Automatic generation and detection of highly reliable fiducial markers under occlusion

S. Garrido-Jurado, R. Muñoz-Salinas, F.J Madrid-Cuevas, M.J. Marín-Jiménez
Department of Computing and Numerical Analysis.
University of Córdoba.
14071 Córdoba (Spain)
{i52gajus,rmsalinas,fjmadrid,mjmarin}@uco.es

Abstract

This paper presents a fiducial marker system specially appropriated for camera pose estimation in applications such as augmented reality, robot localization, etc. Three main contributions are presented. First, we propose an algorithm for generating configurable marker dictionaries (in size and number of bits) following a criterion to maximize the inter-marker distance and the number of bit transitions. In the process, we derive the maximum theoretical inter-marker distance that dictionaries of square binary markers can have. Second, a method for automatically detecting the markers and correcting possible errors is proposed. Third, a solution to the occlusion problem in augmented reality applications is shown. To that aim, multiple markers are combined with an occlusion mask calculated by color segmentation. The experiments conducted show that our proposal obtains dictionaries with higher inter-marker distances and lower false negative rates than state-of-the-art systems, and provides an effective solution to the occlusion problem.

1 Introduction

Camera pose estimation (Fig. 1(a,b)) is a common problem in many applications requiring a precise localization in the environment such as augmented and virtual reality applications, robotics, etc [1, 2, 3, 4]. Obtaining the camera pose from images requires to find the correspondences between known points in the environment and their camera projections. While some approaches seek natural features such as key points or textures [5, 6, 7, 8, 9], fiducial markers are still an attractive approach because they are easy to detect and allows to achieve high speed and precision.

Amongst the several fiducial marker systems proposed in the literature, those based on square markers have gained popularity, specially in the augmented reality community [10, 11, 12]. The reason why is that they allow to extract the camera pose from their four corners, given that the camera is properly calibrated. In most of the approaches, markers encode an unique identification by a binary code that may include error detection and correction bits. In general, each author has proposed its own predefined set of markers (*dictionary*). The problems of setting a predefined dictionary are twofold. First, in some cases, the number of markers required by the application might be higher than the dictionary size. Second, if the number of markers required is smaller, then it is preferable to use a smaller dictionary whose inter-marker distance is as high as possible, so as to reduce the inter-marker confusion rate.

Another common problem in augmented reality applications is related to the occlusion. The problem occurs when a real object appears occluding the virtual scene. In this case, the virtual objects are rendered on the real object, which should be visible (see Fig. 1(c,d)). This is indeed a limitation to the augmented experience since user can not interact freely.

This paper presents a fiducial marker system based on square markers offering solutions to the above mentioned problems. First, we propose a general method for generating configurable dictionaries (both in size and number of bits). Our algorithm creates dictionaries following a criterion to maximize the inter-marker distance and the number of bit transitions. In the process, we derive the maximum theoretical inter-marker distance that a dictionary of square binary markers can have. Then, a method for automatically detecting markers in images and correcting possible errors, based on our generated dictionaries, is presented. Third, we propose a solution to the occlusion problem based on combining multiple markers and an occlusion mask calculated using color information. While using multiple markers provides robustness against occlusion, color information is used to determine the occluded pixels avoiding rendering on them.

The rest of the paper is structured as follows. Section 2 presents the most relevant works related to ours. Section 3 explains the proposed method to generate marker dictionaries. Section 4 shows the process proposed for marker detection and error correction. Section 5 presents our solution to the occlusion problem. Finally, Section 6 shows the experimentation carried out, and Section 7 draws some conclusions.

Finally, it must be indicated that our work has been implemented in the ArUco library which is freely available [13].

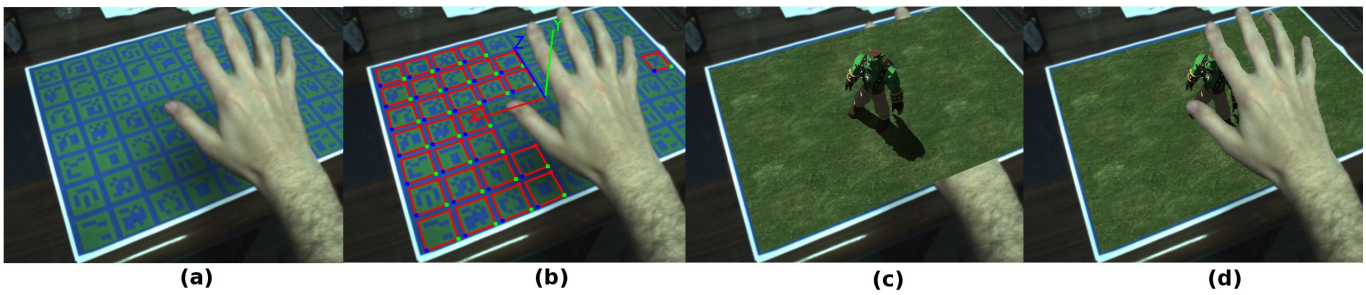

Figure 1: Example of augmented reality scene. (a) Input image containing a set of fiducial markers. (b) Markers automatically detected and used for camera pose estimation. (c) Augmented scene without considering user's occlusion. (d) Augmented scene considering occlusion.

2 Related work

A fiducial marker system is composed by a set of valid markers and an algorithm which performs its detection, and possibly correction, in images. Several fiducial marker systems have been proposed in the literature as shown in Figure 2.

The simplest proposals consist in using points as fiducial markers, such as LEDs, retroreflective spheres or planar dots [14, 15], which can be segmented using basic techniques over controlled conditions. Their identification is usually obtained from the relative position of the markers and often involves a complex process.

Other approaches use planar circular markers where the identification is encoded in circular sectors or concentric rings [16][17]. However, circular markers usually provide just one correspondence point (the center), making necessary the detection of several of them for pose estimation.

Other types of fiducial markers are based on blob detection. Cybercode[18] or VisualCode[19] are derived from 2D-barcodes technology as MaxiCode or QR but can also accurately provide several correspondence points. Other popular fiducial markers are the ReacTIVision amoeba markers [20] which are also based on blob detection and its design was optimized by using genetic algorithms. Some authors have proposed the use of trained classifiers to improve detection in cases of bad illumination and blurring caused by fast camera movement [21].

An alternative to the previous approaches are the square-based fiducial markers systems. Their main advantage is that the presence of four prominent points can be employed to obtain the pose, while the inner region is used for identification (either using a binary code or an arbitrary pattern such as an image). In the arbitrary pattern category, one of the most popular systems is ARToolKit [10], an open source project which has been extensively used in the last decade, especially in the academic community. ARToolkit markers are composed by a wide black border with an inner image which is stored in a database of valid patterns. Despite of its popularity, it has some drawbacks. First, it uses a template matching approach to identify markers, obtaining high false positive and inter-marker confusion rates [22]. Second, the system uses a fixed global threshold to detect squares, making it very sensitive to varying lighting conditions.

Most of the square-based fiducial systems uses binary codes. Matrix[23] is one of the first and simplest proposals. It uses a binary code with redundant bits for error detection. The ARTag [11] system is based on the same principles but improves the robustness to lighting and partial occlusion by using an edge-based square detection method, instead of a fixed threshold. Additionally, it uses a binary coding scheme that includes checksum bits for error detection and correction. It also recommends using its dictionary markers in a specific order so as to maximize the inter-marker distances. Its main drawback is that the proposed marker dictionary is fixed to 36 bits and the maximun number of erroneous bits that can be corrected is two, independently of the inter-marker distances of the subset of markers used.

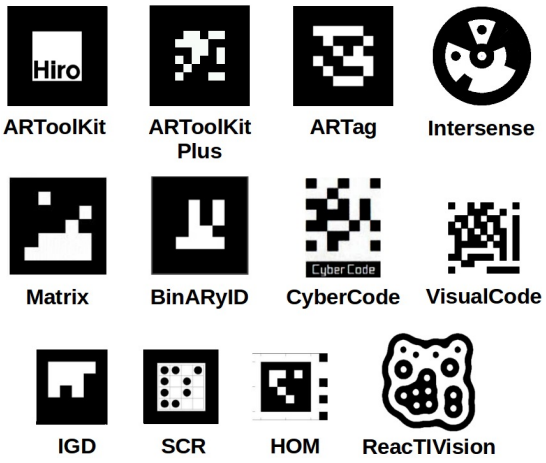

Figure 2: Examples of fiducial markers proposed in previous works.

