IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 21, 2021, accepted October 1, 2021, date of publication October 5, 2021, date of current version October 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3118049

Design, Detection, and Tracking of Customized Fiducial Markers

DAVID JURADO-RODRÍGUEZ[1,2], RAFAEL MUÑOZ-SALINAS[1], SERGIO GARRIDO-JURADO[2], AND RAFAEL MEDINA-CARNICER[1]

[1]Departamento de Informática y Analisis Numérico, Campus de Rabanales, Universidad de Córdoba, 14071 Córdoba, Spain
[2]Seabery R&D, 21005 Huelva, Spain

Corresponding author: Rafael Muñoz-Salinas (rmsalinas@uco.es)

This work was supported in part by the Industrial Ph.D. Program of Córdoba University with Seabery R&D, in part by the Spanish Ministry of Economy, Industry and Competitiveness under Project TIN2019-75279-P, and in part by Fondo Europeo de DEsarrollo Regional (FEDER).

ABSTRACT Fiducial markers such as QR codes, ArUco, and AprilTag have become very popular tools for labeling and camera positioning. They are robust and easy to detect, even in devices with low computing power. However, their *industrial* appearance deters their use in scenarios where an attractive and visually appealing look is required. In these cases, it would be preferable to use customized markers showing, for instance, a company logo. This work proposes a novel method to design, detect, and track customizable fiducial markers. Our work allows creating markers templates imposing few restrictions on its design, e.g., a company logo or a picture can be used. The designer must indicate positions into the template where bits will encode a unique identifier for each marker. Then, our method will automatically create a dictionary of markers, all following the same design, but each with a unique identifier. Finally, we propose a method for detecting and tracking the markers even under occlusion, which is not allowed in traditional fiducial markers. The experiments conducted show that the performance of the customizable markers is similar to the best traditional markers systems without significantly sacrificing speed.

INDEX TERMS Customized markers, fiducial markers, ArUco, AprilTag.

I. INTRODUCTION

Fiducial markers have become a popular and efficient method to solve labeling and monocular localization problems at low cost in indoor environments. Their use has spread in a wide variety of fields, such as surgery [6], [18], robot navigation [34], [43], autonomous aerial vehicle landing [4], augmented reality applications [19], distributed autonomous 3D printing [45], human cognitive processes [2], the study of animal behaviour [1] and patient positioning in radiotherapy treatments [36] among others.

There are several desirable properties a fiducial marker system should have. It must be easy and fast automatically detecting its markers in images. Each marker should have a unique identifier, and it should be possible to estimate its position w.r.t the camera. They should be robustly detected under occlusion, varying lighting conditions, rotation, and scale.

Many markers of different types and shapes been proposed in recent years [17], [20], [31], [37] (Figure 1), of which the squared ones are most widely used nowadays [12], [33], [40]. However, despite the significant advantages of fiducial markers, their visually unappealing design constitutes a

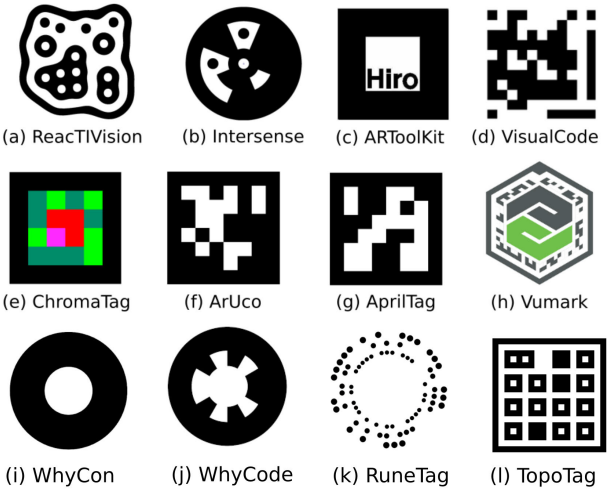


FIGURE 1. Some examples of markers proposed in previous works.

The associate editor coordinating the review of this manuscript and approving it for publication was Son Lam Phung.

deterrent for its adoption in commercial environments. Its industrial look makes them unsuitable for many use cases where a more attractive design is a must. For fiducial markers to be generally adopted in non-industrial environments, they should be able to be customizable.

The main contributions of this paper are three. First, we propose a method to design *customized* fiducial markers. Our proposal offers users the freedom to design markers templates from which a set of unique markers (dictionary) with a very similar appearance is automatically generated. Templates can be created using any vector graphics tool as long as a series of rules are followed. Each marker encodes as two different colors a unique identifier and a Cyclic Redundancy Check code (CRC) to avoid false positives. The designer chooses the position and colors of these bits. As the second contribution, we propose a method to detect these markers in images automatically. As in most existing marker systems, the detection of the marker is based on detecting its border using an adaptive thresholding algorithm, after which the position of the bits in the image is obtained using a homography. Our final contribution is a method for tracking markers by using image keypoints. Once the marker has been detected, we use a tracking process based on keypoints that allow us to know the position of the marker in the event of partial occlusion. Also, when using a calibrated camera, the relative pose w.r.t to the marker can be calculated with our method. The proposed tracking method is faster than detection and robust to partial occlusion.

The rest of this paper is structured as follows. Section II does a blue of related works while Section III introduces the design approach for customized markers templates. Section IV details the procedure for detection and tracking of customized markers, and Section V describes the experiments carried out to validate our proposal. Finally, Section VI draws some conclusions and future works.

II. RELATED WORKS

In the last years, several types of fiducial markers have been proposed (see Figure 1). ReacTIVision [17] presents a highly compact amoeba geometry, designed using genetic algorithms. Some authors have proposed the use of trained classifiers to improve ReacTIVision detection in cases of bad illumination and blurring caused by fast camera movement [7].

InterSense markers [27] are based on circles encoding information as concentric rings. They provide only one correspondence point (the center), making it necessary the use at least four points matching several circular markers for a correct camera pose estimation. WhyCon [28], and WhyCode [20] propose circular markers based on Binary Necklaces [38], a mathematical concept providing a generator for rotation invariant, uniquely identifiable patterns that can scale to a theoretically infinite number of unique markers. Cantag [31] implements two types of markers: circular (CircleTag) and squared (SquareTag). The Fourier markers [37] are based on encoding a pattern in the amplitude spectrum of the Fourier transform of the marker. RuneTag [3] proposes markers characterized by a circular arrangement of dots at fixed angles containing one or several concentric rings. TopoTag [44] uses topological and geometrical information in marker detection. Topological information is extensively used for 2D marker detection and further corresponding geometrical information for ID decoding.

Square fiducial markers have recently become the most popular ones. This type of marker consists of an external black border (easily detectable) and an inner region for employed for identification purposes. It provides four prominent points (i.e., its corners) that can be easily detected and employed to estimate the marker pose w.r.t. the camera. ARToolKit [10] was first released as an open-source project. Its markers are composed of a wide black border with an inner image stored in a database of valid patterns. Despite its success, the main drawback is the high rate of false-positive [10]. VisualCode [32] is based on the same technology as QR, and Maxicode [30].

The two most popular squared fiducial marker systems nowadays are ArUco [33] and AprilTag [29]. Both systems propose an efficient method for markers detection using robust adaptive thresholding and perform error detection and correction of the binary codes. Several works have analyzed their performance showing that speed, robustness, and accuracy are essential factors to consider, where ArUco and AprilTag markers systems show clear advantages [12], [33], [40].

