Tracking Fiducial Markers with Discriminative Correlation Filters

Francisco J. Romero-Ramirez[1], Rafael Muñoz-Salinas[1,2,*], and Rafael Medina-Carnicer[1,2]

[1]Departamento de Informática y Análisis Numérico, Edificio Einstein. Campus de Rabanales, Universidad de Córdoba, 14071, Córdoba, Spain, Tlfn:(+34)957212289
[2]Instituto Maimónides de Investigación en Biomedicina (IMIBIC). Avenida Menéndez Pidal s/n, 14004, Córdoba, Spain, Tlfn:(+34)957213861

Abstract

In the last few years, squared fiducial markers have become a popular and efficient tool to solve monocular localization and tracking problems at a very low cost. Nevertheless, marker detection is affected by noise and blur: small camera movements may cause image blurriness that prevents marker detection.

The contribution of this paper is two-fold. First, it proposes a novel approach for estimating the location of markers in images using a set of Discriminative Correlation Filters (DCF). The proposed method outperforms state-of-the-art methods for marker detection and standard DCFs in terms of speed, precision, and sensitivity. Our method is robust to blur and scales very well with image resolution, obtaining more than 200fps in HD images using a single CPU thread.

As a second contribution, this paper proposes a method for camera localization with marker maps employing a predictive approach to detect visible markers with high precision, speed, and robustness to blurriness. The method has been compared to the state-of-the-art SLAM methods obtaining, better accuracy, sensitivity, and speed. The proposed approach is publicly available as part of the ArUco library.

Keywords— Discriminative Correlation Filter Squared Fiducial Markers Marker MappingSLAM.

*Corresponding author
Email addresses: fj.romero@uco.es (Francisco J. Romero-Ramirez), rmsalinas@uco.es (Rafael Muñoz-Salinas), rmedina@uco.es (Rafael Medina-Carnicer)

Preprint submitted to Image and Vision Computing

1 Introduction

Squared fiducial markers have become a popular and efficient method to solve monocular localization and tracking problems at a very low cost in indoor environments. In medical applications, they are used for tracking of surgical equipment [1, 2, 3]. In augmented reality (AR) problems, it is employed to estimate the camera pose so as to properly render the scene [4, 5]. In autonomous navigation or drone landing, it provides visual references for navigation and landing [6, 7, 8].

The recent works in squared markers [9, 10] make it is possible to estimate the camera pose in the environment (with the correct scale) by just analyzing images where some markers are visible. Given a set of these markers printed on a regular piece of paper and placed randomly in the environment (Fig. 1a), it is possible to estimate their three-dimensional location from a set of images or a video sequence showing them (Fig. 1b). This method allows obtaining motion tracking systems of very low cost, requiring only a camera.

Nevertheless, one of the limitations of these techniques is that the detection of markers is sensitive to blurring. Figure 2 shows the appearance of the markers under different blurring levels obtained by moving the camera at different speeds. Even at low camera speeds, manually recorded videos have blurriness that prevents detection (see Figure 2b). This effect happens either because the camera, which is not placed on a gimbal (e.g. in low-cost AR applications or drone landing), or because the marker moves fast (surgical equipment tracking). The high sensitivity to blurring is a limitation to the spread of that technology in applications of low cost and low computing power.

The contribution of this paper is two-fold. First, this work proposes a novel approach for estimating the location of markers in images, that is both fast and robust to blur, which consists in employing a set of Discriminative Correlation Filters (DCF). In order to speed up computation, our method employs a pyramid of images and selects at each frame the one where tracking can be done at maximum speed. Figure 2 shows the tracking capabilities of the proposed method. As a second contribution, we propose a novel approach for monocular camera pose estimation using marker maps. The proposed method, given a marker map, employs the previous trackers and a predictive approach to detect visible markers with high precision, speed, and robustness to blurriness.

The experiments conducted shows that the proposed marker tracking method is fastest and more robust to blur than the state-of-the-art marker detection algorithms, and more precise than the best DCFs. In addition, our proposal is compared with three state-of-the-art SLAM methods: ORBSlam2 [11], LDSO [12], and UcoSLAM [13]. Our method outperforms them in terms of speed and precision. The proposed method is publicly available as part of the ArUco library[1].

The remainder of this paper is structured as follows. Sect. 2 provides an overview of the related works, while Sect. 3 explains the basis of DCF. Our contributions are explained in Sect. 4 and 5, while Sect. 6 presents the experiments conducted and Sect. 7 draws some conclusions.

2 Related works

2.1 Fiducial marker systems

Fiducial marker systems are commonly used in camera pose estimation and tracking processes, due to their high accuracy, robustness, and speed. Since the appearance of ARToolKit [14], a large number of systems based on square markers have emerged [10, 15, 16, 17]. In general, the detection process involves thresholding

[1] `https://www.uco.es/investiga/grupos/ava/node/26`

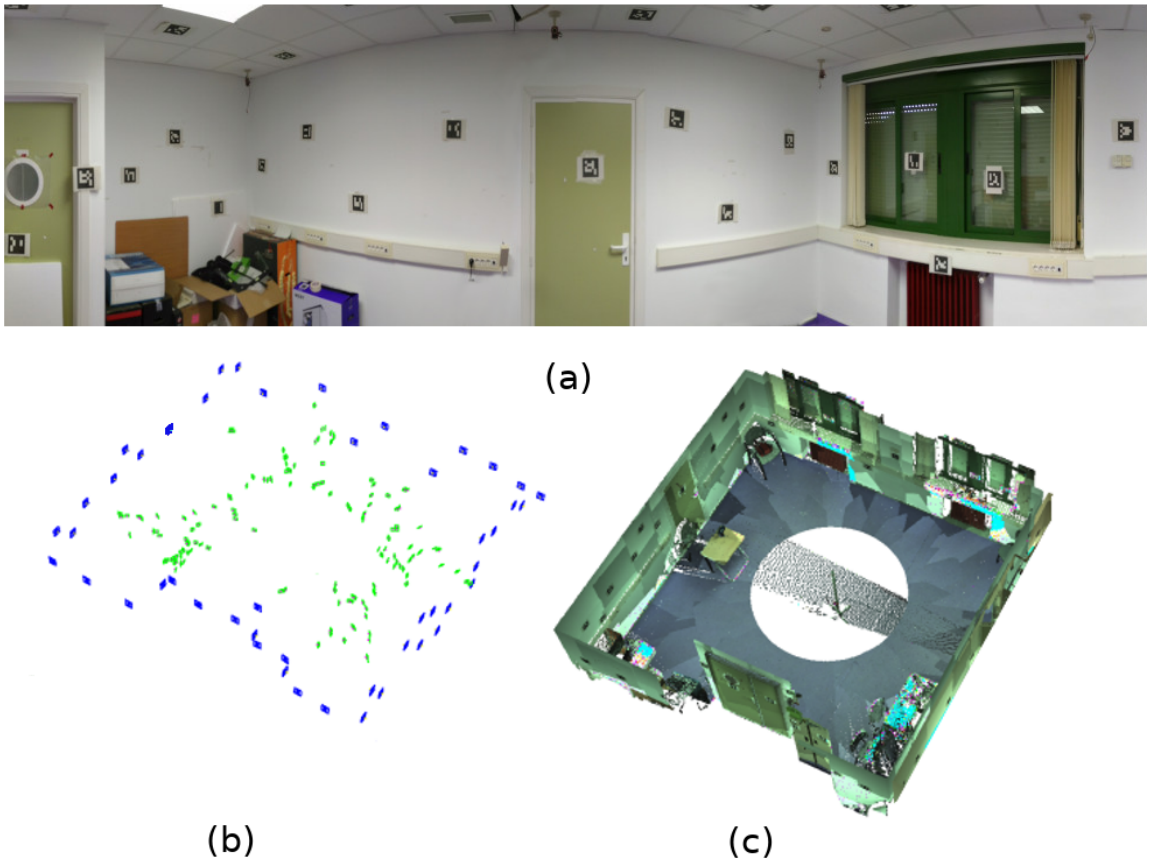

Figure 1: Map of markers generated by the works [9, 10]. (a) Image of tracking room where a set of markers are randomly placed in the walls. (b) Marker map generated with [9]. Blue squares represents the pose of the markers and green ones the pose of the cameras. (c) Laser reconstruction of the room to help to understand the three-dimensional configuration of the room.

the scene in which a set of squared regions (candidate markers) are extracted from the background. Later, the interior of the regions is analyzed, discarding those that cannot be identified. Finally, using the four corners of at least one marker, it is possible to estimate the position of the camera.