ARToolKit Plus [24] improves some of the features of ARToolKit. First, it includes a method to automatically update the global threshold value depending on pixel values from previously detected markers. Second, it employs binary codes including error detection and correction, thus achieving higher robustness than its predecessor. The last known version of ARToolKitPlus employs a binary BCH [25] code for 36 bits markers which presents a minimun Hamming distance of two. As a consequence, ARToolKitPlus BCH markers can detect a maximun error of one bit and cannot perform error correction. ARToolkitPlus project was halted and followed by the Studierstube Tracker[12] project which is not publicly available.

BinARyID[26] proposes a method to generate binary coded markers focused on avoiding rotation ambiguities, however it only achieves Hamming distance of one between two markers and does not present any error correction process. There are also some closed-source systems which employ square markers such as the SCR, HOM and IGD [27] marker systems used by the ARVIKA project [28].

This paper proposes a square-based fiducial marker system with binary codes. However, instead of using a predefined set of markers, we propose a method for generating configurable marker dictionaries (with arbitrary size and number of markers), containing only the number of markers required. Our algorithm produces markers using a criterion to maximize the inter-marker distance and the number of bit transitions. Additionally, a method for detecting and correcting errors, based on the dictionary obtained, is proposed. This method allows error correction of a greater number of erroneous bits compared to the current state of the art systems.

Our last contribution is related to the occlusion problem in augmented reality applications. When designing an augmented reality application, interactivity is a key aspect to consider. So, one may expect users to occlude the markers. ARTag handles the problem in two ways. First, the marker detection method allows small breaks in the square sides. Second, they employ several markers simultaneously, thus, the occlusion of some of them does not affect the global pose estimation. Despite of being robust to occlusion, ARTag still has a main drawback: it can not detect precisely occlusion. As a consequence, if an object moves between the camera and the augmented scene (e.g. user's hands), the virtual objects will be rendered on the hands, hiding it (see Fig. 1(c,d)).

Proposals to detect the occluded regions usually fall into three main categories: depth-based, model-based, and color-based approaches. Depth-based approaches try to calculate the depth of the image pixels to detect occlusions. However, these approaches require depth-based sensors, such as stereo, time of flight or structured light cameras [29, 30, 31]. When a single camera is used, some authors have adopted model-based approaches [32, 33]. The idea is to provide geometric models of the objects which can occlude the scene, and detect their pose. This solution is not practical in many applications where the occluding objects are not known in advance, and imposes very strong performance limitations. Finally, color-based approaches [34], can be employed. The idea is to create a color model of the scene (background) which is then compared to the foreground objects.

In this work, we propose the use of multiple markers to handle occlusion (as in ARTag). However, we also propose the use of a color map for precisely detecting the visible pixels, so that the virtual scene is only rendered on them. In order to improve segmentation, we employ blue and green markers, instead of classical black-and-white ones. As we experimentally show, our proposal is an effective method for improving current augmented reality applications such as in gaming or film industry, although not limited to that.

Figure 3: Examples of markers of different sizes, n, generated with the proposed method. From left to right: $n = 5$, $n = 6$ and $n = 8$.

3 Automatic dictionary generation

The most relevant aspects to consider when designing a marker dictionary are the false positive and negative rates, the inter-marker confusion rate, and the number of valid markers [11]. The first two are often tackled in the literature using error detection and correction bits, which, on the other hand, reduces the number of valid markers. The third one, depends only on the distance between the markers employed. If they are too close, a few erroneous bits can lead to another valid marker of the dictionary, and the error could not be even detected.

Another desirable property of markers is having a high number of bit transitions, so that they are less likely to be confused with environment objects. For instance, the binary codes with only zeros or ones will be printed as completely black or white markers respectively, which would be easily confused with environment objects.

While previous works impose fixed dictionaries, we propose an automatic method for generating them with the desired number of markers and with the desired number of bits. Our problem is then to select m markers, from the space of all markers with $n \times n$ bits, $\mathbb{D}$, so that they are as far as possible from each other and with as many bit transitions as possible. In general, the problem is to find the dictionary $\mathfrak{D}^*$ that maximizes the desired criterion $\hat{\tau}(\mathfrak{D})$:

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

Since a complete evaluation of the search space is not feasible even for a small n, an stochastic algorithm that finds suboptimal solutions is proposed.

3.1 Algorithm overview

Our algorithm starts from an empty dictionary $\mathfrak{D}$ that is incrementally populated with new markers. Our markers are encoded as a $(n+2) \times (n+2)$ grid (Fig. 3) where the external cells are set to black, creating an external border easily detectable. The remaining $n \times n$ cells are employed for coding. Thus, we might define a marker,

$$m = (w_0, w_1, ..., w_{n-1}), \tag{2}$$

as a tuple composed by n binary words w of length n such that

$$w = (b_0, \ldots, b_{n-1} \mid b_i \in \{0, 1\}). \tag{3}$$

Let us also denote $\mathbb{W}$ as the set of all possible words of n bits, whose cardinal is $|\mathbb{W}| = 2^n$.

At each iteration of the algorithm, a marker is selected based on a stochastic process that assigns more probability to markers with a higher number of bit transitions and whose words have not been yet added to $\mathfrak{D}$. If the distance between the generated marker and these in $\mathfrak{D}$ is greater than a minimum value τ, then it is added. Otherwise, the marker is rejected and a new marker is randomly selected. The process stops when the required number of markers is achieved.

Because of the probabilistic nature of the algorithm, the acceptance of new markers could be improbable or even impossible in some cases. To guarantee the convergence of the algorithm, the distance threshold is initially set to the maximum possible inter-marker distance that the dictionary can have τ^0. Along the process, the value of τ is reduced after a number of unproductive iterations ψ. The final value $\hat{\tau}(\mathfrak{D})$ represents the minimum distance between any two markers in $\mathfrak{D}$, and it will be used as the base for error detection and correction (explained in Sect. 4). The proposed algorithm is summarized in Alg. 1.

Algorithm 1 Dictionary generation process

```
𝔇 ← ∅  # Reset dictionary
τ ← τ^0 #Initialize target distance, see Sect. 3.4
ϱ ← 0  # Reset unproductive iteration counter
while 𝔇 has not desired size do
  Generate a new marker m  #Sect. 3.2
  if distance of m to elements in 𝔇 is ≥ τ  then
    𝔇 ← 𝔇 ∪ m  # Add to dictionary
    ϱ ← 0
  else
    ϱ ← ϱ + 1  # It was unproductive
    # maximium unproductive iteration reached ?
    if ϱ = ψ  then
      τ ← τ − 1  # Decrease target distance
      ϱ ← 0
    end if
  end if
end while
```

3.2 Marker generation

As previously pointed out, markers are selected using a random process leaded by a probability distribution that assigns a higher probability to these markers with a high number of transitions and whose words are not yet present in $\mathfrak{D}$. The proposed process for generating a marker consists in selecting n words from $\mathbb{W}$ with replacement. To do so, each word $w_i \in \mathbb{W}$ has a probability of being selected at each iteration that is defined as:

$$P\{w = w_i\} = \frac{T(w_i)O(w_i, \mathfrak{D})}{\sum_{w_j \in \mathcal{W}} T(w_j)O(w_j, \mathfrak{D})}. \quad (4)$$

Eq. 4 defines the probability of selecting a word as the combination of two functions. The first one, $T(w_i) \in [0, 1]$, is related to the number of bit transitions of the word. It is defined as

$$T(w_i) = 1 - \frac{\sum_{j=0}^{n-2} \delta(w_i^{j+1}, w_i^j)}{n-1}, \quad (5)$$

being w_i^j the j-bit of the word w_i, and δ is 1 if both elements are equal and 0 otherwise. So, $T(w_i)$ tends to 1 as the number of transitions between consecutive bits increases and to 0 as the number of transitions decreases. For instance, the words 010110 and 000011 present values of $T = 4/5$ and $T = 1/5$, respectively, which are proportional to the number of bit transitions.

