Refined Pose Estimation for Square Markers Using Shape Fitting

Antonio Zea and **Uwe D. Hanebeck**
Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Institute for Anthropomatics and Robotics
Karlsruhe Institute of Technology (KIT), Germany
antonio.zea@kit.edu, uwe.hanebeck@ieee.org

***Abstract*—We introduce an algorithm to refine the estimation of corners and pose of square fiducial markers, such as Arucos, with focus on mobile augmented reality applications. The idea is to reduce pixel jitter, which causes distracting artifacts such as "vibrating" objects, by exploiting information from the contour pixels of the detected markers. To achieve this, we develop a nonlinear least squares estimator that models a marker explicitly as a polygon and employs ideas from *shape fitting*. This provides not only a best-fitting estimate of the corners, but also a covariance matrix that can be used during further processing. We also implement a pose estimator that incorporates these covariance matrices and show how the effect of pixel jitter is greatly reduced in our approach, without increasing resource usage substantially in mobile devices.**

I. Introduction

Markers, or more accurately *fiducial* markers [1], are artificial landmarks that facilitate correspondences between image points and reference points in the world. The key idea of these markers is that they are easy to find in a captured (usually color) image, with relatively low false positives, and based on prior information we can know the three-dimensional positions that corresponds to the image points. Localizing these markers, and its dual problem, localizing the camera based on these markers, are very common challenges in augmented reality (AR) applications [2], [3]. Recently, platforms that allow for reliable markerless tracking [4] have reached wide acceptance, especially after the entrance of technological giants such as Apple ARKit [5], Google ARCore [6], or Microsoft Hololens [7]. Nonetheless, tracking artificial markers is still highly relevant in many applications, for example in scenarios where it is necessary to track known objects in the environment accurately, or when it is needed to provide an absolute starting position of the device. Given that AR generally runs on mobile devices, marker detection and localization have resource constraints that are not present in other hardware. On the one hand, the CPU load of the tracking algorithms is an important concern, not only because of these devices have lower CPU performance to begin with, but also to reduce battery drain. On the other hand, the image stream usually has a lower resolution, and the aperture is increased to reduce motion blur, both of which have a negative effect on pixel noise, particularly in conditions with low lighting. On the bright side, image streams from the Hololens and ARKit/ARCore platforms employ automatic preprocessing such as correction for lens distortion, and the intrinsic parameters are provided by the hardware.

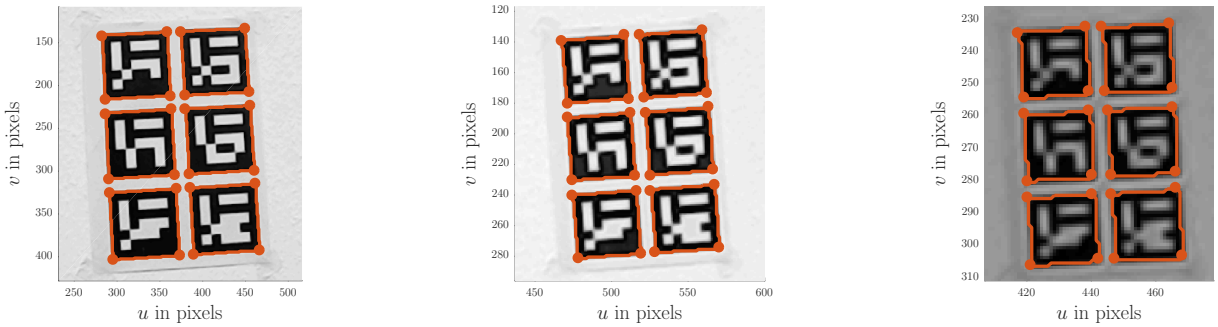
Many forms of fiducial markers have been presented in literature. Square planar markers [8]–[12] are particularly attractive given that they are easy to print on paper their corner positions can be used for localization. The marker interior contains some sort of bit pattern that encodes an identifier and can be used for error detection and correction. Probably one of the best known marker types are Aruco markers [13], [14], which provide relatively fast detection with low false positives. However, the localization of square markers is generally not as robust as other artificial landmarks such as chess boards, as their corner positions tend to suffer more strongly from *pixel jitter*, i.e., noise between frames. This produces a strong negative effect on AR immersion, in drastic cases causing the impression that some objects or even the entire scene is "vibrating".

In order to introduce our contribution to this topic, we will first present an abridged sketch of how square markers such as Arucos are detected from a grayscale image [13]–[15]:

1) First, an adaptive thresholding step is applied.
2) From the resulting binary image, a list of contours is detected [16]. Each contour is represented as an array of integer pixel positions.
3) Contours deemed inappropriate, based on criteria such as an implausible perimeter or dissimilarity to a polygon [17], are discarded.
4) The interior of the contour is divided into cells from which a sequence of bits is extracted, used for error detection and marker identification [13], [14].
5) Finally, the contour corners, together with the marker ID, are returned to the application.

The obtained contour corners, being integer pixel positions, are too unreliable for marker localization. Examples of obtained contours can be seen in Fig. 1, which shows how their accuracy strongly degrades with distance. A common approach to address this is to apply corner refinement based on gradients [13], [15], or for higher accuracy, to apply a linear regression for each contour edge and calculate the intersections [11], [12], [14].

In this paper, we present an alternative approach for corner refinement based on contour data, which employs ideas from shape fitting [18] and least squares shape estimation [19]–[21]. The idea is to model the observed marker explicitly as a tetragon (four-sided polygon) and then develop a nonlinear estimator that minimizes the distance between contour points to the shape. This batch estimator produces not only a best-fitting estimate, but also a covariance matrix that naturally describes how much each corner should be weighted. These terms can then be used to refine the localization step, yielding a pose estimate and covariance matrix that can be employed in (nonlinear)



(a) Markers at 1 meter. (b) Markers at 2 meters. (c) Markers at 4 meters.