Several works have analyzed the performance of marker detection systems [18] showing that speed, robustness, and accuracy are essential factors to be taken into account, where ArUco and AprilTag markers systems take advantage [19, 20, 21].

2.2 Discriminative Correlation Filters

Since the appearance of the work of Bolme et al. [22] with Minimum Output Sum of Squared Error (MOSSE), discriminative correlations filters (DCF) have increased their popularity, becoming one of the main methods of visual tracking due to its efficiency and robustness.

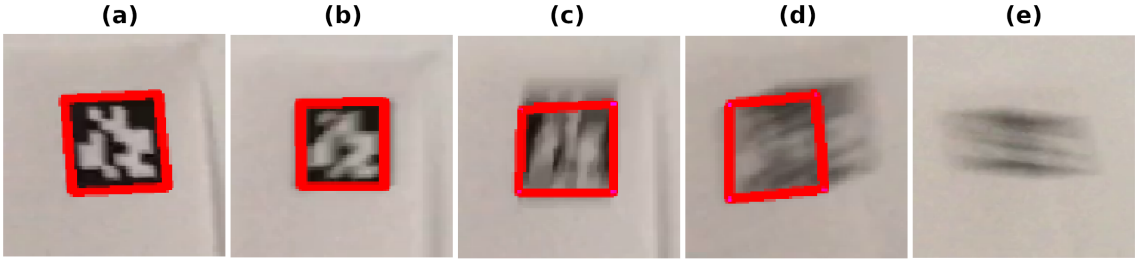

Figure 2: Tracking of a fiducial marker along the a video sequence with the proposed method. From left to right, the marker is observed with increasing blurring levels. The proposed method is capable of tracking the marker in figures a-d but not in figure e. The estimated marker location is drawn as a red rectangle.

Many other researchers have work on improving several aspects of the initial MOSSE proposal. Henriques et al. [23] replaces the use of grayscale filters by using HOG features, Danelljan et al. [24] introduces learning multi-channel filters with Colornames, Li et al. [25] and Lukežič et al. [26] use the integration of both HOG and Colornames. Other works employing convolutional features of CNNs [27, 28, 29] have shown high performance.

DCF usually has limited information about the contour, leading to false positives in some scenarios such as rapid movement, occlusion, or background noise. Mueller et al. [30] use context information in filter training to improve the performance of state-of-art algorithms without incurring in high computational costs. On the other hand, to reduce the boundary effects Danneljan et al. [31] reformulate the learning function by considering larger image regions, penalizing filter values outside the bounding box.

Another limitation of DCFs is the assumption that the target has a fixed size and that it is completely aligned to a rectangular region. However, the shape of the tracked objects and their rotation makes the filter learn the background, leading to errors in tracking. Danelljan et al. [32] presented a method to estimate the scale by training a classifier on a pyramidal scale. Also, Lukežič et al. [26] introduce the channel and spatial reliability concepts. The spatial reliability map adjusts the filter to the object to the object allowing to adapt the size of the search region and improving the tracking in non-rectangular objects. The channel reliability reflects the discriminative power of each filter channel.

3 Mathematical Basis of Discriminative Correlation Filters

Correlation filter based tracking applies a continuous adaptive process to find the filter that when applied on the desired target produces the maximum response. In its simpler form, the filter is a small image patch centered around the object to be tracked. However, in order to increase the robustness to appearance changes, a set of modified images of the target (created using affine transformations) are employed to build the filter. Once the initial filter is created, the filter is applied on the next image at the same location. Then, the position with the maximum response within the region is considered the new target location. Finally, the filter is updated to adapt changes in appearance and the process repeated in the subsequent frames [22].

Let us denote $\mathcal{X} = \{x_1, ..., x_n\}$ the set of gray-scale patches of the target observed under different appearance conditions. It will be used as a training set to create the initial filter h. Also, let us denote

$\mathcal{G} = \{g_1, ..., g_n\}$ the desired response of the filter when applied on the patches, i.e, $h(x_i) = g_i$. Although g_i can have any shape, it is generally generated as a 2D Gaussian ($\sigma = 2$) centered at the center of the patch. Thus, in practice $g_i = g_j \ \forall i, j \in \{1, \ldots, n\}$.

Computing the correlation in the Fourier Domain has demonstrated to be the best way to speed up computation and obtain a certain degree of robustness to misalignment. Correlation in the frequency domain turns into element-wise multiplications expressed as:

$$G = X \odot H^* \tag{1}$$

where G, X and H denotes the Fourier transforms of $g \in \mathcal{G}, x \in \mathcal{X}$ and h respectively, $\odot$ is the element-wise multiplication and $*$ the complex conjugate. In consequence, the estimation of the optimal correlation filter H in the Fourier domain is computed as:

$$\min_H \sum_{i=1}^{n} ||X_i \odot H^* - G_i||^2 + \lambda_1 ||H||^2 \tag{2}$$

where λ_1 is a regularisation term. Since Eq. 2 is convex, it has a single global minimum that can be expressed as:

$$H_1 = \frac{\sum_{i=1}^{n} X_i \odot G_i^*}{\lambda_1 + \sum_{i=1}^{n} X_i \odot X_i^*} \tag{3}$$

which expresses how to obtain the correlation filter in the first frame. The regularisation term λ_1 prevents divisions by zero.

In frame t ($t > 1$), the filter is applied to the previous target location and the location with maximum response is expected to be current target location. We shall define z_t as the image patch centred at the maximum response location in t, and Z_t is its Fourier transform. Then, the filter is updated using a running average so that

$$H_t = \frac{\eta A_t + (1 - \eta) A_{t-1}}{\eta B_t + (1 - \eta) B_{t-1}} \tag{4}$$

$$A_t = G_t \odot Z_t^* \tag{5}$$

$$B_t = Z_t \odot Z_t^* \tag{6}$$

where the parameter $\eta \in [0, 1]$ is the learning rate.

An important aspect to consider is how to detect when the tracking has failed. A method to do so is analyzing the Peak to Sidelobe Ratio (PSR), which is the ratio between the filter value at the point with maximum response and the average response in the rest of the pixels. It has been observed than values below 7.0 indicates tracking failure [22].

In general, the area surrounding the tracked object may contain distracting information for tracking that leads to an erroneous local minimum. An effective approach to alleviate this problem is to include contextual information in the filter [30]. Instead of considering only the target appearance to build the filter, patches surrounding the target are also employed as negative examples. Following this approach, Eq. 2 is updated so that the minimization function takes into account a set of patches surrounding the target.

If $\mathcal{Y} = \{y_1, \ldots, y_m\}$ is the set of contextual patches (blue patches in Fig. 3a) , then Eq.2 becomes:

$$\min_H \sum_{i=1}^{n} ||X_i \odot H^* - G_i||^2 + \lambda_1 ||H||^2 + \lambda_2 \sum_{j=1}^{m} ||Y_j \odot H||^2 \tag{7}$$

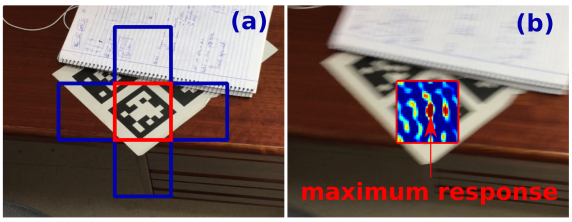

Figure 3: Tracking process with correlative filters. (a) Training process in frame t. The filter is updated using the central patch of the marker in addition to the 4 patches around it. (b) represents the process of tracking in the frame t+1, for it uses the filter updated in t, the maximum value of response indicates the new position of the marker.

where λ_2 modulates the relative importance of the context and Y_j is the Fourier transform of y_j. Using this approach, the update of the filter in frame t ($t > 1$) is expressed as:

$$H_t = \frac{\eta A_t + (1-\eta)A_{t-1}}{(\eta B_t + (1-\eta)B_{t-1}) + \lambda_2 \left(\eta D_t + (1-\eta)D_{t-1}\right)} \tag{8}$$

where

$$D_t = \sum_{i=1}^{m} Y_{i,t} \odot Y_{i,t}^{*} \tag{9}$$

and A_t, B_t are obtained from Eqs. 5 and 6.