On the other hand, the function $O(w_i, \mathfrak{D})$ accounts for the number of times the word w_i appears amongst the markers in $\mathfrak{D}$. The idea is to reduce the probability of choosing words that have already been selected many times. It is defined in the interval $[0, 1]$ as

$$O(w_i, \mathfrak{D}) = \begin{cases} 1 - \frac{\sum_{m_i \in \mathfrak{D}} \sum_{w_j \in m_i} \delta(w_j, w_i)}{n|\mathfrak{D}|} & \text{if } |\mathfrak{D}| \neq 0 \\ 1 & \text{otherwise} \end{cases}. \quad (6)$$

The double sum counts the appearances of w amongst the markers in $\mathfrak{D}$, while the denominator counts the total number of words in $\mathfrak{D}$. Thus, $O(w_i, \mathfrak{D})$ is 1 if w_i is not in $\mathfrak{D}$, and tends to 0 as it appears a higher number of times. Finally, in the first iteration ($|\mathfrak{D}| = 0$), the function is defined as 1 so that all words have the same probability of being selected.

3.3 Distance calculation

As previously indicated, a marker is added to the dictionary if its distance to the markers in the dictionary is below τ. The concept of distance between markers must be defined considering that they are printed as binary grids of $n \times n$ bits that can be observed under rotation. Then, let us define the distance between two markers as

$$D(m_i, m_j) = \min_{k \in \{0,1,2,3\}} \{ H(m_i, R_k(m_j)) \}. \quad (7)$$

The function H is the Hamming distance between two markers, which is defined as the sum of hamming distances between each pair of marker words. The function R_k is an operator that rotates the marker grid $k \times 90$ degrees in clockwise direction. The function D is then the rotation-invariant Hamming distance between the markers.

Let us also define the distance of a marker to a dictionary:

$$D(m_i, \mathfrak{D}) = \min_{m_j \in \mathfrak{D}} \{D(m_i, m_j)\}, \quad (8)$$

as the distance of the marker to nearest one in the dictionary.

Finally, it is not only important to distinguish markers from each other, but also to correctly identify the marker orientation. Otherwise, pose estimation would fail. So, a valid marker must also guarantee that the minimum distance to its own rotations is above τ. Thus, we define the marker self-distance as

$$\mathcal{S}(m_i) = \min_{k \in \{1,2,3\}} \{ H(m_i, R_k(m_i)) \}. \quad (9)$$

In summary, we only add a marker to the dictionary if both $\mathcal{S}(m_i)$ and $D(m_i, \mathfrak{D})$ are greater or equal than τ. Otherwise, the marker is rejected and a new one generated. After a number of unproductive iterations ψ, the value of

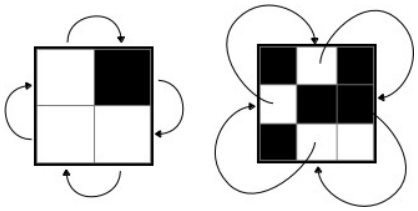

Figure 4: Examples of quartets for a 2×2 and 3×3 marker. Each arrow indicates the destination of a bit after a 90 degrees clockwise rotation.

Group	Quartets	Hamming distances		
		90 deg	180 deg	270 deg
Q_1	0000	0	0	0
Q_2	1000 → 0100 → 0010 → 0001	2	2	2
$\mathbf{Q_3}$	1100 → 0110 → 0011 → 1001	2	4	2
$\mathbf{Q_4}$	0101 → 1010	4	0	4
Q_5	1110 → 0111 → 1011 → 1101	2	2	2
Q_6	1111	0	0	0

Table 1: Quartet groups and quartet Hamming distances for each rotation.

τ is decreased by one so as to allow new markers to be added.

In the end, the markers of the generated dictionary have a minimum distance between them and to themselves, $\hat{\tau}$, that is the last τ employed. This value can be calculated for any marker dictionary (manually or automatically generated) as:

$$\hat{\tau}(\mathfrak{D}) = \min\left\{\min_{m_i\in\mathfrak{D}}\{\mathcal{S}(m_i)\}, \min_{m_i\neq m_j\in\mathfrak{D}}\{D(m_i,m_j)\}\right\}. \tag{10}$$

3.4 Maximum inter-marker distance: τ^0

The proposed algorithm requires an initial value for the parameter τ^0. If one analyzes the first iteration (when the dictionary is empty), it is clear that the only distance to consider is the self distance (Eq. 9), since the distance to other markers is not applicable. So, the maximum self distance for markers of size $n\times n$ (let us denote it by $\mathcal{S}_n^*$) is the maximum distance that a dictionary can have for these type of markers. This section explains how to determine $\mathcal{S}_n^*$, which is equivalent to find the marker of size $n\times n$ with highest self-distance.

If we analyze the path of the bits when applying 90 degrees rotations to a marker, it is clear that any bit (x, y) changes its position to another three locations until it returns to its original position (see Figure 4). It can be understood, that the Hamming distance provided by a marker bit to Eq. 9 is only influenced by these other three bits. So, let us define a *quartet* as the set composed by these positions: $\{(x,y),(n-y-1,x),(n-x-1,n-y-1),(y,n-x-1)\}$.

In general, a marker of size $n\times n$, has a total of C quartets that can be calculated as:

$$C = \left\lfloor \frac{n^2}{4} \right\rfloor, \tag{11}$$

where $\lfloor\cdot\rfloor$ represents the floor function. If n is odd, the central bit of the marker constitutes a quartet by itself which does not provide extra distance to $\mathcal{S}$.

If a quartet is expressed as a bit string, a 90 degrees rotation can be obtained as a circular bit shift operation. For instance, the quartet 1100, becomes (0110 → 0011 → 1001) in successive rotations. In fact, for the purpose of calculating $\mathcal{S}_n^*$, these four quartets are equivalent, and we will refer to them as a *quartet group* Q_i. It can be seen from Eq. 9, that the contribution of any quartet is given by the distance of its successive rotations to the original quartet. For instance, quartet 1100 contributes to Eq. 9 with distances $(2, 4, 2)$ as it rotates:

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

But also, if we start from quartet 0110 and rotate it successively, we obtain the quartets (0011 → 1001 → 1100), that again provide the distances $(2, 4, 2)$:

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

In fact, there are only 6 quartet groups (shown in Table 1), thus reducing the problem considerably.

As previously indicated, calculating $\mathcal{S}_n^*$ is the problem of obtaining the marker with highest self-distance, and we have turned this problem into assigning quartets groups to the C quartets of a maker. It can be seen that it is in fact a multi-objective optimization, where each quartet group Q_i is a possible solution and the objectives to maximize are the distances for each rotation. If the Pareto front is obtained, it can be observed that the groups Q_3 and Q_4 dominates the rest of solutions. Thus, the problem is simplified, again, to assign Q_3 and Q_4 to the C quartets of a marker.

From a brief analysis, it can be deduced that $\mathcal{S}_n^*$ is obtained by assigning the groups $\{Q_3, Q_3, Q_4\}$ (in this order) repeatedly until completing the C quartets. For instance, the simplest marker is a 2×2 marker ($C = 1$), $\mathcal{S}_n^* = 2$ and is obtained by assigning Q_3. For a 3×3 marker ($C = 2$), $\mathcal{S}_n^* = 4$ which is obtained by assigning Q_3 twice. For a 4×4 marker ($C = 4$), $\mathcal{S}_n^* = 10$ obtained by assigning the groups $\{Q_3, Q_3, Q_4, Q_3\}$. This last case is showed in detail in Table 2.