CNNs have been successfully applied to many computer vision tasks in the last years [21]. However, their use for marker detection has been somewhat limited so far. Mondejar *et al.* [24] propose a method to detect ArUco markers using CNNs. They create a synthetic training dataset by applying transformations that emulate real situations and train a CNN using it. The main drawback of their approach is that it can only detect a few markers from the ArUco dictionary. Deep ChArUco [16] presents a real-time pose estimation system that combines two custom deep networks ChArUcoNet and RefineNet. Their method can only detect ArUco markers and is designed for camera calibration. However, they show that their approach outperforms ArUco detection method in challenging illumination conditions such as blurring and low light. Finally, the recent work DeepTag [46] proposes a general deep learning-based framework for fiducial marker design and detection using end-to-end convolutional neural networks. This method decomposes the problem into three stages. First, regions containing markers are spotted. Then, another net refines the region to find the marker corners accurately. Finally, the marker's inner region is analyzed to detect if the region is a marker. One of the advantages of this work over the previous CNN works is that it can detect a large variety of marker types (e.g., ArUco, AprilTag, TopoTag). However, a drawback of CNN approaches is that they need a previous training step with many samples. Another problem is their high computational demands, i.e., they require powerful GPUs to operate. While recent developments in marker detection can reach up to 1000fps

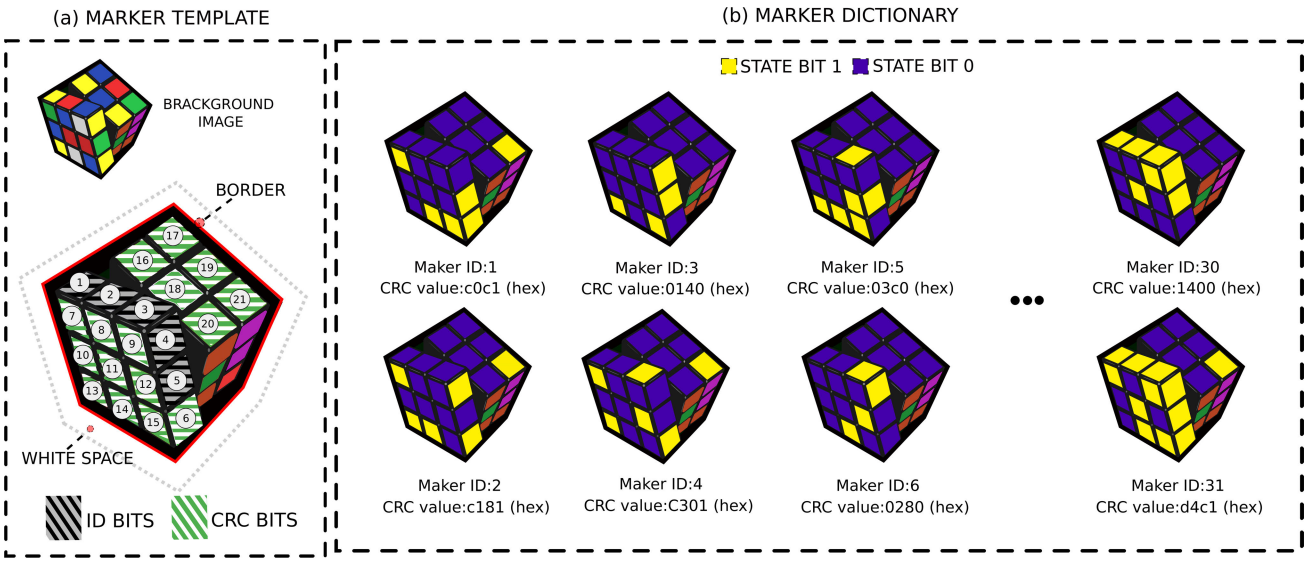


FIGURE 2. Custom marker template. (a) Elements of a marker template. At the top, the background image on which the template is created, and below the template created. Gray regions represent ID bits, which are used for marker identification. Green regions are CRC bits, which are used to include a Cyclic Redundancy Check (CRC) information to avoid false positives. (b) Marker dictionary automatically generated with the template using the yellow and blue colors for encoding.

in 4K images using just a single CPU thread [33], its deep learning counterpart [16] requires a powerful GPU to obtain 66.5fps in images of only 320 × 240 pixels.

While the systems cited above use only black and white colors, ChromaTag [9] introduces a new type of colored fiducial marker Figure 1e). The inner red square and green ring are used to reject false positives, and the outer black is employed to ensure precise marker localization. The main drawback of this approach is that sudden lighting variations decrease the performance of the algorithm detection [44].

As already indicated, very few works have addressed the problem of customizable markers. ARTTag [14] proposes a method to create them incorporating a set of circle pairs in their design. Unique markers can be generated by analyzing the size ratio between the circle pairs, which should be unique among the set of markers generated. However, as indicated by the authors, one of the limitations of their approach is that only a minimal number of identifiers are possible.

To our knowledge, the only similar method is VuMark,[1] a commercial application developed by the company Vuforia. The main drawback of VuMark is that its methodology for customized marker design, detection, and tracking is not public. It is a private company project for which no technical information is available. Nevertheless, since the binary program is available for use, it has been employed in our experiments.

III. DESIGN OF CUSTOMIZED MARKER TEMPLATES

This section presents our proposal for creating customized marker templates. Figure 2 (a) shows the main parts of the proposed customized markers and Figure 2 (b) shows eight different markers, each with a unique identifier, automatically generated using this template. The process starts by selecting an image that acts as background (Figure 2a), which must fulfill two requirements. First, it must be enclosed by a polygonal border with at least four vertices. The reason is that to obtain a good camera pose estimation, at least four points are required [8], [39]. Second, although there is no limitation in the maximum number of vertices of the polygon, it is essential to have a minimum separation between corners so that all polygon sides are correctly detected. We recommend that the minor side's length be greater than a percentage of the most significant side. In our experience, 10% is a good value.

Another aspect to consider is that when a marker is placed in the real environment, there should be enough contrast between its border and the background. It is a natural limitation of fiducial markers, such as ArUco and AprilTags, since it is impossible to distinguish the marker border if it has the same color as the background (e.g., imagine an ArUco marker placed on a black surface). Therefore, if the marker will be placed on a surface with a color similar to the marker border, a solution is printing the marker leaving a white space of at least 10% the size of the background image. In this way, we enforce good border detection.

The designer must then decide which areas of the background image are employed for bit codification. In the example of Figure 2 (a), these areas have been represented by the grey and green striped patterns. In total, 21 bits have been placed by the designer in this example. The number of bits determines how many markers can be created with a template. To avoid false positives, we reserve some bits for Cyclic Redundancy Checking (CRC) [5] (green striped pattern), and the rest for identification (grey striped pattern).

[1] https://library.vuforia.com/articles/Training/VuMark.html

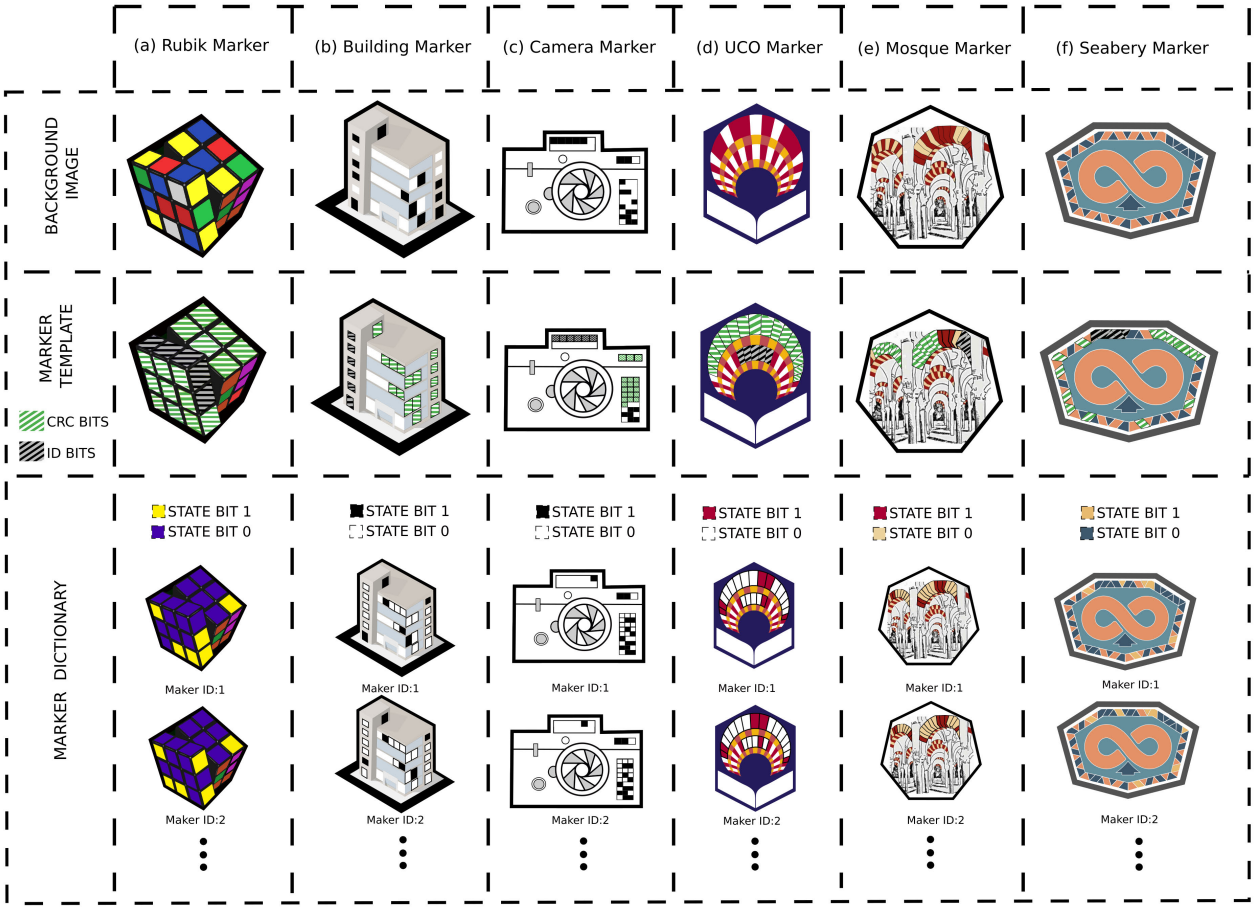


FIGURE 3. Six marker templates designed with our method. The top row shows the background images employed and below the templates created. The bottom row shows some of the markers of each dictionary.