Figure 1: Detected contours at different distances.

Kalman filters [22], [23]. Our approach does not impose further requirements than the information already gathered during detection and the overhead it requires is relatively insignificant, allowing its use in mobile AR applications. We will also show how our contribution improves upon gradient refinement and linear regression techniques. Note that this paper does not introduce new approaches for fiducial marker design or detection, we simply focus on corner and pose refinement from existing data.

The paper is structured as follows. First, we give a brief review of the nonlinear least squares method in Sec. II. Then, we introduce our corner refinement technique in Sec. III, and develop a pose refinement estimator capable to use these results in Sec. IV. Finally, we present our evaluation in V, and the conclusions in Sec. VI.

II. Nonlinear Least Squares

In this section we will present a short description of the nonlinear least squares (NLSQ) estimation technique. We assume that we have a given set of measurements $\underline{y} \in \mathbb{R}^m$, related to the state parameters $\underline{x} \in \mathbb{R}^n$ through the measurement function $\underline{h} : \mathbb{R}^n \to \mathbb{R}^m$ with $m \geq n$, i.e.,

$$\underline{y} = \underline{h}(\underline{x}) + \underline{v} \ ,$$

where $\underline{v}$ is zero-mean noise with known covariance matrix $\mathbf{C}_y$. The idea behind NLSQ estimation is to find the "best" fitting state for those measurements, i.e., the state that minimizes the squared residuals weighted by $\mathbf{W} = \mathbf{C}_y^{-1}$. However, if $\underline{h}$ is nonlinear, finding a general solution is not straightforward. We can address this by linearizing $\underline{h}$ around a given point $\underline{x}_0 \in \mathbb{R}^n$, yielding the approximation

$$\underline{y} - \underline{h}(\underline{x}_0) \approx \mathbf{J}(\underline{x} - \underline{x}_0) + \underline{v} \ , \tag{1}$$

where $\mathbf{J} \in \mathbb{R}^{m \times n}$ is the Jacobian of $\underline{h}$ evaluated in $\underline{x}_0$. The optimal $\hat{\underline{x}}$ for this linear approximation (but not necessarily for $\underline{h}$) is determined in closed form by

$$\hat{\underline{x}} = \underline{x}_0 + \left(\mathbf{J}^T \mathbf{C}_y^{-1} \mathbf{J}\right)^{-1} \cdot \mathbf{J}^T \mathbf{C}_y^{-1} \cdot \left(\underline{y} - \underline{h}(\underline{x}_0)\right) \ .$$

The covariance matrix of this estimate is determined by

$$\begin{aligned} \mathbf{C}_{\hat{x}} &:= \mathrm{cov}(\underline{x}) \\ &= \left[\left(\mathbf{J}^T \mathbf{C}_y^{-1} \mathbf{J}\right)^{-1} \cdot \mathbf{J}^T \mathbf{C}_y^{-1}\right] \mathbf{C}_y \left[\mathbf{C}_y^{-1} \mathbf{J} \cdot \left(\mathbf{J}^T \mathbf{C}_y^{-1} \mathbf{J}\right)^{-1}\right] \\ &= \left(\mathbf{J}^T \mathbf{C}_y^{-1} \mathbf{J}\right)^{-1} \cdot \mathbf{J}^T \mathbf{C}_y^{-1} \mathbf{J} \cdot \left(\mathbf{J}^T \mathbf{C}_y^{-1} \mathbf{J}\right)^{-1} \\ &= \left(\mathbf{J}^T \mathbf{C}_y^{-1} \mathbf{J}\right)^{-1} \ . \end{aligned}$$

If there are multiple measurements $\{\underline{y}_1, ..., \underline{y}_\eta\}$ with corresponding measurement functions $\{\underline{h}_1, ..., \underline{h}_\eta\}$, they can be fused into a single measurement equation by stacking them vertically. However, if the measurements are known to be independent from each other, the calculations can be strongly simplified, yielding

$$\hat{\underline{x}} = \underline{x}_0 + \mathbf{C}_{\hat{x}} \cdot \left(\sum_{i=1}^{\eta} \mathbf{J}_i^T \mathbf{C}_{y_i}^{-1} \cdot \left(\underline{y}_i - \underline{h}_i(\underline{x}_0)\right)\right) \ ,$$

with covariance matrix

$$\mathbf{C}_{\hat{x}} = \left(\sum_{i=1}^{\eta} \mathbf{J}_i^T \mathbf{C}_{y_i}^{-1} \mathbf{J}_i\right)^{-1} \ . \tag{2}$$

The usefulness of $\hat{\underline{x}}$ is determined by how well (1) approximates $\underline{h}$, making the selection of $\underline{x}_0$ critical. By taking an initial guess for $\underline{x}_0$, and then refining it iteratively by using the estimated $\underline{x}^{opt}$ as the next $\underline{x}_0$, we obtain the Gauss-Newton method. A common refinement to this technique, in case that convergence is difficult, is to add a dampening parameter $\lambda \cdot \mathrm{diag}(\mathbf{J}_i^T \mathbf{C}_{y_i}^{-1} \mathbf{J}_i)$ to (2), with a carefully chosen λ. This is the basis of the Levenberg-Marquardt method [25].

