STag: A Stable Fiducial Marker System

Burak Benligiray, Cihan Topal, Cuneyt Akinlar

PII: S0262-8856(19)30090-3
DOI: https://doi.org/10.1016/j.imavis.2019.06.007
Reference: IMAVIS 3779

To appear in: *Image and Vision Computing*

Received date: 24 July 2018
Revised date: 9 June 2019
Accepted date: 27 June 2019

Please cite this article as: B. Benligiray, C. Topal and C. Akinlar, STag: A Stable Fiducial Marker System, Image and Vision Computing, https://doi.org/10.1016/j.imavis.2019.06.007

STag: A Stable Fiducial Marker System

Burak Benligiray[1,*], Cihan Topal[a], Cuneyt Akinlar[b]

[a] *Department of Electrical and Electronics Engineering Eskisehir Technical University, Eskisehir, TURKEY*
[b] *Department of Computer Engineering Eskisehir Osmangazi University, Eskisehir, TURKEY*

Abstract

Fiducial markers provide better-defined features than the ones naturally available in the scene. For this reason, they are widely utilized in computer vision applications where reliable pose estimation is required. Factors such as imaging noise and subtle changes in illumination induce jitter on the estimated pose. Jitter impairs robustness in vision and robotics applications, and deteriorates the sense of presence and immersion in AR/VR applications. In this paper, we propose STag, a fiducial marker system that provides stable pose estimation. STag is designed to be robust against jitter factors, thus sustains pose stability better than the existing solutions. This is achieved by utilizing geometric features that can be localized more repeatably. The outer square border of the marker is used for detection and homography estimation. This is followed by a novel homography refinement step using the inner circular border. After refinement, the pose can be estimated stably and robustly across viewing conditions. These features are demonstrated with a comprehensive set of experiments, including comparisons with the state of the art fiducial marker systems.

Keywords: Fiducial markers, pose estimation stability, geometric calibration, mixed reality.

[☆] Source code, supplementary video: https://github.com/bbenligiray/stag.

[*] Corresponding author
Email address: bbenligiray@gmail.com (Burak Benligiray)

1. Introduction

Fiducial markers are artificial patterns that combine fast and accurate pose estimation with easy and inexpensive deployment [1]. These properties make them particularly useful for prototyping. They were first proposed to be used for augmented reality [2, 3, 4], and are still relevant in the recent virtual reality boom [5]. In addition, they see increasing use in systems that require to interact precisely and responsively with the real world, including robotics [6, 7, 8].

Stability is a key metric of pose estimation, and is especially critical for certain applications. For example, when using the estimated pose in a mixed reality application, instability causes jittery graphics, which is more immersion-breaking than a constant misalignment. Pose estimation instability is also highly undesirable in control applications, as it causes oscillatory behavior. Therefore, fiducial marker systems should aim to improve stability along with the usual performance metrics, such as detection robustness, speed and library size.

The marker pose is estimated using the geometric feature localizations. For better pose stability, these localizations have to be repeatable. Therefore, using geometric features that can be localized better will improve pose stability. In addition, a marker design must provide adequate projective constraints to estimate the pose. To satisfy both of these criteria, we propose a combination of geometric features to be used as the marker design.

In this paper, we propose STag, a fiducial marker system that provides stable localization without using any temporal filtering. This is achieved by the hybrid marker design seen in Figure 1. The outer square border is used for detection and homography estimation, and the inner circular border is used to refine the estimated homography. Since both of these features are simple, they do not degrade when seen from long distances and acute viewing angles, resulting in robust detection. The homography refinement step that provides stability is unique in that it uses a single conic correspondence to optimize the estimated homography. To support the use of a wide range of numbers of markers, we generated various marker libraries by adapting the lexicographic

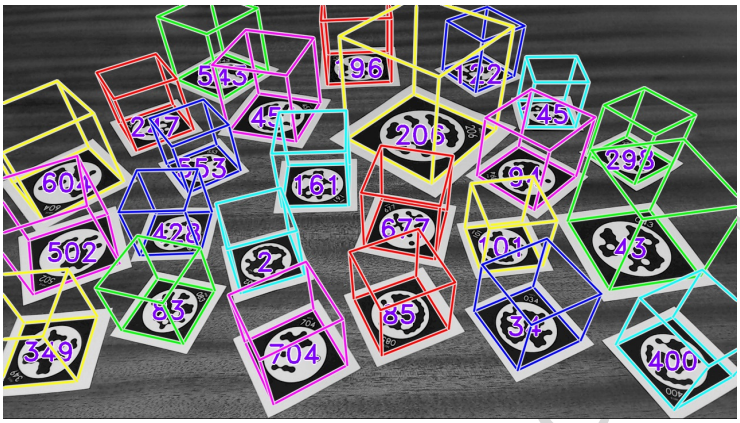


Figure 1: The poses estimated with the proposed marker are used to render 3D boxes.

generation algorithm.

2. Related Work

ARToolkit [9] markers are encoded with patterns chosen by the user, each of which is associated with an ID (see Figure 2a). These patterns are densely sampled and decoded using a nearest neighbor method. This approach has error detection and correction functionality, but its performance depends on the specific set of patterns in the library. Matrix-like coding option is added to ARToolkit with an extension [10].

ARTag [11] is the first marker system with forward error correction capability based on digital coding methods (see Figure 2b). This influenced the following studies to place a heavier emphasis on coding. Additionally, ARTag is the first marker system to detect markers from edges.

AprilTag [6] shares the square barcode type design (see Figure 2c), but the user chooses the size of the encoding grid. Other recent marker systems also provide marker variants with different bandwidths [12, 13]. AprilTag is the first marker system to use lexicodes, greedily generated codes with error detection and correction capabilities.