4 Tracking of a Squared Marker

This section introduces our first contribution, a DCF-based tracker that allows the continuous tracking of a square fiducial marker throughout a video sequence. Since estimating the exact location of the marker corners is required to estimate its three-dimensional pose, our method must be able to track them. Therefore, our approach employs a total of five filters: one filter for tracking the marker general appearance, and four additional filters for tracking the corners. In order to speed up computation while adapting to scale changes, a multi-resolution pyramid tracking approach is proposed. Filters of fixed size are employed (constraining the computation time), but, at each iteration, the scale where the marker dimension best fits the filter size is employed.

Our process can be summarized in the following steps. In the first frame, we find the pyramid level where the filters are created with the desired size. In subsequent frames, the filters are first applied in the neighboring regions of the previous location at the same pyramid level to find the optimal location of the marker and its corners. Then, to adapt to scale changes of the marker, we must find the scale that produces the highest response of the filter. Finally, the filters are updated.

Below, we provide a formal description of the proposed method.

4.1 Tracker definition and initialization

The initial step to track a marker $\mathbf{m}$ along a video sequence is to find it in the image. A marker is a squared matrix in which each element represents a bit (see Fig. 4). The marker is comprised of a black region,

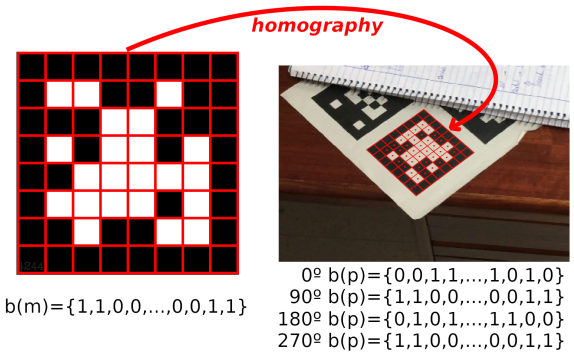

Figure 4: Identification of the tracked marker. The computation of the homography on the detected polygon, allows to take the central value of its identification bits, and analyzed in its four possible orientations.

which helps to detect it, and the inner region containing the bits that uniquely identify the marker. Let us define the sequence of bits of a marker as

$$b(\mathbf{m}) = (b_1, \ldots, b_n) \mid b_i \in \{0, 1\}, \tag{10}$$

which is created row by row starting at the top-left bit of the matrix. The detection of the marker in the image can be efficiently done using the method proposed in [10]. The method extract contours in the image, obtain its polygonal approximation, and discard those that are not quadrilateral. Each remaining polygon $\mathbf{p}$ is analyzed to check if it belongs to a valid marker. Its four corners are employed to compute Homography matrix that determines the central pixel of each bit in the image and its pixel intensities are thresholded using Otsu's [33] algorithm obtaining its bit sequence $b(\mathbf{p})$ in its four main rotations (0^{o}, 90^{o}, 180^{o} and 270^{o}). If the Hamming distance of both is zero in any of the possible rotations, then we have a perfect match and the marker is considered as detected (see Fig. 4).

Let us define

$$c = \{c_k \mid c_k \in \mathbb{R}^2, k \in \{1, \ldots, 4\}\}$$

as the pixel coordinates of the four corners for marker $\mathbf{m}$ in image $\mathbf{I}$, $\mathcal{C}(\mathbf{m}) \in \mathbb{R}^2$ as the location of the marker center, and $\mathcal{A}(\mathbf{m})$ as the observed marker area.

Our aim is to use patches of length side τ_s to create the DCFs for marker and corners. To do so, the patches are obtained from an down-sampled version of the image $\mathbf{I}$ where the marker area $\mathcal{A}(\mathbf{m})$ is most similar to τ_s^2. If we denote

$$\mathcal{I} = (I^0, I^1, ..., I^n)$$

as the pyramid of images ($I^0 = \mathbf{I}$) where the image $I^j, j > 0$ is the original image $\mathbf{I}$ down-sampled by the factor

$$\beta^j \mid \beta \in [0, 1],$$

then, we can define:

$$L(\mathbf{m}) = \begin{cases} 0 & \text{if } \left(\frac{\tau_s^2}{A(\mathbf{m})}\right) \geq 1 \\ \left\lfloor \log_\beta \left(\frac{\tau_s^2}{A(\mathbf{m})}\right) \right\rfloor & \text{otherwise} \end{cases} \tag{11}$$

as the pyramid level where the area $\mathcal{A}(\mathbf{m})$ of the marker is most similar to the desired patch area τ_s^2. In other words, the image $I^{L(\mathbf{m})}$ is where the initial patches of area τ_s^2 will be extracted. Please notice that $\lfloor \cdot \rfloor$ denotes the floor function.

We shall define $P(p, \tau_s)$ as the function that returns a patch of size τ_s^2 centred at $p \in \mathbb{R}^2$ in the image $I^{L(\mathbf{m})}$. Consequently, the patches to generate the DCFs for the marker and its corners are $P(\mathcal{C}(\mathbf{m}), \tau_s), P(c_1, \tau_s), P(c_2, \tau_s), P(c_3, \tau_s)$ and $P(c_4, \tau_s)$, respectively.

Let us then define the tracker for marker $\mathbf{m}$ at time t as:

$$T_t^{\mathbf{m}} = \{T_{0,t}^{\mathbf{m}}, \ldots, T_{4,t}^{\mathbf{m}}, l_t^{\mathbf{m}}\} \tag{12}$$

where $T_{i,t}^{\mathbf{m}}$ represents the Fourier transforms of the DCF for the marker center ($T_{0,t}^{\mathbf{m}}$) and its four corners ($T_{i,t}^{\mathbf{m}}, i \in \{1, \ldots, 4\}$) (see Eq. 7), while $l_t^{\mathbf{m}}$ represents the pyramid level employed for correlation at time t. In the first frame,

$$l_1^{\mathbf{m}} = L(\mathbf{m}).$$

4.2 Tracking and Update

In subsequent frames ($t > 1$), the filters are applied at the previous location, and the location of maximum filter response is obtained:

$$\mathcal{E}_{i,t}^{\mathbf{m}}(l_t^{\mathbf{m}}) = \underset{p \in \mathbb{R}^2}{\operatorname{argmax}} \, PSR(T_{i,t}^{\mathbf{m}}, p, l_t^{\mathbf{m}}), \tag{13}$$

where PSR indicates the response of the filter $T_{i,t}^{\mathbf{m}}$ centred at pixel p in the image $I^{l_t^{\mathbf{m}}}$. If the maximum PSR for the marker tracker $T_{0,t}^{\mathbf{m}}$ is below the established threshold value, the marker is considered as lost.

A very important aspect to consider is the need for an accurate estimation of the marker corners. The corner locations estimated by Eq. 13 do not have the required accuracy for pose estimation. First, because tracking normally is done at a reduced version of the original image. Second, even if the tracker is run at the lowest piramid level I^0, the result is not accurate enough. The corners locations must be refined with sub-pixel accuracy. Thus, in order to obtain a precise corner estimation, we employ an iterative corner upsampling process that produces a precise corner location $\mathcal{S}(T_{i,t}^{\mathbf{m}})$ in the original image I^0. To do so, first, a corner search with sub-pixel accuracy is performed in the vicinity of the estimated corner locations $\mathcal{E}_{i,t}^{\mathbf{m}}(l_t^{\mathbf{m}})$. For that purpose, the refinement method implemented in the OpenCV library [34] is been employed. Then, the corner location is upsampled to the previous pyramid level $l_t^{\mathbf{m}} - 1$, and the search repeated. The process stops when the image I^0 is reached.