As will be explained in the next section, many marker candidates will be extracted from background objects and clutter in the images, and the system must analyze most of them to find valid markers. To prevent false detections, we recommended employing as many CRC bits as possible. In our example, we have employed sixteen bits for CRC, so the remaining $\epsilon = 5$ bits are employed for identification, giving us a total of $2^5 = 32$ different markers.

To choose the colors employed to represent the bits on the marker is the last designer decision. Each bit can have two states, *zero* and *one*, represented by two different colors with, preferably, a high difference between them. Although the colors employed do not have any restrictions, we recommend that they be as different as possible, either in the illumination space (dark-light) or in the chromatic space. Our detection process (as explained later) is color agnostic, i.e., it does not needs to know the exact colors employed for codification. Instead, it extracts the colors in each image and does a clustering assuming a bi-modal distribution. In other words, our detection algorithm does not match template colors with image colors but threshold the observed binary distribution. This approach allows the method to adapt to illumination changes.

Figure 2 (b) shows eight different markers generated using the template. As can be seen, our method only modifies and analyzes the regions where the bits are placed, allowing the rest of the template area to have any desired design and color.

Once the customized marker is designed, it will be exported as a Scalable Vector Graphics (SVG) file, which must include the color of the bits states. Then, using the SVG file as input, we can automatically generate the dictionary of makers, i.e., all the markers with different identifiers that can be generated given the number of bits ϵ available for encoding.

IV. CUSTOMIZED MARKER DETECTION AND TRACKING

This section provides a detailed explanation of our proposal to detect and track customized markers. Detection (subsection IV-A) consists in finding the contours of the image and test if any of them correspond to a valid marker. Once a marker is detected, tracking (subsection IV-B) consists in creating a map of keypoints that is dynamically updated along the video sequence to estimate the marker position (subsection IV-C).

Let us assume we have a customized marker with $m_V \geq 4$ polygon vertices and m_B identification bits. The proposed

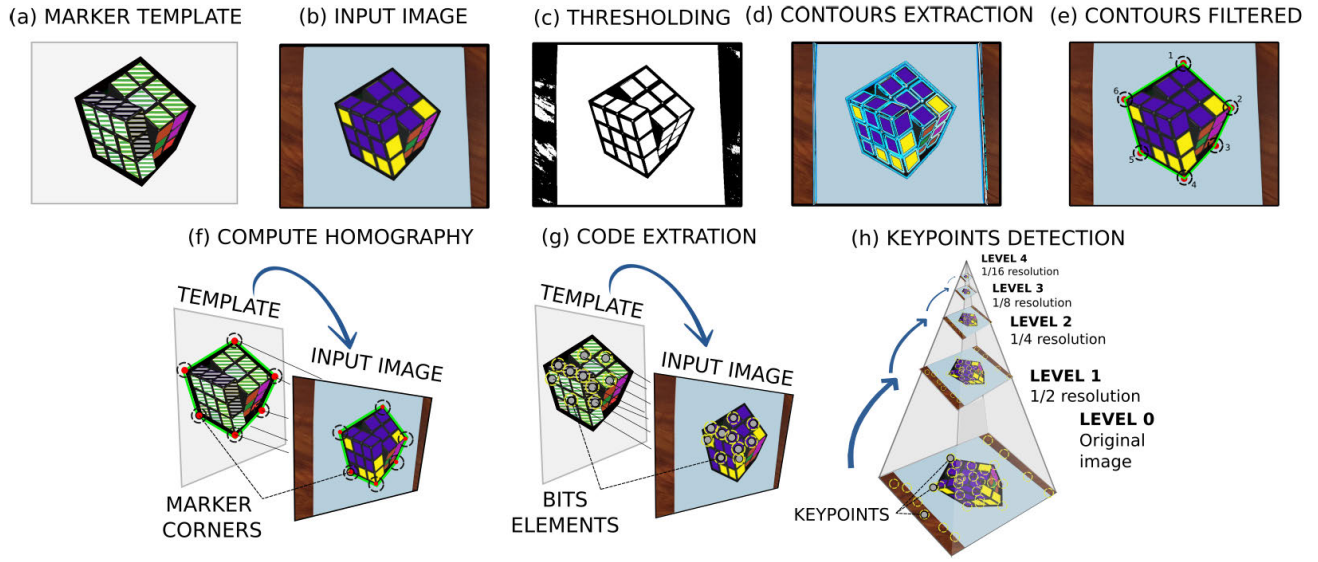


FIGURE 4. Detection pipeline of the proposed method. a) Marker template. b) Image of a marker. c) Results of adaptive thresholding. d) Extracted contours are shown in blue. e) Polygons of six sides found shown in green. f) Homography from the template to the input image. g) Centers of the bits regions computed with the homography. h) For detected markers, key points are extracted from an image pyramid for tracking purposes.

method uses a model $\mathcal{M} = \{V, B\}$ to detect and track a marker, where

$$V = \{\{\mathbf{v}_i\}, i \in \{1, \ldots, m_V\}, \mathbf{v}_i = (x_i, y_i) \in \mathbb{R}^2\},$$

represents the set of marker vertices and

$$B = \{\{\mathbf{b}_j\}, j \in \{1, \ldots, m_B\}, \mathbf{b}_j = (x_j, y_j) \in \mathbb{R}^2\},$$

is the marker bits set. In sets V and B, (x, y) represents the coordinates specified in the SVG file.

A. MARKER DETECTION

The main steps for detection and identification of customized markers presented in this proposal are depicted in Figure 4. Given a customized maker (Figure 4 a) and an input image I (Figure 4 b), the following steps are employed.

1) *Image segmentation:* Since the customized markers have an external contour surrounded by white space, we use a local adaptive thresholding method to find borders (Figure 4 c).
   For every pixel, we analyze its average brightness (in a small neighborhood of size $w_t \times w_t$) and we determine is above a threshold τ_t. This method is robust and obtains good results for a wide range of values τ_t and w_t. Previous experience [13] show that the value $\tau_t = 7$ and $w_t = 6$ are good values in most cases.
2) *Contour detection and filtering:* Contours are extracted using the Suzuki and Abe algorithm [41] (Figure 4 d). Then, a polygonal approximation is obtained using the method proposed in [22] (Figure 4 e), discarding the polygons with a number of sides different from m_V.
   Let

$$\mathcal{P} = \{\{V_p\}, p \geq 1\},$$

be the set of accepted polygons in image I where V_p represents its vertices in the image, i.e.,

$$V_p = \{\{\mathbf{v}_{ip} \in \mathbb{R}^2\}, i \in \{1, \ldots, m_V\}\}.$$

3) *Initial detection of a valid marker:* The objective is to determine if there is a polygon in $\mathcal{P}$ that is a valid marker.
   For each polygon $V_p \in \mathcal{P}$, the m_V possibles circular shifts of its vertices are considered. A shift is equivalent to a polygon rotation, thus obtaining

$$\{\{V_p^r\}, r \in \{1, \ldots, m_V\}\}.$$

For each rotated polygon V_p^r the homography matrix H_p^r transforming the vertices from the image to the model, $V = H_p^r V_p^r$, is calculated (Figure 4 f). The homography is employed to determine the expected image position of the bits into the polygon.

$$\begin{aligned} B_p^r &= (H_p^r)^{-1} B, \\ B_p^r &= \{\{\mathbf{b}_{pj}^r \in \mathbb{R}^2\}, j \in \{1, \ldots, m_B\}\}. \end{aligned} \tag{1}$$

The color of the pixels B_p^r need to be evaluated in order to extract the bits values of the polygon:

$$\mathcal{D}(B_p^r) = (d_1, \ldots, d_j, \ldots, d_{m_B}),$$

where

$$d_j \in \{0, 1\} \forall j \in \{1, \ldots, m_B\}.$$

If the bit sequence correspond to a valid one, then a marker has been found.
To obtain the values d_j, we know that the colors employed for the bits form two different clusters. However, the colors specified in the marker design (indicated in the SVG file) are not reliable. The printing

process as well as the unknown camera color transfer function makes unreliable making assumptions about the colors originally specified in the design. Thus, bit color information for each $\mathbf{b}^r_{pj} \in B^r_p$ is done by analyzing the observed colors in the I input image. We assume that the number of bits in zero and one states (d_j values) is similar for a given marker. Under the previous assumption, the goal is to find the threshold t_c that determines the color (d_j value zero or one). In some cases, the mean brightness can be employed as the threshold, but it is necessary to employ chromatic information in other cases. In particular, we employ the circular mean of the Hue component from the HSV space. For every marker, we employ both approaches to decode the bits.