We observe that to derive a NLSQ estimator we only need to calculate for each measurement i) a measurement function $\underline{h}_i(\underline{y}_i)$, and ii) its Jacobian $\mathbf{J}_i$. In the following sections, we will use this idea to derive nonlinear estimators for corner and pose refinement.

III. Corner Refinement

In this section, we will develop an NLSQ estimator to find the tetragon that best fits a given contour. The state is defined

by the tetragon corners $\underline{c}_1$, $\underline{c}_2$, $\underline{c}_3$, and $\underline{c}_4$ stacked vertically in clockwise order, i.e.,

$$\underline{x}^\tau = \begin{bmatrix} \underline{c}_1 \\ \underline{c}_2 \\ \underline{c}_3 \\ \underline{c}_4 \end{bmatrix} \in \mathbb{R}^8 \ .$$

The contour is assumed to be a list of points $Y = \{\underline{y}_1, ...\underline{y}_\eta\}$. For the sake of simplicity, we assume that the corners and the contour are represented in "world coordinates", i.e., after multiplying with the inverse of intrinsic matrix according to the pinhole model. For instance, if a point $\underline{y}^{px}$ with covariance matrix $\mathbf{C}_y^{px}$ is given in pixels, and the intrinsic parameters have the form

$$\mathbf{K} = \begin{bmatrix} f_x & s \\ 0 & f_y \end{bmatrix} \ , \text{ and } \underline{o} = \begin{bmatrix} o^x \\ o^y \end{bmatrix} \ ,$$

with focal lengths f_x, f_y, skew s, and offset $\underline{o}$, the world coordinates follow as

$$\underline{y} = \mathbf{K}^{-1}\left(\underline{y}^{px} - \underline{o}\right) \ ,$$
$$\mathbf{C}_y = \mathbf{K}^{-1}\mathbf{C}_y^{px}\left(\mathbf{K}^{-1}\right)^T \ .$$

We will now derive a measurement function $\underline{h}^\tau(\underline{x}^\tau) : \mathbb{R}^8 \to \mathbb{R}^2$ and the Jacobian $\mathbf{J}^\tau \in \mathbb{R}^{2\times 8}$ of each $\underline{y} \in Y$. In order to do this, we will employ ideas from shape fitting, where the idea is to minimize the distance between a given set of points and a shape, or more concretely, between the points and the corresponding closest points on the shape boundary.

For the tetragon determined by $\underline{x}^\tau$, the closest point can be found by checking each of the four sides and then choosing the point with the minimum Euclidean distance. Let us, at first, consider only the segment determined by the corners $\underline{c}_j$ and $\underline{c}_k$, with the indices $\{j, k\}$ taken from $I := \{\{1,2\},\{2,3\},\{3,4\},\{4,1\}\}$. The closest point to $\underline{y}$ in that segment is determined by

$$\underline{\pi}_{j,k}(\underline{y}) = \underline{c}_j + \mathrm{clamp}(t_{j,k})\left(\underline{c}_k - \underline{c}_j\right) \ , \tag{3}$$

where $t_{j,k}$ is a helper term defined as

$$t_{j,k} := \frac{\overbrace{\left(\underline{y} - \underline{c}_j\right)^T\left(\underline{c}_k - \underline{c}_j\right)}^{:=t^u_{j,k}}}{\underbrace{\left|\underline{c}_k - \underline{c}_j\right|^2}_{:=t^l_{j,k}}} \tag{4}$$

and $\mathrm{clamp}(t_{j,k}) := \max(\min(t_{j,k}, 1), 0)$. Then, we determine the closest segment by

$$\{j^*, k^*\} = \underset{\{j,k\}\in I}{\arg\min} \left|\underline{\pi}_{j,k}(\underline{y}) - \underline{y}\right| \tag{5}$$

and from that we obtain the measurement function as

$$\underline{h}^\tau(\underline{x}^\tau) = \underline{\pi}_{j^*,k^*}(\underline{y}) \ .$$

If two sides are equally close, then any of them can be chosen. The Jacobian can be calculated using similar ideas by first considering a segment $\{j, k\}$. However, the $\mathrm{clamp}(\cdot)$ function will be ignored, given that it is not derivable at the corners and it is equal to the identity for the vast majority of points. For the sake of convenience, we will first consider a reduced state that contains only two corners, i.e.,

$$\underline{x}^\tau_{j,k} = \begin{bmatrix} \underline{c}_j \\ \underline{c}_k \end{bmatrix} \in \mathbb{R}^4 \ ,$$

First, we calculate the derivative of $t_{j,k}$ w.r.t. $\underline{x}^\tau_{j,k}$ in (4). For the upper term, the derivative is

$$\left(t^u_{j,k}\right)' = \begin{bmatrix} -\underline{y} - \underline{c}_k + 2\underline{c}_j \\ \underline{y} - \underline{c}_j \end{bmatrix}^T \in \mathbb{R}^{1\times 4} \ .$$

Similarly, for the lower term we obtain

$$\left(t^l_{j,k}\right)' = 2\begin{bmatrix} -\left(\underline{c}_k - \underline{c}_j\right) \\ \underline{c}_k - \underline{c}_j \end{bmatrix}^T \in \mathbb{R}^{1\times 4} \ .$$

We combine both terms using the quotient rule to obtain

$$\left(t_{j,k}\right)' = \frac{1}{t^l_{j,k}} \cdot \left(t^u_{j,k}\right)' - \frac{t_{j,k}}{t^l_{j,k}} \cdot \left(t^l_{j,k}\right)' \in \mathbb{R}^{1\times 4} \ .$$