Adapting to scale is another crucial element for a successful tracking. In the first frame, correlation is done at the pyramid level $l_1^{\mathbf{m}}$ where the DCFs were initialized. However, due to scale changes of the marker (when approaching or moving away from it), the initial pyramid level $l_1^{\mathbf{m}}$ may not be the one for which the filters obtain its maximum response. Thus, it is necessary to find the best pyramid level for the next frame. To do so, the response of the filter $T_{0,t}^{\mathbf{m}}$ at the contiguous pyramid scales is analyzed, and the one maximizing the marker filter is selected:

$$l_{t+1}^{\mathbf{m}} = \underset{l \in \{l_t^{\mathbf{m}}+1, l_t^{\mathbf{m}}, l_t^{\mathbf{m}}-1]\}}{\operatorname{argmax}} PSR(T_{0,t}^{\mathbf{m}}, \mathcal{E}_{0,t}^{\mathbf{m}}(l), l) \tag{14}$$

Once the best pyramid level is found, all the filters are updated using the patches extracted from that level.

4.3 Confidence measure

The proposed method can track the marker $\mathbf{m}$ under large appearance changes caused by blur (see Fig. 2). However, in some cases, the blurriness level is so high that the estimated location of the corners is not

reliable enough for three-dimensional pose estimation.

We propose a confidence measure $w^{\mathbf{m}} \in [0,1]$ that indicates how reliable is the estimation provided by our tracker. As it will become evident in the next section, this measure will allow favoring some markers over others when doing localization from multiple makers. Values near to 1 indicate high confidence in the detection while values near to zero indicate low confidence.

The measure is composed of two terms. First, the normalized Hamming H distance between the marker bit sequence $b(\mathbf{m})$ and the bit sequence $b(\mathbf{p})$ observed for the polygon $\mathbf{p}$ formed by the four marker corners estimated by our tracker:

$$\frac{\mathrm{H}(b(\mathbf{m}), b(\mathbf{p}))}{|b(\mathbf{m})|}.$$

But then, this value is modulated by the response of the corner trackers

$$\frac{\sum_{i=1}^{4} \mathcal{PSR}_i}{4},$$

where

$$\mathcal{PSR}_i = \begin{cases} 1 & PSR(T^{\mathbf{m}}_{i,t}, \mathcal{E}^{\mathbf{m}}_{i,t}(l), l) > \chi \\ 0 & otherwise \end{cases} \tag{15}$$

indicates if the tracking of a corner was successful or not.

Thus, the confidence measure is expressed as:

$$w^{\mathbf{m}} = 1 - \left(\frac{\mathrm{H}(b(\mathbf{m}), b(\mathbf{p}))}{|b(\mathbf{m})|} \frac{\sum_{i=1}^{4} \mathcal{PSR}_i}{4} \right). \tag{16}$$

We have found after several experiments that the combination of both terms provides better results than any one of them separately.

5 Robust Marker-Map based Pose Estimation

This section explains our second contribution, an extension of the previous methodology aimed at camera pose estimation with marker maps. A marker map is a set of markers placed in known map locations of the environment that are employed for camera localization in indoor environments. The observation of a single marker can be enough to obtain the pose of the camera on the map. However, the more markers are visible, the better the accuracy that can be obtained (see Fig. 6).

Our goal is estimating the camera pose $\theta_t \in \mathbb{R}^6$ (position and angle) in the map given: (i) a set of markers $\mathcal{M}$ in known map locations, (ii) an image $\mathbf{I}_t$ showing some of them, and (iii) the previous camera location θ_{t-1}.

We shall define the set of markers in our map by

$$\mathcal{M} = \{\mathbf{m} = \{q^{\mathbf{m}}_1, \ldots, q^{\mathbf{m}}_4\}\}, \tag{17}$$

where $q^{\mathbf{m}}_i \in \mathbb{R}^3$ represents the three-dimensional coordinates of the marker corners in the environment. The map can be obtained from images of the environment using any of the methods described in [9, 13, 35].

Given an image showing some of the markers, it is possible to estimate the camera pose by analyzing the set of 2D-3D correspondences. Since the 3D location of the corners is known in advance ($\mathcal{M}$), their 2D image projections can be employed to find the pose between the camera and the global reference system by minimizing the reprojection of the observed markers as will be explained later in Sect. 5.2.

The rest of this Section explains the proposed method to estimate the camera pose θ_t given an input image $\mathbf{I}_t$, which can be summarized in Alg 1.

5.1 Method overview

Our method employs a set of trackers

$$\mathcal{T}_t = \{T_t^{\mathbf{m}}\}, \mathbf{m} \in \mathcal{M}, t \geq 1,$$

to estimate the position of the markers in the image, where $T_t^{\mathbf{m}}$ is the type of tracker defined in the previous Section (Eq. 12). We are proposing a tracking method, and thus, it requires to be initialized. The initial position θ_1 and $\mathcal{T}_1$ are obtained from the markers detected with a marker detector [19, 36, 17].

In subsequent frames $\mathbf{I}_t$, the trackers $\mathcal{T}_t$ are applied in order to find the new markers locations. Tracking of a marker may fail for several reasons: it fall outside the image view, occlusion, high blur, etc. Thus, we remove from $\mathcal{T}_t$ the trackers $T_t^{\mathbf{m}}$ with a low response (PSR) of the central tracker $T_{0,t}^{\mathbf{m}}$. The corners of the remaining markers are employed to obtain an initial estimation of the camera pose $\hat{\theta}_t$ (Sect. 5.2).

As the camera moves along the environment, some markers will fall out of the camera view while others will appear. Since we know both the pose of the camera $\hat{\theta}_t$ and the three-dimensional location of the markers $\mathcal{M}$, we can estimate which markers should be visible in the current image and where (Sect. 5.3). For each expected visible marker, (not in $\mathcal{T}_t$) a quick detection is done on the expected image region where it should be visible. If correctly detected, a new tracker $T_t^{\mathbf{m}}$ is added to $\mathcal{T}_t$. After all the new markers have been added, we calculate the final camera pose θ_t using all the visible markers.

Tracking may fail either because of very fast movement causing a lot of blur (Fig. 2e), or because there are no markers are visible in the image. Thus, as final step we analyze if a tracking confidence measure $w^{\mathcal{T}_t}$ (explained in Sect. 5.4) is high enough. If not, the tracking should stop until a reliable pose can be obtained using a regular marker detector [19, 36, 17] to restart tracking.

Algorithm 1: Tracking algorithm overview for image $\mathbf{I}_t$

Data: $\mathcal{M}, \theta_{t-1}, \mathcal{T}_{t-1}, \mathbf{I}_t$
Result: $\theta_t, \mathcal{T}_t, w^{\mathcal{T}_t}$
begin
  $\mathcal{T}_t \longleftarrow ApplyFilters(\mathcal{T}_{t-1})$ (Sect 4);
  for $T_t^{\mathbf{m}} \in \mathcal{T}_t$ **do**
    if $PSR(T_t^{\mathbf{m}}) < \chi$ **then**
      remove $T_t^{\mathbf{m}}$ from $\mathcal{T}_t$
    end
  end
  Estimate pose $\hat{\theta}_t$ using $\mathcal{T}_t$ (Sect. 5.2);
  Look for new visible marker (Sect. 5.3);
  for *each new maker* **m do**
    add $T_t^{\mathbf{m}}$ to $\mathcal{T}_t$
  end
  Obtain the final pose θ_t using updated $\mathcal{T}_t$;
  Calculate tracking confidence $w^{\mathcal{T}_t}$ (Sect. 5.4);
end

5.2 Camera pose estimation

The estimation of the camera pose given a set of markers detected in the image consists in minimizing the reprojection error of their corners, considering its confidence $w^{\mathbf{m}}$ (Eq. 16):

$$\theta_t = \underset{\theta}{\operatorname{argmin}} \sum_{\mathbf{m} \in \mathcal{M}} w^{\mathbf{m}} \mathcal{H}\left(e_t^{\mathbf{m}}(\theta)\right), \tag{18}$$

where $e_t^{\mathbf{m}}(\theta)$ represents the reprojection error of the corners of marker $\mathbf{m}$ and $\mathcal{H}$ is the Hubber function, employed to minimize the impact of possible outliers:

$$\mathcal{H}(a) = \begin{cases} \frac{1}{2}a^2 & for\ |a| \leq \alpha \\ \alpha(|a| - \frac{1}{2}\alpha) & otherwise \end{cases} \tag{19}$$

The reprojection error of a marker $\mathbf{m}$ is defined as:

$$e_t^{\mathbf{m}}(\theta) = \sum_{i=1}^{4} ||\psi(q_i^{\mathbf{m}}, \theta) - \mathcal{S}(T_{i,t}^{\mathbf{m}})||^2, \tag{20}$$

where the function $\psi(q, \theta) \in \mathbb{R}^2$ projects the three-dimensional point q in the image given the camera pose θ and $\mathcal{S}(T_{i,t}^{\mathbf{m}})$ is the precise corner location in the original image I^0 (section 4.2).