Therefore, for a generic marker with C quartets, the value $\mathcal{S}_n^*$ follows the rule:

$$\mathcal{S}_n^* = 2\left\lfloor \frac{4C}{3} \right\rfloor \tag{12}$$

Then, we employ the value:

$$\tau^0 = \mathcal{S}_n^*, \tag{13}$$

as starting point for our algorithm.

4 Marker detection and error correction

This section explains the steps employed to automatically detect the markers in an image (Fig. 5(a)). The process

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
V_s^*		$\min(10, 12, 10) = 10$		

Table 2: Quartet assignment for a 4×4 marker ($C = 4$) to obtain $\mathcal{S}_n^*$. It can be observed as the sequence $\{Q_3, Q_3, Q_4\}$ is repeated until filling all the quartets in the marker.

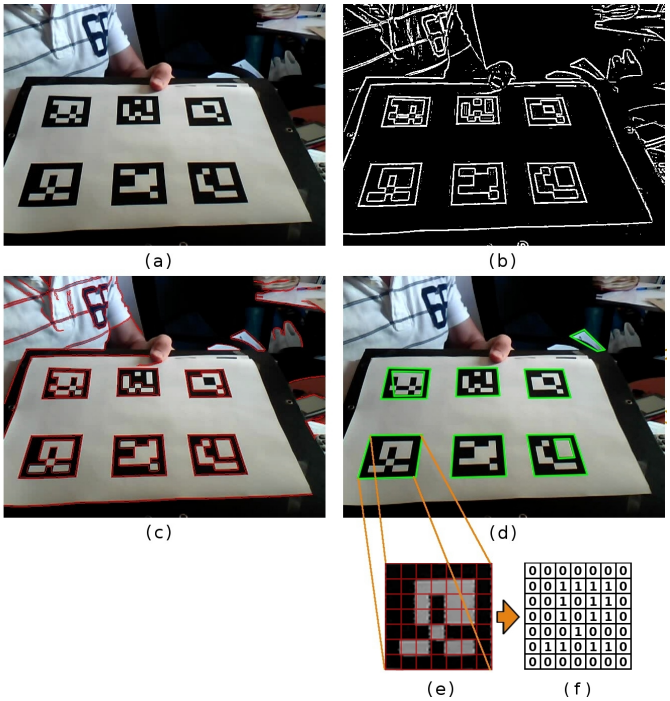

Figure 5: Image Process for automatic marker detection. (a) Original Image. (b) Result of applying local thresholding. (c) Contour detection. (d) Polygonal approximation and removal of irrelevant contours. (e) Example of marker after perspective transformation. (f) Bit assignment for each cell.

is comprised by several steps aimed at detecting rectangles and extracting the binary code from them. For that purpose, we take as input a gray-scale image. While the image analysis is not a novel contribution, the marker code identification and error correction is a new approach specifically designed for the generated dictionaries of our method. Following are described the steps employed by our system.

- Image segmentation: Firstly, the most prominent contours in the gray-scale image are extracted. Our initial approach was employing the Canny edge detector [35], however, it is very slow for our real-time purposes. In this work, we have opted for a local adaptive thresholding approach which has proven to be very robust to different lighting condition (see Fig. 5(b)).

- Contour extraction and filtering: Afterward, a contour extraction is performed on the thresholded image using the Suzuki and Abe [36] algorithm. It produces the set of image contours, most of which are irrelevant for our purposes (see Figure 5(c)). Then, a polygonal approximation is performed using the Douglas-Peucker [37] algorithm. Since markers are enclosed in rectangular contours, these that are not approximated to 4-vertex polygons are discarded. Finally, we simplify near contours leaving only the external ones. Figure 5(d) shows the resulting polygons from this process.

- Marker Code extraction: The next step consists in analyzing the inner region of these contours to extract its internal code. First, perspective projection is removed by computing the homography matrix (Fig. 5(e)). The resulting image is thresholded using the Otsu's method [38], which provides the optimal image threshold value given that image distribution is bimodal (which holds true in this case). Then, the binarized image is divided into a regular grid and each element is assigned the value 0 or 1 depending on the values of the majority of pixels into it (see Fig. 5(e,f)). A first rejection test consists in detecting the presence of the black border. If all the bits of the border are zero, then the inner grid is analyzed using the method described below.

- Marker identification and error correction: At this point, it is necessary to determine which of the marker candidates obtained actually belongs to the dictionary and which are just part of the environment. Once the code of a marker candidate is extracted, four different identifiers are obtained (one for each possible rotation). If any of them is found in $\mathfrak{D}$, we consider the candidate as a valid marker. To speed up this process, the dictionary elements are sorted as a balanced binary tree. To that aim, markers are represented by the integer value obtained by concatenating all its bits. It can be deduced then, that this process has a logarithmic complexity $O(4 \log_2(|\mathfrak{D}|))$, where the factor 4 indicates that it is necessary one search for each rotation of the marker candidate.

  If no match is found, the correction method can be applied. Considering that the minimum distance between any two markers in $\mathfrak{D}$ is $\hat{\tau}$, an error of at most $\lfloor(\hat{\tau}-1)/2\rfloor$ bits can be detected and corrected. Therefore, our marker correction method consists in calculating the distance of the erroneous marker candidate to all the markers in $\mathfrak{D}$ (using Eq. 8). If the distance is equal or smaller than $\lfloor(\hat{\tau}-1)/2\rfloor$, we consider that the nearest marker is the correct one. This process, though, presents a linear complexity of $O(4|\mathfrak{D}|)$, since each rotation of the candidate has to be compared to the entire dictionary. Nonetheless, it is a highly parallelizable process that can be efficiently implemented in current computers.

  Please note that, compared to the dictionaries of ARToolKitPlus (which can not correct errors) and ARTag (only capable of recovering errors of two bits), our approach can correct errors of $\lfloor(\hat{\tau}-1)/2\rfloor$ bits. For instance, for a dictionary generated in the exper-

imental section with 6×6 bits and 30 markers, we obtained $\hat{\tau} = 12$. So, our approach can correct 5 bits of errors in this dictionary. Additionally, we can generate markers with more bits which leads to a larger $\hat{\tau}$, thus increasing the correction capabilities. Actually, our detection and correction method is a general framework that can be used with any dictionary (including ARToolKitPlus and ARTag dictionaries). In fact, if our method is employed with the ARTag dictionary of 30 markers, for instance, we could recover from errors of 5 bits, instead of the 2 bits they can recover from.

- Corner refinement and pose estimation: Once a marker has been detected, it is possible to estimate its pose respect to the camera by iteratively minimizing the reprojection error of the corners (using for instance the Levenberg-Marquardt algorithm[39, 40]). While, many approaches have been proposed for corner detection [41, 42, 43], we have opted for doing a linear regression of the marker side pixels to calculate their intersections. This approach was also employed in ARTag [11], ARToolKit [10] and ARToolKitPlus [24].

5 Occlusion detection

Detecting a single marker might fail for different reasons such as poor lighting conditions, fast camera movement, occlusions, etc. A common approach to improve the robustness of a marker system is the use of marker boards. A marker board is a pattern composed by multiple markers whose corner locations are referred to a common reference system. Boards present two main advantages. First, since there are more than one marker, it is less likely to lose all of them at the same time. Second, the more markers are detected, the more corner points are available for computing the camera pose, thus, the pose obtained is less influenced by noise. Figure 1(a) shows the robustness of a marker board against partial occlusion.

Based on the marker board idea, a method to overcome the occlusion problem in augmented reality applications (i.e., virtual objects rendered on real objects as shown in Fig. 1(c,d)) is proposed. Our approach consists in defining a color map of the board that is employed to compute an occlusion mask by color segmentation.

Although the proposed method is general enough to work with any combinations of colors, we have opted in our tests to replace black and white markers by others with higher chromatic contrast so as to improve color segmentation. In our case, blue and green have been selected. Additionally we have opted for using only the hue component of the HSV color model, since we have observed that it provides the highest robustness to lighting changes and shadows.