Then, we calculate the derivative of (3) by employing the product rule, yielding

$$\mathbf{J}^\tau_{j,k} = \left(\underline{c}_k - \underline{c}_j\right) \cdot \left[\left(t_{j,k}\right)'\right]^T + \left[(1 - t_{j,k})\mathbf{I}_2 \quad t_{j,k}\mathbf{I}_2\right] \ ,$$

where $\mathbf{J}^\tau_{j,k} \in \mathbb{R}^{2\times 4}$. Finally, we split $\mathbf{J}^\tau_{j,k}$ into a left part and a right part, i.e.,

$$\mathbf{J}^\tau_{j,k} = \begin{bmatrix} \mathbf{J}^\tau_j & \mathbf{J}^\tau_k \end{bmatrix} \ ,$$

where $\mathbf{J}^\tau_j \in \mathbb{R}^{2\times 2}$ and $\mathbf{J}^\tau_k \in \mathbb{R}^{2\times 2}$ represent the partial derivatives for $\underline{c}_j$ and $\underline{c}_k$ respectively.

Taking the minimum indices $\{j^*, k^*\}$ from (5), we now construct the derivative of the entire state $\mathbf{J}^\tau \in \mathbb{R}^{2\times 8}$ as a block of four 2×2 matrices, where the block for j^* is occupied by $\mathbf{J}^\tau_j$, the block for k^* is taken by $\mathbf{J}^\tau_k$, and all other blocks are zero. For example, let us assume that the closest segment corresponds to $j^* = 2$ and $k^* = 3$. Then, $\mathbf{J}^\tau$ takes the form

$$\mathbf{J}^\tau = \begin{bmatrix} 0_{2\times 2} & \mathbf{J}^\tau_2 & \mathbf{J}^\tau_3 & 0_{2\times 2} \end{bmatrix} \in \mathbb{R}^{2\times 8} \ .$$

IV. Pose Refinement

Obtaining a pose from measured image points is usually known as perspective-n-point (PnP). Informally speaking, the idea of PnP is that, given a set of measured image points in $\mathbb{R}^2$ and a set of reference points in $\mathbb{R}^3$ known a priori, we need to find a pose so that, by transforming the reference points, and then projecting them onto the screen, the resulting points are as close as possible to the image points. In literature, this problem has been explored for a long time [26], with several solutions depending on the amount of information [27]–[29] or the hardware being used [30], [31]. However, incorporating covariance matrices as weights in these approaches is not straightforward. For the sake of completeness, in this subsection we will introduce a simple NLSQ estimator that solves PnP and can incorporate the estimates from Sec. III.

More concretely, we obtain as measurements four image points which represent the corners of a tetragon

$$\underline{y}^\tau = \begin{bmatrix} \underline{c}_1 \\ \underline{c}_2 \\ \underline{c}_3 \\ \underline{c}_4 \end{bmatrix} \in \mathbb{R}^8 \ ,$$

with associated covariance matrix $\mathbf{C}_y^\tau$. In general, these values are taken from the corners estimated in Sec. III and the corresponding covariance matrix, i.e.,

$$\begin{aligned}\underline{y}^\tau &= \hat{\underline{x}}^\tau \\ \mathbf{C}_y^\tau &= \mathbf{C}_{\hat{x}}^\tau \ .\end{aligned}$$

We also have a set of known reference points $\underline{c}_1^r$, $\underline{c}_2^r$, $\underline{c}_3^r$, and $\underline{c}_4^r$ in $\mathbb{R}^3$ that correspond to each tetragon corner, and are independent from $\underline{x}^p$ and $\underline{y}^\tau$. The pose transformation is determined by the parameters

$$\underline{x}^p := \begin{bmatrix}\underline{x}_r^p \\ \underline{x}_t^p\end{bmatrix} \in \mathbb{R}^6 \ ,$$

where $\underline{x}_r^p \in \mathbb{R}^3$ represents the rotation and $\underline{x}_t^p \in \mathbb{R}^3$ the translation. For this paper, we choose the *Rodrigues* representation for the rotation, which encodes a rotation with angle θ_r around a unitary axis vector $\underline{u}_r$ as $\underline{x}_r^p = \theta_r \cdot \underline{u}_r$. Analogously, given a $\underline{x}_r^p$ we can regain both parameters from

$$\begin{aligned}\theta_r &= |\underline{x}_r^p| \\ \underline{u}_r &= \frac{1}{\theta_r}\underline{x}_r^p \ .\end{aligned}$$

The rotation matrix $\mathbf{R}(\underline{x}_r^p) \in \mathbb{R}^{3\times3}$ results from the Rodrigues formula

$$\mathbf{R}(\underline{x}_r^p) = \cos(\theta_r)\,\mathbf{I}_3 + \sin(\theta_r)\,[\underline{u}_r]_\times + (1-\cos(\theta_r))\,\underline{u}_r\underline{u}_r^T$$

employing the cross-product matrix representation $[\cdot]_\times$ defined as

$$[\underline{a}]_\times = \left[[a_1,a_2,a_3]^T\right]_\times := \begin{bmatrix}0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0\end{bmatrix} \ .$$

For the special case of $\underline{x}_r^p = \underline{0}$, we obtain $\mathbf{R}(\underline{x}_r^p) = \mathbf{I}_3$.