ArUco [13] has the unique feature of calculating an occlusion mask with

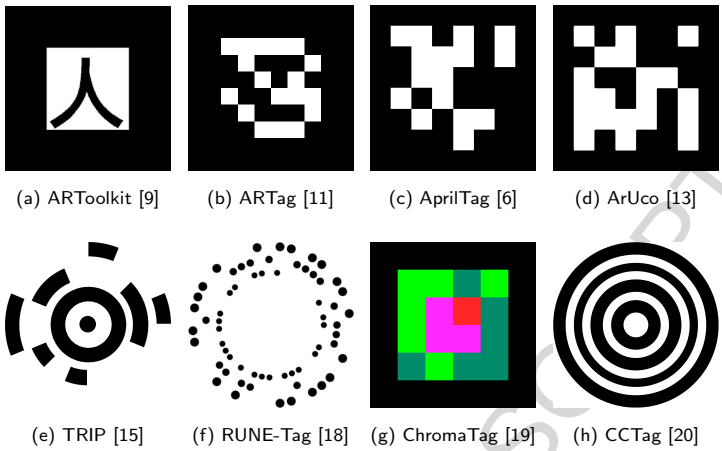


Figure 2: Marker designs of some of the notable fiducial marker systems in the literature.

tiled markers, which can prove useful in mixed reality applications (see Figure 2d). The original ArUco marker library is created using a stochastic lexicode generation algorithm. Better libraries are added using a mixed integer linear programming approach [14].

TRIP [15] uses a circular design (see Figure 2e). Markers have a ring in the middle for detection, and two outer rings for binary encoding. Outer rings are divided into aligned slices. The black slice pair indicates the starting point of the encoding. Relying on singular features to resolve rotation ambiguity for circular designs is not unique to TRIP [16, 17]. Having such regions is not preferable, as occluding them will prevent the marker from being detected.

RUNE-Tag [18, 12] does not have a border in the traditional sense. The coding dots are arranged in a circular shape, and act as candidate generating features (see Figure 2f). The pose is estimated by using the coding dots as point correspondences. This approach results in a finite pose ambiguity similar to square markers, which can be resolved by decoding. The finely detailed design provides many correspondence points, resulting in stable pose estimation. However, these fine details are prone to degradation under suboptimal viewing conditions.

The literature tends to use monochromatic designs, as using colors will require certain photometric assumptions, or a calibration step [1]. ChromaTag [19] modifies AprilTag to use colors for both candidate detection and decoding (see Figure 2g). The detection algorithm only runs on areas where both violet and green are present, which decreases the detection time significantly.

The systems we have mentioned until now are designed for pose estimation from a single marker, which is useful for applications such as augmented reality and robotics. Square designs are dominant among these markers, as their corners provide the four correspondences required to estimate the homography. Photogrammetry applications come with a slightly different requirement: Each marker needs to provide a single highly accurate correspondence. In these applications, circular markers, and especially markers composed of concentric circles excel.

CCTag [20] is an improvement over the concentric circles commonly used in photogrammetry (see Figure 2h). Its main contribution is robustness against motion blur, an important issue in tracking applications. Due to the reasons described above, CCTag does not estimate the pose of the marker, but only locates the projection of the marker center, similar to some earlier circular fiducial markers [21].

3. Stable Marker Design

To understand how stability can be improved, let us discuss what causes instability. To estimate the pose of a fiducial marker, we localize its geometric features, and use them as correspondences. Therefore, the uncertainty in the estimated pose is a result of the uncertainty in feature localizations. We are going to investigate if we can improve stability further by utilizing geometric features that can be localized more certainly.

3.1. Geometric Features

Various geometric features can be used to generate correspondences for pose estimation:

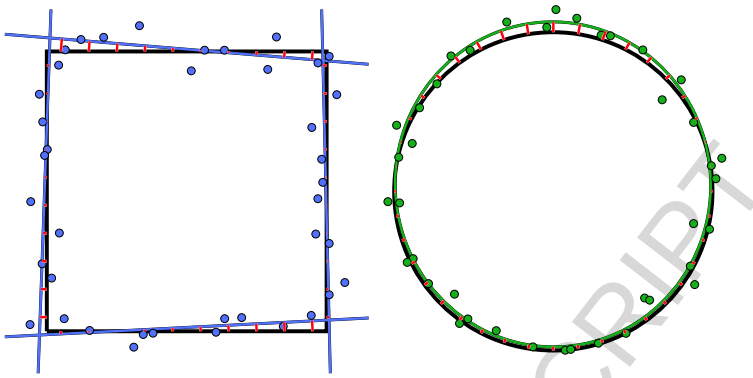
- Lines are used in square markers [11, 13]. This is by far the most common solution in the literature, and does not have any obvious disadvantages.
- Points are used as correspondences in dot patterns [22, 12]. These markers are composed of fine dots, which are difficult to detect robustly from acute viewing angles and long distances.
- Conics are used as correspondences in circular markers [15, 20]. A single conic correspondence is not adequate to estimate the pose, which is why these markers either use additional geometric features [15], or do not allow pose estimation with a single marker [21, 20].

We are going to argue that the traditional square border has a critical drawback compared to the circular border: It cannot be localized as stably. Therefore, circular borders should be utilized for utmost pose stability. The problem of a single conic correspondence under-defining the pose still stands, to which we are going to propose a novel solution to in Section 4.3.

The square border of the fiducial marker appears as a quadrilateral (shortened as quad) on the image. To localize the quad, we sample points from each vertex of the quad (e.g., by edge detection), and fit lines to individual point groups. Either these lines, or their intersections can be used to estimate the marker pose. Each line represents an exclusive quarter of the information that is used to estimate the pose, which means that the localization error of a line cannot be compensated for by the others.

The circular border of the fiducial marker appears as an ellipse on the image. Differently from the quad, the ellipse cannot be represented as a combination of independent parts. For this reason, a single fitting operation is applied to all samples. This can be seen as the pose information being distributed to the entire shape, which allows it to be recovered more robustly.

Let us consider a square and a circle, both centered on the origin with an area of 1. We uniformly sample 40 points along these shapes, and perturb them with Gaussian noise. The quad is localized by fitting a line to each 10 samples with least squares. The ellipse is localized by fitting an ellipse to all samples



(a) Quad localization (b) Ellipse localization

Figure 3: A square and a circle are uniformly sampled at 40 points, which are perturbed with a Gaussian noise with 0.04 standard deviation along each axis. A quad is fitted to the samples from the square, and an ellipse is fitted to the samples from the circle. Localization errors are shown with red lines.

with a least squares method [23]. To measure the mean localization error, the original shape is uniformly sampled at 40 points, and the average distance of these samples to the fitted shape is calculated (see Figure 3).

The described experiment is run 10,000 times with standard deviation values between 0 and 0.04. See Figure 4 for the results. As expected, the ellipse is localized better than the quad for all standard deviation values. See Section 5.4 for an experiment with real images.

