stochastic search strategy. Secondly, as the marker size increases, its memory requirements grow exponentially, limiting the maximum size to which it can be applied. As ARTag, ArUco sorts the generated markers in a list so as to maximize the inter-marker distance.

This paper proposes two novel approaches to generate square-based fiducial marker dictionaries based on Mixed Integer Linear Programming (MILP) [28]. MILP methods can achieve the optimal results of a mathematical model represented by linear relationships and where some unknowns are constrained to be integers. MILP problems receive special attention from the community since they fit many real-life situations, such as embedded system design [29], industrial processes [30], automatic scheduling [31], distribution systems [32] or trajectory planning [33]. However, these kind of problems are known to be NP-hard and, as a consequence, there have been many efforts in developing techniques to speed up the convergence process.

In contrast to previous works, our first proposed method achieves the optimal dictionary in terms of inter-marker distance for any number of markers and bits. It is the first approach in the literature that guarantees the optimal inter-marker distances, up to our knowledge. However, it suffers from the course of dimensionality and the computing times are too long as the size of the dictionaries or number of bits increase. Thus, we propose a second approach that achieves suboptimal dictionaries within restricted computing times which compares very favorable to the dictionaries obtained by previous works.

As shown in the experimental section, our methods obtain state of the art dictionaries (in terms of inter-marker distances), which are a relevant improvement to the error correction capabilities of square fiducial marker systems.

3. Problem formulation

The most relevant aspects to consider during the design of a marker dictionary are the false positive rate, the false negative rate and the inter-marker confusion rate. The first two are usually handled using error detection and correction techniques. On the other hand, the inter-marker confusion rate depends only on the distance among the dictionary markers. If the distance between two markers is short, a marker could be confused with another one with just a few bit modifications, and the error could not be detected. *As a consequence, this value also affects to the maximum number of erroneous bits that can be corrected.*

Let us denote by $\mathbb{D}$ the set of all possible markers of $n \times n$ bits and by $\mathbb{D}^d \subset \mathbb{D}$ the subset of all vectors formed by exactly d markers. An element $\mathfrak{D} = (m^1, m^2, m^3 \ldots, m^d)$ of the set $\mathbb{D}^d$ is named a dictionary, where the super-index indicates the marker position in the dictionary.

An automatic dictionary generation process consists in selecting a dictionary $\mathfrak{D}$ from the set $\mathbb{D}^d$, so that each m^i, $i \in \{1, ..., d\}$, is as far from each m^j, $j \in \{1, ..., d\}$, $j \neq i$, as possible. In general, the problem is to find the dictionary $\mathfrak{D}^*$ that maximizes the desired criterion $\tau(\mathfrak{D})$:

$$\mathfrak{D}^* = \underset{\mathfrak{D} \in \mathbb{D}^d}{\operatorname{argmax}} \{\tau(\mathfrak{D})\}. \tag{1}$$

The criterion $\tau(\mathfrak{D})$ employed in this work is the dictionary inter-marker distance, which is the minimum Hamming distance between any two markers of $\mathfrak{D}$. This value is of great importance since it indicates the minimum number of erroneous bits that can be corrected: $\lfloor(\tau(\mathfrak{D}) - 1)/2\rfloor$. If the number of erroneous bits of a marker is lower than or equal to this value, it can be guaranteed that the closest marker in the dictionary is the correct one. However, if the number of erroneous bits is higher than $\tau(\mathfrak{D})$, the assumption does not hold and hence, the error correction cannot be performed because the closest marker could not be the correct one (inter-marker confusion mistake). Thus, the objective of a dictionary generation method is to maximize the function $\tau(\mathfrak{D})$.

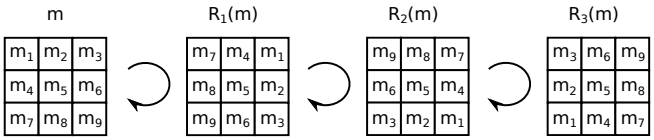


Figure 3: Example of 90 degrees rotations of a marker, m, composed by 3×3 bits. The marker m is represented by the bits in row-major order $(m_1, \ldots, m_9)$. As can be observed, after one rotation, the bits are permuted and the obtained marker, $R_1(m)$, is represented by $(m_7, m_4, m_1, m_8, m_5, m_2, m_9, m_6, m_3)$. The same process repeats for the rest of rotations.

Since an exhaustive evaluation of the entire search space is not feasible, a MILP model to find the optimal solution is proposed in this work.

Let us start defining the marker named i, i.e. a component of a dictionary $\mathfrak{D}$, as a binary matrix m^i of size $n \times n$:

$$m^i = (m^i_1, m^i_2, m^i_3 \ldots, m^i_{n \times n}) \mid m^i_k \in \{0, 1\}, \tag{2}$$

where m^i_k denotes the k-th bit of the marker matrix m^i assuming a row-major order.

Note that a marker detected in an image can be rotated respect to its original position and, thus, the marker bits will also be rotated. As it is shown in Fig. 3, a marker rotation can be formulated as a permutation of the marker bits. For a marker m^i, let us define its analogous set, $\mathcal{A}(m^i)$, as the set of the markers obtained after the three possible rotations plus the marker itself:

$$\mathcal{A}(m^i) = \bigcup_{l=0}^{3} R_l(m^i), \tag{3}$$

being R_l an operator that rotates the marker bits $l \times 90$ degrees in clockwise direction. In fact, all the markers in an analogous set can be considered as equivalent solutions of the search space.

Since our goal is to obtain a dictionary that maximizes the inter-marker distance, the distance between two markers, $m^i, m^j, \mid i, j \in \{1, ..., d\}$, $i \neq j$, must be properly defined. Considering that all the elements in an analogous set are equivalent, the distance between markers from different analogous set is defined as:

$$D(m^i, m^j) = \min_{m^k \in \mathcal{A}(m^j)} \{\ H(m^i, m^k)\ \}, \tag{4}$$

where the function H is the Hamming distance between two markers.

Furthermore, since we want to obtain the camera pose with respect to the marker, its corners must be identified unequivocally. Therefore, the distance of a marker to the rest of elements in its analogous set must be considered too. This distance, referred to as marker *self-distance*, is defined as:

$$\mathcal{S}(m^i) = \min_{m^k \in \mathcal{A}'(m^i)} \{\ H(m^i, m^k)\ \}, \tag{5}$$

where $\mathcal{A}'(m^i)$ is the analogous set of m^i without considering the marker itself:

$$\mathcal{A}'(m^i) = \mathcal{A}(m^i) - \{m^i\}. \tag{6}$$

In the end, the objective function $\tau(\mathfrak{D})$ in Eq. 1 is the minimum distance among the marker self-distances and the distances between any pair of markers in the dictionary:

$$\tau(\mathfrak{D}) = \min \left\{ \min_{m^i \in \mathfrak{D}} \left\{ \mathcal{S}(m^i) \right\}, \min_{\substack{m^i, m^j \in \mathfrak{D} \\ m^i \neq m^j}} \left\{ D(m^i, m^j) \right\} \right\}. \tag{7}$$

This function represents the inter-marker distance of a dictionary and the goal of the dictionary generation methods proposed in this work is to maximize it.