Equation 18 is a non-linear function that can be efficiently minimized using the Levenberg–Marquardt's (LM) algorithm [37].

5.3 Look for visible markers

As the video sequence progresses, and the camera moves, markers will appear and disappear from the scene. The initialization of these markers is essential to achieve continuous tracking and accurate pose estimation.

Given that the three-dimensional locations of the marker corners in $\mathcal{M}$ are known, and an initial camera pose $\hat{\theta}_t$ for the $\mathbf{I}_t$ image is available, we can calculate which markers should be visible in the image $\mathbf{I}_t$ image and where their corner should project.

For each expected marker, we apply a detection process in the region where it should be visible. First, the Otsu's thresholding algorithm [33] is applied, and contours are extracted using the Suzuki and Abe algorithm [38]. Using the Douglas and Peucker algorithm [39], the largest a squared polygon $\mathbf{p}$ is selected. Then, the bits $b(\mathbf{p})$ of the polygon are extracted and if they match the predicted marker $b(\mathbf{m})$, using the Hamming distance, the marker is considered found and a tracker initialized and added to $\mathcal{T}_t$ to be employed for the next image.

5.4 Calculate Tracking Confidence

As previously mentioned, tracking may fail due to the absence of markers, blur, bad lightning conditions, marker occlusion or any other reason. Therefore, it is important to provide a confidence value indicating how reliable the estimated pose θ_t is. It allows determining whether tracking has failed, and in that case, the system can stop tracking and use a slower but more conservative method for detecting the markers [19, 36, 17].

In this paper, we propose a confidence measure based on the following principle. If a single marker is spotted very near to the camera, occupying a large region of the image, the estimation of the pose is reliable. However, if the same marker is detected far from the camera, occupying only a very small region

Table 1: Nomenclature and values of the main parameters used by the proposed method.

Parameter	Default value	Description
λ_1	10^{-4}	Filter regulation parameter (Sect 3)
λ_2	20	Context-Aware parameter (Sect 3)
η	0.2	Learning rate (Sect 3)
τ_s	32	Filter size (Sect 4.1)
β	0.7	Pyramid scale factor (Sect 4.1)
χ	5.7	Peak to Sidelobe Ratio (Alg. 1)
α	2.5	Hubber function cut-off value (Sect 5.2)
τ_c	0.1	Tracking Confidence Threshold (Sect 5.4)

of the image, the estimation is very unreliable. In the end, the reliability of the estimated pose depends mainly on the total area of the points employed for computing Eq. 18. If the points are far apart, occupying a large region of the image, the estimated pose is reliable, and vice versa. So, let us define the confidence measure $w^{\mathcal{T}_t}$ as the relative area of the convex hull formed by the marker corners employed in Eq. 18. This value is one of the points cover all the image, and tends to zero as they are more concentrated in a region. If the confidence $w^{\mathcal{T}_t}$ is below a threshold τ_c, we consider the tracking has failed.

6 Experiments and results

This section explains the experiments carried out to validate our proposal. The goal of the experiments is to evaluate the robustness, speed, and accuracy of the proposed method for marker tracking. Experiments have been divided in two categories. First, the individual marker tracking algorithm (Sect. 4) is tested, comparing it with state-of-the-art marker detection methods (Sect. 6.1), and correlation filter trackers (Sect. 6.2). Afterward, our method for camera pose estimation using marker maps (Sect. 5) is compared with the state-of-the-art SLAM methods in challenging video sequences (Sect. 6.3).

All experiments have been performed using an Intel® processor Core™ i7-7500U CPU @ 2.70GHz × 4, with 8Gb of Ram, and the Ubuntu 18.04 operating system. Although some of the processing could be parallelized, only one thread has been used.

Several parameters that control the behavior of the proposed algorithms we have introduced along the paper. The values used for these parameters have been experimentally selected and are shown in Table 1.

Finally, we must indicate that the code has been integrated as part of the public library ArUco. We will refer to the proposed method as TR-ArUco. The code and the videos recorded to conduct the experiments are publicly available [2].

[2] `https://www.uco.es/investiga/grupos/ava/node/69`

Table 2: For each method and image resolution: number of frames per second (FPS) and true positive rate (TPR).

Method	480p		720p		1080p	
	FPS	TPR	FPS	TPR	FPS	TPR
TR-ArUco	348.986	0.719	297.750	0.901	234.796	0.946
AprilTag	132.567	0.479	49.581	0.644	26.978	0.629
ArUco_NORMAL	760.168	0.333	227.289	0.416	102.737	0.460
ArUco_FAST	806.543	0.358	570.364	0.454	198.292	0.532

6.1 Comparison with Fiducial Squared Marker Detectors

This section makes a comparison in terms of speed and detection rate of the proposed method TR-ArUco against the main state-of-the-art marker detection and tracking algorithms: ArUco [19] and AprilTag [17]. Nowadays, both methods are widely used due to their high performance in terms of speed detecting fiducial markers.

Both AprilTag and ArUco detector has different configurable parameters establishing a balance between speed and detection range. For the AprilTag detector, this parameter is the decimation factor, and for the ArUco detector, it is the minMarkerSize. To do a fair comparison, parameter values that maximize the number of detections are chosen. Thus, for AprilTag the decimate factor 2 has been employed, and the minMarkerSize is set to 0 for ArUco. Additionally, for the ArUco method two versions have been used: the *ArUco_NORMAL* detection method, which employs an adaptive image threshold, and the *ArUco_FAST* detection method that uses a global threshold.

A set of video sequences have been recorded showing a squared marker (of size 6×6 cm) printed on a piece of paper. Along the sequences, the marker remains static, while the camera moves at different speeds and distances from the marker. Throughout the sequences, the marker is seen with different sizes, lighting conditions, and degrees of deformations produced by blurring. In total, 10 video sequences, containing a total of 3326 video frames, of resolution 1920×1080, have been recorded using a mobile phone. The location of the corners can not be estimated in all the images of the sequence (see Fig. 2 (c-e)). However, we know the marker is visible in all the images. As a consequence, we can analyze the True Positive Ratio of detections. Besides, we have analyzed the processing speed of the methods. Table 2 shows the results of the experiment for several resolutions of the video sequences (namely 1080p, 720p and 480p.)

As can be seen, our method obtains the highest TPR in all the tests performed. Also, although the proposed method is not the fastest one for all resolutions, it becomes the fastest one as the resolution increases. The speed of our method TR-ArUco is less affected by the image resolution because it employs filters of fixed size. The proposed pyramid method is the key to obtain high frame rates as the resolution increases. In the end, our method obtains the highest TPR an is an order of magnitude faster than AprilTag, the second one with highest TPR.