3.2. Proposed Marker Design

In Section 3.1, we argued that a circular border can be localized more stably. By this motivation, the proposed stable marker design contains an inner circular border. Since a single conic correspondence is not adequate to estimate the pose, additional geometric features need to be used. An outer square border supplies the additional constraints, as seen in Figure 5.

The outer square border of the proposed marker is utilized to detect the marker and estimate a rough homography. After detecting the marker and validating it with its encoding, the circular border is localized and used to

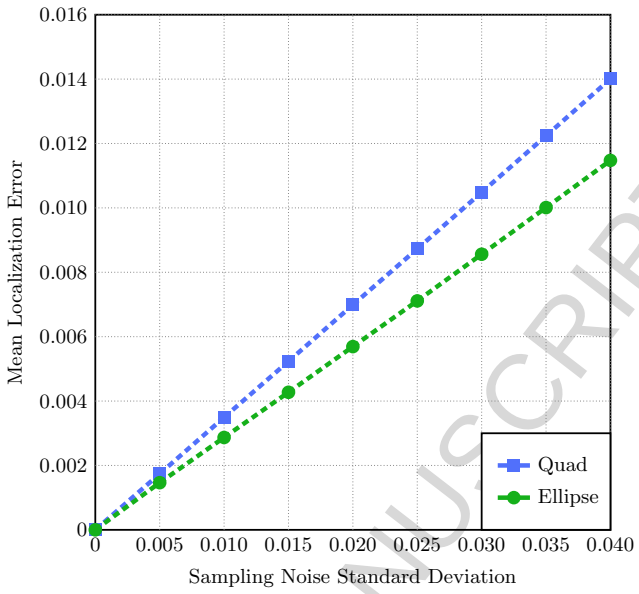


Figure 4: Mean localization errors with varying sampling noise standard deviation. Ellipse is localized better across all parameters

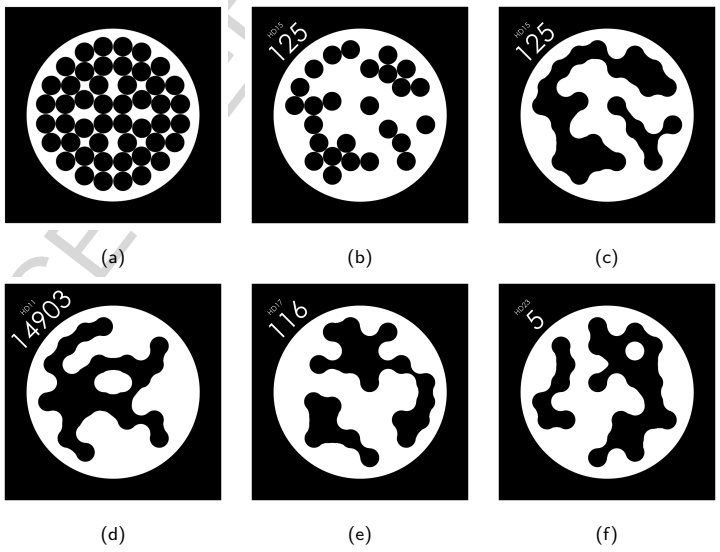


Figure 5: (a) shows the tiling of bit representations in the marker. (b) and (c) are markers before and after morphological operations, respectively. (d)-(f) are example markers from libraries with different minimum Hamming distances.

Table 1: STag library sizes with respective minimum Hamming distances.

Minimum HD	11	13	15	17	19	21	23
Library Size	22,309	2,884	766	157	38	12	6

refine the previously estimated homography. The encoding area inside the inner circular border is filled with disk-shaped bit representations, similar to [4]. 48 disks are packed efficiently inside the circular area using a simulated annealing method [24] (see Figure 5a).

After encoding the marker as in Figure 5b, the code pattern is morphologically dilated and eroded repeatedly, resulting in the encoding pattern in Figure 5c. Filling the gaps between neighboring bit representations allows the code to be read correctly in the case of slight localization errors. It also reduces high frequency elements in the pattern, resulting in less edge detections. Finally, it improves robustness against blooming and reflection effects, especially when the printing material is glossy. See Figure 5d-f for additional example markers from libraries with different minimum Hamming distances.

3.3. Encoding

We generated lexicographic marker libraries, as first done in [6]. The lexicode generation algorithm tests potential codewords and chooses the ones that satisfy an arbitrary constraint. For the traditional version, this constraint is that if a codeword is to be selected, it should be at least a predetermined Hamming distance away from the previously chosen codewords [25]. This can be adapted to marker lexicode generation by also testing for the circular permutations of the codewords.

The described lexicode generation algorithm is an exhaustive search of an n-dimensional binary space, n being the length of the codewords. This corresponds to a time complexity of $O(2^n)$. With 48-bit long codewords, the algorithm requires an unreasonable amount of time to finish, even with a GPU implementation. To overcome this problem, we took a two-step hierarchical approach. We first generated 12-bit long codewords, then generated the full-length

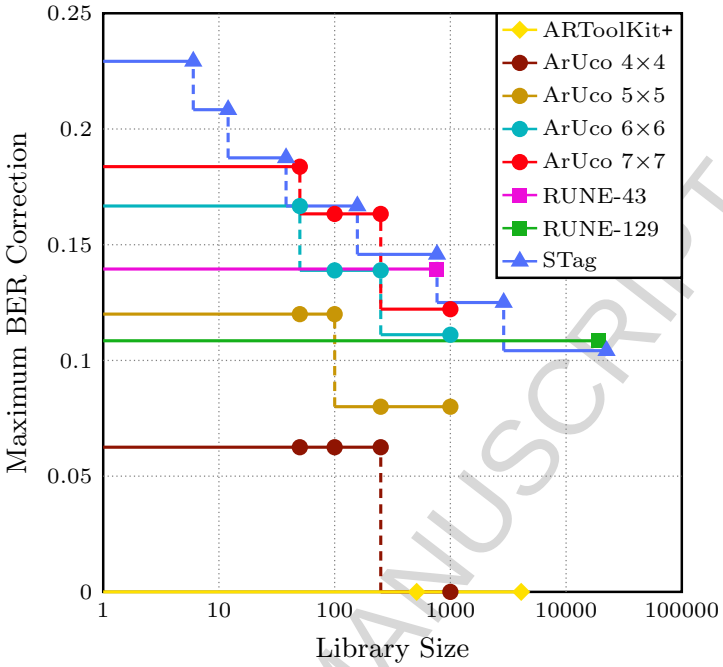


Figure 6: Marker libraries with different sizes and maximum bit error ratio (BER) correction capabilities.

codewords as 4-combinations of these.

