sensors

MDPI

*Article*

# Reflection-Aware Generation and Identification of Square Marker Dictionaries

**Sergio Garrido-Jurado [1,*], Juan Garrido [2], David Jurado-Rodríguez [1,3], Francisco Vázquez [2] and Rafael Muñoz-Salinas [3]**

[1] Seabery R&D, Aldebarán Building, Córdoba Science and Technology Park, 14014 Córdoba, Spain
[2] Department of Electrical Engineering and Automation, Rabanales Campus, University of Córdoba, 14071 Córdoba, Spain
[3] Department of Computer Science and Numerical Analysis, Rabanales Campus, University of Córdoba, 14071 Córdoba, Spain
* Correspondence: sgj@seaberyat.com

**Abstract:** Square markers are a widespread tool to find correspondences for camera localization because of their robustness, accuracy, and detection speed. Their identification is usually based on a binary encoding that accounts for the different rotations of the marker; however, most systems do not consider the possibility of observing reflected markers. This case is possible in environments containing mirrors or reflective surfaces, and its lack of consideration is a source of detection errors, which is contrary to the robustness expected from square markers. This is the first work in the literature that focuses on reflection-aware square marker dictionaries. We present the derivation of the inter-marker distance of a reflection-aware dictionary and propose new algorithms for generating and identifying such dictionaries. Additionally, part of the proposed method can be used to optimize preexisting dictionaries to take reflection into account. The experimentation carried out demonstrates how our proposal greatly outperforms the most popular predefined dictionaries in terms of inter-marker distance and how the optimization process significantly improves them.

**Keywords:** square markers; fiducial markers; camera localization; object tracking; augmented reality



**Citation:** Garrido-Jurado, S.; Garrido, J.; Jurado-Rodríguez, D.; Vázquez, F.; Muñoz-Salinas, R. Reflection-Aware Generation and Identification of Square Marker Dictionaries. *Sensors* **2022**, *22*, 8548. https://doi.org/10.3390/s22218548

Academic Editor: Hyun Myung

Received: 30 August 2022
Accepted: 3 November 2022
Published: 6 November 2022



## 1. Introduction

Camera localization is a fundamental process in many computer vision applications, such as augmented reality [1,2], autonomous driving [3,4], or robotics [5,6]. This problem is usually tackled by finding correspondences between the real environment and their projections on the camera image, followed by a Perspective-n-Point (PnP) optimization [7] to estimate the 3D rotation and translation of the camera. The correspondences search can be performed using natural features, such as keypoints or textures [8–10], or using artificial markers that facilitate their detection and guarantee higher robustness, such as retroreflective spheres, VLC transmitters, LEDs, or planar markers [11–14].

Among the different types of flat markers, square ones are among the most widely used [15–17]. Square markers are composed of a black outline, which simplifies their detection in the image, and a central area for identification, usually a binary grid code that allows the application of error detection and correction techniques (see Figure 1).

Marker detection can be complex and error-prone due to different causes, such as inadequate illumination, occlusions, motion blur, or reflected observations. Some of these scenarios have been addressed in previous contributions [18,19]; however, the detection of reflected markers, i.e., seen through a mirror or reflective surface, has not received adequate attention despite being a potential source of errors. This work focuses on the detection of binary square markers in the presence of reflections.

Detection of square markers involves two steps [20,21]. The first one looks for candidates in the image, i.e., square shapes that resemble a marker, while the second one checks

whether candidates are markers or something else. To perform this identification, it is necessary to verify whether the code is part of the set of valid codes in the system. This set is known as the dictionary.

A key characteristic of a dictionary is its inter-marker distance [22], which is equivalent to the minimum Hamming distance between all its markers. This value is directly related to the error correction capability and, therefore, to the false negative rate and to the inter-marker confusion rate, i.e., rate of confusing one marker into another.

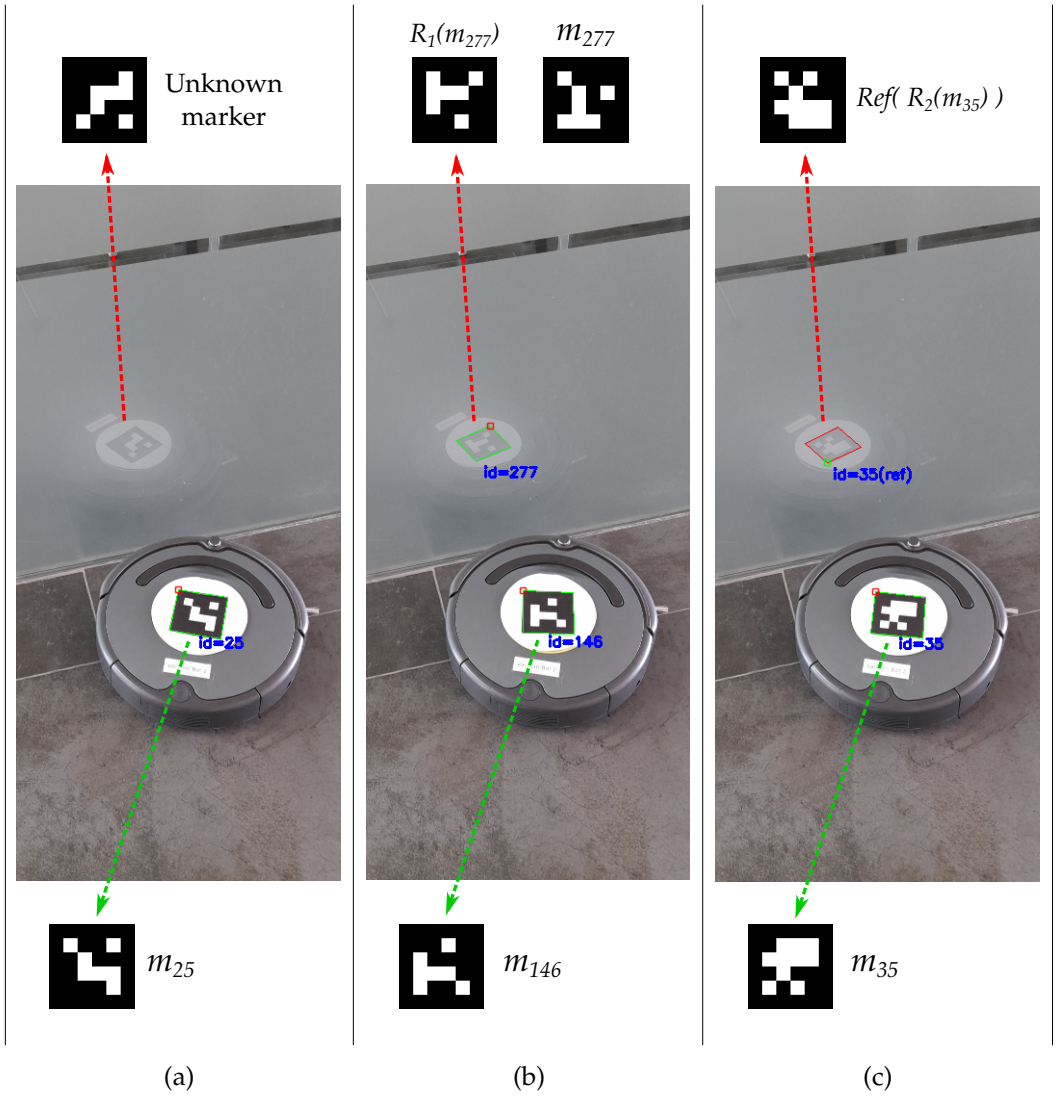


**Figure 1.** Examples of marker detection when observed through a reflective surface in a robot tracking system. In (**a**,**b**), a non-reflection-aware dictionary is used, in particular ArUco OpenCV $4 \times 4$ (see Table 1). Panel (**a**) shows a false negative since the reflected marker code (top) is not recognized within the dictionary. In (**b**) an inter-marker confusion error occurs because the reflected code of marker $m_{146}$ is identical to the code of marker $m_{277}$ after a rotation. Panel (**c**) employs one of the reflection-aware dictionaries generated by our proposal and the reflection-aware identification process. Thanks to that, it is possible to detect the marker even when it is reflected and also determine the kind of detection, i.e., direct or reflected.

The literature contains several proposals for square marker detection, usually including a public implementation. Almost all of these systems provide predefined dictionaries that cover most of the common situations. However, in some applications the predefined dictionaries are not appropriate and it is necessary to generate a custom one. Some reasons for the use of custom dictionaries include requiring a higher number of markers to cover a

large area [23], using a minimum dictionary size to maximize the inter-marker distance [24], using an unusual number of bits [25], or requiring custom binary patterns, for example, to maximize the number of bit transitions [26]. Therefore, some of the available systems also propose methods to generate custom dictionaries.