Let us define the color map $\mathcal{M}$ as a $n_c \times m_c$ grid, where each cell c represents the color distribution of the pixels of a board region. If the board pose is properly estimated, it is possible to compute the homography H_m that maps

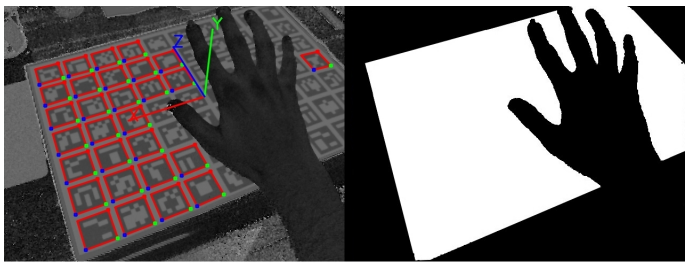

Figure 6: Occlusion mask example. (a) Hue component of Fig. 1(a) with the detected markers (b) Occlusion mask: white pixels represent visible regions of the board.

the board image pixels p into the map space

$$p_m = H_m p.$$

Then, the corresponding cell p_c is obtained by discretizing the result to its nearest value $p_c = [p_m]$. Let us denote by $\mathcal{I}_c$ the set of image board pixels that maps onto cell c.

If the grid size of $\mathcal{M}$ is relatively small compared to the size of the board in the images, $\mathcal{I}_c$ will contain pixels of the two main board colors. It is assumed then, that the distribution of the colors in each cell can be modeled by a mixture of two Gaussians [44], using the Expectation-Maximization algorithm [45] to obtain its parameters. Therefore, the pdf of the color u in a cell c can be approximated by the expression

$$P(u, c) = \sum_{k=1,2} \pi_k \mathcal{N}_k^c(u|\mu_k^c, \Sigma_k^c), \tag{14}$$

where $\mathcal{N}_k^c(u|\mu_k^c, \Sigma_k^c)$ is the k-th Gaussian distribution and π_k^c is the mixing coefficient, being

$$\sum_{k=1,2} \pi_k^c = 1.$$

In an initial step, the map must be created from a view of the board without occlusion. In subsequent frames, color segmentation is done analyzing if the probability of a pixel is below a certain threshold θ_c. However, to avoid the hard partitioning imposed by the discretization, the probability of each pixel is computed as the weighted average of the probabilities obtained by the neighbor cells in the map:

$$\bar{P}(p) = \frac{\sum_{c \in \mathcal{H}(p_c)} w(p_m, c) P(p_u, c)}{\sum_{c \in \mathcal{H}(p_c)} w(p_m, c)}, \tag{15}$$

where p_u is the color of the pixel, $\mathcal{H}(p_c) \subset \mathcal{M}$ are the nearest neighbor cells of p_c, and

$$w(p_m, c) = (2 - |p_m - c|_1)^2 \tag{16}$$

is a weighting factor based on the L^1-norm between the mapped value p_m and the center of the cell c. The value 2 represents the maximum possible L^1 distance between neighbors. As a consequence, the proposed weighting value is very fast to compute and provides good results in practice.

Considering that the dimension of the observed board in the image is much bigger than the number of cells in the color map, neighbor pixels in the image are likely to have similar probabilities. Thus, we can speed up computation by downsampling the image pixels employed for calculating the mask and assigning the same value to its neighbors.

Figure 6 shows the results of the detection and segmentation obtained by our method using as input the hue channel and a downsampling factor of 4. As can be seen, the occluding hand is properly detected by color segmentation.

Finally, it must considered that the lighting conditions might change, thus making it necessary to update the map. This process can be done with each new frame, or less frequently to avoid increasing the computing time excessively. In order to update the color map, the probability distribution of the map cells are recalculated using only the visible pixels of the board. The process only applies to cells with a minimum number of visible pixels γ_c, i.e., only if $|\mathcal{I}_c| > \gamma_c$.

6 Experiments and results

This section explains the experimentation carried out to test our proposal. First, the processing times required for marker detection and correction are analyzed. Then, the proposed method is compared with the state-of-the-art systems in terms of inter-marker distances, number of bit transitions, robustness against noise and vertex jitter. Finally, an analysis of the occlusion method proposed is made.

As already indicated, this work is available under the BSD license in the ArUco library [13].

6.1 Processing Time

Processing time is a crucial feature in many real time fiducial applications (such as augmented reality). The marker detection process of Sect. 4 can be divided in two main steps: finding marker candidates and analyzing them to determine if they actually belong to the dictionary.

The detection performance of our method has been tested for a dictionary size of $|\mathfrak{D}| = 24$. The processing time for candidate detection, marker identification and error correction was measured for several video sequences. The tests were performed using a single core of a system equipped with an Intel Core 2 Quad 2.40 Ghz processor, 2048 MB of RAM and Ubuntu 12.04 as the operating system with a load average of 0.1. Table 3 summarizes the average obtained results for a total of 6000 images with resolution of 640×480 pixels. The sequences include indoor recording with several markers and marker boards arranged in the environment.

In addition, we have evaluated the computing time required for generating dictionaries with the proposed method for 6×6 markers. The value of τ was reduced after $\psi = 5000$ unproductive iterations. The computing times for dictionaries of sizes 10, 100 and 1000 elements are approximately 8, 20 and 90 minutes, respectively. Since this is an off-line process done only once, we consider that the computing times obtained are appropriated for real applications. It must be considered, though, that generating the first elements of the dictionary is more time consuming because the high inter-distances imposed. As τ decreases, the computation speed increases.

Finally, the time required for creating the color map and the occlusion mask in the sequences reported in Sect. 6.6, are 170 and 4 ms, respectively. In these sequences, the board has an average dimension of 320×240 pixels.

Candidates detection	8.17 ms/image
Marker identification	0.17 ms/candidate
Error correction	0.71 ms/candidate
Total time ($\|\mathfrak{D}\| = 24$)	11.08 ms/image

Table 3: Average processing times for the different steps of our method.

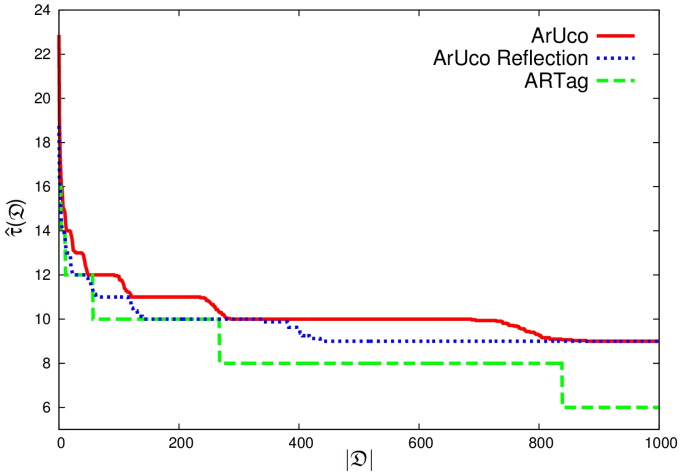

Figure 7: Inter-marker distances of ARTag dictionaries and ours (Eq. 10) for an increasing number of markers. ArUco values correspond to the mean of 30 runs of our algorithm (with and without considering reflection). Higher distances reduces the possibility of inter-marker confusion in case of error.

6.2 Analysis of Dictionary distances

The inter-marker confusion rate is related to the distances between the markers in the dictionary $\hat{\tau}(\mathfrak{D})$ (Eq. 10). The higher the distance between markers, the more difficult is to confuse them in case of error. The marker dictionary proposed by Fiala in the ARTag [11] system improves the distances of other systems such as ARToolKitPlus [24] or BinARyID [26]. His work recommends using its dictionary (of 6×6 markers) in a specific order so as to maximize the distance.