The assumption of having a similar number of bits in zero and one state has proven to work. The CRC bits represent the majority of the bits, and their value can be considered random. The only exception happens for the value zero (all bits to zero), which produces also produces the CRC value zero. For that reason, the marker with that value is not employed in our codification.

Once the bit sequence $\mathcal{D}(B^r_p)$ is obtained, we analyze if it is valid by checking the CRC. If the ϵ data bits used for marker identification

$$\mathbf{D}(B^r_p) = (d_1, \ldots, d_\epsilon),$$

and its CRC bits matches, i.e.,

$$CRC(\mathbf{D}(B^r_p)) = (d_{\epsilon+1}, \ldots, d_{m_B}), \tag{2}$$

the marker with identification $\mathbf{D}(B^r_p)$, which we will denote by $\mathbf{D}$ in the rest of the paper, has been detected. The corresponding homography H^r_p, which we will denote by H, will also be employed in next step (see Figure 4 g).

4) *Tracking model initialization:* Our tracking proposal is based on keypoint detection and matching. For that purpose, a model of key points is created representing the marker texture. The model will be dynamically updated along the video sequence to account for light, scale, and viewpoint changes.

Once a marker has been detected, a collection of ORB keypoints [35] are obtained from the image I employing the same rationale as in the recent UcoSLAM [25] and ORB-SLAM [26] methods.

First, a image pyramid of the original image I is created and keypoints k'_l are detected at different scales levels (p_l). Let

$$K' = \{\mathbf{k}'_l, l \in \{1, \ldots, N_{kp}\}, \mathbf{k}'_l = (x'_l, y'_l, p_l)\}, \tag{3}$$

be the keypoints set. key points in the first levels (higher resolutions), represents smaller areas than these in the lowest levels (see Figure 4 h). Second, to avoid the concentration of key points in small regions of the image, each image is subdivided into a grid, and a proportional number of key points is obtained from each one of them.

Then we use the homography matrix H to obtain K, which transforms the keypoint locations to the marker reference system, i.e.:

$$\begin{aligned} K &= (H^{-1}K') \cap \Delta(V), \\ K &= \{\{\mathbf{k}_l\}, l \in \{1, \ldots, N_{kp}\}, \mathbf{k}_l = (x_l, y_l, p_l)\}, \end{aligned} \tag{4}$$

where $\Delta(V)$ are the keypoints within the polygon V. Let us denote

$$\mathcal{M}^{\mathbf{D}} = \{V, K, H\},$$

the initial tracking model of the marker $\mathbf{D}$. As it will be explained later, the model is dynamically updated to deal with image changes along the video sequence.

B. MARKER TRACKING

This section explains the tracking process in detail. Let $\mathcal{F} = \{\{f\}, f > 1\}$ represent the frames of a video sequence, and $f = 1$ be the first frame of the sequence where the marker was detected as previously explained.

The aim of the tracking algorithm is to find the marker in frame f assuming it was detected at frame $f-1$, for which the model

$$\mathcal{M}^{\mathbf{D}}_{f-1} = \{V, K_{f-1}, H_{f-1}\},$$

is known. These are the steps followed by our algorithm:

1) *Image KeyPoint Extraction:* Keypoints of frame f are obtained as explained before obtaining

$$K' = \{(\mathbf{k}'_l), l \in \{1, \ldots, N_{kp}\}, \mathbf{k}'_l = (x'_l, y'_l, p'_l)\}.$$

2) *Image-Model KeyPoint Matching:* The keypoints locations are translated to the marker reference system using the homography computed in the previous frame:

$$K^0_f = H^0_f K' \tag{5}$$

where $H^0_f = H_{f-1}$. These are initial estimations of their position around which the matching process will be done. For each keypoint in K^0_f, the corresponding model keypoints $\mathbf{k}_l \in K_{f-1}$ within a circle of radius r are selected. Amongst them, the one minimizing the Hamming distance (with a maximum α threshold) is considered as the best match (Figure 4 h). However, to consider a match, the model and image key points must be in similar pyramid levels, allowing only a difference of one level. The set of matches obtained is employed to compute an updated homography H^1_f, along with updated locations of the image keypoints

$$K^1_f = H^1_f K'.$$

This provides an initial estimation refined further by repeating the matching process, but this time reducing the radius search to $r/2$. The matches obtained then are employed to compute the final homography $H_f = H^2_f$

and the corresponding set K_f^2. If the number of matches is more significant than a threshold β, the marker is considered as found.

3) *Keypoints model update:* To adapt the model to viewpoint and illumination changes, it must be dynamically updated every frame. Please notice that many of the model key points are only matched once and never matched again. Thus, we perform a pruning process, inspired by UcoSLAM [25], to avoid keeping in the model useless key points. The idea is that a key point is removed from the model if it is not matched in at least 10 frames, after the next 20 frames since its insertion in the model. If, as a consequence of the pruning process, the total number of key points in the model falls below a certain threshold γ, new key points are added to the model, selecting amongst these inside the polygon that has not been matched. As a result, the updated keypoint model K_f is obtained.

As consequence from previous steps the updated model is obtained from frame f:

$$\mathcal{M}_f^{\mathbf{p}} = \{V, K_f, H_f\}.$$

C. POSE ESTIMATION

The pose $\theta = (\mathbf{r}, \mathbf{t})|\mathbf{r}, \mathbf{t} \in \mathbb{R}^3$ of a marker w.r.t the camera reference system is defined by its three rotational and translational components $\mathbf{r} = (r_x, r_y, r_z)$ and $\mathbf{t} = (t_x, t_y, t_z)$. Using the Rodrigues' rotation formula, the rotation matrix R can be obtained from $\mathbf{r}$.

The estimation of the marker pose is known as the PnP problem, which can be solved by minimizing the reprojection error of the marker vertices, as explained below. Let us denote

$$U = \{\mathbf{u}_i \in \mathbb{R}^2\}$$

the image position of the marker vertices V in the current frame f. It can be obtained using the calculated homography H_f as:

$$\mathbf{u}_i = H_f \mathbf{v}_i, \mathbf{v}_i \in V. \tag{6}$$

The position $\mathbf{u}_i$ is an initial estimation that can be further refined with subpixel accuracy using the method proposed in [11]. The algorithm analyzes a window of length w_{vr} around the vertices to find the maximum gradient within the region. To refine the vertices, there must be a minimum separation between them. This the reason why a minimum separation p_{vs} between consecutive vertices was required in the marker design (see section III). As a result, we obtain the refined vertices positions

$$U' = \{\mathbf{u}_\mathbf{i}'\}.$$

On the other hand, using the standard pin-hole camera model, the marker vertices $\mathbf{v_i}$ can be projected on the camera plane into the pixel $\mathbf{p_i} \in \mathbb{R}^2$ as

$$\mathbf{p_i} = \Psi(\delta, \theta, \mathbf{v}_\mathbf{i}') \tag{7}$$

where $\mathbf{v}_\mathbf{i}' = (\mathbf{v_i}, 0)$ is the three-dimensional representation of the vertex, $\delta = (f_x, f_y, c_x, c_y, l_1, \ldots, l_n)$ refers to the camera intrinsic parameters, being (f_x, f_y) focal distances, (c_x, c_y) optical center and $(l_1, \ldots, l_n)$ the distortion parameters.

Then, estimating the pose of the marker $\hat{\theta}$ is the problem of minimizing the reprojection error of the observed marker vertices, i.e.:

$$\hat{\theta} = argmin \sum_i [\Psi(\delta, \theta, \mathbf{v}_\mathbf{i}') - \mathbf{u}_\mathbf{i}']^2. \tag{8}$$

Equation 8 is a non-linear function that can be efficiently minimized using the Levenberg–Marquardt's (LM) algorithm [23].