Table 3 shows a summary of the average time consumed by the different steps of the TR-ArUco method. Notice that steps $1.1 - 1.2$ are only performed when the marker is not being tracked, i.e. in the first frame of the video sequence, or when a marker that was being tracked is lost. The steps $2.1 - 2.4$ are performed on all frames. As can be seen, the computation times for the resolutions used are similar, with an average computation time of 4.19 ms. Among the different phases, the selection of the optimal scale is the most time-consuming one. Also, it is affected by the visible area of the marker, i.e., the larger the marker appears in the image, the more computation is required.

Table 3: Mean computing times (milliseconds) of the different steps of the proposed method for different resolutions.

	Resolution		
	480p	720p	1080p
Step 1.1:ArUco detect	0.204	0.704	0.947
Step 1.2:Creating filters	0.028	0.029	0.031
Time Step 1 (ms)	0.232	0.733	0.978
Step 2.1:Convert to grey	0.267	0.492	1.365
Step 2.2:Optimal scale	1.191	1.382	1.645
Step 2.3:Track corners	0.984	1.006	1.038
Step 3.3:Track marker	1.050	1.111	1.142
Time Step 2 (ms)	3.492	3.991	5.190

6.2 Comparison with Discriminative Correlation Filters

This Section compares the proposed method with the state-of-the-art Discriminative Correlation Filters trackers, namely, KCF [40], CSRT [26], MIL [41], TLD [42], MEDIANFLOW[43], MOSSE[22] and BOOSTING[44]. The implementations provided in the public library OpenCV[3] library have been employed.

The key aspect when detecting a squared fiducial marker is correctly detecting the position of its four corners in the image. Consequently, this experiment aims at evaluating the capability of the above indicated DCFs to track the four corners of a marker.

The video sequences employed in the previous experiment have been used for this one. The ground truth has been obtained using the ArUco library, obtaining the location of the marker corners in these frames where the marker is detectable. For each one of the selected DCFs trackers, we have applied the following methodology. A total of four independent trackers have been employed to track the marker corners. The trackers are initialized in the first frame to the center of each corner, and then, the trackers are applied to the subsequent frames. The size of the filters is half the size of the marker (see Fig. 5). Whenever the tracking error becomes higher than a number of pixels ϵ, the trackers are initialized so as to avoid the trackers to become completely lost for the rest of the sequence. For our tracker, we proceed in a similar way, re-initializing the tracker if the error in the estimation of the corners becomes greater than ϵ.

The results obtained for different values of ϵ are shown in Table 4, where each row represents a method. The columns show the total number of re-initialization required in the sequences evaluated (init), the average frames per second employed by each method (fps), and the average tracking error (err) which is expressed in pixels. In this set of experiments, only images of resolution $1080p$ have been employed.

As can be observed, the proposed method TR-ArUco outperforms the rest of the methods in the three parameters evaluated. Our method obtains a stable frame rate, which is an order of magnitude faster than the rest of the methods (except for MOSSE). The same can be said about the number of re-initializations, which is much lower than in the rest of the algorithms. Finally, the tracking error of the corners of our method is the lowest of all. The main conclusion is that the proposed method outperforms the naive approach (i.e., using individual DCFs) for the given problem.

Figure 2 shows some of the images evaluated in this experiment, overlaying in red the estimations obtained. Figure 2-(e) shows a case in which our method fails and requires re-initialization. As can be seen, our method requires re-initialization only in very extreme cases.

[3] `https://opencv.org/`

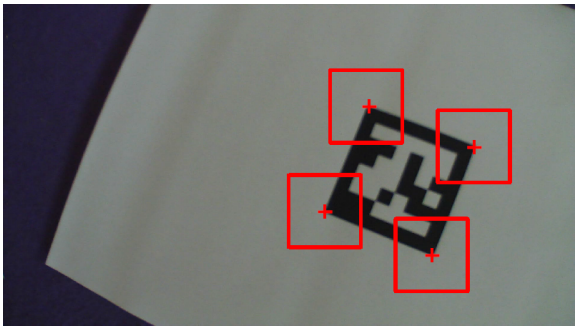

Figure 5: Naive approach employed to track a marker consist in using four independent DCFs: one for each corner.

Table 4: Results obtained by different state-of-the-art DCF trackers. We evaluate the total number of tracking re-initializations (init), the computation time (fps), and the average tracking error (err).

	$\epsilon < 5$			$\epsilon < 10$			$\epsilon < 15$		
	init	fps	err	init	fps	err	init	fps	err
TR-ARUCO	24	287.42	0.82	18	266.14	0.98	13	294.24	1.13
CSRT	72	7.12	1.63	60	8.15	1.76	57	8.35	1.85
BOOSTING	98	13.16	1.83	91	12.68	1.92	84	12.85	2.30
MEDIANFLOW	116	19.55	1.92	77	25.26	2.92	61	21.26	4.06
MIL	245	5.99	2.68	163	6.00	3.97	112	5.90	4.65
MOSSE	270	1105.06	2.31	214	1035.57	3.38	166	1020.56	4.20
KCF	338	91.29	2.74	236	79.82	4.97	186	75.23	6.73
TLD	734	4.90	4.08	539	2.37	7.45	409	2.39	10.48

6.3 Comparison with SLAM methods

This section analyzes the TR-ArUco method for camera pose estimation using marker maps (Sec. 5), with the state-of-art SLAM methods. The following SLAM algorithms have been tested:

- ORBSlam2 [11]: a SLAM method based on keypoints.
- LDSO [12]: a SLAM method based on photoconsistency.
- *ArUco_MM* [35]: a SLAM method based on fiducial squared markers.
- UcoSLAM [13]: a SLAM method using both keypoints and fiducial squared markers.

For evaluation purposes we have employed two different datasets: the publicly available SPM dataset [9], and a new dataset created for this paper (the DCF dataset [4]).

Both datasets have been recorded in our laboratory where a set of fiducial squared markers have been placed at random locations. The SPM dataset consists of eight video sequences recorded with a PtGrey FLEA3 camera capturing 1920×1080 images at 60Hz. The videos show up to fifty different fiducial markers of 16.5 cm, distributed in the walls and ceiling of the room. The DCF dataset has nine video sequences recorded with an ELP camera capturing at 30 Hz frame rate with a resolution of 1920×1080 pixels. In this case, a total of 102 markers of a smaller size (7.9 cm), have been distributed by the walls and ceiling of the room. The videos of the DCF dataset have been recorded moving the camera fast and with brusque movements with the aim of achieving different degrees of blurring. In both cases, the ground truth camera poses are obtained using an Optitrack motion capture system equipped with six cameras (see Fig. 6).

[4] `https://mega.nz/folder/LiRCDYYb#aAOjirkUt54-0CGr3C6-1g`

While the ORBSlam2 and LDSO makes no use of the markers explicitly, the $ArUco\_MM$ and UcoSLAM methods use the markers for tracking. However, our method, TR-ArUco, requires the location of the marker to be know in advance (i.e. the marker map). The map has been created with the UcoSLAM method using a long video sequence that covers all markers in the room.

For the SLAM methods, the following methodology has been employed to analyze the video sequences. The sequence has been first processed to obtain the map and then, using the generated map, it is processed again to estimate the camera poses at each frame. In this way, the SLAM methods are evaluated after correcting possible loops in the sequence and obtains better accuracy. In consequence, a fair comparison with our method, that has a known map of the environment build in a previous phase, can be made.

Table 5 show the results obtained. For each video sequence (row) and method (column), three measures have been obtained. First, the computing time (FPS). Second, the Absolute Trajectory Error (ATE), which is the translational RMSE after $Sim(3)$ alignment [45] of the estimated poses with the ground truth. And third, the percentage of the video sequence frames for which the method provides a pose estimation (%Trck). It must be indicated that SLAM systems do not provide estimations in all the frames of a sequence: in some cases, they get lost due to fast movement or lack of texture.

Two conclusions can be drawn from Table 5. First, the proposed method outperforms the others in terms of speed and percentage of tracked frames. Second, that the LDSO method performs poorly in most of the sequences tested.

Table 5: Results obtained for each method in the SMP [9] and DCF datasets. For each sequence, the frames per second (FPS), absolute trajectory error (ATE), and percentage of tracked frames (%Trck) are reported.