We have compared the dictionaries generated with our method to these obtained by incrementally adding the first 1000 recommended markers of ARTag. For our algorithm, the initial distance employed is $\tau^0 = 24$ (Eq. 13), which has been decremented by one after $\psi = 5000$ unproductive iterations. Since ARTag considers the possibility of marker reflection (i.e. markers seen in a mirror), we have

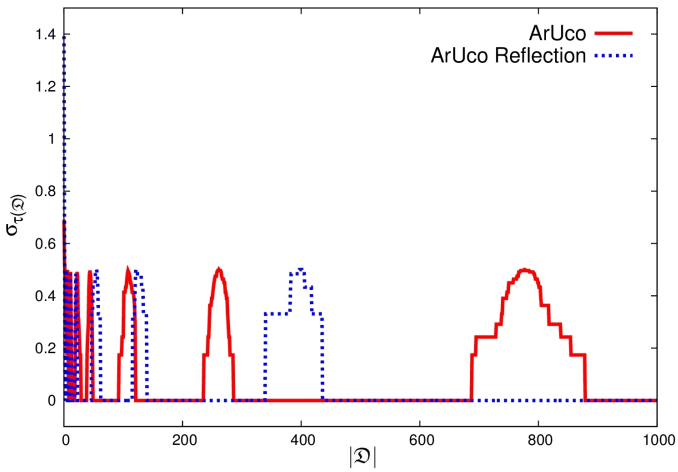

Figure 8: Standard deviations of inter-marker distances obtained by our method in Figure 7 (with and without considering reflection).

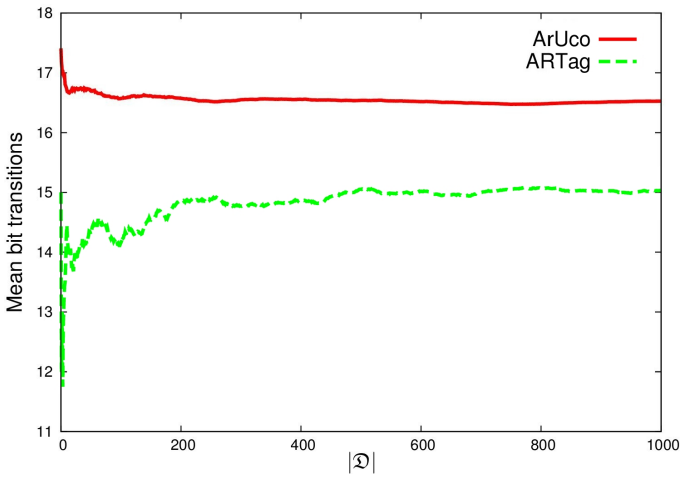

Figure 9: Number of bit transitions of ARTag dictionaries and ours for an increasing number of markers. Higher number of transitions reduces the possibility of confusion with environment elements.

also tested our method including the reflection condition. However, we consider this is as an uncommon case in fiducial marker applications.

Figure 7 shows the values $\hat{\tau}(\mathfrak{D})$ for the dictionaries as their size increases. The results shown for our method represent the average values of 30 runs of our algorithm. As can be seen, our system outperforms the ARTag dictionaries in the majority of the cases and obtains the same results in the worst ones. Even when considering reflection, our method still outperforms the ARTag results in most cases. ARToolKitPlus system has not been compared since it does not include a recommended marker order as ARTag. However, the minimum distance in ARToolKitPlus considering all the BCH markers is 2, which is a low value in comparison to our method, or ARTag.

Figure 8 shows standard deviations for 30 runs of the tests shown in Figure 7. It can be observed that there are two patterns in the deviation results: (i) peaks which correspond to the slopes in Figure 7, and (ii) intervals without deviation where the inter-marker distance remains the same in all runs. As can be observed, the higher deviations occurs at the transitions of $\hat{\tau}(\mathfrak{D})$ in Figure 7. It must be noted, though, that in most of the cases, the maximun deviation is 0.5. Just in the generation of the first markers, the deviation ascends up to 1.4 and 0.7 (with and without considering reflection respectively).

6.3 Evaluation of the bit transitions

Our marker generation process encourages markers with a high number of bit transitions, thus, reducing the possibility of confusion with environment elements. Figure 9 shows the number of bit transitions of the dictionaries generated in the previous section with our method and with ARTag. The number of transitions are obtained as the sum of the transitions for each word in the marker. As in the previous case, our results represent the average values obtained for 30 different marker dictionaries generated with our algorithm. It must be indicated that the maximum standard deviation obtained in all cases was 1.7.

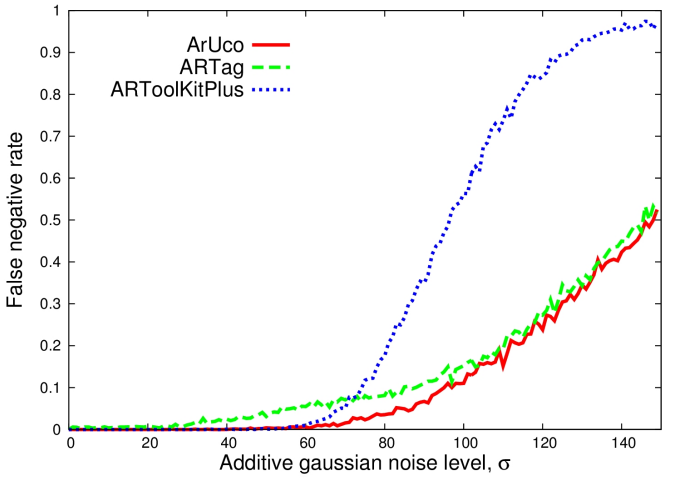

Figure 10: False negative rates for different levels of additive Gaussian noise.

It can be observed that our approach generates markers with more transitions than ARTag. Also, the number of transitions does not decrease drastically as the number of markers selected grows. The mean bit transitions for all the BCH markers in ARToolKitPlus is 15.0 which is also below our method.

6.4 Error detection

The false positive and false negative rates are related to the coding scheme and the number of redundant bits employed for error detection and correction. In our approach, however, false positives are not detected by checking redundant bits but analyzing the distance to the dictionary markers. A comparison between the correction capabilities of ARToolKitPlus, ARTag and our method has been performed by comparing the false negative rates from a set of 100 test images for each system. The images showed markers of each system from different distances and viewpoints. The images were taken in the same positions for

Figure 11: Image examples from video sequences used to test the proposed fiducial marker system. First row shows cases of correct marker detection. Second row shows cases where false positives have not been detected.

each of the tested systems. Different levels of additive Gaussian noise have been applied to the images to measure the robustness of the methods. Figure 10 shows the false negative rates obtained as a function of the noise level.

As can be observed, the proposed method is more robust against high amounts of noise than the rest. ARToolKitPlus false negative rate increases sharply for high levels of noise. ARTag presents a higher sensitivity for low levels of noise, however it is nearly as robust as our method for high levels. Figure 11 shows some examples of the sequences used to test the proposed system. It must be indicated, though, that no false positives have been detected by any method in the video sequences tested during our experimentation.

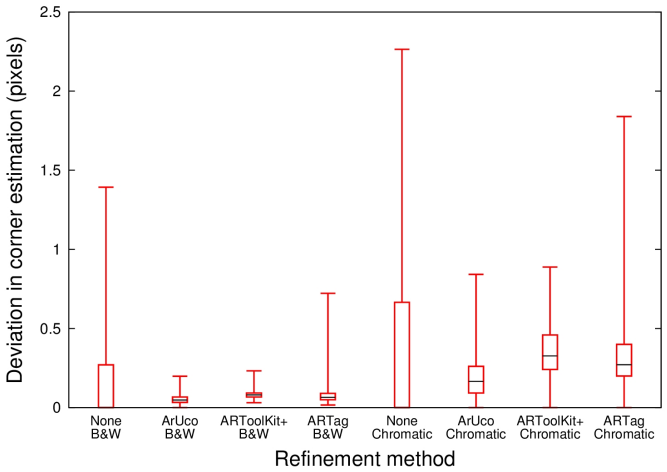

Figure 12: Vertex jitter measures for different marker systems.