Figure 1 shows a marker detection example which illustrates the benefits of using dictionaries with large inter-marker distances. A dictionary composed by 3 markers of 6×6 bits is shown in Fig.1a,b. While the distance between m^0 and m^1 is only one bit, the distances to m^2 are larger, $D(m^0, m^2) = 5$ and $D(m^1, m^2) = 6$. The markers have been detected using a standard marker detection process like the one in [12]. Figure 1c shows the images obtained after removing the perspective distortion of each marker and Figure 1d shows the extracted bits from each image. It can be observed that the images in Figure 1c have an important amount of noise, which usually produces errors during the bit extraction process. Erroneous bits are highlighted in red in Figure 1d. Finally, Figure 1e shows the assigned identifiers to each of the markers applying a maximum error correction of 2 bits. The marker m^1 has been erroneously identified as m^0 due to the erroneous extracted bit and the short marker distance. Note that a single error is enough to make m^1 identical to m^0. On the other hand, m^2 has been correctly identified despite presenting two erroneous bits. Due to the large marker distance, it is less unlikely that m^2 gets erroneously identified during the identification step.

3.1. Maximum inter-marker distance: τ^n_{max}

In order to reduce the search space and accelerate the MILP convergence, it is of great importance to known the maximum possible value of the objective function $\tau(\mathfrak{D})$. Let us denote by τ^n_{max} the maximum possible inter-marker distance of a dictionary with markers of $n \times n$ bits. In this section, the derivation of this value is presented as already done in [12].

If we think of the simplest possible dictionary, we realize that it is composed by a single marker. Then, the marker self-distance constitutes the inter-marker distance of the dictionary. If a second marker is added to the dictionary, the new inter-marker distance will be smaller than or equal to the previous one. As a consequence, the maximum theoretical value of τ^n_{max} is given only by the self-distance (Eq. 5) of the first marker.

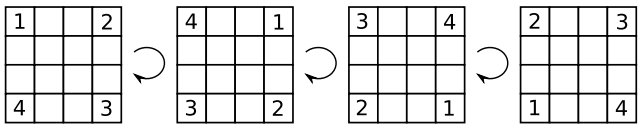


Figure 4: Example of quartet in a 4×4 bits marker.

Group	Quartets	Hamming distances		
		90 deg	180 deg	270 deg
Q_1	0000, 1111	0	0	0
Q_2	1000, 0100, 0010, 0001, 1110, 0111, 1011, 1101	2	2	2
$\mathbf{Q_3}$	1100, 0110, 0011, 1001	2	4	2
$\mathbf{Q_4}$	0101, 1010	4	0	4

Table 1: Quartet groups and Hamming distances they provide in each rotation.

Quartet	Group	Hamming distances		
		90 degrees	180 degrees	270 degrees
1	Q_3	2	4	2
2	Q_3	2	4	2
3	Q_4	4	0	4
4	Q_3	2	4	2
Total distances		10	12	10
τ^4_{max}		$\min(10, 12, 10) = 10$		

Table 2: Quartet assignment for a 4×4 marker ($C = 4$) to obtain $\tau^4_{max} = 10$. It can be observed that the sequence $\{Q_3, Q_3, Q_4\}$ is repeated until filling all the quartets in the marker.

The key point to understand the derivation of τ^n_{max} is the concept of *quartet*, which is the set of four bits that interchange their positions at each rotation (see Figure 4). As can be observed, these four bits do not *interact* with the rest of bits when the marker is rotated. Hence, a quartet contributes to the marker self-distance independently from the rest of quartets. In general, the number of quartets of a marker is $C = \left\lfloor \frac{n^2}{4} \right\rfloor$. If n is odd, the central bit constitutes a quartet by itself that can be ignored since it does not influence on the self-distance.

Since a quartet is composed by 4 bits, there is a total of 16 different possible quartets. From a brief study, it can be observed that some of the quartets provide the same Hamming distances in each rotation. For the purpose of calculating τ^n_{max}, these quartets can be considered equivalent and they can be grouped into the same *quartet group*, Q_i. Table 1 shows the 4 different quartet groups and the Hamming distances they provide in each rotation.

For instance, the quartet 1100 contributes with Hamming distances $(2, 4, 2)$ as it rotates:

$$H(1100, 0110) = 2; H(1100, 0011) = 4; H(1100, 1001) = 2,$$

and quartet 1001, which belongs to the same quartet group, contributes with the same Hamming distances:

$$H(1001, 1100) = 2; H(1001, 0110) = 4; H(1001, 0011) = 2.$$

Hence, the problem of obtaining the maximum marker self-distance consists in assigning each quartet to a quartet

group, so that the minimum marker distance of the three rotations is maximized. If this is understood as a multiobjective problem, the Pareto front is composed by the quartet groups Q_3 and Q_4, since they dominate all the other solutions. Thus, the problem is simplified in assigning each quartet to any of the two quartet groups Q_3 and Q_4. Then, it can be easily deduced that the maximum value τ^n_{max} is obtained by assigning the groups $\{Q_3, Q_3, Q_4\}$ (in this order) repeatedly until completing all the quartets of a marker.

For instance, for a 4×4 marker ($C = 4$), τ^4_{max} is obtained by assigning the groups $\{Q_3, Q_3, Q_4, Q_3\}$, as it is shown in Table 2.

In general, the maximum inter-marker distance is calculated as:

$$\tau^n_{max} = 2 \left\lfloor \frac{4C}{3} \right\rfloor . \tag{8}$$

4. Proposed solutions

This section presents our proposals to generate marker dictionaries using MILP. First, a short introduction to MILP is given. Then, our first model is presented, which obtains optimal solutions. Finally, our second model, which obtains suboptimal solutions within restricted time is presented.

4.1. Mixed Integer Linear Programming (MILP)

An integer linear programming (ILP) problem is a mathematical optimization or feasibility program in which some or all of the decision variables are restricted to be integers and the objective function and the constraints are linear.

The canonical form of a integer linear program is:

$$\begin{aligned} &\text{maximize } c^t x \\ &\text{Subject to} \\ &\quad Ax \leq b \\ &\quad x \geq 0 \\ &\quad x \in \mathbb{Z}, \end{aligned} \tag{9}$$

where x is the vector of decision variables, c is the coefficient vector of the objective function, A is the coefficient matrix of the constraints and b is the constant terms vector. The last constraint forces the decision variables to be integers, although in practice this constraint can be applied to some or all of the variables. In case all variables are constrained to be integers, the problem is known as a Pure Integer Linear Programming Problem (PILP), otherwise it is known as Mixed Integer Linear Programming (MILP).