V. EXPERIMENTS AND RESULTS

This section presents the experiments conducted to validate the performance of the proposed method, which along this section will be named JuMarker. We analyze its accuracy, speed and robustness, comparing the results with the state-of-the-art alternatives, namely ArUco [33], AprilTag [29], TopoTag [44], DeepTag [46] and VuMark.[2] We must remark that VuMark is a commercial project for which the code is not available. For our method, the values indicated in Table 1 have been employed, while in the rest of the methods, default settings have been employed.

TABLE 1. Parameters values used in quantitative evaluation of our proposal.

Parameter	Value	Comment
p_{vs}	10	Percentage for marker design (section III)
w_t	6	Neighbourhood size for image segmentation (subsection IV-A)
τ_t	7	Threshold for image segmentation section IV)
t_c	0.6	Threshold for bits states colors (section IV)
N_{kp}	6000	Maximum number of keypoints (section IV)
r	(5, 30)	Circle radius for matching(subsection IV-B)
α	50	Maximum threshold in matching (subsection IV-B)
β	20	Matches to consider the marker has been found (subsection IV-B)
w_{vr}	0.5	Window length around vertices(subsection IV-B)
γ	14	Minimum matches to continue the tracking process (subsection IV-B)

To properly test the detection and tracking capabilities of the JuMarker method, we have designed six different customized markers templates that are shown in Figure 3(a-f). They all have 16 CRC bits, and the number of ID bits is 5, 6, 6, 5, 3 and 5 respectively. These markers are also detectable by the VuMark system.

To validate ArUco, AprilTag, and DeepTag methods, an AprilTag marker (Figure 1(g)) has been employed, since both ArUco and DeepTag system is trained to detect an AprilTag marker. To test the TopoTag method, it has been necessary to use one of its markers as shown in Figure 1(l).

In order to achieve a fair comparison between the different methods, we have printed an AprilTag and a TopoTag marker on the same piece of paper for each customized marker (see Figure 5). In this way, the video sequences recorded for evaluation show the three markers, and the methods can be compared under similar distances and illumination conditions.

Since we have used six different customized markers for the JuMarker and VuMark methods, analyzing the

[2]https://library.vuforia.com/articles/Training/VuMark.html

performance individually for each customized marker allows us to identify the main differences between JuMarker and VuMark. In addition, the average of the six is used to compare against the ArUco, AprilTag, TopoTag, and DeepTag methods.

The experiments have been conducted using a camera with resolution 1280 × 720, and the methods have been run in a single CPU of an Intel Core^{TM} i7-10510U 2.30GHz with 8GB RAM running Ubuntu 20.04.1 operating system.

The rest of this Section is structured as follows. Section V-A compares the accuracy of each method in camera pose estimation while V-B analyses how the distance affects the precision. Section V-C studies the vertex jitters and V-C the robustness to occlusion. Finally, Section V-E shows the computing times of the tested methods.

A. METHOD ACCURACY

The goal of this experiment is to analyze the precision of our method in estimating the pose of a camera. For each customized marker designed (see Figure 3(a-f)), we have recorded four video sequences spotting at the markers from different distances and angles as it is shown in Figure 5. A total of 12 video sequences with 1265 images each one as average have been recorded.

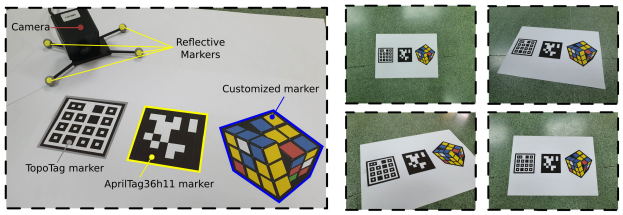


FIGURE 5. Experiments Setup. A camera attached with reflective markers has been employed for recording a piece of paper with three markers printed on it. The OptiTack motion capture system has been used to obtain the ground truth camera poses in the environment while recording.

To obtain the ground truth, the commercial motion capture system OptiTack [3] has been employed. It has a set of infrared cameras and requires us to attach several spherical infrared reflective markers to our camera to estimate its pose in the environment with sub-millimeter accuracy. The system provided us with the ground truth camera pose in every recorded frame.

The Absolute Trajectory Error (ATE) (the difference between the estimated camera position and the ground truth along the complete recorded video sequence) is used as an error measure. To compute the ATE, it is necessary to align and scale the trajectories computed by each method w.r.t. the ground truth, which is done using the Horn's algorithm [15].

Figure 6 shows (1) the errors for each customized marker employing VuMark and JuMarker and (2) the average of ATE achieved by each method. The VuMark method has the option to process the videos at 30 and 60 fps, and both have

[3]https://www.optitrack.com/software/

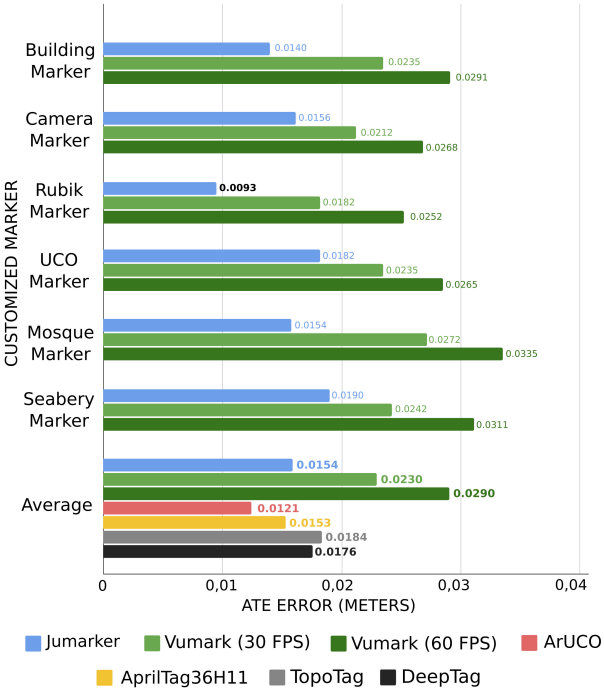


FIGURE 6. The Absolute Trajectory Errors (ATE) obtained in camera pose estimation, using JuMarker and VuMark methods for each marker designed in Figure 3 (a-f) and the average error, with respect to ground truth.

been employed to observe the differences. As can be seen, depending on the customized marker evaluated, the results obtained are different. However, it is clear that the proposed method (JuMarker) outperforms VuMark in all the tests and that VuMark processing at 60 fps is worst than at 30 fps. It must be remembered that both methods are employed exactly with identical input video sequences. At the bottom of Figure 6, the average errors for all methods is shown. As we can see, the ArUco and AprilTag methods outperform customized markers in the tests performed. However, we would like to draw attention to the Rubik marker, which offers the minor error of all. Its error (0.0093 *meters*) is smaller than the average obtained employing ArUco or AprilTag methods. Regarding the accuracy of TopoTag and DeepTag, we should mention that both perform worse than ArUco, AprilTag, and JuMarker but better than VuMark.

Despite the better performance of the AprilTag and ArUco methods, the average difference with our proposal is only 2.5 mm. To assess if the differences observed are relevant, we have employed the non-parametric Wilcoxon test [42]. At a 0.99 as confidence level, the test indicates that JuMarker method does not have statistically significant differences w.r.t AprilTag and ArUco methods. However, if we compare the results of JuMarker w.r.t VuMark, the differences are statistically significant in favor of our method. The statistical tests show that our method outperforms TopoTag and DeepTag too. Additionally, there are statistically significant differences between the two VuMark versions. As a consequence,

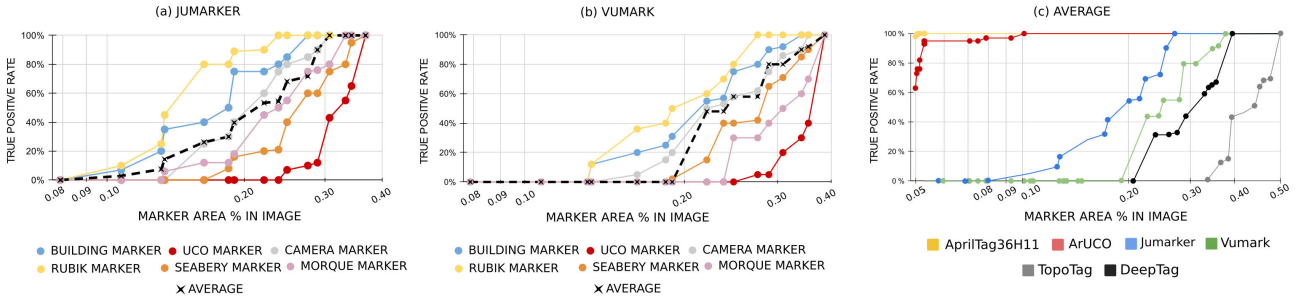


FIGURE 7. True Positive Ratio (TPR) as function of the marker area observed in the total images. The images (a) and (b) display the TPR for each customized using JuMarker and VuMark methods respectively and image (c) show the average performance for each method.

only the 30 fps version will be employed in the following experiments.