We generated libraries with different sizes and Hamming distances to accommodate for a variety of applications. Smaller libraries with higher minimum Hamming distances can be used for small-scale AR applications where occlusion is common. Larger libraries are more suitable for navigation through large interior spaces or inventory applications. The resulting library sizes are presented in Table 1.

The main performance metric for a coding scheme is the amount of error correction it can provide for various library sizes. Bit error ratio (BER) is the number of error bits divided by the number of total bits. A marker library that can correct 0.1 BER will work when at most 10% of the coding area is read incorrectly. By using this metric, we can compare libraries with different code lengths.

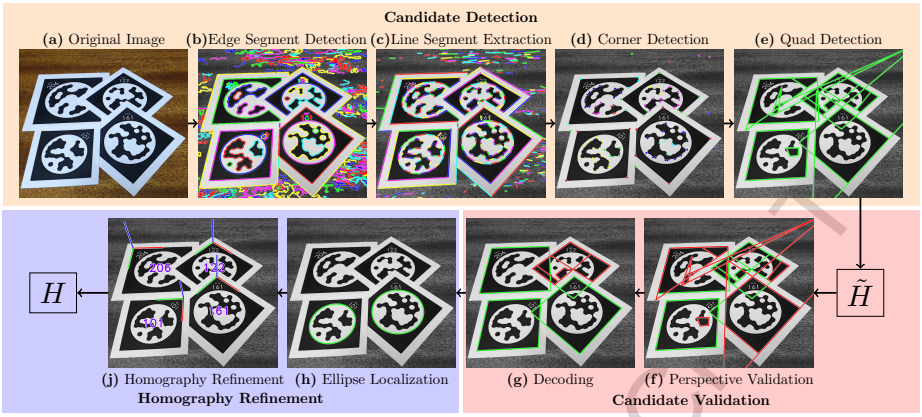


Figure 7: Flowchart of the detection algorithm for the proposed marker system (best viewed in color).

See Figure 6 for a comparison of ARToolKit+ [10], ArUco [14], RUNE-Tag [18], and STag libraries. Marker variants with different code lengths are plotted separately. The horizontal axis is in logarithmic scale, because improving error correction capability reduces library size exponentially.

ARToolKit+ implementation provides no error correction capability. ArUco variants grow more efficient with longer codewords. Both RUNE-Tag libraries are efficient, but they do not provide the best performance across the whole scale. ArUco 7×7, RUNE-129 and STag libraries lie on the same line, which implies a theoretical upper bound. STag libraries cover a large range of library sizes, and provide the best or near-best performance.

4. Detection Algorithm

The detection algorithm is composed of three main parts, as shown in Figure 7. In the candidate detection part, which is discussed in Section 4.1, we detect all quads in the image. These candidates are validated firstly by their shape, and secondly by their encoding, which is discussed in Section 4.2. For the candidates that are validated to be markers, the computationally intensive homography refinement step is run, as described in Section 4.3. After homography

refinement, the pose (i.e., $[R|\mathbf{t}]$) can be estimated using [26].

4.1. Candidate Detection

The first step of the candidate detection algorithm is to detect edge segments as contiguous arrays of pixels, which significantly eases further processing. For this task, we employ the EDPF algorithm [27], which determines anchor points in the image, and joins them by aiming to maximize the total gradient response along its path. Then, it validates these edge chains according to the Helmholtz principle to obtain native and well-localized edge segments. The edge segments detected in a sample image are shown in Figure 7b.

As the next step, we process the edge segments to find the linear borders of the markers. This is done by fitting a line to the beginning pixels of an edge segment, and extending this line as long as the following edge pixels are on it. This operation is repeated until all edge segments are processed. This approach avoids repeated line fitting operations, which become expensive with a large number of iterations [28]. In Figure 7c, the extracted line segments are shown.

Corners are detected by intersecting the line segments extracted consecutively from the same edge segment. Detected corners for the sample image is shown in Figure 7d. Following this, three consecutive corners are used to detect the quads on the image. This allows the detection algorithm to detect quads even when a corner is occluded. See all quads detected on the sample image in Figure 7e.

As seen in Figure 6, STag libraries provide the highest BER correction capability. This means that the markers can be decoded robustly against occlusion. Together with the detection algorithm described in this section, we can detect markers even under significant occlusion. See Figure 8 and the supplementary video for examples.

4.2. Candidate Validation

We propose a novel candidate validation algorithm that utilizes the candidate shape. Specifically, we eliminate candidates based on the imposed perspective distortion. Here, *perspective distortion* is in reference to the aspect of

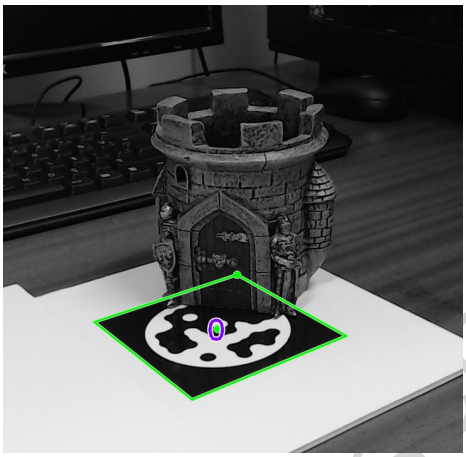


Figure 8: The marker is detected successfully under heavy occlusion.

a projective transformation excluding the part that can be represented as an affine transformation. The objective is to eliminate as much false candidates as possible, which will decrease the false positive rate.

Depth (α) is the distance from a point to the camera. The relative depth (α_{rel}) of an object is the ratio of the largest depth (α_{max}) to the smallest depth (α_{min}) of the object. The effect of perspective distortion becomes increasingly apparent as α_{rel} of the object increases. If we can find α_{rel} of the candidates, we can eliminate them accordingly, assuming α_{rel} will be bounded by some constraints.