We can now use these terms to derive a measurement function $\underline{h}^p(\underline{x}^p) : \mathbb{R}^6 \to \mathbb{R}^8$ and the corresponding Jacobian $\mathbf{J}^p \in \mathbb{R}^{8\times6}$. First, the transformation of a single reference point $\underline{c}_i^r$ for $i \in \{1,2,3,4\}$ is determined by $\mathrm{trans}_i(\underline{x}^p) : \mathbb{R}^3 \to \mathbb{R}^3$ where

$$\mathrm{trans}_i(\underline{x}^p) := \mathbf{R}(\underline{x}_r^p)\cdot\underline{c}_i^r + \underline{x}_t^p \ . \tag{6}$$

Second, we define the projection function $\mathrm{proj} : \mathbb{R}^3 \to \mathbb{R}^2$ as

$$\begin{aligned}\mathrm{proj}(\underline{a}) &:= \mathrm{proj}([a_1,a_2,a_3]^T) \\ &= \frac{1}{a_3}\begin{bmatrix}a_1 \\ a_2\end{bmatrix}\end{aligned}$$

Finally, the measurement function is given by

$$\underline{h}_i^p(\underline{x}^p) := \mathrm{proj}(\mathrm{trans}_i(\underline{x}^p)) \in \mathbb{R}^2 \ , \tag{7}$$

and by incorporating all four reference points,

$$\underline{h}^p(\underline{x}^p) = \begin{bmatrix}\underline{h}_1^p(\underline{x}^p) \\ \underline{h}_2^p(\underline{x}^p) \\ \underline{h}_3^p(\underline{x}^p) \\ \underline{h}_4^p(\underline{x}^p)\end{bmatrix} \in \mathbb{R}^8 \ .$$

The task is now to derive a Jacobian for this expression. First, we define the helper function $\mathrm{rot}_i : \mathbb{R}^3 \to \mathbb{R}^3$ as

$$\mathrm{rot}_i(\underline{x}_r^p) := \mathbf{R}(\underline{x}_r^p)\cdot\underline{c}_i^r \ ,$$

with derivative [32]

$$\mathrm{rot}_i'(\underline{x}_r^p) = -\mathbf{R}(\underline{x}_r^p)\,[\underline{c}_i^r]_\times\ \frac{\underline{x}_r^p\cdot(\underline{x}_r^p)^T + \left(\mathbf{R}(\underline{x}_r^p)^T - \mathbf{I}_3\right)[\underline{x}_r^p]_\times}{|\underline{x}_r^p|^2} \ ,$$

and for the special case of $\underline{x}_r^p = \underline{0}$

$$\mathrm{rot}_i'(\underline{x}_r^p) = -\,[\underline{c}_i^r]_\times \ .$$

It follows from (6) that

$$\mathrm{trans}_i'(\underline{x}^p) = [\mathrm{rot}_i'(\underline{x}_r^p) \quad \mathbf{I}_3] \in \mathbb{R}^{3\times6} \ .$$

Now, we introduce the following helper terms, defined informally as:

- $z_i \in \mathbb{R}$ is the third component of $\mathrm{trans}_i(\underline{x}^p)$,
- $\mathbf{T}_i^u \in \mathbb{R}^{2\times6}$ is the first two rows of $\mathrm{trans}_i'(\underline{x}^p)$, and
- $\mathbf{T}_i^d \in \mathbb{R}^{1\times6}$ is the last row of $\mathrm{trans}_i'(\underline{x}^p)$.

Using the quotient rule in (7), we finally obtain

$$\mathbf{J}_i^p := \frac{\mathbf{T}_i^u - \underline{h}_i^p(\underline{x}^p)\cdot\mathbf{T}_i^d}{z_i} \ ,$$

with $\mathbf{J}_i^p \in \mathbb{R}^{2\times6}$, and with all four reference points together,

$$\mathbf{J}^p = \begin{bmatrix}\mathbf{J}_1^p \\ \mathbf{J}_2^p \\ \mathbf{J}_3^p \\ \mathbf{J}_4^p\end{bmatrix} \in \mathbb{R}^{8\times6} \ .$$

In applications of extrinsic camera calibration, it is a common approach to find the camera pose by first finding the pose of the markers in relation to the camera, and then calculating the inverse of that pose. Given the parameters $\underline{x}^p$, the inverse pose can be obtained easily as

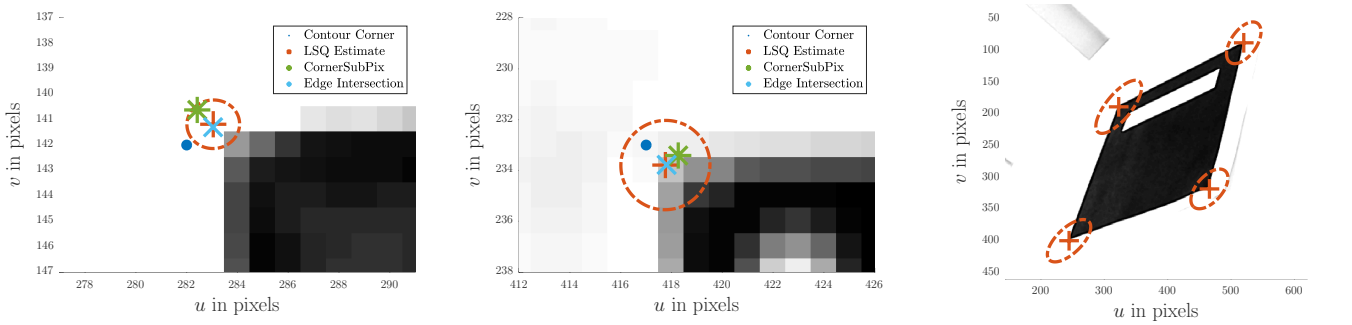
$$\underline{x}^{p,inv} := \begin{bmatrix}\underline{x}_r^{p,inv} \\ \underline{x}_t^{p,inv}\end{bmatrix} = \begin{bmatrix}-\underline{x}_r^p \\ \mathbf{R}(-\underline{x}_r^p)\cdot-\underline{x}_t^p\end{bmatrix} \ . \tag{8}$$

V. Evaluation

In this section, we will evaluate our proposed approach and compare it to similar state-of-the-art techniques. First, we evaluate the effect of pixel jitter in the the corner estimate, and then we proceed to evaluate how this noise affects the pose estimate. We will also present a brief discussion of the results.