In conclusion, we can indicate that ArUco and AprilTag are the most accurate methods in camera pose estimation. Moreover, the performance of our method outperforms VuMark, TopoTag and DeepTag in terms of accuracy.

B. ANALYSIS OF DETECTION RANGES

This experiment compares the detection and tracking ranges of the different methods, i.e., the range of distances at which the marker is detected and tracked. For that purpose, we evaluate the True Positive Rate (TPR) in 6 video sequences (accounting for a total of 6452 images) that have been recorded, spotting the marker at distances ranging from 2.5 to 0.10 meters.

It must be considered that the method's success depends on the observed size of the marker in the image. The larger the marker is seen on the image, the higher the success. However, the observed size of the marker depends on three factors: the actual marker size, the camera focal length, and the distance from the marker to the camera.

Therefore, to make the analysis independent from these parameters, we have reported the success of each method. We have taken into account the observed area of the marker in the image divided by the total image area, which is a normalized value in the range [0, 1].

Figure 7(a, b) shows the True Positive Rate (TPR) obtained using JuMarker and VuMark methods for each customized marker in Figure 3 (a-f). Please notice that the results of each marker are represented with colored lines, while the black line represents the average of all of them. Figure 7(c) shows the TPR for all methods used (ArUco, AprilTag, JuMarker, TopoTag, and DeepTag). Several conclusions can be drawn from the results obtained. First, AprilTag is the method that achieves the best results, reaching a 100% of TPR when the marker occupies as little as 0.050% of the image, followed by ArUco with 0.054%. Second, JuMarker obtains a 100% TPR when the marker occupies at least 0.26% of the image, while VuMark requires 0.38% of the image area for the same result. Finally, TopoTag and DeepTag methods require the marker to occupy 0.40% and 0.50% respectively to achieve a 100% TPR. In other words, they cannot deal properly with small markers.

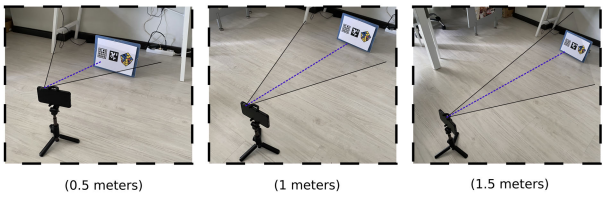


FIGURE 8. Test sequences. Distances between the marker and camera employed in video sequences for the Vertex Jitter evaluation.

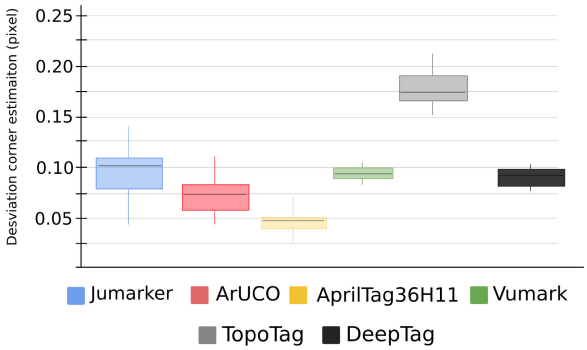


FIGURE 9. Average, minimum and maximum errors in corner estimation of JuMarker, ArUco, AprilTag, VuMark, TopoTag and DeepTag methods.

C. VERTEX JITTER ANALYSIS

Vertex jitter refers to the inaccuracy in the estimation of the marker corners, which is an important aspect affecting negatively the quality of the camera pose estimated. This is why the methods tested implement an algorithm to estimate the corner's locations with sub-pixel accuracy.

We have recorded five video sequences for each customized marker at three different distances {0.50, 1.00, 1.50} *meters*. In total, we account for 18 videos comprising 8572 images. Figure 8 shows the setup employed. For each sequence, both the camera and the marker are static. Then, the ground truth location of a corner can be precisely estimated as the average observed position along the whole

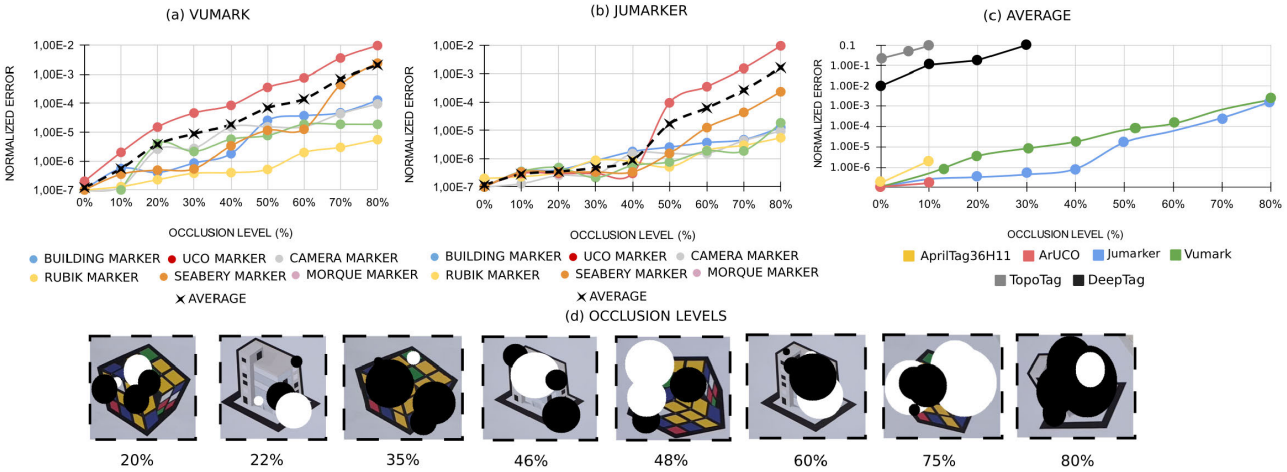


FIGURE 10. **Normalized translational error of each method as a function of the occlusion level. (a) Errors of the JuMarker method for each customized marker, and the average results in black. (b) Errors of the VuMark method. (c) Average results of all methods. (d) Some of the images employed in the experiment.**

video sequence, and the error measured as the standard deviation [13].

Figure 9 shows the average, minimum and maximum errors for all methods. As can be observed, AprilTag and ArUco achieve the best results: 0.045 and 0.059 pixels, respectively. JuMarker obtains an error of 0.1 pixels while the VuMark method 0.09. Finally, TopoTag and DeepTag obtain errors of 0.18 and 0.08 pixels, respectively

The non-parametric Wilcoxon test [42], using a 0.99 confidence level, shows that there are statistically significant differences between JuMarker, ArUco, and AprilTag methods. However, JuMarker and VuMark do not show significant differences between them. Using the same confidence level in the Wilcoxon test, JuMarker has been compared to TopoTag and DeepTag. As a result, it has been obtained that JuMarker does show statistical differences to TopoTag, but not with DeepTag. Therefore, we conclude that ArUco and AprilTag are the best methods. Then, JuMarker, VuMark, and DeepTag exhibit similar performance, while TopoTag is the worst one.

D. OCCLUSION ANALYSIS

The main goal of this experiment is to analyze the robustness and precision of the methods under different degrees of occlusion. To produce systematic occlusion, we have printed the markers and taken snapshots at different locations. Then circles of random radius have been overlaid at random locations into the marker area (see Figure 10 (d)). The color of the circles is randomly selected as white or black. Since it is a synthetic occlusion, the percentage of occlusion of the marker is known.

We have captured ten images at different viewpoints for each marker within a distance ranging from 0.2 to 0.8 meters. Because there are six markers, a total of 60 images have been captured under controlled indoor illumination conditions. Afterward, we randomly generated 60 synthetic images for each original image by superimposing a random number of circles of random color and random radius obtaining occlusion levels in the interval $[0\%, 80\%]$. As a consequence, we have used a total of 3600 images for this experiment.