Dataset	Sequence	TR-ArUco			ArUco_MM			LDSO			ORB_SLAM2			UcoSLAM		
		FPS	ATE	%Trck	FPS	ATE	%Trck	FPS	ATE	%Trck	FPS	ATE	%Trck	FPS	ATE	%Trck
SPM	video1	58.5	0.068	99.8	48.8	0.062	99.6	2.97	0.769	46.2	12.6	2.360	99.3	9.96	0.378	65.1
SPM	video2	62.0	0.111	99.8	53.4	0.103	98.5	1.47	1.250	99.8	10.8	0.575	97.4	22.1	0.054	99.8
SPM	video3	49.1	0.061	99.8	42.8	0.058	98.2	1.65	2.320	99.8	12.6	0.054	99.8	24.7	0.098	99.8
SPM	video4	44.6	0.015	99.8	45.5	0.013	99.2	0	0.000	0.03	12.2	0.020	99.8	24.9	0.011	99.8
SPM	video5	38.8	0.023	99.8	41.4	0.019	98.6	0	0.000	0.04	11.8	1.410	94.7	23.1	0.026	98.0
SPM	video6	34.2	0.145	99.8	45.1	0.018	98.6	0	0.000	0.04	11.4	0.527	96.8	6.46	0.670	52.2
SPM	video7	36.0	0.950	99.8	31.3	1.020	99.2	0	0.000	0.05	9.62	1.280	99.4	17.3	1.860	100.
SPM	video8	36.5	0.077	99.9	41.8	0.077	99.6	0	∞	0	3.05	0.437	55.0	17.9	0.049	99.8
DCF	video1	44.2	0.116	97.4	27.5	0.108	73.6	0	∞	0	1.11	0.499	30.4	5.46	0.095	57.6
DCF	video2	43.6	0.095	96.5	31.1	0.114	80.2	0	∞	0	0	∞	0	9.18	0.109	73.1
DCF	video3	51.7	0.085	99.9	42.6	0.082	91.9	0	∞	0	0	∞	0	12.3	0.105	87.6
DCF	video4	46.5	0.072	99.9	44.2	0.076	93.2	0	∞	0	0	∞	0	12.7	0.074	88.8
DCF	video5	29.0	0.163	81.9	19.4	0.106	60.3	0	∞	0	1.56	0.293	38.0	4.07	0.081	53.7
DCF	video6	38.7	0.093	99.9	31.6	0.101	82.3	0	∞	0	0	∞	0	7.82	0.092	75.8
DCF	video7	42.8	0.116	94.7	27.6	0.114	72.3	0	∞	0	0.04	0.040	6.62	5.61	0.102	65.0
DCF	video8	52.1	0.067	99.9	52.6	0.071	98.5	0	∞	0	7.39	0.303	84.4	14.8	0.065	96.6
DCF	video9	41.5	0.074	99.9	45.2	0.082	96.0	0	∞	0	0	∞	0	11.0	0.067	88.5

However, comparing the results of two SLAM methods is not a trivial task. Imagine a method that only estimates the pose of the camera in the first ten frames while a second method estimates poses in the whole sequence. Because of the reduced drift in the first frames, the total ATE of the first method will be smaller than the ATE of the second method (which evaluates the whole sequence). This is why (%Trck) is also an important aspect to consider.

The work [13] proposes an evaluation methodology to compare two SLAM methods A and B combining both the ATE and the %Trck. It defines a measure $S_p(A, B) \in [-1, 1]$ that employs a confidence level $p \in [0, 1]$. When $S_p(A, B)$ is close to 1, it indicates that the A method is better than B, while values close

Table 6: Measure $S_p(A, B)$ according to different confidence levels p of the analyzed methods. The final ranking shows TR-ArUco as the best, while LDSO provides the worst scores.

method A / method B	TR-ArUco			ArUco_MM			UcoSLAM			ORB_SLAM2			LDSO		
	$p = 0.01$	$p = 0.1$	$p = 0.25$	$p = 0.01$	$p = 0.1$	$p = 0.25$	$p = 0.01$	$p = 0.1$	$p = 0.25$	$p = 0.01$	$p = 0.1$	$p = 0.25$	$p = 0.01$	$p = 0.1$	$p = 0.25$
TR-ArUco	–	–	–	−0.5	−0.21	−0.15	−0.56	−0.5	−0.29	−0.75	−0.62	−0.58	−0.25	−0.25	−0.25
ArUco_MM	0.5	0.21	0.15	–	–	–	−0.26	−0.21	−0.18	−0.62	−0.58	−0.58	−0.12	−0.25	−0.25
UcoSLAM	0.56	0.5	0.29	0.26	0.21	0.18	–	–	–	−0.38	−0.29	−0.25	−0.25	−0.25	−0.25
ORB_SLAM2	0.75	0.62	0.58	0.62	0.58	0.58	0.38	0.29	0.25	–	–	–	−0.062	−0.12	−0.12
LDSO	0.25	0.25	0.25	0.12	0.25	0.25	0.25	0.25	0.25	0.062	0.12	0.12	–	–	–
Times winner	4	4	4	3	3	3	2	2	2	1	1	1	0	0	0

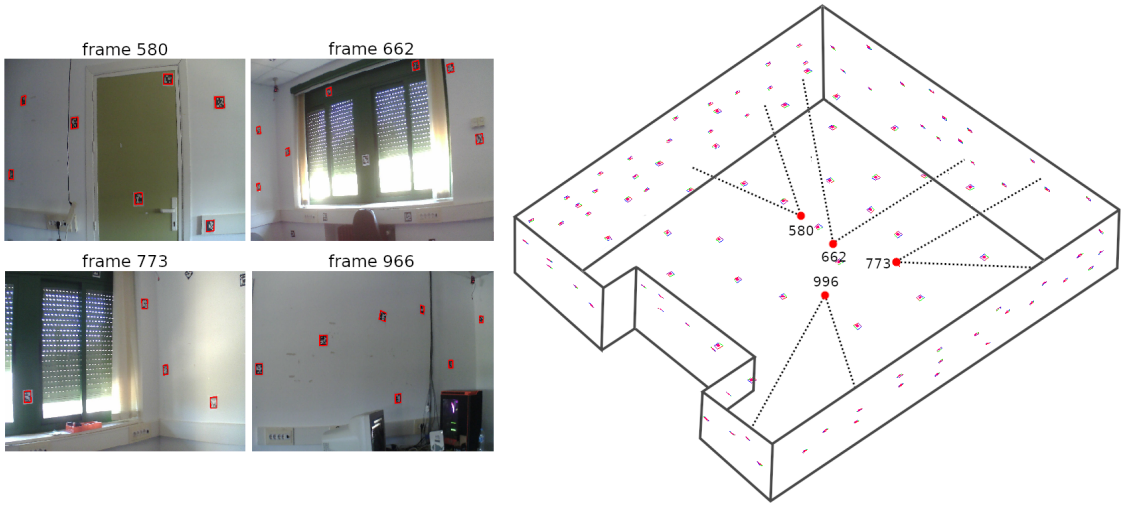

Figure 6: Map of markers displayed in the laboratory for experimentation. Some scenes of the environment corresponding to the first video are shown in it.

to −1 indicates that the B method is better than A.

Table 6 shows the values of $S_p(A, B)$ for each pair of methods, using the 17 sequences of the SPM and DCF datasets, for different confidence values. As can be seen, the proposed method $TR - ArUco$ obtains best scores than the rest of the methods for different confidence levels. The last row of the Table indicates how many times a method obtains better results than other methods. In our case, the value 4 means that proposed method wins to the other four tested methods.

The main conclusion that can be obtained from this experiment is that the proposed method outperforms the state-of-the-art SLAM methods in terms of speed, accuracy and sensitivity, for this particular problem.

7 Conclusions

This paper has proposed methods for tracking squared fiducial markers under challenging conditions. Our first contribution is a method for tracking squared marker using a set of Discriminative Correlation Filters which combines a proper scale selection and a corner upsampling strategy. The proposed method outper-

forms state-of-the-art methods for marker detection and standard DCFs in terms of speed, precision and sensitivity. In addition, our method scales very well with image resolution, obtaining more than 200fps in HD images using a single CPU thread.