6.5 Vertex jitter

An important issue in many augmented reality applications is the vertex jitter, which refers to the noise in the localization of the marker corner. Errors in the location of corners are propagated to the estimation of the camera extrinsic parameters, leading to an unpleasant user experiences. This section analyzes the obtained vertex jitter of i) the result of the polygonal approximation (see Sect. 4), ii) our method implemented in the ArUco library, iii) the ARToolKitPlus library and iv) the ARTag library. The first method is the most basic approach (i.e., no corner refinement) and is applied to analyze the impact of the other methods. Then, since the techniques used by ARToolKitPlus, ARTag and our method are based on the same principle (linear regression of marker side pixels), it is expected that they obtain similar results.

For the experiments, the camera has been placed at a fixed position respect to a set of markers and several frames have been acquired. Then, the camera has been moved farther away from the marker thus obtaining several view points at different distances. The standard deviation of the corners locations estimated by each method has been measured in all the frames. The experiment has been repeated both for black-and-white markers and green-and-blue markers. Please note that the hue channel employed for detecting the latter presents less contrast than the black-and-white markers (see Fig. 6(a)). Thus, evaluating the different corner refinement systems is especially relevant in that case.

Figure 12 shows the results obtained as a box plot [46] for both, black-and-white markers and green-and-blue markers. The lower and upper ends of the whiskers represent the minimum and maximum distribution values respectively. The bottom and top of the boxes represent the lower and upper quartiles, while the middle band represents the median.

It can be observed that the jitter level is lower in black-and-white markers than in green-and-blue ones. Nonetheless, it is small enough to provide a satisfactory user's experience. As expected, not performing any refinement produces higher deviations. It can also be noted that our method obtains similar results than these obtained by ARToolKitPlus and ARTag libraries. We consider that dif-

ferences obtained between the three methods can be attributed to implementation details.

6.6 Analysis of Occlusion

Along with the marker system described, a method to overcome the occlusion problem in augmented reality applications has been proposed. First, we employ marker boards so as to increase the probability of seeing complete markers in the presence of occlussion. Then, we propose using a color map to calculate an occlusion mask of the board pixels. We have designed two set of experiments to validate our proposal. Firstly, it has been analyzed how different occlusion levels affects to the estimation of the camera pose. While ARTag introduces the idea of multiple markers, no analysis of occlussion is made in their work. Secondly, a qualitative evaluation of the occlusion mask generated has been performed under different lighting conditions. It must be noticed that the estimation of the occlusion mask is not present in any of the previous works (ARTag, ARToolKit or ARToolKitPlus), thus a comparison with them is not feasible.

For our tests, the parameters

$$\theta_c = 10^{-4}, \gamma_c = 50, n_c = m_c = 5,$$

have been employed, providing good results in a wide range of sequences.

6.6.1 Occlusion tolerance

In this experiments we aim at analyzing the tolerance to occlusion of our system. To do so, a video sequence is recorded showing a board composed by 24 markers without occlusion so that all markers are correctly detected. Assuming gaussian noise, the ground truth camera pose is assumed to be the average in all the frames. Then, we have artificially simulated several degrees of oclussion by randomly removing a percentage of the detected markers in each frame and computing the pose with the remaining ones. Thus, the deviation from the ground truth at each frame is the error introduced by occlusion. This process has been repeated for three distances from the board to analyze the impact of distance in the occlussion handling.

The 3D rotation error is computed using the inner product of unit quaterions[47]

$$\phi(q_1, q_2) = 1 - |q_1 \cdot q_2|$$

which gives values in the range $[0, 1]$. The traslation error has been obtained using the Euclidean distance.

Figures 13-14 show the obtained results for different camera distances to the marker board. It can be observed that, both in rotation and traslation, the error originated by the occlusion are insignificant until the occlusion degree is above 85%. It can also be noted that the error increases as camera is farther from the board.

6.6.2 Qualitative evaluation of the occlusion mask

Figure 15 shows some captures from a user session using the green-and-blue marker board. The augmented objects

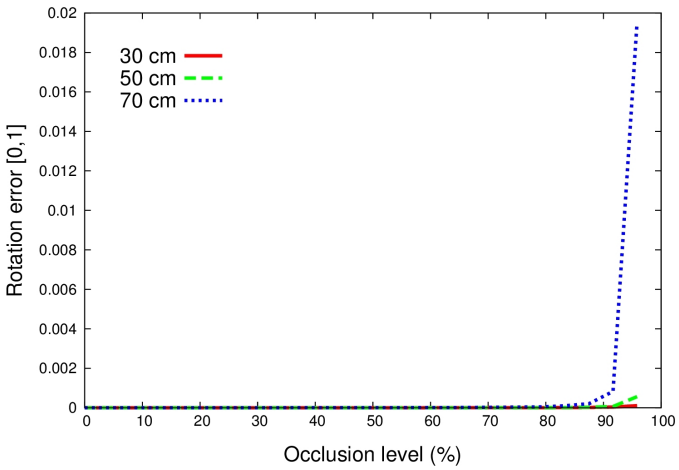

Figure 13: Rotation error for different degrees of marker board occlusion and for three camera distances.

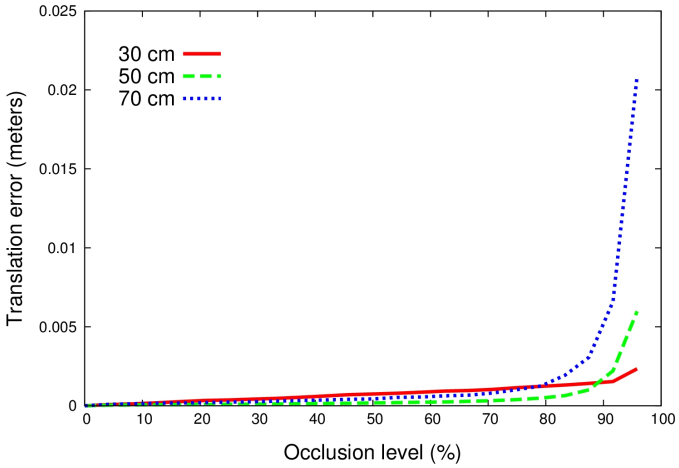

Figure 14: Traslation error for different degrees of marker board occlusion and for three camera distances.

consist in a piece of virtual floor and a virtual character doing some actions around. It can be observed that the user hand and other real objects are not occluded by virtual objects since they have different tonalities than the board and thus can be recognized by our method.

Nonetheless, as any color-based method, it is sensitive to lighting conditions, i.e., too bright or too dark regions makes it impossible to detect the markers nor to obtain a precise occlusion mask. Fig. 16 shows an example of scene where a lamp has been placed besides the board. It can be seen that there is a bright spot saturating the lower right region of the board, where markers can not be detected. Additionally, because of the light saturation, the chromatic information in that region (hue channel) is not reliable, thus producing segmentation errors in the board.

7 Conclusions

This paper has proposed a fiducial marker system specially appropriated for camera localization in applications such as augmented reality applications or robotics. Instead of employing a predefined set of markers, a general method

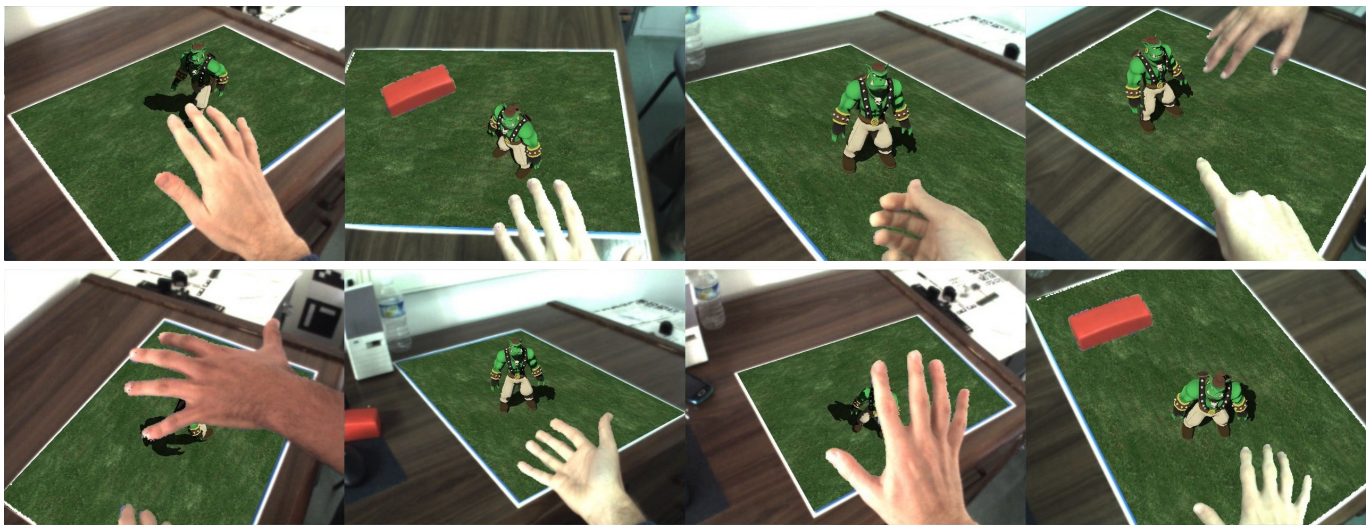

Figure 15: Examples of users' interaction applying the occlusion mask. Note that hands and other real objects are not occluded by the virtual character and the virtual floor texture.