Both opposite edges of a square are parallel, and lines extending along these parallel edges intersect at two distinct points at infinity. The line at infinity ($\mathbf{l}_\infty$) passes through these two points at infinity. When a projective transformation is applied, each of these points at infinity are projected on a respective finite vanishing point. The line passing through this vanishing point pair is the image of $\mathbf{l}_\infty$ (see Figure 9). The projection of $\mathbf{l}_\infty$ can also be found by using the respective homography matrix [29]. The points closest to and farthest to $\mathbf{l}'_\infty$ will be two of the four corners of the quad. The distance between $\mathbf{l}'_\infty$ and points on the object (d_i) has a linearly negative relationship with the depth of the said point (α_i) [30]. Using this relationship, we can find the relative depth of

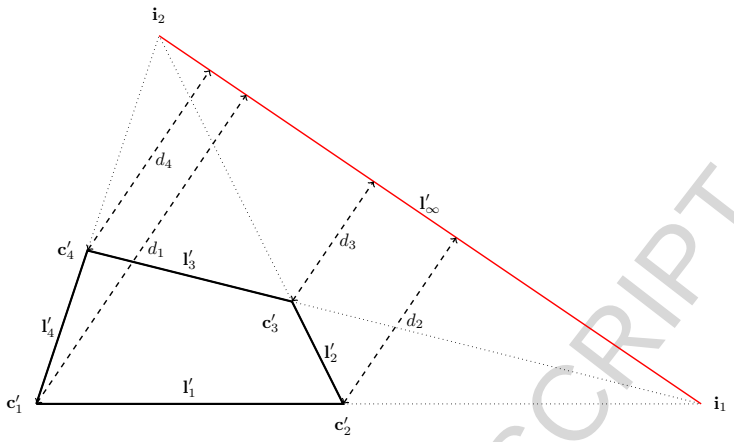


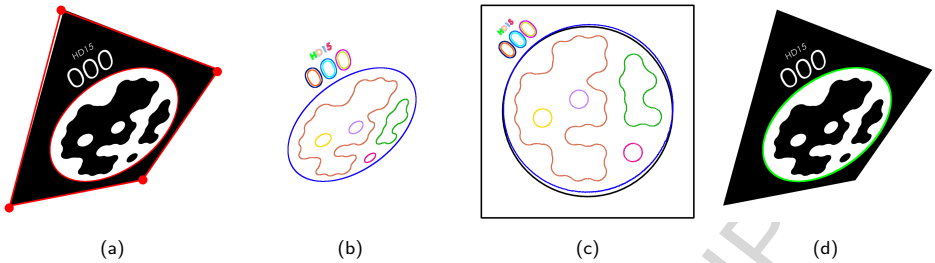
Figure 9: A marker candidate with corners $\mathbf{c}'_i$ and vertices $\mathbf{l}'_i$. The projective transformation have projected the line at infinity to $\mathbf{l}'_\infty$, which can be found by using the intersections $\mathbf{i}_1$ and $\mathbf{i}_2$. Distances from $\mathbf{c}'_i$ to $\mathbf{l}'_\infty$ are shown with d_i.

a planar object using Equation 1 (k is a scalar, refer to Figure 9 for further notation).

$$\alpha_{rel} = \frac{\alpha_{max}}{\alpha_{min}} = \frac{kd_{min}^{-1}}{kd_{max}^{-1}} = \frac{d_{max}}{d_{min}} \tag{1}$$

α_{rel} of the projected quad indicates the amount of perspective distortion it underwent. A threshold is needed to eliminate candidates based on this metric. Assume that 10 cm×10 cm markers will be used in an application and none of them will approach the camera more than 20 cm (α_{min}). We can calculate α_{max} to be $20 + 10\sqrt{2}$ cm, which results in the maximum α_{rel} to be 1.707. Therefore, we can safely eliminate all quads that have a larger α_{rel} than this threshold. If the eliminated quads indicated with red in Figure 7f are examined, it is seen that only the ones that can not be a valid projection of a square shape are eliminated. We are going to show the benefit of this validation step experimentally in Section 5.2.

We estimate the homography matrix, H, of a candidate by using its corners as correspondence points [29]. There are four ways of matching the corresponding corners, of which an arbitrary one is used. The projections of bit representations are sampled using this homography to obtain the embedded codeword.



(a) (b) (c) (d)

Figure 10: (a) A marker with incorrectly localized corners and the respective expected ellipse location. (b) Edge segment loops inside the marker detection. (c) Edge segment loops backprojected to the marker plane. (d) The ellipse fitted to the chosen edge segment loop.

We keep all codewords in the library and their circular rotations in a list. The read codeword is compared with each element in the list by XORing and doing a population count. If the Hamming distance between the read codeword and a codeword in the list is smaller than or equal to the maximum number of bits to be corrected, the respective rotation and ID is associated with the candidate. If the arbitrarily chosen rotation is found to be incorrect, the homography is updated with the correct correspondence. If the codeword can not be found in the ID list, then that marker is also eliminated after decoding (see the red quads in Figure 7g).

4.3. Homography Refinement

At this point, we have a set of markers that are validated by decoding. We are going to localize the inner circular border as an ellipse, and utilize the conic information to refine the estimated homography. Let us illustrate the ellipse localization process. In Figure 10a, we present an example with exaggerated localization error for better viewing. We first find the edge segment loops inside the marker detection (see Figure 10b) and backproject them to the marker plane (see Figure 10c). It is expected that among the edge segment loops, the one from the border ellipse will be the most similar to the inner circular border. We estimate this similarity by sparsely sampling the distances between the inner circular border and the back projected edge segment loops. An ellipse is fitted

to the edge segment loop whose back projection is the most similar to the inner circular border [23] (see Figure 10d).

We detect the absence of a suitable edge segment loop by thresholding the similarity between the backprojected edge segment loops and the inner circular border. Since this metric is calculated in the marker plane, the shape and size of the detection is normalized. Thus, the threshold acts equivalently in all poses. If a suitable ellipse could not be detected, homography refinement is omitted (e.g., markers #122 and #206 in Figure 7j). This is a deliberate design choice, because we have observed that homography refinement with a partially occluded ellipse does not benefit stability, and in some cases even degrades it.