**Table 1.** Summary of the public dictionaries evaluated in the experimentation.

| Name | Library | Generation Method | Marker Size ($n \times n$) | Dictionary Size ($p$) | Reflection-Aware |
|---|---|---|---|---|---|
| ARTag | ARTag | Fixed | $6 \times 6$ | 1023 | Yes |
| ARToolKitPlus Simple | ARToolKitPlus | Fixed | $6 \times 6$ | 512 | No |
| ARToolKitPlus BCH | ARToolKitPlus | Fixed | $6 \times 6$ | 4096 | No |
| AprilTag 16h5 | AprilTag | [20] | $4 \times 4$ | 30 | No |
| AprilTag 25h7 | AprilTag | [20] | $5 \times 5$ | 242 | No |
| AprilTag 25h9 | AprilTag | [20] | $5 \times 5$ | 35 | No |
| AprilTag 36h10 | AprilTag | [20] | $6 \times 6$ | 2320 | No |
| AprilTag 36h11 | AprilTag | [20] | $6 \times 6$ | 587 | No |
| ArUco Original | ArUco | Fixed | $5 \times 5$ | 1024 | No |
| ArUco OpenCV $4 \times 4$ | ArUco (OpenCV) | [21] | $4 \times 4$ | 1000 | No |
| ArUco OpenCV $5 \times 5$ | ArUco (OpenCV) | [21] | $5 \times 5$ | 1000 | No |
| ArUco OpenCV $6 \times 6$ | ArUco (OpenCV) | [21] | $6 \times 6$ | 1000 | No |
| ArUco OpenCV $7 \times 7$ | ArUco (OpenCV) | [21] | $7 \times 7$ | 1000 | No |
| ArUco MIP 16h3 | ArUco | [22] | $4 \times 4$ | 250 | No |
| ArUco MIP 25h7 | ArUco | [22] | $5 \times 5$ | 100 | No |
| ArUco MIP 36h12 | ArUco | [22] | $6 \times 6$ | 250 | No |

ARTag [27] is one of the first libraries to propose binary-coded markers and the only one, to our knowledge, whose dictionary contemplates marker reflection, although it does not elaborate on this aspect. ARTag's dictionary uses a predefined CRC code [28] to encode the marker identifiers and does not allow the use of other custom dictionaries, i.e., with a user-defined number of markers or number of bits. A feature introduced by ARTag, which has been adopted in some later proposals, is to provide the dictionary in a specific order so that a smaller number of markers can be picked to maximize the inter-marker distance.

ARToolKitPlus library [29] is based on its predecessor, the well-known ARToolKit [15], but introduces some improvements, including the usage of binary encoding. In its early versions, it included a predefined dictionary, known as Simple, based on repeating a 9-bit code four times. In later versions, it provides a new, more robust dictionary that uses a BCH encoding [30].

Most modern libraries, besides providing some pregenerated dictionaries, also propose dictionary generation algorithms based on some heuristics with the objective of maximizing the inter-marker distance. AprilTag [20] proposes an iterative method in which each new marker is generated by searching for a code that satisfies a minimum inter-marker distance as well as a minimum geometric complexity to increase the number of bit transitions. One of the main drawbacks is that the generation time is significantly long. AprilTag's library contains several predefined dictionaries generated with this method.

The first versions of the ArUco library [31] included a predefined dictionary whose encoding was based on using five words of 5 bits, with 2 bits being for information and 3 bits being for redundancy. Its main drawback was that it did not account for marker rotation, resulting in a low inter-marker distance. In spite of this, this dictionary has been maintained as its use is still widespread.

In [21], ArUco authors propose a new heuristic method for generating customized dictionaries that considers the inter-marker distance. This method consists of an iterative process in which each new marker is generated based on a probability distribution that tries to maximize the inter-marker distance and the number of bit transitions. Its main disadvantage is that generation times and memory requirements are too high. The predefined dictionaries of the ArUco fork included in OpenCV [32] were generated using this method.

Finally, in [22], two methods are proposed to generate customized dictionaries using mixed integer programming (MIP) [33]. The former is the first proposal in the literature

that guarantees optimal dictionaries in terms of inter-marker distance; however, it suffers from high time and space complexity, making it impractical for generating dictionaries with marker sizes larger than $3 \times 3$ bits. The second proposal is a suboptimal method in which a new marker is generated in each iteration, maximizing the inter-marker distance relative to the previously generated markers. This proposal achieves the best inter-marker distances to date considering only rotation, although the generation times are high for large marker sizes. The main ArUco library [31] includes several predefined dictionaries generated with this method.

Most of the above methods and dictionaries attempt to maximize the inter-marker distance, which is an NP-complete problem [34]. This complexity is partly because a marker can be observed in any of its four rotations (see Figure 2a) and those rotations must be considered when calculating the marker distances.

Besides being rotated, a marker could also be reflected, which would modify the disposition of its internal code, as shown in Figure 2b. This situation occurs when a marker is seen through a mirror or reflective surface, as in Figure 1, creating a source of potential errors in the pose estimation if neglected. Although previous contributions have overlooked this possibility, it can arise depending on the type of environment and the control we have over it, i.e., absence of reflective surfaces cannot always be guaranteed. Considering that artificial markers are chosen mainly for their robustness, the reflected marker scenario should also be addressed to avoid errors.

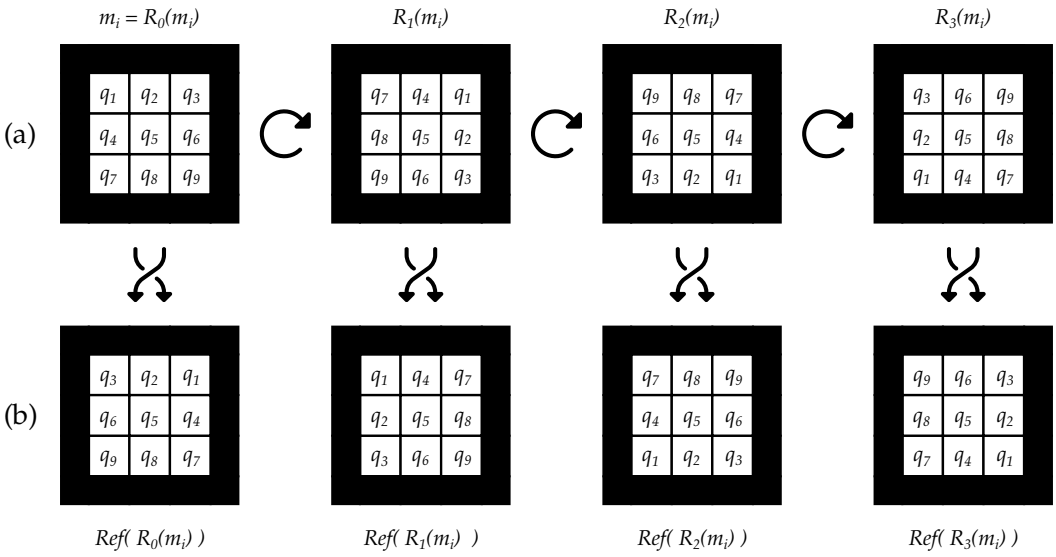


**Figure 2.** Bit permutations when applying the rotation and reflection transformations on a marker $m_i$ of $3 \times 3$ bits. (**a**) Original marker $m_i$ and its other three rotations after rotating 90, 180, and 270 degrees, respectively. (**b**) Resulting markers after applying reflection to each of the markers in (**a**). Note that reflection is applied with respect to the vertical axis but it could be performed with respect to the vertical or horizontal axis indifferently since it is applied to all 4 rotations and the result would be equivalent. The complete set of 8 markers corresponds to $\mathcal{A}(m_i)$, the first row corresponds to $\mathcal{A}_{rot}(m)_i$, the second row corresponds to $\mathcal{A}_{ref}(m_i)$, and the set of all markers except the original one, $m_i$, represents $\mathcal{A}_s(m_i)$ (see Equations (4) and (7)).

None of the previous systems, except ARTag's predefined dictionary, contemplate the possibility of marker reflection. Because of this, in this work, we present a new method for generating and detecting reflection-aware dictionaries, with this being the first work in the literature that delves into this topic.

The rest of this paper is structured as follows. Section 2 presents a mathematical formalization of the problem. Section 3 details the proposed generation and identification methods for reflection-aware dictionaries. Finally, Section 4 presents the experimentation carried out, and Section 5 draws some conclusions.

## 2. Problem Formulation

A dictionary is the set of valid markers that can be detected in an application. In the case of binary markers, the dictionaries vary depending on the number of bits and the encoding of each marker.

An advantage of using binary markers is that error detection and correction techniques can be applied to avoid false negatives. However, if the encoding of two or more markers in the dictionary is too similar, the system could become confused in the correction and misidentify one marker as another; this is known as inter-marker confusion error [27]. To avoid this, during dictionary generation we try to maximize the inter-marker distance, which is the minimum distance within a dictionary and is directly related to the maximum number of bits that can be corrected during the identification. In the following, we derive the calculation of the inter-marker distance using a similar approach to that of [21] but adding the condition of reflected markers.

First, we define a marker, $m_i$, of size $n \times n$ as a vector of bits (see Figure 2):

$$m_i = (q_1, q_2, q_3 \ldots, q_{n \times n}) \mid q_k \in \{0, 1\}. \tag{1}$$

where $i$ is the marker identifier. Note that to simplify the notation we treat a marker as a binary vector rather than as a matrix.