Whereas Linear Programming problems belong to complexity class P [34], which means they are efficiently solvable, ILP or MILP problems are known to be NP-hard due to the integer restriction and, thus, they cannot be solved in polynomial time [35]. In fact, the particular case where the decision variables are binary is one of the problems in the well known list of Karp's 21 NP-complete problems [36]. This also implies that the computational complexity cannot be determined, neither analytically or experimentally.

However, these kind of problems have been extensively studied in the literature and many techniques have been proposed in order to obtain the optimal solution efficiently. *The main techniques are based in the Branch and Cut method [37], which is an iterative process that combines the Branch and Bound [38] algorithm and the use of cutting planes [39].*

Branch and Bound is an optimization algorithm based on a search tree which is explored by partitioning the search space on each node. It is comprised by the branching step and the bounding step.

During the branching step, a node is split in several branches by dividing the possible values of a specific variable, so that the union of all the branches covers all the possibilities. In the MILP procedure, this is performed by splitting the different values that a decision variable can take.

The bounding step determines the lower and upper bounds of the optimization function in a particular branch. If these bounds cannot surpass the current best solution, the branch is pruned, reducing the search space. In the MILP case, non-integral solutions to LP relaxations, i.e. the problem without considering the integer constraints, serve as upper bounds and integral solutions serve as lower bounds.

Finally, cutting planes can be applied during the optimization process to further reduce the search space. Cutting planes generate new restrictions for the model that are satisfied for any feasible solution, i.e. any integer solution, but violated by the current solution of the LP relaxation, so that the next non-integer optimal solution should be closer to the integer one.

The Branch and Cut algorithm explores the tree until finding the optimal solution, nevertheless many feasible non-optimal solutions can also be found during the process.

4.2. Optimal Dictionary

In order to obtain dictionaries with optimal inter-marker distances, we propose a MILP model to obtain the maximum value of the cost function $\tau(\mathfrak{D})$. This model is processed by a MILP solver so that Branch and Cut algorithm is applied to reduce the search space and speed up the convergence to the optimum.

The decision variables of the proposed model are the bits m^i_j of the dictionary markers, i.e., one binary variable per bit. In order to formulate the MILP problem, let us rewrite the objective function in Eq. 7 as:

$$\tau(\mathfrak{D}) = \min \left\{ \begin{array}{cc} \min\limits_{\substack{m^i \in \mathfrak{D} \\ m^k \in \mathcal{A}'(m^i)}} \{H(m^i, m^k)\}, & \min\limits_{\substack{m^i, m^j \in \mathfrak{D} \\ m^i \neq m^j \\ m^k \in \mathcal{A}(m^j)}} \{H(m^i, m^k)\} \end{array} \right\} . \tag{10}$$

Thus, our goal can be enunciated as maximizing the minimum of a set of Hamming distances, some of which are self-distances and the others are distances between pair of

markers. The Hamming distance between two markers can be expressed as:

$$H(m^i, m^j) = \sum_{k=1}^{n \times n} m_k^i \otimes m_k^j, \tag{11}$$

being $\otimes$ the exclusive-or operator. Since this is a non-linear operation, it must be reformulated as a linear one in order to be represented in a MILP model. This is accomplished by introducing, for each exclusive-or operation, a new auxiliary binary decision variable, δ, and the following set of constraints:

$$\begin{aligned} & m_k^i \otimes m_k^j = m_k^i + m_k^j - 2\delta \\ & \delta \leq m_k^i \\ & \delta \leq m_k^j \\ & \delta \geq m_k^i + m_k^j - 1. \end{aligned} \tag{12}$$

Finally, since the objective function is the minimum of a set of values, a new auxiliary decision variable, τ^*, is added to represent the minimum of all the Hamming distances (Eq. 10). The proposed problem formulation is then defined as:

$$\begin{aligned} & \text{maximize } \tau^* \\ & \text{Subject to, } \forall m^i \in \mathfrak{D}, \\ & \text{(I)} \quad H(m^i, m^k) - \tau^* \geq 0 \quad \forall m^k \in \mathcal{A}(m^j),\ \forall m^j \in \mathfrak{D},\ j \neq i \\ & \text{(II)} \quad H(m^i, m^k) - \tau^* \geq 0 \quad \forall m^k \in \mathcal{A}'(m^i) \\ & \text{(III)} \quad 0 \leq \tau^* \leq \tau_{max}^n \\ & \text{(IV)} \quad \mathcal{I}(m^i) - \mathcal{I}(m^k) \geq 0 \quad \forall m^k \in \mathcal{A}'(m^i) \\ & \text{(V)} \quad \mathcal{I}(m^i) - \mathcal{I}(m^{i+1}) \geq 0 \quad m^{i+1} \in \mathfrak{D} \\ & \text{(VI)} \quad \sum_{m^j \in \mathfrak{D}} \sum_{k=0}^{n \times n} m_k^j \geq \frac{dn^2}{2}, \end{aligned} \tag{13}$$

so that the minimum distance τ^* is maximized, ensuring that every Hamming distance is larger or equal to this value. Note that after the optimization process, the variable τ^* will contain the value of $\tau(\mathfrak{D})$.

Constraint (I) guarantees that the distance between any pair of markers in the dictionary is greater than or equal to τ^*, while constraint (II) guarantees the same condition for all the self-distances. Note that the previous model is a simplified version since each Hamming distance is represented by a sum of exclusive-or operations (which include the decision variables associated to the marker bits, see Eq. 11). Furthermore, each exclusive-or operation requires the addition of the inequalities in Eq. 12 and the auxiliary variables δ. We have decided not to represent all these auxiliary information in Eq. 13 for the sake of clarity.

Constraint (III) defines an upper bound for the value of τ^*. This bound is the maximum inter-marker distance τ_{max}^n, which is theoretically obtained in Sec. 3.1. However, if an optimal dictionary with fewer markers has been previously generated, its optimal objective value can be employed as upper-bound, since it is not possible to surpass the optimal solution of a dictionary with fewer markers. This modification accelerates the convergence process.

Finally, in order to reduce the search space, constraints (IV), (V) and (VI) are added to remove symmetric solutions. Note that for an optimal solution of the model, another optimal solution can be obtained by just rotating any of the markers (i.e., selecting another marker from its analogous set $\mathcal{A}(m^i)$). To avoid this, only the markers with the highest encoded value in the analogous sets are considered. This is achieved by constraint (IV), where $\mathcal{I}(m^i)$ represents the number encoded by the marker bits:

$$\mathcal{I}(m^i) = \sum_{k=1}^{n \times n} 2^{k-1} m_k^i. \tag{14}$$

Another symmetry arises from the fact that a dictionary is a vector of markers, thus, permutations of its elements lead to equivalent solutions. Constraint (V) is added to avoid this symmetry by forcing an strict ascending order of the markers.