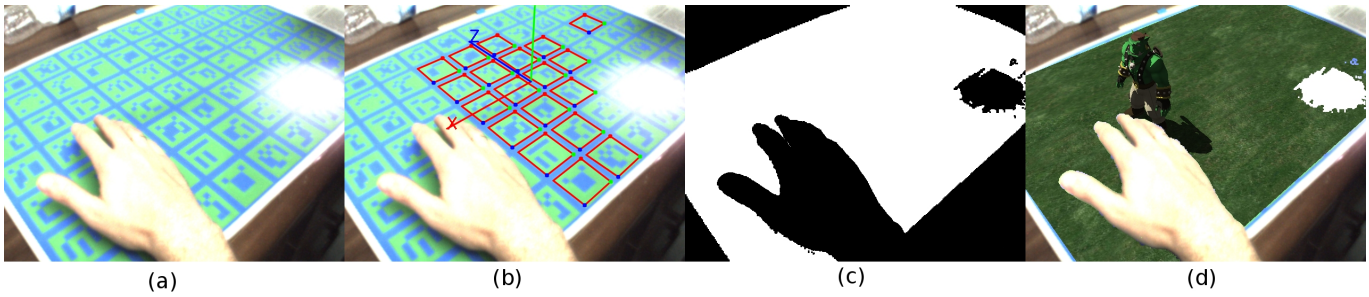

Figure 16: Example of occlusion mask errors due to light saturation. (a) Original input image. (b) Markers detected. (c) Occlusion mask. (d) Augmented scene.

to generate configurable dictionaries in size and number of bits has been proposed. The algorithm relies on a probabilistic search maximizing two criteria: the inter-marker distances and the number of bit transitions. Also, the theoretical maximum inter-marker distance that a dictionary with square makers can have has been derived. The paper has also proposed an automatic method to detect the markers and correct possible errors. Instead of using redundant bits for error detection and correction, our approach is based on a search on the generated dictionary. Finally, a method to overcome the occlusion problem in augmented reality applications has been presented: a color map employed to calculate the occlusion mask.

The experiments conducted have shown that the dictionaries generated with our method outperforms state-of-the-art systems in terms of inter-marker distance, number of bit transitions and false positive rate. Finally, this work has been set publicly available in the ArUco library [13].

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

[2] H. Kato, M. Billinghurst, Marker tracking and HMD calibration for a Video-Based augmented reality conferencing system, Augmented Reality, International Workshop on 0 (1999) 85–94.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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–.

[9] P. Bhattacharya, M. Gavrilova, A survey of landmark recognition using the bag-of-words framework, in: Intelligent Computer Graphics, Vol. 441 of Studies in Computational Intelligence, Springer Berlin Heidelberg, 2013, pp. 243–263.

[10] 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–.

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

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

[13] R. Munoz-Salinas, S. Garrido-Jurado, Aruco library, `http://sourceforge.net/projects/aruco/`, [Online; accessed 01-December-2013] (2013).

[14] K. Dorfmller, H. Wirth, Real-time hand and head tracking for virtual environments using infrared beacons, in: in Proceedings CAPTECH98. 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–.

[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] 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.

[21] D. Claus, A. Fitzgibbon, Reliable automatic calibration of a marker-based position tracking system, in: Workshop on the Applications of Computer Vision, 2005, pp. 300–305.

[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, July 15-17, 1998, Kangawa, Japan, Proceedings, IEEE Computer Society, 1998, pp. 63–69.

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

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

[26] 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.

[27] X. Zhang, S. Fronz, N. Navab, Visual marker detection and decoding in ar systems: A comparative study, in: Proceedings of the 1st International Symposium on Mixed and Augmented Reality, ISMAR '02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 97–.

[28] W. Friedrich, D. Jahn, L. Schmidt, Arvika - augmented reality for development, production and service, in: DLR Projektträger des BMBF für Informationstechnik (Ed.), International Status Conference - Lead Projects Human-Computer Interaction (Saarbrücken 2001), DLR, Berlin, 2001, pp. 79–89.

[29] S. Zollmann, G. Reitmayr, Dense depth maps from sparse models and image coherence for augmented reality, in: 18th ACM symposium on Virtual reality software and technology, 2012, pp. 53–60.

[30] M. o. Berger, Resolving occlusion in augmented reality: a contour based approach without 3d reconstruction, in: In Proceedings of CVPR (IEEE Conference on Computer Vision and Pattern Recognition), Puerto Rico, 1997, pp. 91–96.

[31] J. Schmidt, H. Niemann, S. Vogt, Dense disparity maps in real-time with an application to augmented reality, in: Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision, WACV '02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 225–.

[32] A. Fuhrmann, G. Hesina, F. Faure, M. Gervautz, Occlusion in collaborative augmented environments, Tech. Rep. TR-186-2-98-29, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria (Dec. 1998).

[33] V. Lepetit, M. odile Berger, A semi-automatic method for resolving occlusion in augmented reality, in: In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2000, pp. 225–230.

[34] R. Radke, Image change detection algorithms: a systematic survey, Image Processing, IEEE Transactions on 14 (3) (2005) 294–307.

[35] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. 8 (6) (1986) 679–698.

[36] S. Suzuki, K. Be, Topological structural analysis of digitized binary images by border following, Computer Vision, Graphics, and Image Processing 30 (1) (1985) 32–46.

[37] D. H. Douglas, T. K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Cartographica: The International Journal for Geographic Information and Geovisualization 10 (2) (1973) 112–122.

[38] N. Otsu, A threshold selection method from gray-level histograms, IEEE Transactions on Systems, Man and Cybernetics 9 (1) (1979) 62–66.

[39] D. W. Marquardt, An algorithm for Least-Squares estimation of nonlinear parameters, SIAM Journal on Applied Mathematics 11 (2) (1963) 431–441.

[40] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, New York, NY, USA, 2003.

[41] C. Harris, M. Stephens, A combined corner and edge detector, in: In Proc. of Fourth Alvey Vision Conference, 1988, pp. 147–151.

[42] W. Förstner, E. Gülch, A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centres of Circular Features (1987).

[43] S. M. Smith, J. M. Brady, Susan - a new approach to low level image processing, International Journal of Computer Vision 23 (1995) 45–78.

[44] A. Sanjeev, R. Kannan, Learning mixtures of arbitrary gaussians, in: Proceedings of the thirty-third annual ACM symposium on Theory of computing, STOC '01, ACM, New York, NY, USA, 2001, pp. 247–257.

[45] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society. Series B (Methodological) 39 (1) (1977) 1–38.

[46] D. F. Williamson, R. A. Parker, J. S. Kendrick, The box plot: a simple visual method to interpret data., Ann Intern Med 110 (11).

[47] D. Q. Huynh, Metrics for 3d rotations: Comparison and analysis, J. Math. Imaging Vis. 35 (2) (2009) 155–164.