We define $\mathbb{D}$ as the set of all possible binary markers with size $n \times n$. A dictionary of markers, $\mathfrak{D}$, consisting of $p$ markers is an element of the set $\mathbb{D}^p$:

$$\mathfrak{D} = (m_1, m_2, m_3 \ldots, m_p) \in \mathbb{D}^p. \tag{2}$$

The goal, therefore, is to find a dictionary, $\mathfrak{D}$, that maximizes its inter-marker distance, $\tau(\mathfrak{D})$:

$$\mathfrak{D}^* = \underset{\mathfrak{D} \in \mathbb{D}^p}{\operatorname{argmax}} \{\tau(\mathfrak{D})\}. \tag{3}$$

Note that, when detecting a marker in an image, a marker may be rotated with respect to its original position and, hence, the extracted bits will also be rotated (see Figure 2a). Similarly, when a marker is reflected, the bits are also mirrored (see Figure 2b). When comparing the distance between two markers, we must consider all these possible transformations. We define the analogous set of a marker, $\mathcal{A}(m_i)$, as the set of markers obtained by applying the rotation and reflection transformations:

$$\begin{aligned} \mathcal{A}(m_i) &= \mathcal{A}_{rot}(m_i) \cup \mathcal{A}_{ref}(m_i), \\ \mathcal{A}_{rot}(m_i) &= \bigcup_{l=0}^{3} R_l(m_i), \\ \mathcal{A}_{ref}(m_i) &= \bigcup_{l=0}^{3} Ref(R_l(m_i)), \end{aligned} \tag{4}$$

where $\mathcal{A}_{rot}(m_i)$ and $\mathcal{A}_{ref}(m_i)$ correspond to the marker sets obtained from rotation and reflection transformations, respectively, $R_l$ applies a rotation of $l \times 90$ degrees clockwise to a marker, and $Ref$ reflects a marker. Both transformations consist of a permutation in the bit positions of the marker, as shown in Figure 2. A marker is considered equivalent to all markers in its analogous set, since it can be observed in any of its forms.

Hence, the distance between two markers considering all possible transformation is the minimum Hamming distance between any markers in each other's analogous sets:

$$d(m_i, m_j) = \min_{m_k \in \mathcal{A}(m_j)} \{\, ham(m_i, m_k) \,\}, \tag{5}$$

$ham()$ being the function that computes the Hamming distance between two markers.

Besides distinguishing between different markers, we often want to identify each of the marker corners unambiguously in order to perform a correct pose estimation. In other words, it is necessary to identify the specific rotation or reflection at which a marker has been detected. To achieve this, we must consider the distance between a marker and the rest of the elements of its own analogous set. We call this measure the marker self-distance:

$$d_s(m_i) = \min_{m_k \in \mathcal{A}_s(m_i)} \{\, ham(m_i, m_k) \,\}, \tag{6}$$

where $\mathcal{A}_s(m_i)$ is the analogous set of $m_i$ without considering the marker itself:

$$\mathcal{A}_s(m_i) = \mathcal{A}(m_i) - \{m_i\} \tag{7}$$

Finally, we can define the inter-marker distance of a dictionary, $\tau(\mathfrak{D})$, as the minimum distance between all self-distances and distances between pairs of markers in the dictionary:

$$\tau(\mathfrak{D}) = \min \left\{ \min_{\substack{m_i, m_j \in \mathfrak{D} \\ m_i \neq m_j}} \{d(m_i, m_j)\}, \min_{m_i \in \mathfrak{D}} \{d_s(m_i)\}, \right\}. \tag{8}$$

As defined in Equation (3), this is the measure we want to maximize when generating a dictionary since it is directly related to the error correction capability. The longer the distance, the further apart the markers are and the lower the probability of confusing one marker with another one when correcting bits. Specifically, the maximum number of bits that can be corrected in a dictionary without the danger of causing an inter-marker confusion error is $\lfloor (\tau(\mathfrak{D}) - 1)/2 \rfloor$.

## 3. Proposed Solution

This section presents the proposed method for generating reflection-aware dictionaries. The method is divided into two parts. In the first part, an initial dictionary is generated (Section 3.1), while in the second part, the previous dictionary is optimized by selecting a subset of markers to maximize the inter-marker distance (Section 3.2). Finally, we present the reflection-aware identification process for detecting markers in an image (Section 3.3).

### *3.1. Initial Dictionary Generation*

Our proposal starts generating an initial dictionary whose marker codes are as far from each other as possible. The generation of optimal dictionaries is an NP-complete problem [22]; hence, a heuristic approach is necessary. We propose an iterative algorithm that adds a new marker at each iteration until the desired dictionary size is reached.

In each iteration, the new marker is compared with the dictionary generated up to that moment. Therefore, it is necessary to define some concepts related to the distance between a marker and a dictionary. We call $\mathcal{H}(m_i, \mathfrak{D})$ to the multiset composed by all Hamming distances of a marker $m_i$ with respect to all markers in a dictionary $\mathfrak{D}$:

$$\mathcal{H}(m_i, \mathfrak{D}) = \bigcup_{\substack{m_j \in \mathfrak{D} \\ m_k \in \mathcal{A}(m_j)}} ham(m_i, m_k). \tag{9}$$

We further define $d_{\mathcal{H}}(m_i, \mathfrak{D})$ as the minimum distance within $\mathcal{H}(m_i, \mathfrak{D})$ and $f_{\mathcal{H}}(m_i, \mathfrak{D})$ to the multiplicity of this minimum distance, i.e., the number of times it is repeated in the multiset:

$$\begin{aligned} d_{\mathcal{H}}(m_i, \mathfrak{D}) &= min(\mathcal{H}(m_i, \mathfrak{D})), \\ f_{\mathcal{H}}(m_i, \mathfrak{D}) &= \sum_{\substack{h \in \mathcal{H}(m_i, \mathfrak{D}) \\ h = d_{\mathcal{H}}(m_i, \mathfrak{D})}} 1. \end{aligned} \tag{10}$$

At each iteration $t$, a new marker, $m_t$, is randomly initialized by assigning a 0 or a 1 to each bit $q_k$ with the same probability:

$$\left.\begin{matrix} P(q_k{=}0) = 0.5 \\ P(q_k{=}1) = 0.5 \end{matrix}\right\} \forall q_k \in m_t. \tag{11}$$

Then, a greedy algorithm is used to increase the distances of $m_t$ to the current dictionary, $\mathfrak{D}_t$. A greedy algorithm consists of taking small steps that improve the result until no step is found that produces a better solution. In our approach, each of these steps corresponds to a modification of a single bit of $m_t$. We call $m_t^b$ to the resulting marker after modifying the bit in position $b$ of $m_t$:

$$q_k = \begin{cases} \neg r_k & if\ k = b \\ r_k & otherwise \end{cases} \quad \forall q_k \in m_t^b \ \ and \ \ \forall r_k \in m_t \tag{12}$$

where $q_k$ represents the bits in $m_t^b$ and $r_k$ the bits in $m_t$. Among all the bits that can be modified at each step, we choose the one that maximizes the sum of distances in $\mathcal{H}(m_t, \mathfrak{D}_t)$:

$$b^* = \underset{b \in n \times n}{\text{argmax}} \{ \sum_{h \in \mathcal{H}(m_t^b, \mathfrak{D}_t)} h \ \}, \tag{13}$$

provided that, after the bit modification, the marker meets the following constraints that guarantee a minimum quality:

$$d_s(m_t^b) \geq d_{\mathcal{H}}(m_t^b, D_t), \tag{14}$$

$$d_{\mathcal{H}}(m_t^b, D_t) \geq d_{\mathcal{H}}(m_t, D_t), \tag{15}$$

$$d_{\mathcal{H}}(m_t^b, D_t) > d_{\mathcal{H}}(m_t, D_t) \ \vee \ f_{\mathcal{H}}(m_t^b, D_t) < f_{\mathcal{H}}(m_t, D_t). \tag{16}$$

The first condition guarantees that the self-distance of the new marker is not less than the distance to any other marker in the dictionary. The second condition ensures that the minimum distance to the dictionary does not decrease after the bit change. Finally, the third condition guarantees that, if the minimum distance has not increased, at least the number of its occurrences has decreased.

The bit modification process is repeated until there are no modifications that meet the above constraints. In that case, the marker cannot be improved any further, so it is added to the dictionary and the next iteration, $t + 1$, is started. The dictionary generation process continues until the desired number of markers has been reached. The proposed method pseudocode is shown in Algorithm 1.