In the following discussion, we are going to be considering circles and ellipses as conic sections, which are represented as matrices. Capital letters refer to matrices, bold-face letters refer to vectors, and plain letters refer to scalars. Where $\mathbf{x}$ is a point in homogeneous coordinates and $\mathbf{x}'$ is its projection, the 2D transformation is represented as such [29]:

$$\mathbf{x}' = H\mathbf{x} \tag{2}$$

Under the same transformation as Equation 2, the projection of a conic section is commonly represented as the following two equivalent equations:

$$C' = H^{-T}CH^{-1} \tag{3}$$

$$C = H^{T}C'H \tag{4}$$

Let C be the inner circular border, whose projection is localized as an ellipse C'. When we back project C' to the marker plane with Equation 4, we get an ellipse $\tilde{C}$, which does not coincide with C perfectly. This is because the initial homography estimated using the marker corners is slightly incorrect. Our approach is going to be to optimize H to make C and $\tilde{C}$ as similar as possible.

To optimize H, we need a similarity metric between the ellipse $\tilde{C}$ and the circle C. Where a is the semi-major axis, b is the semi-minor axis , (e_x, e_y) is

Table 2: Running times of detection algorithm steps in milliseconds.

EDPF [27]	EDLines [28]	Candidate Detection	Candidate Validation	Ellipse Localization	Homography Refinement	**Total**
22.6	1.0	0.7	0.6	1.1	0.8	**26.8**

the center of $\tilde{C}$, and r is the radius, (c_x, c_y) is the center of C, respectively, the metric we propose is:

$$\epsilon = \sqrt{(e_x - c_x)^2 + (e_y - c_y)^2 + (a - r)^2 + (b - r)^2} \tag{5}$$

These distances are in the marker plane, thus they are normalized for pose.

A single circle–ellipse correspondence is not adequate to estimate the homography of a marker. We use the homography estimated using the corners as a starting point, and minimize ϵ in Equation 5 using the Nelder–Mead method [31]. This way, the previously estimated homography is refined such that the ellipse detection is back projected directly on the inner circular border. Since the ellipse is localized more correctly than the marker corners, this improves the stability of the localization. See Section 5.4 for an experiment that demonstrates the benefits of homography refinement.

4.4. Running Time and Parallelization

Since marker detection algorithms have to run in real time, it is important for them to both run fast and be parallelizable in case a further speed up is needed. In this section, we are going to analyze the proposed detection algorithm in this regard. See Table 2 for the steps of the detection algorithm for a 1280×720 image with a cluttered scene containing a single marker.

The first thing that meets the eye is that EDPF constitutes more than 80% of the running time. In fact, the total running time is higher than Table 7 simply because of the larger number of edge segments extracted from the background clutter. As the name implies, the Edge Drawing algorithm is partly sequential.

However, a CUDA implementation of this algorithm has resulted in up to ×12 speedup even with older model GPUs [32]. This would correspond to a ×4.4 speedup in the STag detection algorithm by itself.

The rest of the detection algorithm is considerably more parallelizable than the edge segment detection step. EDLines and candidate detection process edge segments independently. Considering that there are hundreds of edge segments in a typical image, processing of these can be parallelized with arbitrary granularity. Following this, candidate validation operates on candidates independently, and around 3.7 candidates can be expected on a typical scene (derived from the results in Table 3). Finally, ellipse localization and homography refinement only operate on successfully decoded markers. In the case that there are multiple markers on the scene, these steps are also easily parallelizable. Alternatively, these two final steps can be omitted if there are many markers on the scene, as a large number of correspondences from multiple markers would already provide a stable pose.

5. Experiments

We compared the proposed marker system with ARToolKit+ [10], ArUco [13] and RUNE-Tag [18] in a series of experiments. These marker systems are comparable with ours, in that they both provide large libraries and allow pose estimation with a single marker. The literature uses warped synthetic images for pose experiments, which is not an adequate approximation. All our experiments are done on real images.

5.1. Detection

We used an indoor scene recognition dataset with 15,620 images from 67 categories to represent markerless scenes [33]. See Table 3 for library characteristics and respective numbers of candidates and false positives. We used the error correction capability of each library to its fullest extent. While both ArUco and STag are trying to detect square markers, STag detects significantly less false candidates. RUNE-Tag detects no candidates in the entire dataset.

Table 3: Marker library characteristics and detection performance at the Indoor Scene Recognition Dataset [33].

Marker System	Marker Library	Bits	Error Correction	Library Size	Number of Candidates	Number of False Positives	Validation Failure Probability[1]
ARToolKit+ [10]	simple-id	36	0	512	11739	0	N/A
	BCH-id	36	0	4096	11739	3	6.2E-8
ArUco [13, 14]	ARUCO_ORIGINAL	16	0	1024	317741	785	2.4E-06
	4X4_50	16	1	50	317741	2349	7.4E-05
	4X4_100	16	1	100	317741	3250	5.1E-05
	4X4_250	16	1	250	317741	13018	8.2E-05
	4X4_1000	16	0	1000	317741	1875	5.9E-06
	5X5_50	25	3	50	317741	269	2.1E-06
	5X5_100	25	3	100	317741	934	3.7E-06
	5X5_250	25	2	250	317741	294	9.3E-07
	5X5_1000	25	2	1000	317741	1821	1.4E-06
	6X6_50	36	6	50	317741	979	9.6E-07
	6X6_100	36	5	100	317741	177	1.7E-07
	6X6_250	36	5	250	317741	939	3.7E-07
	6X6_1000	36	4	1000	317741	212	4.2E-08
	7X7_50	49	9	50	317741	27	3.3E-09
	7X7_100	49	8	100	317741	2	2.5E-10
	7X7_250	49	8	250	317741	112	5.53E-09
	7X7_1000	49	6	1000	317741	4	2.0E-10
RUNE-Tag [12]	Rune-43[2]	43	6	762	N/A	N/A	N/A
	Rune-129	129	14	17000	0	0	N/A
STag	HD11	48	5	22309	57893	7	1.7E-10
	HD13	48	6	2884	57893	6	5.6E-10
	HD15	48	7	766	57893	23	4.1E-09
	HD17	48	8	157	57893	11	4.7E-09
	HD19	48	9	38	57893	5	4.4E-09
	HD21	48	11	12	57893	2	2.8E-09
	HD23	48	13	6	57893	38	5.3E-08

[1] This metric shows the probability of a false positive for a marker library size of 1 with no error correction. Its objective is to assess false positive rates without penalizing larger marker libraries and higher error correction capabilities. Lower is better.