Finally, an optimal solution can be converted into another one by just inverting all its bits. To avoid this symmetry, constraint (VI) forces that the total number of ones has to be higher than or equal to the total number of zeros. Thus, only one of the two opposite solutions is valid. The parameter d is the cardinal of the dictionary.

4.3. Suboptimal Dictionary

The previous model achieves the optimal inter-marker distance results. However, as shown in Sec. 5, the convergence times are too long despite the efforts made to reduce the search space. As a consequence, it can only be applied to generate dictionaries with relatively small number of markers and bits. In this section, an alternative formulation that obtains suboptimal results in much less time is proposed.

In this case, instead of generating all the markers in a single MILP optimization step, an iterative method, where markers are generated incrementally, is proposed. At each iteration, t, a new MILP model is defined to generate the next marker m^t, that maximizes its distance to all the previously generated markers and its self-distance. As in the previous case, the decision variables of this model are the marker bits. Then, the objective function at the t-th iteration is defined as:

$$\begin{aligned} & \tau(\mathfrak{D}_t) = \min\left\{\mathcal{S}(m^t), \min_{m^i \in \mathfrak{D}_{t-1}}\{D(m^t, m^i)\}\right\} = \\ & \min\left\{\min_{m^k \in \mathcal{A}'(m^t)} H(m^t, m^k), \min_{\substack{m^k \in \mathfrak{D}_{t-1} \\ m^i \in \mathcal{A}(m^k)}}\{H(m^t, m^k)\}\right\}. \end{aligned} \tag{15}$$

Once again, our goal is equivalent to the maximization of the minimum of a set of Hamming distances, some of them related to the new marker self-distance and the rest related to its distance to the previous markers. The self-distances can be expressed using the same transformation described for the previous model (Eq. 12). However, this transformation is not required to calculate the distance between two markers since only the bits of one of them are variables. As a consequence, the transformation in Eq. 12 can be reformulated as:

$$m_k^i \otimes m_k^j = \begin{cases} m_k^i, & \text{if } m_k^j = 0 \\ 1 - m_k^i, & \text{otherwise} \end{cases}, \tag{16}$$

where m_k^i is an unknown bit represented by a decision variable and m_k^j is a bit from a marker in $\mathfrak{D}_{t-1}$.

The proposed model is similar to the previous one, including the τ^* variable which represents the $\tau(\mathfrak{D})$ value. However, in this case the number of involved Hamming distances is smaller and so is the total number of constraints. The proposed MILP formulation is as follows:

$$\begin{aligned} &\text{maximize } \tau^* \\ &\text{Subject to} \\ &\text{(I)} \quad H(m^t, m^k) - \tau^* \geq 0 \quad \forall m^k \in \mathcal{A}'(m^t) \\ &\text{(II)} \quad H(m^t, m^k) - \tau^* \geq 0 \quad \forall m^k \in \mathcal{A}(m^i),\ \forall m^i \in \mathfrak{D}_{t-1} \\ &\text{(III)} \quad 0 \leq \tau^* \leq \tau(\mathfrak{D}_{t-1}) \\ &\text{(IV)} \quad \mathcal{I}(m^t) - \mathcal{I}(m^k) \geq 0 \quad \forall m^k \in \mathcal{A}'(m^t). \end{aligned} \tag{17}$$

Similarly to the optimal model, constraint (I) guarantees that the marker distance (Eq. 4) between the new generated marker, m^t, and any marker previously generated is greater than or equal to τ^*. Constraint (II) guarantees the same condition for the self-distance of m^t.

The upper bound of τ^* in constraint (III) refers to the fact that the maximum cost of a solution is lower than or equal to the maximum cost of the previous dictionary. This assumption is only valid if the previous markers have been created using the same suboptimal iterative method. For the first generated marker, the value τ_{max}^n (Sec. 3.1) can be employed for constraint (III). Finally, constraint (IV) is the same than in the optimal model, i.e., selecting the marker with highest encoded value from its analogous set.

The proposed method works iteratively, i.e., a new MILP model is generated and solved to obtain a new marker at each iteration. This process repeats until the desired number of markers, d, is generated. It must be noted that, at each iteration, the optimal solution may not be unique, and that the selection of a solution will condition the subsequent iterations. As a result, the marker dictionary obtained by this method is not optimal, nor unique, contrary to the previous model. However, the processing time spent by a MILP solver is much shorter compared to the optimal formulation and the results, as shown in Sec. 5, are still remarkable. *The whole process is summarized in Algorithm 1* .

Algorithm 1 *Suboptimal dictionary generation.*

```
1: 𝔇_0 ← ∅    # Empty dictionary
2: for t from 1 to d do
3:     Generate MILP Model for 𝔇_t    # See Model in Eq. 17
4:     m^t ← Solve MILP Model    # Get optimal marker
5:     𝔇_t ← 𝔇_{t-1} ⋃ m^t    # Add to previous markers
6: end for
7: Return last dictionary, 𝔇_d
```

Additionally, in order to ensure the convergence within a restricted amount of time, a time limit can be set to each MILP model. Thus, if the optimization is not finished when the limit is reached, the best feasible solution obtained at that moment is selected. This is a common strategy in Mixed Integer Programming to guarantee convergence when suboptimal solutions are allowed.

5. Experiments and results

This section shows the results of the experimentation carried out to validate our proposals. First, the optimal formulation is studied. Then, the suboptimal model is analyzed in terms of inter-marker distances and generation times. The obtained results have been compared to those produced by the best alternatives in the literature, *ArUco [12], AprilTags [13], ARTag [10] and ARToolKitPlus [11].*

The Gurobi optimizer v5.6 [40] has been chosen to solve the MILP models since it is the solver achieving the fastest convergence times for our models. The default Gurobi configuration has shown to be adequate for our proposals, since most of the parameters are configured automatically based on the model characteristics. All tests were performed using the twelve cores of a system equipped with an Intel Core i7-3930K 3.20 Ghz processor, 16 GB of RAM and Ubuntu 14.04 as operating system with a load average of 0.1.

It must be indicated that the generated dictionaries by our proposals have been set publicly available as a part of the ArUco library [12].

5.1. Optimal formulation

The generation of a marker dictionary is an off-line process which is typically performed only once. As a consequence, the generated time is not a critical aspect of the process. However, the main problem of the proposed optimal model is that its generation times are too long for high number of markers and number of bits. Note that the search space for the optimal model is $2^{n \times n \times d}$ (being d the number of generated markers) which indicates an exponential growth.