**Algorithm 1** Initial dictionary generation.

**Input:** int $n$, int $p$ ▷ Marker size and dictionary size
**Output:** Dictionary $\mathfrak{D}_{out}$

$\mathfrak{D}_1 \leftarrow \emptyset$
**for** $t \leftarrow 1$ to $p$ **do**
  $m_t \leftarrow$ generate_random_marker($n$) ▷ See Equation (11)
  bool improvement_found $\leftarrow$ false
  **do**
    improvement_found $\leftarrow$ false
    int best_b $\leftarrow 0$
    int best_sum_hamming $\leftarrow 0$
    **for** $b \leftarrow 1$ to $n \times n$ **do**
      $m_t^b \leftarrow$ apply_bit_change($m_t, b$) ▷ See Equation (12)
      **if** $d_s(m_t^b) < d_{\mathcal{H}}(m_t^b, \mathfrak{D}_t)$ **then continue** ▷ See Equation (14)
      **if** $d_{\mathcal{H}}(m_t^b, \mathfrak{D}_t) < d_{\mathcal{H}}(m_t, \mathfrak{D}_t)$ **then continue** ▷ See Equation (15)
      **if** $d_{\mathcal{H}}(m_t^b, \mathfrak{D}_t) = d_{\mathcal{H}}(m_t, \mathfrak{D}_t)$ **and**
        $f_{\mathcal{H}}(m_t^b, \mathfrak{D}_t) \geq f_{\mathcal{H}}(m_t, \mathfrak{D}_t)$ **then continue** ▷ See Equation (16)
      int sum_hamming $\leftarrow 0$;
      **for each** $h \in \mathcal{H}(m_t^b, \mathfrak{D}_t)$ **do** ▷ See Equation (9)
        sum_hamming $\leftarrow$ sum_hamming + $h$
      **end for**
      **if** sum_hamming $>$ best_sum_hamming **then**
        best_sum_hamming $\leftarrow$ sum_hamming
        best_b $\leftarrow b$
        improvement_found $\leftarrow$ true
      **end if**
    **end for**
    **if** improvement_found **then** $m_t \leftarrow m_t^{best\_b}$
  **while** improvement_found
  $\mathfrak{D}_{t+1} \leftarrow \mathfrak{D}_t \bigcup m_t$
**end for**

### *3.2. Dictionary Optimization*

Although Algorithm 1 allows to generate a dictionary maximizing the distances between markers, it is not optimal and can produce low-quality markers that penalize the inter-marker distance. It is therefore advisable to generate more markers than required and apply a second step to select a subset of markers that maximize the inter-marker distance. This section explains the proposed method for marker selection.

The algorithm consists of an iterative process in which, in each iteration, we look for the optimal subset of markers for a specific value of inter-marker distance.

It starts with a large value of inter-marker distance, which is relaxed, i.e., decremented, on each iteration. In this manner, we begin by looking for small sets with large inter-marker distance, and as we progress we seek larger sets with smaller inter-marker distance. The process finishes when a subset with sufficient markers has been found or when all the markers in the dictionary have been selected. For the initial inter-marker distance, we use the largest self-distance of any marker in the dictionary since, by definition, it is always greater than or equal to the maximum theoretical inter-marker distance (see Equation (8)).

More precisely, for each iteration we have to solve the problem of, given an input dictionary $\mathfrak{D}$, obtaining the largest subset of markers, $\mathfrak{D}_u \subseteq \mathfrak{D}$, such that $\tau(\mathfrak{D}_u) \geq u$, where $u$ is the target inter-marker distance for that iteration.

To solve this problem we represent the dictionary as an undirected graph, $\mathcal{G}_u = (V_u, E_u)$. The graph will contain a vertex $v_i \in V_u$ for each marker $m_i \in \mathfrak{D}$ whose self-distance, $d_s(m_i)$, is greater than or equal to $u$. Furthermore, two vertices, $v_i$ and $v_j$, will be connected by an edge, $e_{i,j} \in E_u$, if the distance between their corresponding markers, $m_i$ and $m_j$, is greater than or equal to $u$:

$$\begin{aligned} V_u &= \{v_i \ : \ m_i \in \mathfrak{D} \ \wedge \ d_s(m_i) \geq u\}, \\ E_u &= \{e_{i,j} \ : \ m_i, m_j \in \mathfrak{D} \ \wedge \ d(m_i, m_j) \geq u\}. \end{aligned} \tag{17}$$

The solution to our problem is obtained by finding the maximum clique of the graph. A clique is a subset of vertices that are all connected to each other, which in our problem means that the distances between all markers in the clique are greater than or equal to $u$; in other words, $\tau(\mathfrak{D}_u) \geq u$. The maximum clique is the largest clique within a graph and thus the largest subset of markers with inter-marker distance greater than or equal to $u$.

Estimating the maximum clique is a classical problem in graph theory and there are numerous algorithms to solve it [35–37]. In our case, we have opted for the approximate coloring algorithm proposed in [38] due to its availability, simplicity, and efficiency.

The pseudocode of the dictionary optimization is shown in Algorithm 2.

**Algorithm 2** Dictionary optimization.

**Input:** Dictionary $\mathfrak{D}_{in}$, int target_dictionary_size
**Output:** Dictionary $\mathfrak{D}_{out}$ ▷ Optimized dictionary

```
u ← max{⋃_{m_i ∈ D_in} d_s(m_i)}        ▷ Maximum self-distance
while u ≥ 0 do
    G_u ← generate_graph(D_in, u)
    D_u ← find_maximum_clique(G_u)
    if |D_u| ≥ target_dictionary_size then
        D_out ← D_u
        return
    end if
    u ← u − 1
end while
```

Additionally, the optimization algorithm can also be applied to any predefined dictionary from public libraries. In this manner, they can be optimized by taking the subset of markers that maximizes the inter-marker distance when considering reflection. This is especially interesting for those applications that are already using a marker system and do not want to switch to a new dictionary. The results of these optimizations are shown in Section 4.3.

Finally, although this work focuses on marker reflection, it must be noted that the proposed algorithms, Algorithms 1 and 2, can be easily adapted to only consider rotation and ignore reflection. This only requires removing the reflection term, $\mathcal{A}_{ref}(m_i)$, from Equation (4).

### *3.3. Identification of Reflected Markers*

While each marker system proposes its own pipeline for analyzing an image and detecting markers, they typically share two fundamental steps. First, the image is processed to find candidates, i.e., square shapes with a black border that resemble a marker, and then each candidate is examined to verify whether it is a marker or not. As part of this second step, it is necessary to extract the bits of the marker from the image and determine if they belong to a valid dictionary code or not.

In libraries such as ArUco or AprilTag, the extracted code, $m_i$, is compared with all the markers in the dictionary, $\mathfrak{D}$, and it is identified as a valid marker if there is any marker in the dictionary, $m_j$, whose distance, $d(m_i, m_j)$, is less than or equal to $\lfloor(\tau(\mathfrak{D}) - 1)/2\rfloor$, which is the maximum number of bits that can be corrected without risking an inter-marker confusion error.

The novelty of our proposal is that it also accounts for reflection in the identification process. In particular, the algorithm finds the closest marker in the dictionary when estimating the distances $d(m_i, m_j)$ by considering the reflection analogous set, $\mathcal{A}_{ref}(m_j)$, in Equation (4).

Furthermore, not only can we identify a marker, but we can also distinguish whether it has been detected directly or reflected. To achieve this, it is sufficient to check whether the minimum Hamming distance between $m_i$ and $m_j$ is obtained with respect to the set $\mathcal{A}_{ref}(m_j)$, indicating reflected detection, or to the set $\mathcal{A}_{rot}(m_j)$, indicating direct detection. This information is necessary to identify each of the marker corners uniquely, but it can also be useful for other purposes. For example, an augmented reality application could display the virtual objects inverted in accordance with the reflected marker, or a robotic navigation application could ignore the reflected markers to avoid introducing potential errors in the trajectory estimation. Algorithm 3 summarizes the reflection-aware identification process.

**Algorithm 3** Reflection-aware marker identification.

**Input:** Dictionary $\mathfrak{D}$, Marker Candidate $m_i$
**Output:** bool is_marker, Identifier marker_id, bool is_reflected

**for each** $m_j \in \mathfrak{D}$ **do**
  **if** $d(m_i, m_j) \leq \lfloor (\tau(\mathfrak{D}) - 1)/2 \rfloor$ **then** ▷ See Equation (5)
    is_marker $\leftarrow$ true
    marker_id $\leftarrow j$
    **if** $d_{\mathcal{H}}(m_i, \mathcal{A}_{ref}(m_j)) < d_{\mathcal{H}}(m_i, \mathcal{A}_{rot}(m_j))$ **then** ▷ See Equation (10)
      is_reflected $\leftarrow$ true
    **else**
      is_reflected $\leftarrow$ false
    **end if**
    **return**
  **end if**
**end for**
is_marker $\leftarrow$ false

Figure 1 shows several cases of reflected marker detection. In the first two, Figure 1a,b, a non-reflection-aware dictionary is used, namely, the ArUco OpenCV $4 \times 4$ dictionary (see Table 1). In the first scenario, Figure 1a, a false negative is produced as the detection process does not consider reflection and the code of the reflected marker does not correspond to any marker in the dictionary. The second case, Figure 1b, is more critical as the code of the original marker ($m_{146}$), when reflected, is identical to that of another marker in the dictionary after being rotated ($m_{277}$), confusing the system and causing an inter-marker confusion error.

On the other hand, in Figure 1c, the same situation is presented with a dictionary generated by our proposal, which does consider reflection, and the identification process of Algorithm 3. In this case, the marker is correctly identified and, in addition, it is possible to distinguish whether it is reflected or not.

## 4. Experimentation and Results

This section details the experimentation carried out to validate our proposed dictionary generation compared to the public dictionaries of the most popular libraries for marker detection. Table 1 lists the dictionaries of other libraries against which we compared and some of their most relevant features. Note that the ArUco OpenCV dictionaries refer to those available in the OpenCV fork of ArUco [32] while ArUco MIP dictionaries refer to those available in the main ArUco library [31]. ArUco Original refers to the first ArUco dictionary, which is included in both implementations. Figure 3 shows the first marker of each of the dictionaries listed in Table 1.