[2] Rune-43 is not provided by the authors

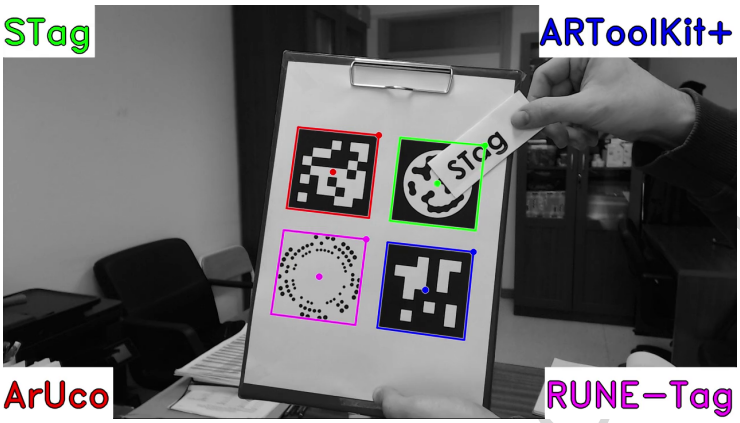


Figure 11: A frame from the supplementary video that demonstrates detection performance. See `https://github.com/bbenligiray/stag` for the video.

Looking at the number of false positives in Table 3, we see that STag returns few false positives even with extreme error correction. Compared to STag, ArUco has returned more false positives. To eliminate the effect of the number of candidates, amount of error correction and library size to the discussion, we developed a metric named Validation Failure Probability:

$$\text{Validation Failure Probability} = \frac{\text{No. False Positives}}{\text{No. Candidates} \times \text{Library Size} \times 2^{\frac{\text{Error Correction}}{\text{Bits}}}} \tag{6}$$

By this metric, we see that the reduced number of false positives from STag is not only due to the lower number of candidates, but also a better performing marker library generation scheme. Moreover, the validation performance is consistent for different library sizes and error correction rates.

For a qualitative experiment about detection with markers on the scene, see the supplementary video (see Fig. 11). A marker of ARToolKit+, ArUco, RUNE-Tag and STag are printed on a sheet. They are moved around, rotated and occluded to compare the detection algorithms of the respective systems. RUNE-Tag cannot be detected when the markers are further away. The temporal tracking of ARToolKit+ results in mislocalization when the marker is moving rapidly. ArUco and ARToolKit+ are not very robust against occlusion. STag is

Table 4: The number of false positive detections with maximum error correction in the indoor scene recognition dataset [33] with and without the candidate validation method described in Section 4.2.

Minimum HD	11	13	15	17	19	21	23
w/o validation	15	16	34	13	6	6	98
w/ validation	7	6	23	11	5	2	38

detected consistently throughout the video, is robust against occlusion, and is localized stably due to homography refinement. Note that the processing was done offline, as RUNE-Tag cannot be run with this frame rate in real time.

5.2. Perspective Validation

Let us demonstrate the benefits of the proposed shape-based candidate validation method described in Section 4.2 using the same indoor scene recognition dataset [33]. A total of 94,427 quads were detected in the entire dataset, and this number was reduced to 57,893 after being validated based on perspective. To clearly see the effect of this reduction, we ran the decoding algorithm with maximum error correction capability (half of the minimum Hamming distance), which typically returns a large number of false positives. See Table 4 for the results. Indeed, a significant portion of the eliminated candidates were going to be detected as false positives.

5.3. Setup for Stability Experiments

For stability experiments, we captured images of a stationary marker using a stationary camera. Ideally, the localization and pose differences between these images should be zero. Although the marker detection algorithms are deterministic, imaging noise causes localization differences between the images, which result in pose differences. This reflects to the end user as instability and jitter in pose. We calculated the standard deviation of these differences to compare the stability of different marker systems.

See Figure 12 for the experimental setup of the stability experiments. A 15 cm-wide marker printed on paper is tightly affixed to a jig built for easy

Figure 12: A jig is built to manipulate the viewing angle without moving the camera. The marker printed on paper is affixed tightly to the jig by taping its corners. The webcam is mounted on a tripod at marker height, aimed straight at the center of the marker. The viewing angle and distance is estimated using additional markers.

pose manipulation. We chose to print the markers on paper rather than rigid material such as polystyrene tiles, because this better represents the typical usage scenario. A Logitech C920 webcam is mounted on a tripod at marker level and aimed directly at the center of the marker. The viewing angle and distance are adjusted by keeping the camera stationary and manually manipulating the jig, taking the pose estimated from additional markers as reference. The pose is set up to 1° and 1 cm accuracy. For each marker system, distance and viewing angle, 1000 frames were captured (i.e., each point in the following figures is the standard deviation over 1000 frames).

The camera's focus and exposure were set manually, and kept the same for all experiments. Anti-flicker was on and lens distortion was rectified beforehand. The captured images were grayscale and of 1280×720 resolution. The lighting was completely artificial, coming from overhead sources. Since the camera did not move, its relative position with respect to the lighting sources did not change. In addition, we made sure that there was no significant air flow in the environment during the experiments that could move the markers.

5.4. Benefits of Homography Refinement

In this section, we are going to demonstrate that the circular border is localized more stably than the square border in real images, and the proposed

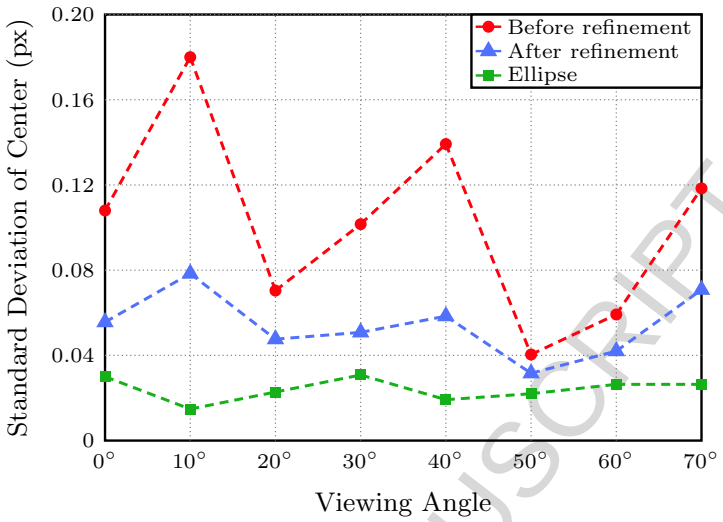


Figure 13: Localization stability before refinement and after refinement. Before refinement, marker center stability is very low. Refining with the well-localized ellipse results in a significant improvement in stability.

homography refinement step improves localization stability. For this specific experiment, a 10 cm-wide marker is positioned 100 cm away from the camera. Images with a resolution of 1920×1080 are captured from a series of viewing angles. The stability of the detection is represented by the standard deviation of the center localization. The marker centers before and after refinement are localized by projecting the marker center on the image with the respective homography matrix. See Figure 13 for the results. Before refinement, the marker center is unstable. Moreover, the localization is not robust against the changes in viewing angles. The center of the ellipse we have located is significantly more stable. After homography refinement, the marker center gains stability across all viewing angles.