From our experimentation, it has been observed that for markers of sizes bigger than 5×5 bits, the convergence times are too long to study the results, thus, our experimentation has been restricted to smaller sizes. Even for marker sizes of 3×3 and 4×4 bits, we have been limited to dictionaries of 37 and 8 markers respectively.

Figure 5 shows the generation times for the optimal model. For each dictionary size, the $\tau(\mathfrak{D})$ value obtained

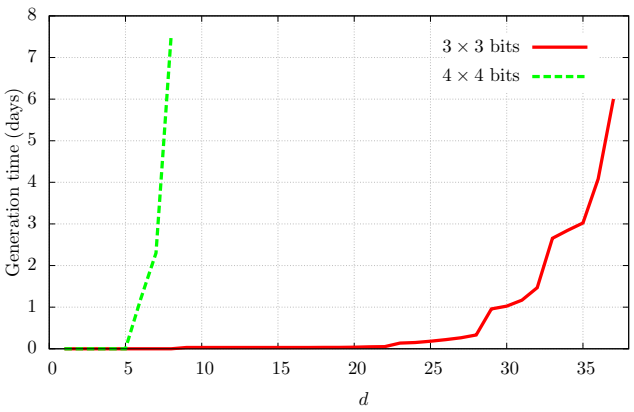


Figure 5: Generation times for the optimal formulation proposal as a function of the dictionary size for 3×3 and 4×4 bits. As it can be observed, the generation times increase considerably with the dictionary size and the number of bits. A formal study for bigger marker sizes or number of markers is not feasible.

in the previous generation was used as an upper bound of the objective function, as it is explained in the model description in Sec. 4.2.

It can be observed that the convergence times are indeed considerably long. For instance, the generation of an optimal dictionary of 37 markers and 3×3 bits lasted 6 days, and the generation of a dictionary of only 8 markers and 4×4 bits lasted more than 7 days. Due to this time limitation, we have not been able to study the optimal dictionaries for a higher number of bits or dictionary sizes, neither to compare the optimal results with the rest of methods. Nevertheless, it must be noted that the formulation is suitable for those applications where the required number of markers and marker size are not too high, keeping in mind that dictionary generation is an off-line process which is necessary to perform only once.

Furthermore, these long times justify the suboptimal model proposal which converges notably faster, allowing the generation of bigger dictionaries, both in number of markers and bits.

5.2. Suboptimal formulation

5.2.1. Analysis of dictionary distances

This section compares the distances obtained by the suboptimal formulation with those obtained by the *ArUco, AprilTags, ARTag and ARToolKitPlus methods*. To that end, dictionaries with up to 250 markers have been generated, covering in our opinion the requirements of most fiducial marker applications. The marker sizes have been selected from a range of sizes from 4×4 to 25×25 bits.

A time restriction of 150 seconds has been set to solve each MILP model in order to ensure the convergence of the optimization process within restricted time. Once the limit is reached, the best feasible solution at that moment is selected.

The ArUco method is an iterative process which employs an objective distance value. This value is decremented after an amount of unproductive iterations. To compare the results in the same conditions, the ArUco method was also configured to decrement this value after 150 seconds of unproductive iterations.

In the AprilTags method, the objective distance has to be specified by the user and its method does not propose any specific condition to reduce this value. Thus, we have employed the same condition than in the ArUco case, i.e. reducing the objective distance after 150 seconds of unproductive iterations.

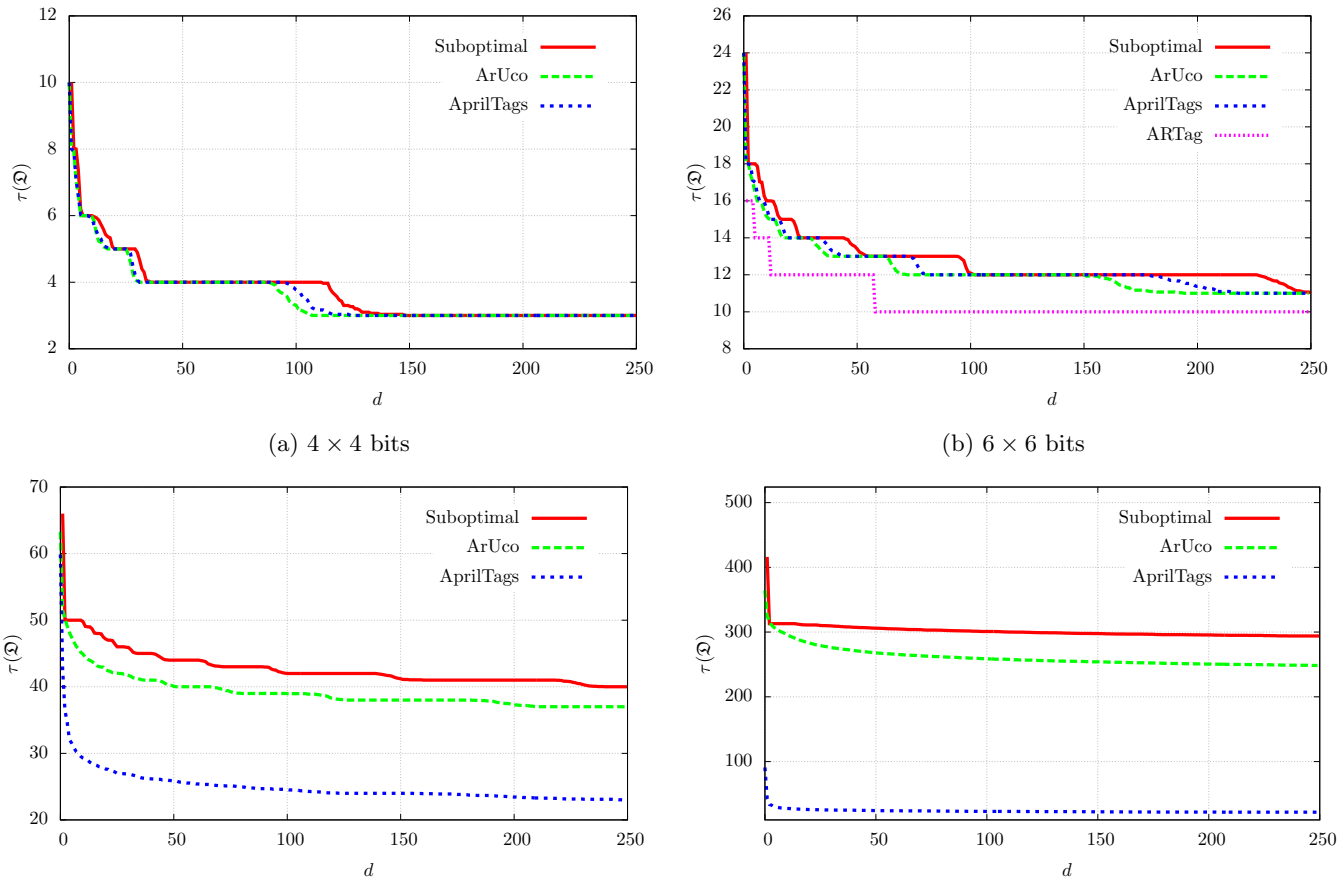
The marker dictionaries of ARTag and ARToolKitPlus are fixed and their markers are composed by 6×6 bits. Thus, they cannot been compared for different marker sizes. In the ARTag case, different dictionary sizes are obtained by taking the specific subset of markers in the order recommended by the authors. On the other hand, ARToolKitPlus does not provide a recommended order and hence, its dictionary size is also fixed.