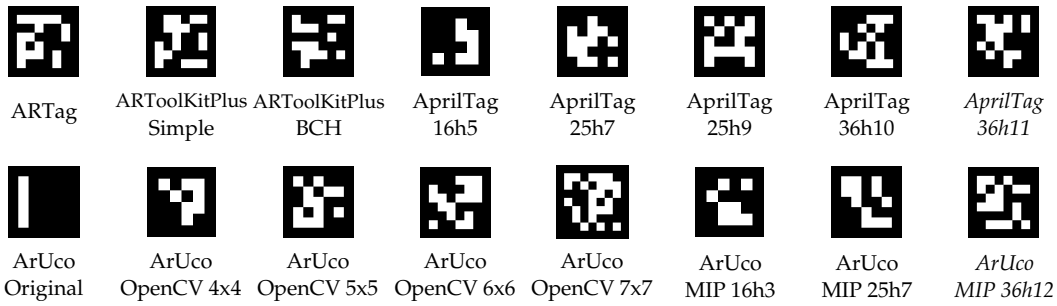


**Figure 3.** First marker of each of the dictionaries listed in Table 1.

Section 4.1 compares the inter-marker distance of the dictionaries generated by our proposal with those in Table 1. In Section 4.2, we study the generation times of our proposal. Finally, in Section 4.3, the dictionary optimization algorithm is applied to the dictionaries in Table 1 and the results are presented.

### *4.1. Inter-Marker Distance*

This section analyzes the inter-marker distance of the dictionaries generated by our proposal and the public dictionaries listed in Table 1.

To evaluate our proposal, we generated dictionaries with markers of the most common sizes, namely $4 \times 4$, $5 \times 5$, and $6 \times 6$ bits. Regarding the dictionary size, we consider that 250 markers are enough to cover most of the use cases and to compare with the rest of the dictionaries in the literature. In our experience, it is sufficient to generate six to eight times the desired number of markers with Algorithm 1. Therefore, for each size, a total of 2000 $(250 \times 8)$ markers were generated using Algorithm 1, and a subset of 250 markers was selected using Algorithm 2. For each size, 30 different dictionaries were generated and the results were averaged. We chose 30 samples, as this is the minimum number to assure a normal distribution according to the central limit theorem [39]. To limit the search time of the maximum clique algorithm, a timeout of 150 s was configured. After the timeout, the largest clique found up to that moment is returned.

The inter-marker distance was estimated for each dictionary size from 1 to 250 markers. For the public dictionaries in Table 1, subsets of markers up to size 250 were taken in increasing order. This is appropriate since some dictionaries provide a specific order that maximizes their inter-marker distance.

The inter-marker distance was calculated considering the formulation in Section 2, which introduces the reflection term in order to evaluate the error correction capability in the presence of reflection. Figure 4 shows the results.

It can be observed that the inter-marker distance of the dictionaries generated by our approach outperforms the other dictionaries in virtually all cases. This is expected, as the rest of the dictionaries are not reflection-aware; however, it is worth noting that the advantage is highly significant since the results are doubled in most cases. For example, if we take a dictionary of 50 markers of size $5 \times 5$ bits, we can observe that the maximum distance achieved by the rest of the dictionaries is two; this means that erroneous bits cannot be corrected ($\lfloor (\tau(\mathfrak{D}) - 1)/2 \rfloor = 0$) and it would only take two incorrect bits to produce an inter-marker confusion error. On the other hand, in that same case, our proposal achieves an inter-marker distance of seven, which means that it can correct up to three erroneous bits and it would need four incorrect bits for an inter-marker confusion error to occur.

The only dictionary that comes close to the results of our proposal is that of ARTag, as it is the only one that considers the possibility of reflection. Even so, our proposal outperforms ARTag in nearly all cases. Although it is difficult to appreciate, the only exception is for dictionary sizes between 52 and 57, where ARTag reaches an inter-marker distance of 12 and some executions of our proposal drop to 11. In any case, this only represents 0.0013% of the total number of executions, so we can state that our method clearly outperforms ARTag's dictionary. Apart from ARTag and our proposal, the best

performing dictionary families are AprilTag and ArUco MIP. The other ArUco dictionaries and, in particular, the ARToolKitPlus dictionaries show worse results.

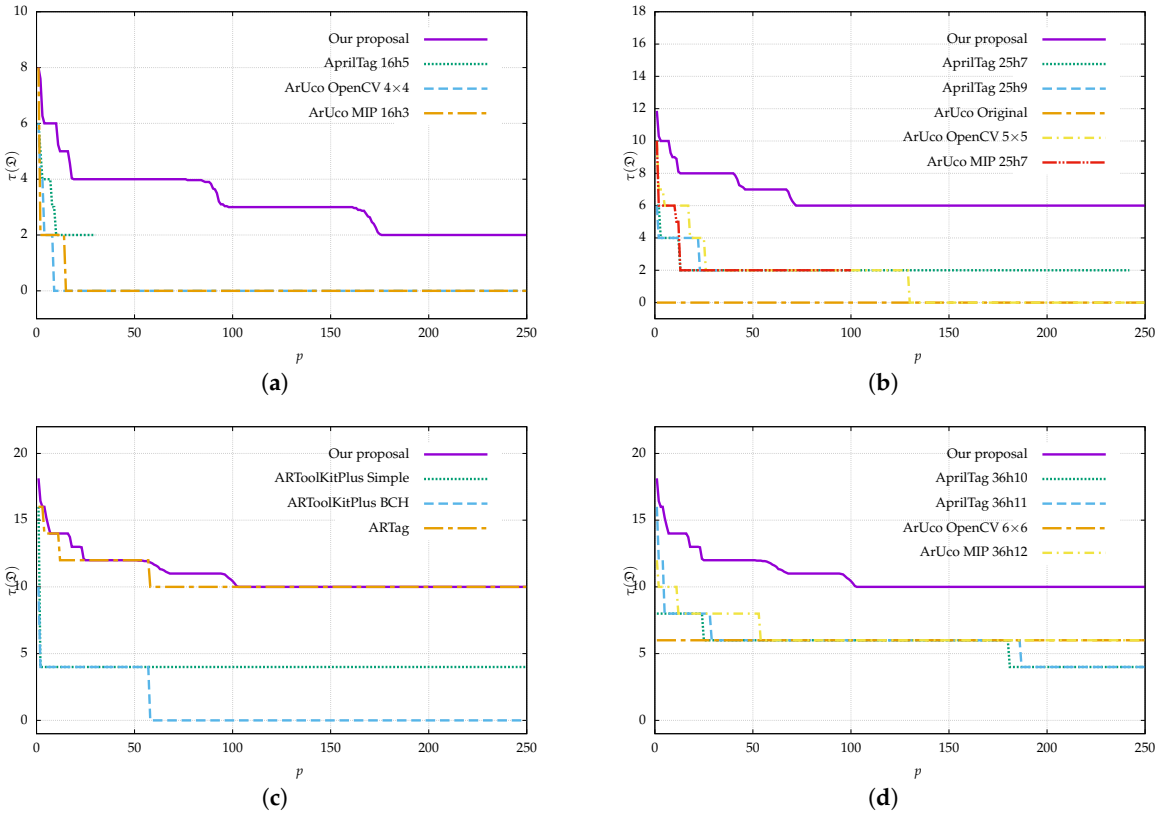


**Figure 4.** Inter-marker distance as a function of dictionary size for the proposed generation method compared to public dictionaries (see Table 1). The results are shown considering reflection and are broken down for different marker sizes. The $6 \times 6$ bit dictionaries are divided into two plots for easier visualization. Higher values indicate a larger inter-marker distance and, therefore, a higher error correction capability. Our proposal clearly outperforms all methods, including ARTag, which is the only one that considers reflection. (**a**) $4 \times 4$ bits, (**b**) $5 \times 5$ bits, (**c**) $6 \times 6$ (1/2), (**d**) $6 \times 6$ (2/2).

It is worth noting that some dictionaries have an inter-marker distance of 0. This is especially critical as it means that one marker could be confused with another simply by being reflected, even if there is no error in the extracted bits. Some examples are the ArUco Original, ARToolKitPlus BCH, or some of the ArUco OpenCV dictionaries. An example of this error can be seen in Figure 1b.

Most of the plots have a staircase shape in which each step down represents the decrease of the inter-marker distance as the dictionary size increases. In general, these steps are larger as we reduce the inter-marker distance requirement since it is easier to find more markers that meet that requirement. In addition, these steps are more frequent in our method and in ARTag since they are the only ones that consider reflection. In the rest of the dictionaries, the decrease of the inter-marker distance is faster and less progressive since they are not designed for reflection.

Finally, regarding marker size, we observe that, as expected, the inter-marker distance increases when using larger markers. In general, a large marker size allows for larger dictionaries and larger inter-marker distances. For example, using our proposal, a dictionary of 150 markers of $4 \times 4$ bits reaches an inter-marker distance of 3 while a dictionary of the same size and markers of $6 \times 6$ bits reaches a value of 10. On the other hand, a small marker size permits markers to be more easily detected in the image since the bits occupy more pixels. In the end, the choice of marker size will depend on the specific application

and aspects such as the working distance, the physical size of the marker, or the number of markers needed.