A. Corner Refinement

We start by evaluating the corner refinement step. The images were captured by the RGB camera of a Microsoft Hololens device, with a resolution of 896×504 pixels, as recommended by the manufacturer for image processing tasks. As mentioned before, the Hololens applies image preprocessing to remove distortions, but this introduces some blur into the marker edges. The intrinsic calibration matrix was automatically provided by the hardware. While we employed Aruco markers, we emphasize that the approaches discussed here can be used with any similar square marker. The marker size was 8×8 cm^2. For the contour and Aruco detection, the library OpenCV version 3.4.5 was employed, compiled from source for the UWP Hololens platform.



(a) Results for top-left corner at 1 meter. (b) Results for top-left corner at 4 meters. (c) LSQ estimates with covariance matrices.

Figure 2: Results and covariance matrices for different setups. The covariance matrices have been upscaled for legibility.

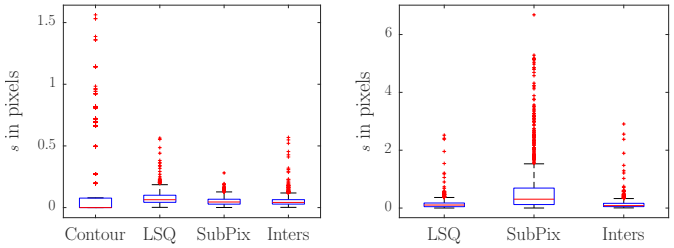
Three techniques were considered:

- the direct contour corners provided by the OpenCV detector ('Contour'),
- the LSQ estimate from our proposed approach ('LSQ'),
- the results of the OpenCV *cornerSubPix* function ('CornerSubPix'), which refines a corner by exploiting image gradient information from a small neighborhood around it,
- and the edge linear regression approach used in [11], [12], [14] ('Edge Intersection').

For the LSQ approach, a measurement covariance matrix of $\mathbf{C}_y^{pix} = 4 \cdot \mathbf{I}_2$ was employed. The starting state was the four corners provided by OpenCV. For the *cornerSubPix* function, default parameters were used. For the line regression approach in Edge Intersection, each contour point was associated with the closest edge, and then total least squares regression was employed to estimate the parameters of each edge line. The corners result as the intersection of these lines.

Fig. 2 shows a visual representation of the results. Note that contrast has been increased strongly for legibility (see Fig. 1 for the originals), which makes the marker exteriors look cleaner than in reality. In Fig. 2a we see the results for the top-left corner of the top-left marker, taken at a distance of 1 meter. We observe that Edge Intersection and LSQ provide very similar results, which is not surprising given that both are the result of edge distance minimization. The CornerSubPix approach, which only employs local information instead of the contour, is closer to the LSQ estimate than to the contour corner.

In Fig. 2b, we see the results that correspond to the same marker in Fig. 1c, this time taken at a distance of 4 meters. Image quality has degraded considerably, and we can even see parts of the marker interior. Once more, LSQ and Edge Intersection are almost identical, with CornerSubPix coming close. The covariance matrix of LSQ, however, is now twice as large. As a reminder, the weight of each measurement is the inverse of its covariance matrix, meaning that this marker would be weighted half as much as the one in Fig. 2a. Also, as an aside, we want to point out that the corner covariance matrices do not need to be isotropic, or even identical within the same marker. For example, in Fig. 2c, we observe another



(a) Deviations at 1 meter. (b) Deviations at 3 meters.

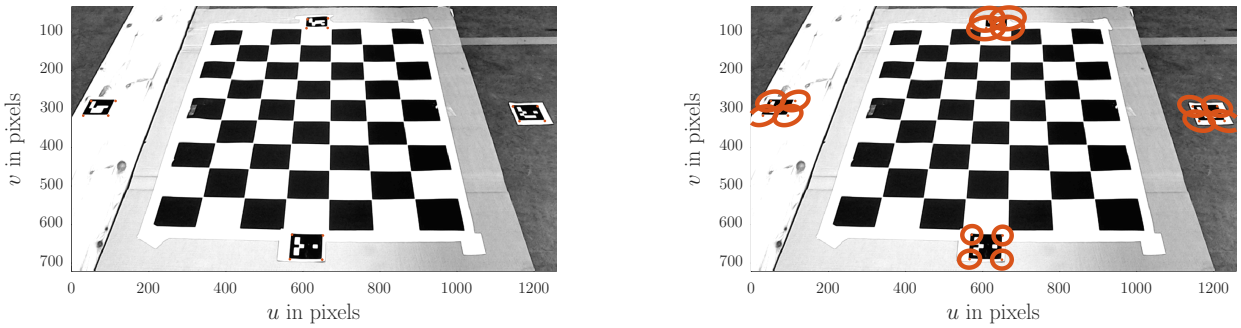
Figure 3: Distance deviations from the mean at different distances. The red line denotes the median, the blue box represents the range between 25th and 75th percentiles. Red dots are assumed outliers.

marker being captured from a skewed angle, with each corner having an individual covariance matrix (and weight).