Figure 6 shows the mean $\tau(\mathfrak{D})$ value for 30 executions as a function of the dictionary size and for different marker sizes. The results of the different executions only present deviations in the intervals where the objective distance is reduced, which correspond to the slopes of the curves. In the flat regions, there is no deviation and the same result is achieved for all the executions.

As it can be observed, the proposed suboptimal method outperforms the results of all the other proposals. For the smallest size, 4×4 bits, there are not remarkable differences since the search space is smaller. However, the improvements increase with the marker size. For marker sizes of 6×6 bits and bigger, the suboptimal method achieves results which clearly surpass the other methods. For instance, for a dictionary composed by 22 markers of 10×10 bits, the suboptimal model achieves a $\tau(\mathfrak{D})$ value of 47 while the second best alternative, the ArUco method, achieves a value of 42. This implies that the suboptimal dictionary can correct up to 23 erroneous bits whereas the ArUco method can only correct up to 20 bits, a difference of 3 bits. For 25×25 markers, the difference increases to 15 bits.

It is also remarkable how the results of the AprilTags method notably degrade as the marker size increases. This indicates that the employed strategy, which is based in an ascending order search, is not suitable for large search spaces.

The ARToolKitPlus library proposes two different dictionaries, also known as ARToolKitPlus Simple and ARToolKitPlus BCH. However, both of them are fixed and, contrary to ARTag, they do not provide a recommended order and, consequently, the inter-marker distance cannot be analyzed as a function of the dictionary size. Instead, we have compared against the ARToolKitPlus dictionaries by generating dictionaries with the same characteristics in terms of dictionary size and number of bits. ARToolKitPlus Simple dictionary is composed by 512 markers of 6×6 bits and achieves a $\tau(\mathfrak{D})$ value of 4, while the ARToolKitPlus BCH dictionary is composed by 4096 markers of 6×6 bits achieving a $\tau(\mathfrak{D})$



(a) 4×4 bits

(b) 6×6 bits

(c) 10×10 bits

(d) 25×25 bits

Figure 6: Minimum inter-marker distances $\tau(\mathfrak{D})$ for the suboptimal formulation and *the ArUco, AprilTags and ARTag methods*, as a function of the dictionary size for different number of bits. ARTag dictionary is only shown for 6×6 bits since its dictionary is fixed. It can be observed that the suboptimal model outperforms the results of the other methods.

	Dictionary Size	**Marker Size**	$\tau(\mathfrak{D})$			
			Original	**Suboptimal**	**ArUco**	**AprilTags**
ARToolKitPlus Simple	512 markers	6×6 bits	4	**11**	10	10
ARToolKitPlus BCH	4096 markers	6×6 bits	2	**9**	8	8

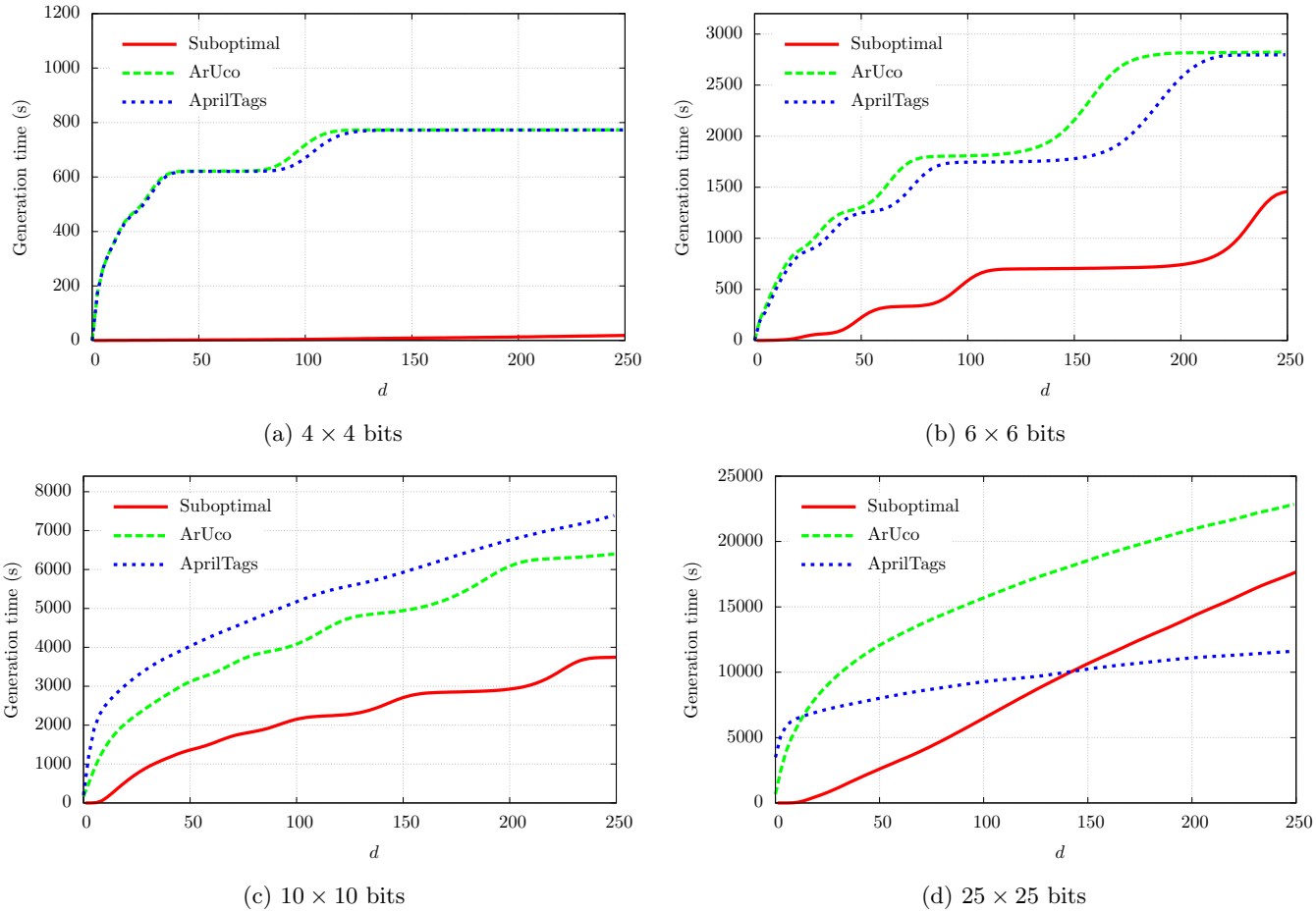
Table 3: *Inter-marker distance comparison between the dictionaries from the ARToolKitPlus library and those generated by ArUco, AprilTags and our suboptimal proposal with the same characteristics, i.e. same number of markers and bits. The column Original indicates the inter-marker distance of the original ARToolKitPlus dictionaries. The results of the original dictionaries are significantly lower than those obtained by the rest of methods. The results of our suboptimal proposal surpass the other approaches, allowing the correction of one more bit in comparison to ArUco and AprilTags.*

value of 2.