### *4.2. Generation Time*

In this section, we study the dictionary generation times of our proposal and compare them with the most popular dictionary generation algorithms in the literature. In particular, we compare against the AprilTag generation method [20], the ArUco generation method based on mixed-integer programming (ArUco MIP) [22], and the ArUco iterative generation method [21], which we refer to as ArUco OpenCV since the dictionaries of the ArUco fork in OpenCV were created using this approach. Note that these methods are the ones used to generate the custom dictionaries of Table 1. It is also noteworthy that ARTag, whose dictionary is the only one that considers reflection, just provides a predefined dictionary and not a method to generate custom dictionaries.

The generation was configured in a similar way to that proposed in [22]. All tests were performed using a system equipped with an Intel Core i7-3930K 3.20 GHz processor and 16 GB of RAM. The results of our proposal correspond to the dictionaries generated for the experimentation of Section 4.1. Figure 5 shows the generation times for two typical marker sizes of $4 \times 4$ and $6 \times 6$ bits.

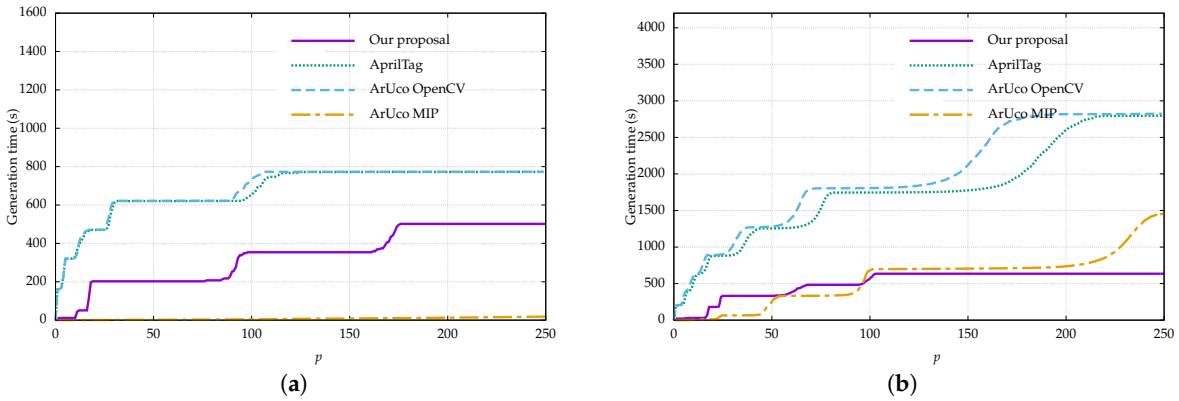


**Figure 5.** Generation times as a function of dictionary size for the proposed generation method compared to AprilTag, ArUco OpenCV, and ArUco MIP counterparts. Results are shown for marker sizes of (**a**) $4 \times 4$ and (**b**) $6 \times 6$ bits.

The methods with the shortest generation times are our proposal and ArUco MIP. For small marker sizes, i.e., $4 \times 4$ bits, ArUco MIP achieves the best results with generation times of a few seconds, while our method requires more than 8 min to generate a dictionary of 250 markers. On the other hand, for larger marker sizes, namely, $6 \times 6$ bits, our proposal has the shortest generation times, taking about 10 min, while the second best option, ArUco MIP, takes slightly more than 24 min. The slopes in the plots where the times increase sharply correspond to those cases where the timeout has been reached. It can be seen that the steps in the plots are larger as the dictionary size increases. This is because at each slope the inter-marker distance requirement is reduced, and it is easier and faster to find new markers that meet that constraint.

In any case, the dictionary generation is normally performed offline and only once; thus, the generation times are not critical as long as they are not prohibitive.

### *4.3. Optimizing Public Dictionaries*

This section evaluates the effect of the optimization algorithm from Section 3.2 when applied to the public dictionaries in Table 1. The goal is to check whether we can reuse existing dictionaries that a priori do not consider reflection by taking a subset of markers that maximizes the inter-marker distance. This is especially interesting for those applications

that intend to use a preexisting dictionary from a public library but still want to consider the possibility of reflection.

In addition to applying the optimization for reflection, we also evaluated the improvement considering only the possibility of rotation by eliminating the term $\mathcal{A}_{ref}(m)$ from Equation (4). This analysis is interesting since almost all the predefined dictionaries were generated considering only rotation, and we can verify if our optimization process is able to improve the dictionary even in this case.

Figures 6 and 7 show the inter-marker distance results for each dictionary and as a function of the dictionary size. For each case, the results are shown before and after applying the optimization process and considering reflection or only rotation. Optimization was applied to the full size of each dictionary. Figure 6 shows the results for all the dictionaries of the ArUco family, while Figure 7 shows the same for the rest of the dictionaries in Table 1.

It can be seen that, when considering reflection, we obtain better results in nearly all cases and in the rest we equal them, but we never obtain worse results. The improvement is almost always highly significant. For instance, the ArUco OpenCV $6 \times 6$ dictionary of size 800 achieves an inter-marker distance of 0 before optimizing, which makes it unusable since one marker can be confused with another simply by being reflected even if the bit extraction is perfect. After applying the optimization, an inter-marker distance of seven is achieved, which allows correcting up to three incorrect bits.

The only dictionary where the improvement is not as remarkable is the case of ARTag, since it already considers reflection. Still, we achieve equal or better results in all cases. Some improvements can be observed, for example, for dictionaries sizes close to 300.

Note that some dictionaries, such as ARToolKitPlus BCH and several dictionaries of the ArUco family, achieve an inter-marker distance of zero before optimization, which makes them unusable if we want to take reflection into account. After applying the optimization, almost none of the dictionaries reach a minimum distance of zero. The only exceptions are ArUco Original, ArUco OpenCV $4 \times 4$, and ArUco MIP 16*h*3; hence, it is not recommended to use these dictionaries when the input images might contain reflections, even after applying the optimization process.

When we only take rotation into account, the optimization results also improve or equal the inter-marker distance in most cases, although the improvement is not as significant as when reflection is considered. It only obtains worse results for some dictionary sizes in ArUco OpenCV, namely, $5 \times 5$, $6 \times 6$, and $7 \times 7$ bits. The reason for the worsening is that the maximum clique search has reached the timeout more frequently, resulting in a suboptimal result. In several cases, the shapes of the plots are similar before and after optimization. This occurs because those dictionaries are close to the optimum and have little room for improvement when considering only rotation, such as in ArUco OpenCV or ArUco MIP families.