Our second contribution is a method for low-cost camera pose estimation using fiducial marker maps. The proposed method is able to estimate the pose of a camera by tracking the position of the already visible markers and predicting the location of the markers appearing in the scene. Our method has been compared to state-of-the-art SLAM methods obtaining, better accuracy, sensitivity, and speed.

The proposed methods are publicly available for other researchers as part of the ArUco library[5], and the datasets employed in this paper are available to ease the reproduction of the experiments.

Acknowledgments

This project has been funded under projects TIN2019-75279-P and IFI16/00033 (ISCIII) of Spain Ministry of Economy, Industry and Competitiveness, and FEDER.

References

[1] H. Nakawala, G. Ferrigno, E. D. Momi, Development of an intelligent surgical training system for thoracentesis, Artificial Intelligence in Medicine 84 (2018) 50 – 63.

[2] P. Matthies, B. Frisch, J. Vogel, T. Lasser, M. Friebe, N. Navab, Inside-Out Tracking for Flexible Hand-held Nuclear Tomographic Imaging, in: IEEE Nuclear Science Symposium and Medical Imaging Conference, San Diego, USA, 2015.

[3] P. K. Kanithi, J. Chatterjee, D. Sheet, Immersive augmented reality system for assisting needle positioning during ultrasound guided intervention, in: Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing, ICVGIP '16, ACM, New York, NY, USA, 2016, pp. 65:1–65:8.

[4] E. Marchand, H. Uchiyama, F. Spindler, Pose estimation for augmented reality: A hands-on survey, IEEE Transactions on Visualization and Computer Graphics 22 (12) (2016) 2633–2651.

[5] H. Duan, Q. Zhang, Visual measurement in simulation environment for vision-based uav autonomous aerial refueling, IEEE Transactions on Instrumentation and Measurement 64 (9) (2015) 2468–2480.

[6] A. Marut, K. Wojtowicz, K. Falkowski, Aruco markers pose estimation in uav landing aid system, in: 2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace), 2019, pp. 261–266.

[7] M. F. Sani, G. Karimian, Automatic navigation and landing of an indoor ar. drone quadrotor using aruco marker and inertial sensors, in: 2017 International Conference on Computer and Drone Applications (IConDA), 2017, pp. 102–107.

[8] R. Polvara, S. Sharma, J. Wan, A. Manning, R. Sutton, Towards autonomous landing on a moving vessel through fiducial markers, in: 2017 European Conference on Mobile Robots (ECMR), 2017, pp. 1–6.

[9] R. Muñoz-Salinas, M. J. Marín-Jiménez, R. Medina-Carnicer, Spm-slam: Simultaneous localization and mapping with squared planar markers, Pattern Recognition 86 (2019) 156 – 171.

[5] `https://www.uco.es/investiga/grupos/ava/node/26`

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

[11] R. Mur-Artal, J. D. Tardós, Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras, IEEE Transactions on Robotics 33 (5) (2017) 1255–1262.

[12] X. Gao, R. Wang, N. Demmel, D. Cremers, Ldso: Direct sparse odometry with loop closure, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 2198–2204.

[13] R. Muñoz-Salinas, R. Medina-Carnicer, Ucoslam: Simultaneous localization and mapping by fusion of keypoints and squared planar markers, Pattern Recognition 101 (2020) 107193.

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

[15] Q. Bonnard, S. Lemaignan, G. Zufferey, A. Mazzei, S. Cuendet, N. Li, A. Özgür, P. Dillenbourg, Chilitags 2: Robust fiducial markers for augmented reality and robotics. (2013).
URL `http://chili.epfl.ch/software`

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

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

[18] A. Sagitov, K. Shabalina, R. Lavrenov, E. Magid, Comparing fiducial marker systems in the presence of occlusion, in: 2017 International Conference on Mechanical, System and Control Engineering (ICMSC), 2017, pp. 377–382.

[19] F. J. Romero-Ramirez, R. Muñoz-Salinas, R. Medina-Carnicer, Speeded up detection of squared fiducial markers, Image and Vision Computing 76 (2018) 38–47.

[20] F. J. Romero-Ramirez, R. Muñoz-Salinas, R. Medina-Carnicer, Fractal markers: A new approach for long-range marker pose estimation under occlusion, IEEE Access 7 (2019) 169908–169919.

[21] M. Krogius, A. Haggenmiller, E. Olson, Flexible layouts for fiducial tags, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 1898–1903.

[22] D. S. Bolme, J. R. Beveridge, B. A. Draper, Y. M. Lui, Visual object tracking using adaptive correlation filters, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010, pp. 2544–2550.

[23] J. F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (3) (2015) 583–596.

[24] M. Danelljan, F. S. Khan, M. Felsberg, J. v. d. Weijer, Adaptive color attributes for real-time visual tracking, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1090–1097.

[25] Y. Li, J. Zhu, A Scale Adaptive Kernel Correlation Filter Tracker with Feature Integration, in: Computer Vision - ECCV Workshops, 2014, pp. 254–265.

[26] A. Lukežič, T. Vojíř, L. Čehovin Zajc, J. Matas, M. Kristan, Discriminative correlation filter tracker with channel and spatial reliability, International Journal of Computer Vision 126 (7) (2018) 671–688.

[27] C. Ma, J. Huang, X. Yang, M. Yang, Hierarchical convolutional features for visual tracking, in: 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 3074–3082.

[28] M. Danelljan, G. Häger, F. S. Khan, M. Felsberg, Convolutional features for correlation filter based visual tracking, in: 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), 2015, pp. 621–629.

[29] M. Danelljan, A. Robinson, F. Khan, M. Felsberg, Beyond correlation filters: Learning continuous convolution operators for visual tracking, Springer International Publishing, 2016, pp. 472–488.

[30] M. Mueller, N. Smith, B. Ghanem, Context-aware correlation filter tracking, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1387–1395.

[31] M. Danelljan, G. Häger, F. S. Khan, M. Felsberg, Learning spatially regularized correlation filters for visual tracking, in: 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 4310–4318.

[32] M. Danelljan, G. Häger, F. Khan, M. Felsberg, Accurate scale estimation for robust visual tracking, in: Proceedings of the British Machine Vision Conference, BMVA Press, 2014, pp. 1–11.

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

[34] G. Bradski, A. Kaehler, Learning OpenCV: Computer Vision in C++ with the OpenCV Library, 2nd Edition, O'Reilly Media, Inc., 2013.

[35] R. Muñoz-Salinas, M. J. Marín-Jimenez, E. Yeguas-Bolivar, R. Medina-Carnicer, Mapping and localization from planar markers, Pattern Recognition 73 (2018) 158 – 171.

[36] M. Fiala, Designing highly reliable fiducial markers, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (7) (2010) 1317–1324.

[37] K. Madsen, H. B. Nielsen, O. Tingleff, Methods for non-linear least squares problems (2nd ed.) (2004).

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

[39] 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 2 (10) (1973) 112 – 122.

[40] J. F. Henriques, R. Caseiro, P. Martins, J. Batista, Exploiting the circulant structure of tracking-by-detection with kernels, in: Computer Vision – ECCV 2012, Springer Berlin Heidelberg, 2012, pp. 702–715.

[41] B. Babenko, M.-H. Yang, S. Belongie, Visual tracking with online multiple instance learning, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 983–990.

[42] Z. Kalal, K. Mikolajczyk, J. Matas, Tracking-learning-detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (7) (2012) 1409–1422.

[43] Z. Kalal, K. Mikolajczyk, J. Matas, Forward-backward error: Automatic detection of tracking failures, in: 2010 20th International Conference on Pattern Recognition, 2010, pp. 2756–2759.

[44] H. Grabner, M. Grabner, H. Bischof, Real-time tracking via on-line boosting, Vol. 1, 2006, pp. 47–56.

[45] J. Engel, T. Schöps, D. Cremers, LSD-SLAM: Large-scale direct monocular SLAM, in: European Conference on Compututer Visision (ECCV), 2014.