In the context of corner refinement, an important criterion is the amount of *pixel jitter*, i.e., the change in the corner position between frames caused by pixel noise. In Fig. 3, pixel jitter was measured by taking 60 frames and calculating the Euclidean distance from the mean. As there are 6 markers with 4 corners, we obtained 1440 measurements. Fig. 3 shows the results taken at different distances.

First, in the results for 1 meter shown in Fig. 3a we see a somewhat paradoxical result: the contour corners ('Contour') have the lowest mean but the highest amount of outliers. This is an artifact from the fact that the contour corners are always on the exact pixel center, and thus, the coordinates are always integers. In turn, most distances are going to be clamped down to 0, making the mean and variance unrepresentatively small. From the remaining approaches, LSQ and Edge Intersection have similar values. Unexpectedly, CornerSubPix has the smallest spread, even if it only uses local information. This is a consequence from the fact that, at 1 meter, the corner neighborhood has relatively high contrast, meaning that the corner will not move much between frames.

However, the SubPix spread changes strongly in Fig. 3b, taken at 3 meters where pixel noise dominates more. While



(a) Example image showing setup.

(b) Corner covariance matrices for the example image.

Figure 4: Evaluation setup showing the four used markers. The covariance matrices have been upscaled for legibility.

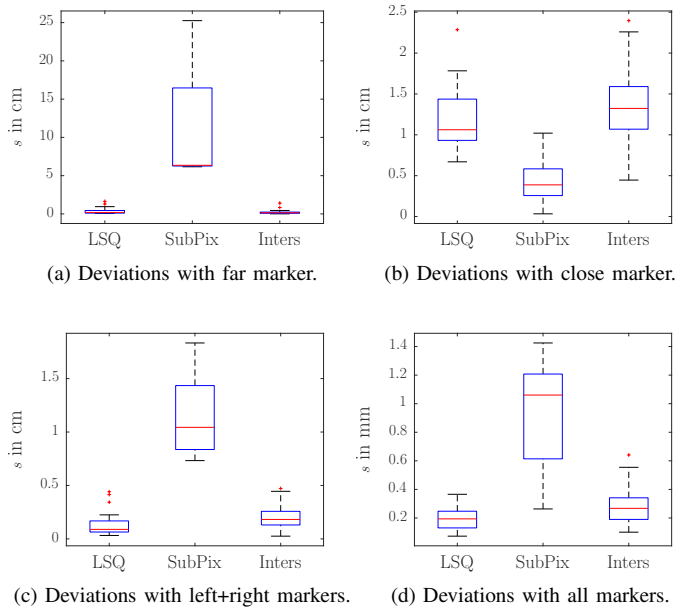
the variance remains low, the number of outliers is much higher, reaching up to 6 pixels. The spread of LSQ and Edge Intersection remain low, because they use information from multiple contour points instead. This shows that, in the general case, either LSQ or Edge Intersection should be preferred.

B. Pose Refinement

In this subsection, we evaluate the quality of the pose estimates that result from the corner refinement in the previous section. Fig. 4 shows the experiment setup, with four prominent Aruco markers (Fig. 4a) denoted as 'far', 'close', 'left' and 'right'. The far marker was about 3 meters from the camera, and the close marker about 1.5 meters. The chess board in the middle was used to calculate the starting pose for the estimates. The image resolution was 1280×720 pixels, to ensure that the far marker was always detectable. Once more, 60 images were used. For CornerSubPix and Edge Intersection, the OpenCV function *solvePnP* was used, while LSQ employed the approach introduced in Sec. IV with the estimated corners and their covariance matrices reinterpreted as measurements. For reference, Fig. 4b shows the resulting covariance matrices, illustrating how the marker next to the camera has the highest weight and the farthest the lowest weight.

Four setup variants were employed: i) only the far marker, ii) only the close marker, iii) the left and right markers together, and iv) all markers together. In order to quantify the results, we will employ the translation of the inverse pose shown in (8). The reasons are as follows. On the one hand, given that it is calculated from the rotations and translations of the markers, it illustrates the noise of both estimates. On the other hand, the inverse pose corresponds to the pose of the camera, which is in turn indicative of the results one would expect in AR applications. The values were processed a similar way as Fig. 3, i.e., by taking the Euclidean distances to the mean. The results are shown in Fig. 5.

For the first variant, by using only the far marker, the pose estimates were highly unreliable (Fig. 5a). Because of the low contrast, CornerSubPix had results with a high spread, spanning a range of up to 25 centimeters. LSQ and Edge Intersection,



(a) Deviations with far marker.

(b) Deviations with close marker.

(c) Deviations with left+right markers.

(d) Deviations with all markers.

Figure 5: Distance deviations from the mean for the different setups. The red line denotes the median, the blue box represents the range between 25th and 75th percentiles. Red dots are assumed outliers.

however, had a much lower variance. Still, the variance of LSQ was about half as large. The spread of CornerSubPix was reversed in Fig. 5b for the second variant, for a similar reason as Fig. 3a. Here, the marker was close to the camera, meaning that contrast was high and pixel noise was low. In turn, this means that the refined corner estimate did not change much between frames. The LSQ and Edge Intersection approaches did not produce bad results, however, with jitter of about 1 centimeter. For the third variant using the left and right markers, LSQ and Edge Intersection had once more the lower variances. Still, LSQ had a slightly lower spread. For the fourth variant using

all markers, the spread was much lower with all approaches, with all estimates falling within a range of 1.5 millimeters. Once more, the LSQ spread is the lowest by about 30%.