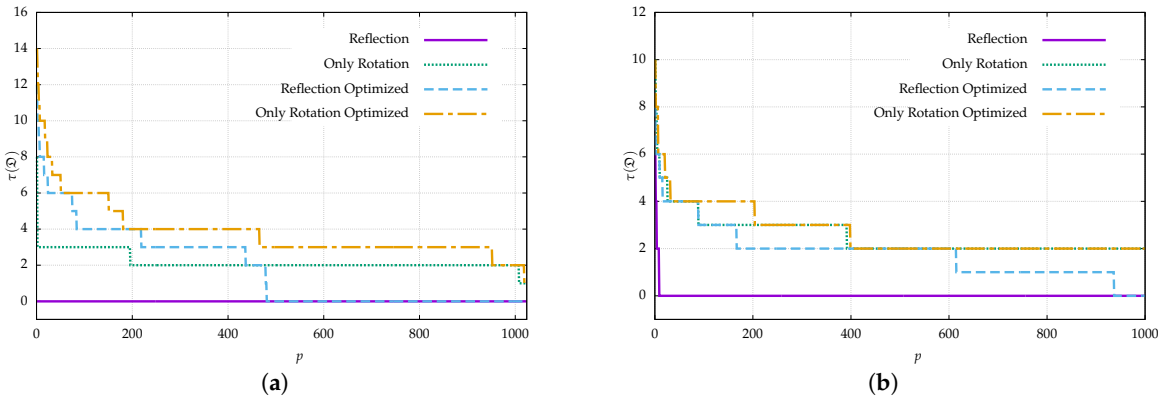


**Figure 6.** *Cont.*

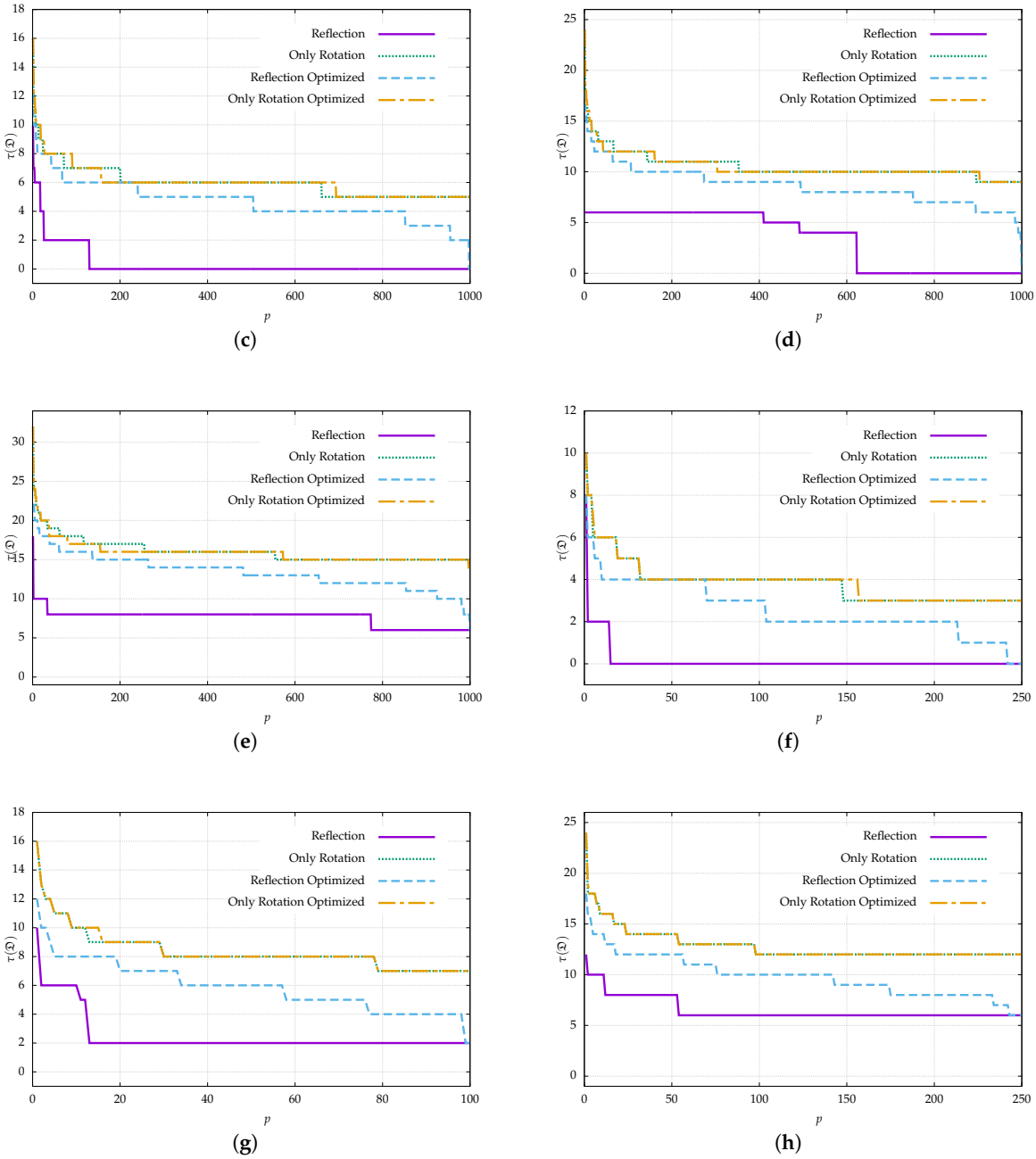


**Figure 6.** Inter-marker distance as a function of dictionary size for the ArUco family dictionaries (see Table 1) before and after applying the proposed optimization (Algorithm 2). Results are shown considering reflection as well as considering only rotation. When considering reflection, the optimized dictionaries obtain equal or better results in all cases. See Figure 7 for the rest of the dictionaries. (**a**) ArUco Original, (**b**) ArUco OpenCV 4 × 4, (**c**) ArUco OpenCV 5 × 5, (**d**) ArUco OpenCV 6 × 6, (**e**) ArUco OpenCV 7 × 7, (**f**) ArUco MIP 16h3, (**g**) ArUco MIP 25h7, (**h**) ArUco MIP 36h12.

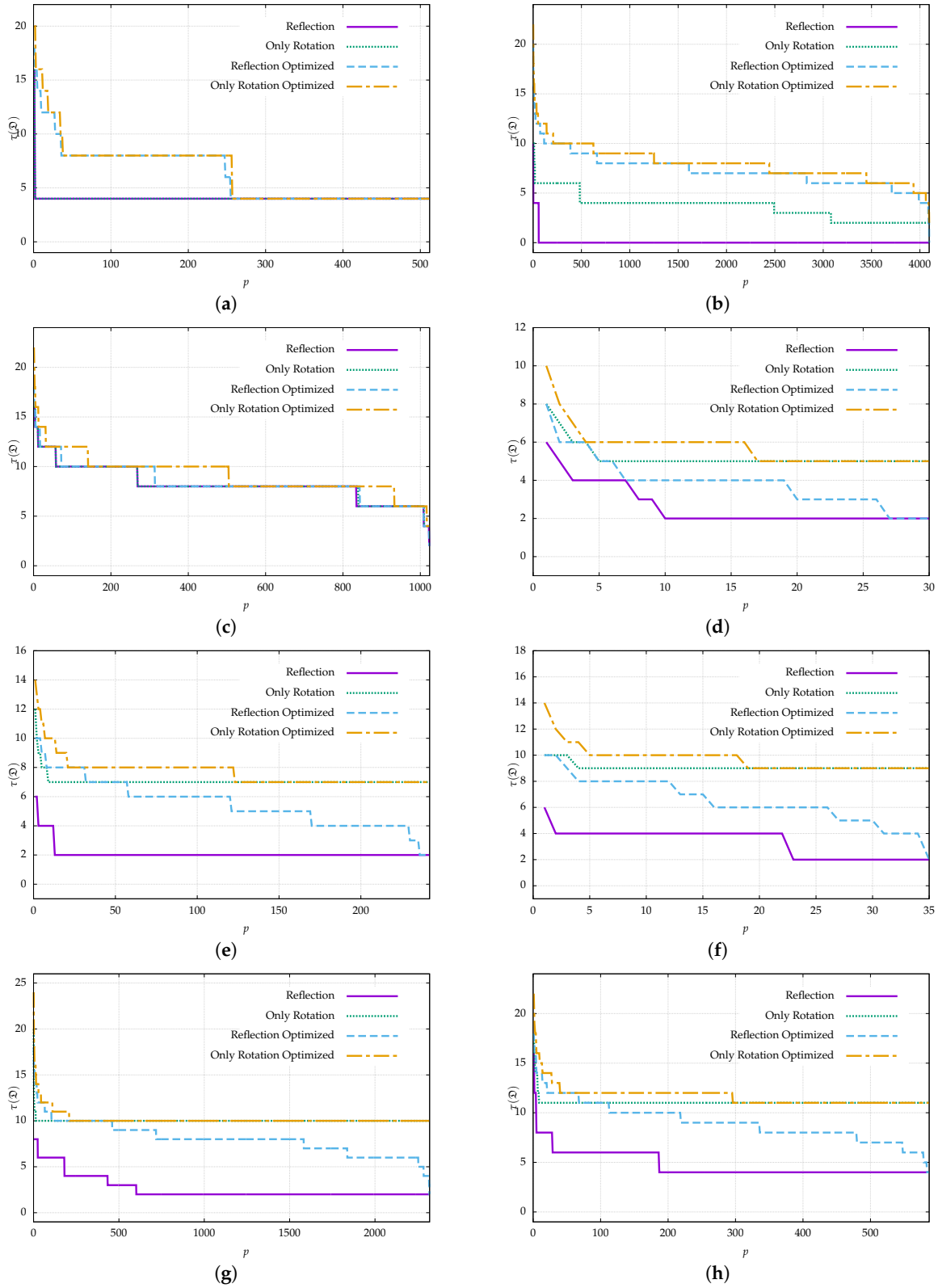


**Figure 7.** Inter-marker distance as a function of dictionary size for several of the public dictionaries in Table 1 before and after applying the proposed optimization (Algorithm 2). Results are shown considering reflection as well as considering only rotation. When considering reflection, the optimized dictionaries obtain equal or better results in all cases, even when comparing with the reflection-aware dictionary of ARTag. See Figure 6 for the rest of the dictionaries. (**a**) ARToolKit+ Simple, (**b**) ARToolKit+ BCH, (**c**) ARTag, (**d**) AprilTag 16h5, (**e**) AprilTag 25h7, (**f**) AprilTag 25h9, (**g**) AprilTag 36h10, (**h**) AprilTag 36h11.

## 5. Conclusions

This paper presents the first study specifically focused on the generation and detection of reflection-aware dictionaries of square markers.

First, we presented a new formulation for the inter-marker distance which considers the possibility of marker reflection. Secondly, a new method for generating reflection-aware dictionaries maximizing the inter-marker distance was proposed. As proved in the experimentation section, our proposal clearly outperforms all other tested dictionaries, doubling the minimum distance in almost all cases and, therefore, doubling the bit-correction capacity. Moreover, part of the proposed method consists of a dictionary optimization that can also be applied to preexisting dictionaries. Applying this method to the public dictionaries yields a significant improvement of the inter-marker distances. The optimization can also be applied considering only rotation and still improves most of the preexisting dictionaries. Furthermore, in conjunction with the dictionary generation process, we proposed an algorithm to identify reflection-aware markers that allows, in addition to identifying the marker, to report whether it has been detected as reflected or not.

Finally, the source code of the proposed algorithms has been made publicly available as Supplementary Material to this paper.