Table 3 shows the results obtained by the suboptimal, ArUco and AprilTags methods in comparison to the ARToolKitPlus dictionaries. It can be observed that the results of the original ARToolKitPlus dictionaries are considerably low in comparison to the rest of methods. For instance, in the case of ARToolKitPlus Simple, the original dictionary can perform an error correction of 1 bit, whereas the ArUco and AprilTags dictionaries can correct up to 4 bits due to the larger inter-marker distance of 10. However, our suboptimal proposal achieves a inter-marker distance of 11, allowing a maximum error correction of 5 bits and surpassing the rest of methods.

The same situation occurs for ARToolKitPlus BCH case, the original dictionary cannot perform error correction while ArUco and AprilTags can correct up to 3 bits due to an inter-marker distance of 8. Once again, our suboptimal approach surpasses the rest of methods achieving a maximum inter-marker distance of 9 and allowing error correction up to 4 bits.

The experimentation shows that the proposed suboptimal model produces dictionaries with the longest inter-marker distances in the literature, incrementing the error correction capabilities and only surpassed by the results of the optimal model, which is not applicable to a high number of markers and bits.



(a) 4×4 bits

(b) 6×6 bits

(c) 10×10 bits

(d) 25×25 bits

Figure 7: Generation times for the suboptimal, ArUco and AprilTags methods as a function of the dictionary size for different number of bits. The suboptimal times are, in general, shorter than the times of the other approaches, although the difference decreases as the marker size increase.

5.2.2. Generation Time

This section analyses the times employed by the suboptimal, ArUco and AprilTags methods to generate the dictionaries shown in the previous section. As for ARTag and ARToolKitPlus, the comparison is not feasible since their dictionaries are fixed and, hence, there is no generation process.

Figure 7 shows the mean generation times for 30 executions as a function of the dictionary size for different number of bits. As can be observed, the generation times are notably shorter compared to the optimal case. For instance, the generation of an optimal dictionary composed by 6 markers and 4×4 bits needs more than 1 day, while the suboptimal model employed less than 20 seconds to generate a dictionary of 250 markers with the same marker size.

Also, the generation times of the suboptimal method are, in general, shorter than those of ArUco and AprilTags. These differences are specially relevant for smaller marker sizes. *The generation times are only shorter in the AprilTags case for 25×25 bits. However, as it has been shown in Sec. 5.2.1, the inter-marker distances obtained by AprilTags in this case are completely unsatisfactory in comparison to those obtained by our proposal or the ArUco method.*

As for the ArUco and AprilTags results, it can be noted that there are some slopes in the plots, where the times increase sharply. This is especially remarkable for small marker sizes (4×4 and 6×6 bits). These peaks correspond to those dictionary sizes where the objective distance is decreased. Since the objective distance can only be reduced by reaching the time limit, the generation time increases considerably with each reduction.

On the other hand, the suboptimal proposal reduces the objective distance by adjusting their bounds during the MILP optimization based on the linear relaxation of the problem. This means that the time limit does not need to be reached every time the objective distance is reduced and it explains why the suboptimal times are significantly shorter than the times of the other approaches for 4×4 bits. However, as the marker size increases, the times of the suboptimal method start to grow and some sharply slopes appear (similarly to those on the ArUco or AprilTags curves). These peaks also correspond to the dictionary sizes where the time limit is reached. In these cases, the solver takes the best feasible solution found until that moment and continues

with the next generation.

Note that for the three methods, after the upper bound of the objective function is reduced, the next generated markers are less restricted and they can be generated faster, which explains the flat lines after each slope.

For the biggest marker size, 25×25 bits, there is a high number of objective distance reductions so that the slopes become less distinguishable in the three cases.

5.2.3. Comparison to optimal dictionary

In this section, the distances obtained by the suboptimal method are compared to those obtained by the optimal model. However, as it is shown in Section 5.1, the experimentation carried out with the optimal model is limited to small dictionaries and number of bits due to its time complexity. As a consequence, the results are not enough to draw any conclusion. Table 4 summarizes the distance results for dictionaries up to 8 markers and 4×4 bits. The distances of the suboptimal model correspond to the mean of 30 executions.

As it can be observed, the maximum difference between the distances of the optimal and suboptimal models is 2, although, as it has been stated, we cannot draw a conclusion due to the reduced number of results.

d	$\tau(\mathfrak{D})$	
	Optimal	**Suboptimal**
1	10	10
2	8	8
3	8	8
4	8	7.37
5	8	6.2
6	8	6
7	8	6
8	8	6

Table 4: Suboptimal distances compared to the optimal distances for 4×4 bits and dictionary size up to 8 markers.

6. Conclusions

This paper has proposed two novel methods to obtain fiducial marker dictionaries based on the Mixed Integer Linear Programming paradigm. The first model, contrary to any of the previous methods, guarantees the optimality of the dictionary in terms of inter-marker distance for any number of bits and markers. However, the generation times are too long for many practical situations. The second method, proposes an iterative formulation that, although does not guarantee optimality, achieves better results than the state-of-the art methods within restricted time.

As a consequence, the dictionaries generated with our proposals allow the detection and correction of a higher number of erroneous bits than previous approaches. *These results lead to a direct improvement in the marker detection process.*

Finally, it must be indicated that the generated dictionaries by our proposals have been set publicly available as a part of the ArUco library [12].

7. Acknowledgments

We are grateful to the financial support provided by Science and Technology Ministry of Spain and FEDER (projects TIN2012-32952 and BROCA).

References

[1] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, J. Tardós, A comparison of loop closing techniques in monocular SLAM, Robotics and Autonomous Systems (2009) 1188–1197.

[2] E. Royer, M. Lhuillier, M. Dhome, J.-M. Lavest, Monocular vision for mobile robot localization and autonomous navigation, International Journal of Computer Vision 74 (3) (2007) 237–260.