For every image, the methods have been applied to compute the camera pose in original and synthetically generated images, considering the pose of the original image as the ground truth. Then, the translational error between the two poses has been employed as the error measure. However, to normalize the error so that it is not affected by the distance, it has been divided by the visible marker area.

Figure 10(a, b) shows the errors of the JuMarker and VuMark methods as a function of the occlusion levels. Each colored line corresponds to a custom marker, while the black one represents the average error. As can be seen, the UCO Marker is the worst for both methods, and the Rubik one is the best for VuMark. For JuMarker, there are very similar results, and it is impossible to claim which one is the best.

Figure 10(c) compares the methods using the average values of each one. It can be concluded that JuMarker and VuMark perform better than the rest under occlusion. Both methods detect the marker when the occlusion level is less than 80%. In addition, it can be seen that JuMarker performs better than VuMark. The marker is not detected by DeepTag when the occlusion is higher than 30%, and the same is true for TopoTag, ArUco, and AprilTag when occlusion is higher than 10%.

E. COMPUTING TIME

Estimating the pose of markers in real-time is an essential objective of any method. Then, this section aims to analyze the processing times of our proposal and compare it with the rest of the methods. To that end, we have recorded fifty-five video sequences comprising a total of 6.580 images with a resolution of 1280×720.

As explained, our method automatically chooses between detection and tracking depending on the situation. Table 2

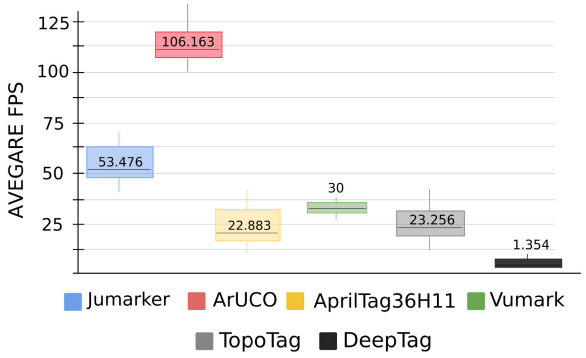


FIGURE 11. Computing times of Jumarker, ArUco, AprilTag, Vumark, TopoTag and DeepTag.

TABLE 2. Computing times of the detection steps of JuMarker.

Initial detection	Avrg(ms)
Step 1: Image segmentation	4.2
Step 2: Contour extraction and filtering	5.9
Step 3: Initial detection of a valid marker	3.6
Step 4: Tracking model initialization	3.4
Total	17.1

TABLE 3. Computing times (in milliseconds) of the different steps involved in Tracking.

Tracking based on keypoint	Avrg(ms)
Step 1: Image keypoint extraction	3.2
Step 2: Image-model keypoint matching	4.9
Step 3: Keypoint model update	3.6
Total times	11.7

shows the average computing times employed in each one of the detection steps (see subsection IV-A). As can be observed, the *image segmentation* and the *contour extraction and filtering* account for almost 50% (10 ms) of the processing time. Both steps are crucial to ignore irrelevant information in the image and to detect the customized marker.

Table 3 shows the processing times for each step involved in the tracking approach employed (see subsection IV-B). As we can see, the *Match between model and image* step requires around 28% of processing time. It must be remarked that this value depends on the number of key points detected in the image, which is a parameter. In our case, we have set this value to 6000 as already indicated (see Table 1).

Finally, Figure 11 shows the computing times of each method in video sequences with a resolution of 1280 × 720. Please, notice that for JuMarker and VuMark methods, the type of customized marker used does not affect the computing times. As can be seen, the fastest method is ArUco with 106 fps, and the slowest is DeepTag with 1.3 fps. Regarding TopoTag, this method presents a computation time of 23.256 fps. Also, notice that the JuMarker method is the second-fastest one.

VI. CONCLUSION AND FUTURE WORK

This paper has proposed a novel method to design, detect and track customized markers as an alternative to the traditional squared makers that have a more *industrial* appearance. The proposed methodology allows creating markers, including any image in it, adapting better to the particularities of the use case it is applied to, allowing its adoption in many commercial areas.

To our knowledge, there is only a private project named VuMark, with a similar capability, but it is not possible to know the way it operates. As shown experimentally, the proposed method outperform VuMark, TopoTag, and DeepTag methods in terms of accuracy, detection range, occlusion, and computing time. Nonetheless, the performance of our proposal is worse than ArUco and AprilTag in terms of accuracy, detection range, and vertex jitter, but it is better considering occlusion. As far as computing time is concerned, our proposal ranks second.

However, the proposed method has some disadvantages. First, compared to traditional marker systems, such as ArUco and AprilTags, its performance is worse as the distance to the marker increases since the total area of the marker used for bit encoding is smaller. Second, the detection of our markers is slower than that of ArUco markers. Finally, our current implementation is limited to polygons and does not allow curves.

In conclusion, we must say that the price one has to pay to use customized markers is a slight reduction in the accuracy and speed. However, this paper proves that the reduction is insignificant and that the proposed method is a valid option for commercial and research applications.

In future work, we consider using the proposed markers for Simultaneous Localization and Mapping tasks. Our idea is to take advantage of customized markers to locate the camera position in a controlled indoor environment.

REFERENCES

[1] G. Alarcón-Nieto, J. M. Graving, J. A. Klarevas-Irby, A. A. Maldonado-Chaparro, I. Mueller, and D. R. Farine, "An automated barcode tracking system for behavioural studies in birds," *Methods Ecol. Evol.*, vol. 9, no. 6, pp. 1536–1547, 2018.
[2] M. Behroozi, A. Lui, I. Moore, D. Ford, and C. Parnin, "Dazed: Measuring the cognitive load of solving technical interview problems at the white-board," in *Proc. 40th Int. Conf. Softw. Eng., New Ideas Emerg. Results*, May 2018, pp. 93–96.
[3] F. Bergamasco, A. Albarelli, E. Rodola, and A. Torsello, "RUNE-tag: A high accuracy fiducial marker with strong occlusion resilience," in *Proc. CVPR*, Jun. 2011, pp. 113–120.
[4] M. Bhargavapuri, A. K. Shastry, H. Sinha, S. R. Sahoo, and M. Kothari, "Vision-based autonomous tracking and landing of a fully-actuated rotorcraft," *Control Eng. Pract.*, vol. 89, pp. 113–129, Aug. 2019.
[5] G. Castagnoli, S. Bräuer, and M. Herrmann, "Optimization of cyclic redundancy-check codes with 24 and 32 parity bits," *IEEE Trans. Commun.*, vol. 41, no. 6, pp. 883–892, Jun. 1993.
[6] H. Choi, Y. Park, S. Lee, H. Ha, S. Kim, H. S. Cho, and J. Hong, "A portable surgical navigation device to display resection planes for bone tumor surgery," *Minimally Invasive Therapy Allied Technol.*, vol. 26, no. 3, pp. 144–150, 2017.
[7] D. Claus and A. W. Fitzgibbon, "Reliable automatic calibration of a marker-based position tracking system," in *Proc. 7th IEEE Workshops Appl. Comput. Vis. (WACV/MOTION)*, Jan. 2005, pp. 300–305.

[8] T. Collins and A. Bartoli, "Infinitesimal plane-based pose estimation," *Int. J. Comput. Vis.*, vol. 109, no. 3, pp. 252–286, Sep. 2014.

[9] J. DeGol, T. Bretl, and D. Hoiem, "ChromaTag: A colored marker and fast detection algorithm," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1472–1481.

[10] M. Fiala, "Comparing ARTag and ARToolkit plus fiducial marker systems," in *Proc. Int. Worksho Haptic Audio Vis. Environ. Appl. (IREE)*, 2005, pp. 6–7.

[11] W. Förstner and E. Gülch, "A fast operator for detection and precise location of distinct points, corners and centres of circular features," in *Proc. Intercommission Conf. Fast Process. Photogrammetric Data (ISPRS)*, 1987, pp. 281–305.

[12] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognit.*, vol. 51, pp. 481–491, Mar. 2016.

[13] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognit.*, vol. 47, no. 6, pp. 2280–2292, 2014.

[14] S. Higashino, S. Nishi, and R. Sakamoto, "ARTTag: Aesthetic fiducial markers based on circle pairs," in *Proc. ACM SIGGRAPH Posters (SIGGRAPH)*, New York, NY, USA, 2016, pp. 1–2.

[15] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 4, no. 4, pp. 629–642, Apr. 1987.