In the context of the evaluation, we will briefly discuss two relevant topics. On the one hand, for the sake of conciseness, we omitted an analysis of pose accuracy in this work. For the Edge Intersection proposed in [11], [12], [14], there are multiple analyses of this topic in literature such as [9], [33]. Given the closeness of the results between Edge Intersection and LSQ, it is very likely that those works also apply to our introduced approach. Nonetheless, it is important to evaluate the effects of the weighting mechanism in different circumstances, such as lightning and distance, and for this reason this analysis is part of our immediate future work. On the other hand, given our focus in mobile AR applications, it is necessary to find whether our minimization approach can be reliably used in low-resource devices. Based on experiments used on the Microsoft Hololens, the Apple iPhone SE, and the Samsung Galaxy Tab A (all considered dated in 2018), we noticed that our approach consistently remained below 1 millisecond, and still far lower than the marker detection itself, which varied between 5 and 20 milliseconds. All other approaches were also below 1 millisecond.

VI. Conclusions

For this paper, we introduced a new approach for corner and pose refinement for square fiducial markers, with focus on AR applications. The idea was to reduce the effect of pixel jitter by exploiting information from marker contours, which are part of standard marker detection techniques. First, we developed a NLSQ estimator for corner refinement that provided us with a tetragon, together with a covariance matrix that showed us how much weight each marker corner in the image should have. Then, we derived a second estimator for pose refinement that exploits this information. Evaluations showed that the proposed corner refinement approach produced similar results to the edge intersection techniques used in multiple off-the-shelf libraries. However, when incorporating the weights into the pose refinement step, results showed that poses calculated with our approach were more robust to the effect of pixel jitter. Future work will focus on comparing the pose accuracy of our approach compared with related state-of-the-art techniques.

References

[1] M. Fiala, "Designing highly reliable fiducial markers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1317–1324, jul 2010.

[2] M. Billinghurst, A. Clark, G. Lee *et al.*, "A survey of augmented reality," *Foundations and Trends® in Human–Computer Interaction*, vol. 8, no. 2-3, pp. 73–272, 2015.

[3] B. H. Thomas, "A survey of visual, mixed, and augmented reality gaming," *Comput. Entertain.*, vol. 10, no. 1, pp. 3:1–3:33, Dec. 2012. [Online]. Available: http://doi.acm.org/10.1145/2381876.2381879

[4] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for augmented reality: a hands-on survey," *IEEE transactions on visualization and computer graphics*, vol. 22, no. 12, pp. 2633–2651, 2016.

[5] Apple Inc., "ARKit," 2017. [Online]. Available: https://developer.apple.com/arkit/

[6] Google LLC, "ARCore," 2018. [Online]. Available: https://developers.google.com/ar/

[7] Microsoft Corp., "Hololens," 2016. [Online]. Available: https://www.microsoft.com/de-de/hololens

[8] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3400–3407.

[9] R. Munoz-Salinas, M. J. Marin-Jimenez, E. Yeguas-Bolivar, and R. Medina-Carnicer, "Mapping and localization from planar markers," *Pattern Recognition*, vol. 73, pp. 158–171, 2018.

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

[11] D. Wagner and D. Schmalstieg, *Artoolkitplus for pose tracking on mobile devices*. na, 2007.

[12] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. IEEE, 2005, pp. 590–596.

[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 Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

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

[15] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[16] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.

[17] D. H. Douglas and 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*, vol. 10, no. 2, pp. 112–122, 1973.

[18] P. Lancaster and K. Salkauskas, "Curve and surface fitting. an introduction," *London: Academic Press, 1986*, vol. 1, 1986.

[19] A. Fitzgibbon, M. Pilu, and R. B. Fisher, "Direct least square fitting of ellipses," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 5, pp. 476–480, 1999.

[20] W. Gander, G. H. Golub, and R. Strebel, "Least-squares fitting of circles and ellipses," *Bulletin of the Belgian Mathematical Society Simon Stevin*, vol. 3, no. 5, pp. 63–84, 1996.

[21] M. Werman and D. Keren, "A Bayesian method for fitting parametric and nonparametric models to noisy data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 5, pp. 528–534, 2001.

[22] G. Welch, G. Bishop *et al.*, "An introduction to the kalman filter," 1995.

[23] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, Mar. 2004.

[24] M. J. Powell, "On search directions for minimization algorithms," *Mathematical programming*, vol. 4, no. 1, pp. 193–201, 1973.

[25] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*. Springer, 1978, pp. 105–116.

[26] J. A. Grunert., "Das Pothenotische Problem in erweiterter Gestalt nebst ber seine Anwendungen in der Geodsie," *Archiv der Mathematik und Physik, Volume 1*, 1841.

[27] L. Quan and Z. Lan, "Linear n-point camera pose determination," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 8, pp. 774–780, 1999.

[28] B. Triggs, "Camera pose and calibration from 4 or 5 known 3d points," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1. IEEE, 1999, pp. 278–284.

[29] L. Zhi and J. Tang, "A complete linear 4-point algorithm for camera pose determination," *AMSS, Academia Sinica*, vol. 21, pp. 239–249, 2002.

[30] C. Albl, Z. Kukelova, and T. Pajdla, "Rolling shutter absolute pose problem with known vertical direction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3355–3363.

[31] G. Hee Lee, M. Pollefeys, and F. Fraundorfer, "Relative pose estimation for a multi-camera system with known vertical direction," in *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 540–547.

[32] G. Gallego and A. Yezzi, "A compact formula for the derivative of a 3-d rotation in exponential coordinates," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 3, pp. 378–384, 2015.

[33] V. Agnus, S. Nicolau, and L. Soler, "Illumination independent and accurate marker tracking using cross-ratio invariance," *IEEE computer graphics and applications*, vol. 35, no. 5, pp. 22–33, 2015.