[3] R. T. Azuma, A survey of augmented reality, Presence 6 (1997) 355–385.

[4] H. Kato, M. Billinghurst, Marker tracking and HMD calibration for a video-based augmented reality conferencing system, in: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality, IWAR '99, IEEE Computer Society, Washington, DC, USA, 1999, pp. 85–94.

[5] V. Lepetit, P. Fua, Monocular model-based 3d tracking of rigid objects: A survey, in: Foundations and Trends in Computer Graphics and Vision, 2005, pp. 1–89.

[6] W. Daniel, R. Gerhard, M. Alessandro, T. Drummond, S. Dieter, Real-time detection and tracking for augmented reality on mobile phones, IEEE Transactions on Visualization and Computer Graphics 16 (3) (2010) 355–368.

[7] G. Klein, D. Murray, Parallel tracking and mapping for small AR workspaces, in: Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 1–10.

[8] K. Mikolajczyk, C. Schmid, Indexing based on scale invariant interest points., in: ICCV, 2001, pp. 525–531.

[9] D. G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2, ICCV '99, IEEE Computer Society, Washington, DC, USA, 1999, pp. 1150–1157.

[10] M. Fiala, Designing highly reliable fiducial markers, IEEE Trans. Pattern Anal. Mach. Intell. 32 (7) (2010) 1317–1324.

[11] D. Wagner, D. Schmalstieg, ARToolKitPlus for pose tracking on mobile devices, in: Computer Vision Winter Workshop, 2007, pp. 139–146.

[12] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, M. Marín-Jiménez, Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recognition 47 (6) (2014) 2280 – 2292.

[13] E. Olson, AprilTag: A robust and flexible visual fiducial system, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2011, pp. 3400–3407.

[14] K. Dorfmüller, H. Wirth, Real-time hand and head tracking for virtual environments using infrared beacons, in: in Proceedings CAPTECH'98. 1998, Springer, 1998, pp. 113–127.

[15] M. Ribo, A. Pinz, A. L. Fuhrmann, A new optical tracking system for virtual and augmented reality applications, in: In Proceedings of the IEEE Instrumentation and Measurement Technical Conference, 2001, pp. 1932–1936.

[16] V. A. Knyaz, R. V. Sibiryakov, The development of new coded targets for automated point identification and non-contact surface measurements, in: 3D Surface Measurements, International Archives of Photogrammetry and Remote Sensing, Vol. XXXII, part 5, 1998, pp. 80–85.

[17] L. Naimark, E. Foxlin, Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker, in: Proceedings of the 1st International Symposium on Mixed and Augmented Reality, ISMAR '02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 27–36.

[18] J. Rekimoto, Y. Ayatsuka, CyberCode: designing augmented reality environments with visual tags, in: Proceedings of DARE 2000 on Designing augmented reality environments, DARE '00, ACM, New York, NY, USA, 2000, pp. 1–10.

[19] M. Rohs, B. Gfeller, Using camera-equipped mobile phones for interacting with real-world objects, in: Advances in Pervasive Computing, 2004, pp. 265–271.

[20] E. Ouaviani, A. Pavan, M. Bottazzi, E. Brunelli, F. Caselli, M. Guerrero, A common image processing framework for 2d barcode reading, in: Image Processing and Its Applications. Seventh International Conference on (Conf. Publ. No. 465), Vol. 2, 1999, pp. 652–655.

[21] M. Kaltenbrunner, R. Bencina, reacTIVision: a computer-vision framework for table-based tangible interaction, in: Proceedings of the 1st international conference on Tangible and embedded interaction, TEI '07, ACM, New York, NY, USA, 2007, pp. 69–74.

[22] M. Fiala, Comparing ARTag and ARToolKit Plus fiducial marker systems, in: IEEE International Workshop on Haptic Audio Visual Environments and their Applications, 2005, pp. 147–152.

[23] J. Rekimoto, Matrix: A realtime object identification and registration method for augmented reality, in: Third Asian Pacific Computer and Human Interaction, Kangawa, Japan, IEEE Computer Society, 1998, pp. 63–69.

[24] W. Peterson, D. Brown, Cyclic codes for error detection, Proceedings of the IRE 49 (1) (1961) 228–235.

[25] S. Lin, D. Costello, Error Control Coding: Fundamentals and Applications, Prentice Hall, 1983.

[26] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnação, M. Gervautz, W. Purgathofer, The Studierstube augmented reality project, Presence: Teleoper. Virtual Environ. 11 (1) (2002) 33–54.

[27] D. Flohr, J. Fischer, A Lightweight ID-Based Extension for Marker Tracking Systems, in: Eurographics Symposium on Virtual Environments (EGVE) Short Paper Proceedings, 2007, pp. 59–64.

[28] A. Schrijver, Theory of Linear and Integer Programming, John Wiley & Sons, Inc., New York, NY, USA, 1986.

[29] R. Niemann, P. Marwedel, An algorithm for hardware/software partitioning using mixed integer linear programming, Design Automation for Embedded Systems 2 (2) (1997) 165–193.

[30] C.-W. Hui, Y. Natori, An industrial application using mixed-integer programming technique: A multi-period utility system model, Computers and Chemical Engineering 20, Supplement 2 (0) (1996) 1577–1582.

[31] H. Morais, P. Kádár, P. Faria, Z. A. Vale, H. Khodr, Optimal scheduling of a renewable micro-grid in an isolated load area using mixed-integer linear programming, Renewable Energy 35 (1) (2010) 151–156.

[32] T. Gönen, Distribution-system planning using mixed-integer programming, Generation, Transmission and Distribution, IEE Proceedings C 128 (1981) 70–79(9).

[33] A. Richards, J. P. How, Aircraft trajectory planning with collision avoidance using mixed integer linear programming, American Control Conference (ACC) 3 (2002) 1936–1941 vol.3.

[34] J. A. Nelder, R. Mead, A simplex method for function minimization, The computer journal 7 (4) (1965) 308–313.

[35] A. Schrijver, Theory of Linear and Integer Programming, John Wiley & Sons, Chichester, 1986.

[36] M. R. Garey, D. S. Johnson, Computers and intractability: a guide to the theory of NP-completeness. 1979, San Francisco, LA: Freeman.

[37] M. Padberg, G. Rinaldi, A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, SIAM review 33 (1) (1991) 60–100.

[38] E. L. Lawler, D. E. Wood, Branch-and-bound methods: A survey, Operations research 14 (4) (1966) 699–719.

[39] H. Marchand, A. Martin, R. Weismantel, L. Wolsey, Cutting planes in integer and mixed integer programming, Discrete Applied Mathematics 123 (1) (2002) 397–446.

[40] I. Gurobi Optimization, Gurobi optimizer reference manual, `http://www.gurobi.com` (2014).