**Supplementary Materials:** An implementation of the proposed methods based on the OpenCV ArUco module is available at: https://github.com/sergarrido/opencv_contrib/tree/aruco_reflection (accessed on 30 August 2022).

**Author Contributions:** Conceptualization, S.G.-J., J.G., and F.V.; methodology, R.M.-S. and S.G.-J.; software, D.J.-R. and S.G.-J.; validation, R.M.-S. and J.G.; formal analysis, J.G. and F.V.; investigation, S.G.-J. and D.J.-R.; writing—original draft preparation, S.G.-J., J.G., and F.V.; writing—review and editing, R.M.-S., J.G., and F.V.; visualization, D.J.-R. and S.G.-J.; supervision, S.G.-J.; funding acquisition, S.G.-J. and R.M.-S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This project was funded under the Industrial PhD Program of Córdoba University with Seabery R&D.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Marchand, E.; Uchiyama, H.; Spindler, F. Pose estimation for augmented reality: A hands-on survey. *IEEE Trans. Vis. Comput. Graph.* **2015**, *22*, 2633–2651. [CrossRef] [PubMed]
2. Frikha, R.; Ejbali, R.; Zaied, M. Camera pose estimation for augmented reality in a small indoor dynamic scene. *J. Electron. Imaging* **2017**, *26*, 053029. [CrossRef]
3. Fu, J.; Pertuz, S.; Matas, J.; Kämäräinen, J.K. Performance analysis of single-query 6-DoF camera pose estimation in self-driving setups. *Comput. Vis. Image Underst.* **2019**, *186*, 58–73. [CrossRef]
4. Venator, M.; Bruns, E.; Maier, A. Robust camera pose estimation for unordered road scene images in varying viewing conditions. *IEEE Trans. Intell. Veh.* **2019**, *5*, 165–174. [CrossRef]
5. Ali, I.; Suominen, O.J.; Morales, E.R.; Gotchev, A. Multi-view camera pose estimation for robotic arm manipulation. *IEEE Access* **2020**, *8*, 174305–174316. [CrossRef]
6. Lee, T.E.; Tremblay, J.; To, T.; Cheng, J.; Mosier, T.; Kroemer, O.; Fox, D.; Birchfield, S. Camera-to-robot pose estimation from a single image. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 9426–9432.
7. Lu, X.X. A review of solutions for perspective-n-point problem in camera pose estimation. *Proc. J. Phys. Conf. Ser. IOP Pub.* **2018**, *1087*, 052009. [CrossRef]
8. Klein, G.; Murray, D. Parallel Tracking and Mapping for Small AR Workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; IEEE Computer Society: Washington, DC, USA, 2007; pp. 1–10.

9. Buch, A.G.; Kraft, D.; Kamarainen, J.K.; Petersen, H.G.; Krüger, N. Pose estimation using local structure-specific shape and appearance context. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 2080–2087.
10. Cheng, M.L.; Matsuoka, M. An Efficient and Precise Remote Sensing Optical Image Matching Technique Using Binary-Based Feature Points. *Sensors* **2021**, *21*, 6035. [CrossRef]
11. Ribeiro, L.G.; Suominen, O.J.; Durmush, A.; Peltonen, S.; Ruiz Morales, E.; Gotchev, A. Retro-reflective-marker-aided target pose estimation in a safety-critical environment. *Appl. Sci.* **2020**, *11*, 3. [CrossRef]
12. Zhang, W.; Kavehrad, M. Comparison of VLC-based indoor positioning techniques. *Proc. SPIE* **2013**, *8645*, 152–157.
13. Zhuang, Y.; Hua, L.; Qi, L.; Yang, J.; Cao, P.; Cao, Y.; Wu, Y.; Thompson, J.; Haas, H. A survey of positioning systems using visible LED lights. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 1963–1988. [CrossRef]
14. Benligiray, B.; Topal, C.; Akinlar, C. STag: A stable fiducial marker system. *Image Vis. Comput.* **2019**, *89*, 158–169. [CrossRef]
15. Kato, H.; Billinghurst, M. 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, San Francisco, CA, USA, 20–21 October 1999; IEEE Computer Society: Washington, DC, USA, 1999; pp. 85–94.
16. Wang, J.; Olson, E. AprilTag 2: Efficient and robust fiducial detection. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; pp. 4193–4198.
17. Romero-Ramirez, F.J.; Muñoz-Salinas, R.; Medina-Carnicer, R. Speeded up detection of squared fiducial markers. *Image Vis. Comput.* **2018**, *76*, 38–47. [CrossRef]
18. Mondéjar-Guerra, V.; Garrido-Jurado, S.; Muñoz-Salinas, R.; Marín-Jiménez, M.J.; Medina-Carnicer, R. Robust identification of fiducial markers in challenging conditions. *Expert Syst. Appl.* **2018**, *93*, 336–345. [CrossRef]
19. Li, B.; Wang, B.; Tan, X.; Wu, J.; Wei, L. Corner location and recognition of single ArUco marker under occlusion based on YOLO algorithm. *J. Electron. Imaging* **2021**, *30*, 033012. [CrossRef]
20. Olson, E. AprilTag: A robust and flexible visual fiducial system. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011; pp. 3400–3407.
21. Garrido-Jurado, S.; Muñoz-Salinas, R.; Madrid-Cuevas, F.; Marín-Jiménez, M. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognit.* **2014**, *47*, 2280–2292. [CrossRef]
22. Garrido-Jurado, S.; Muñoz-Salinas, R.; Madrid-Cuevas, F.J.; Medina-Carnicer, R. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognit.* **2016**, *51*, 481–491. [CrossRef]
23. Dujany, M. Localization of an Underwater Swimming Robot. 2018. Available online: https://www.epfl.ch/labs/biorob/wp-content/uploads/2019/02/final_report_semester_project_MatthieuDujany.pdf (accessed on 22 October 2022).
24. Mostashiri, N.; Dhupia, J.S.; Verl, A.W.; Xu, W. A Novel Spatial Mandibular Motion-Capture System Based on Planar Fiducial Markers. *IEEE Sens. J.* **2018**, *18*, 10096–10104. [CrossRef]
25. Wang, C.; Komninos, C.; Andersen, S.; D'Ettorre, C.; Dwyer, G.; Maneas, E.; Edwards, P.; Desjardins, A.; Stilli, A.; Stoyanov, D. Ultrasound 3D reconstruction of malignant masses in robotic-assisted partial nephrectomy using the PAF rail system: A comparison study. *Int. J. Comput. Assist. Radiol. Surg.* **2020**, *15*, 1147–1155. [CrossRef]
26. Schoun, B.; Oagaz, H.; Choi, M.H. Corner-based Square Fiducial Marker Detection for Hand-manipulated AR Objects. In Proceedings of the 2021 IEEE International Conference on Intelligent Reality (ICIR), Piscataway, NJ, USA, 12–13 May 2021; pp. 31–38.
27. Fiala, M. Designing highly reliable fiducial markers. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *32*, 1317–1324. [CrossRef]
28. Peterson, W.; Brown, D. Cyclic Codes for Error Detection. *Proc. IRE* **1961**, *49*, 228–235. [CrossRef]
29. Wagner, D.; Schmalstieg, D. ARToolKitPlus for Pose Tracking on Mobile Devices. In Proceedings of the Computer Vision Winter Workshop, St. Lambrecht, Austria, 6–8 February 2007; pp. 139–146.
30. Lin, S.; Costello, D. *Error Control Coding: Fundamentals and Applications*; Prentice Hall: Hoboken, NJ, USA, 1983.
31. Munoz-Salinas, R.; Garrido-Jurado, S.; Romero-Ramirez, F.J. ArUco: A Minimal Library for Augmented Reality Applications. Available online: https://www.uco.es/investiga/grupos/ava/portfolio/aruco/ (accessed on 15 August 2022).
32. ArUco Contrib Module in OpenCV. Available online: https://github.com/opencv/opencv_contrib/tree/4.x/modules/aruco (accessed on 15 August 2022).
33. Schrijver, A. *Theory of Linear and Integer Programming*; John Wiley & Sons, Inc.: New York, NY, USA, 1986.
34. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; Freeman: San Francisco, CA, USA, 1979.
35. Wu, Q.; Hao, J.K. A review on algorithms for maximum clique problems. *Eur. J. Oper. Res.* **2015**, *242*, 693–709. [CrossRef]
36. Guo, P.; Wang, X.; Zeng, Y.; Chen, H. MEAMCP: A membrane evolutionary algorithm for solving maximum clique problem. *IEEE Access* **2019**, *7*, 108360–108370. [CrossRef]
37. Rossi, R.A.; Gleich, D.F.; Gebremedhin, A.H. Parallel maximum clique algorithms with applications to network analysis. *SIAM J. Sci. Comput.* **2015**, *37*, C589–C616. [CrossRef]
38. Konc, J.; Janezic, D. An improved branch and bound algorithm for the maximum clique problem. *Proteins* **2007**, *4*, 590–596.
39. Kwak, S.G.; Kim, J.H. Central limit theorem: The cornerstone of modern statistics. *Korean J. Anesthesiol.* **2017**, *70*, 144–156. [CrossRef]