[16] D. Hu, D. DeTone, and T. Malisiewicz, "Deep ChArUco: Dark ChArUco marker pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8428–8436.

[17] M. Kaltenbrunner and R. Bencina, "ReacTIVision: A computer-vision framework for table-based tangible interaction," in *Proc. 1st Int. Conf. Tangible Embedded Interact. (TEI)*, 2007, pp. 69–74.

[18] M. Koeda, D. Yano, N. Shintaku, K. Onishi, and H. Noborio, "Development of wireless surgical knife attachment with proximity indicators using ArUco marker," in *Human-Computer Interaction. Interaction in Context*, M. Kurosu, Ed. Cham, Switzerland: Springer, 2018, pp. 14–26.

[19] V. Lepetit and P. Fua, "Monocular model-based 3D tracking of rigid objects: A survey," *Found. Trends Comput. Graph. Vis.*, vol. 1, pp. 1–89, Jan. 2005.

[20] P. Lightbody, T. Krajník, and M. Hanheide, "A versatile high-performance visual fiducial marker detection system with scalable identity encoding," in *Proc. Symp. Appl. Comput.*, Apr. 2017, pp. 276–282.

[21] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, Jan. 2020.

[22] F. J. Madrid-Cuevas, E. J. Aguilera-Aguilera, A. Carmona-Poyato, R. Muñoz-Salinas, R. Medina-Carnicer, and N. L. Fernández-García, "An efficient unsupervised method for obtaining polygonal approximations of closed digital planar curves," *J. Vis. Commun. Image Represent.*, vol. 39, pp. 152–163, Aug. 2016.

[23] K. Madsen, H. B. Nielsen, and O. Tingleff, *Methods for Non-Linear Least Squares Problems*, 2nd ed. Lyngby, Denmark: Informatics and Mathematical Modelling, Technical University of Denmark, 2004.

[24] V. Mondéjar-Guerra, S. Garrido-Jurado, R. Muñoz-Salinas, M.-J. Marín-Jiménez, and R. Medina-Carnicer, "Robust identification of fiducial markers in challenging conditions," *Expert Syst. Appl.*, vol. 93, no. 1, pp. 336–345, 2018.

[25] R. Muñoz-Salinas and R. Medina-Carnicer, "UcoSLAM: Simultaneous localization and mapping by fusion of keypoints and squared planar markers," *Pattern Recognit.*, vol. 101, May 2020, Art. no. 107193.

[26] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[27] L. Naimark and E. Foxlin, "Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker," in *Proc. Int. Symp. Mixed Augmented Reality*, 2002, p. 27.

[28] M. Nitsche, T. Krajnik, P. Cizek, M. Mejail, and T. Duckett, "WhyCon: An efficient, marker-based localization system," in *Proc. IEEE/RAS IROS Workshop Open Source Aerial Robot.*, Hamburg, Germany, Sep. 2015, pp. 1–2.

[29] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3400–3407.

[30] E. Ouaviani, "A common image processing framework for 2D barcode reading," in *Proc. 7th Int. Conf. Image Process. Appl.*, vol. 465, 1999, pp. 652–655.

[31] A. C. Rice, A. R. Beresford, and R. K. Harle, "Cantag: An open source software toolkit for designing and deploying marker-based vision systems," in *Proc. 4th Annu. IEEE Int. Conf. Pervas. Comput. Commun. (PERCOM)*, Mar. 2006, p. 10.

[32] M. Rohs and B. Gfeller, "Using camera-equipped mobile phones for interacting with real-world objects," in *Advances in Pervasive Computing*. Vienna, Austria: Österreichische Computer Gesellschaft, 2004, pp. 265–271.

[33] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image Vis. Comput.*, vol. 76, pp. 38–47, Aug. 2018.

[34] E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *Int. J. Comput. Vis.*, vol. 74, no. 3, pp. 237–260, Jul. 2007.

[35] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.

[36] H. Sarmadi, R. Muñoz-Salinas, M. Á. Berbís, A. Luna, and R. Medina-Carnicer, "3D reconstruction and alignment by consumer RGB-D sensors and fiducial planar markers for patient positioning in radiation therapy," *Comput. Methods Programs Biomed.*, vol. 180, Oct. 2019, Art. no. 105004.

[37] J. Sattar, E. Bourque, P. Giguere, and G. Dudek, "Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction," in *Proc. 4th Can. Conf. Comput. Robot Vis. (CRV)*, May 2007, pp. 165–174.

[38] J. Sawada, "Generating bracelets in constant amortized time," *SIAM J. Comput.*, vol. 31, no. 1, pp. 259–268, Jan. 2002.

[39] G. Schweighofer and A. Pinz, "Robust pose estimation from a planar target," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2024–2030, Dec. 2006.

[40] K. Shabalina, A. Sagitov, H. Li, and E. Magid, "Comparing fiducial marker systems occlusion resilience through a robot eye," in *Proc. 10th Int. Conf. Develop. eSystems Eng. (DeSE)*, Jun. 2017, pp. 273–278.

[41] S. Suzuki and K. Be, "Topological structural analysis of digitized binary images by border following," *Comput. Vis., Graph., Image Process.*, vol. 30, no. 1, pp. 32–46, 1985.

[42] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.

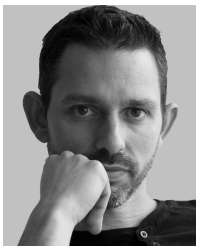
[43] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular SLAM," *Robot. Auton. Syst.*, vol. 57, no. 12, pp. 1188–1197, Dec. 2009.

[44] G. Yu, Y. Hu, and J. Dai, "TopoTag: A robust and scalable topological fiducial marker system," *IEEE Trans. Vis. Comput. Graphics*, vol. 27, no. 9, pp. 3769–3780, Sep. 2021.

[45] X. Zhang, M. Li, J. Lim, Y. Weng, Y. W. D. Tay, H. Pham, and Q. C. Pham, "Large-scale 3D printing by a team of mobile robots," *Autom. Construct.*, vol. 95, pp. 98–106, Nov. 2018.

[46] Z. Zhang, Y. Hu, G. Yu, and J. Dai, "DeepTag: A general framework for fiducial marker design and detection," 2021, *arXiv:2105.13731*. [Online]. Available: https://arxiv.org/abs/2105.13731

DAVID JURADO-RODRÍGUEZ received the degree in computer science from the University of Jaén, Spain, in 2019. He is currently pursuing the Ph.D. degree with the research group of Artificial Vision Applications, University of Córdoba. His research interests include artificial vision, computer graphics, and augmented reality.



RAFAEL MUÑOZ-SALINAS received the bachelor's degree in computer science and the Ph.D. degree from the University of Granada, Spain, in 2003 and 2006, respectively. Since then, he has been working with the Department of Computing and Numerical Analysis, Córdoba University, where he is currently a Full Professor. His research is focused on computer vision, soft computing techniques applied to robotics, and human–robot interaction. He has coauthored more than 100 papers in conferences, books, and top-ranked journals. One of his papers was the most-cited of the prestigious journal *Pattern Recognition* and is considered a highly-cited paper. He has advised seven Ph.D. students and participated in more than 20 projects, both industrial and scientific. He has been a Visiting Researcher at the universities of DeMontfort (U.K), Orebro (Sweden), TUM (Munich), INRIA (France), Groningen (The Netherlands), and Luxembourg. As a teacher, he has been teaching for more than 12 years, advised more than 30 final degree projects, and taught three international courses in the universities of Groningen, Luxembourg, and Brno. In addition, he has been part of the Erasmus STA teaching mobility projects in the universities of Malta, Coimbra (Portugal), and Dubrovnik (Croatia).

SERGIO GARRIDO-JURADO received the Ph.D. degree in computer vision from the University of Córdoba, Spain, in 2016. Since then, he has been dedicated to the development of augmented and virtual reality commercial products for educational purposes at Seabery R&D, where he currently leads the Computer Vision Department. During the last years, he has also collaborated in research projects with the AVA Group, University of Córdoba, and the Fluid Interfaces Group, MIT Media Lab. His research interests include the areas of SLAM, object tracking, and 3D reconstruction.

RAFAEL MEDINA-CARNICER received the bachelor's degree in mathematics from the University of Sevilla, Spain, and the Ph.D. degree in computer science from the Polytechnic University of Madrid, Spain, in 1992. Since 1996, he has been the Head of the Computer Vision Group AVA, University of Córdoba, Spain, and a Full Professor since 2012. He has been a principal investigator of more than ten research projects and his current research interests include computer vision techniques applied to robotics, biomedicine, and augmented reality.