5.5. Pose Estimation Stability

ArUco [13] and RUNE-Tag [18] uses OpenCV's Perspective-n-Point implementation to estimate the pose from a single marker. For coplanar correspondence points, this implementation estimates the homography matrix, decom-

poses it as described in [34] and refines it to minimize the projection error if there are redundantly many correspondence points. The method described in [34] requires the homography matrix to be estimated using at least two images where the orientation of the planar object is different. With a single image of a planar object, the pose estimation has four different solutions of which two satisfy the cheirality constraints [35]. Therefore, there is an ambiguity between two solutions. Especially when the marker is seen from far away or an acute viewing angle, the estimated pose switches between the two possible solutions randomly, causing a significant amount of pose jitter.

ARToolkit+ uses the method proposed in [26] to find both possible solutions and select the one that returns the smaller reprojection error in the object-space. We should note that in practice, this method does not always converge to the correct solution when only 4 marker corners are used, as we have observed jittering due to pose ambiguity while using it. Nevertheless, it is obviously superior over traditional homography decomposition for difficult viewing conditions, which is why we have also used it to estimate the pose of a single marker for STag.

The estimated pose has 6 degrees of freedom, thus it is difficult to quantify the amount of jitter in the pose with a single metric. For this reason, we investigated the jitter with different metrics. Namely, we checked jitter in rotation, translation, and center localization. To quantify the jitter in rotation, we found an average rotation across the frames. Axis–angle representation of rotation is composed of a direction and a magnitude. By using the magnitude part, we can represent the difference from the mean rotation, which was used to calculate the deviation in degrees. Quantifying the jitter in translation is rather straightforward. A mean translation vector is found for all frames, then the L2 distance from this mean is used to calculate the standard deviation in metric units. Finally, the center of the marker is localized by using the camera matrix K and the pose matrix $[R|\mathbf{t}]$ to project the marker center on the image. The standard deviation is found using the L2 distance on the image in pixel units. Note that the pose ambiguity between the two possible solutions only causes

Table 5: Number of false negative detections out of 1000 frames.

Viewing Angle	0°	5°	10°	15°	20°	25°	30°	35°	40°	45°	50°	55°	60°	65°	70°	75°	80°	85°
ARToolKit+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000	1000	1000	1000
ArUco	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	416	1000
RUNE-Tag	0	0	0	0	420	0	11	970	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
STag	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	1000

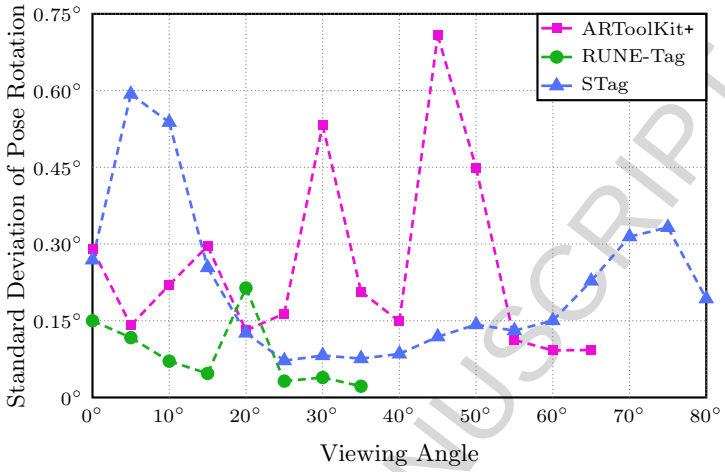
jitter in rotation, and should not affect translation or marker center localization.

5.5.1. Different Viewing Angles

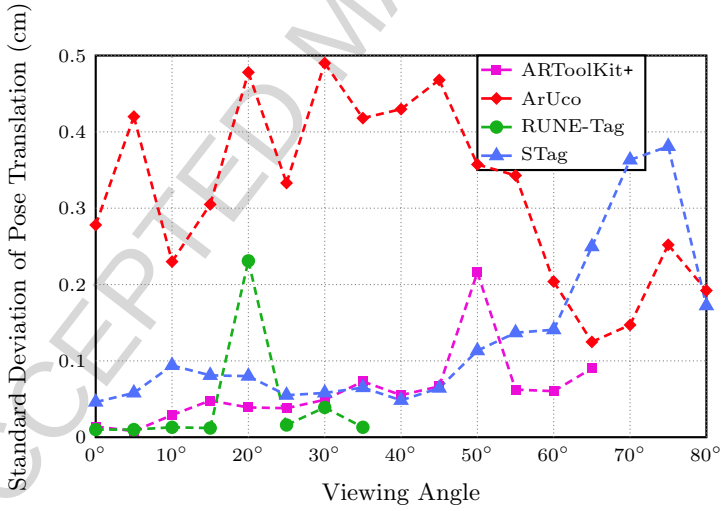
For this experiment, we captured images of the markers from different viewing angles using the setup described in Section 5.3. Let us start by the detection accuracies (see Table 5). We highlighted the cases where more than half of the frames returned false negative detections in red, and we will not report stability metrics for these cases. For the rest, we did not penalize the false negative detections in our stability calculations.

Both ArUco and STag are highly robust against difficult viewing angle conditions, yet STag performs significantly better from 80°. ARToolKit+ cannot detect markers seen from an angle of 70° or more, yet is rather robust for lower viewing angles. RUNE-Tag quickly stops being detectable as viewing angle increases, and is not very robust even in smaller viewing angles. Seeing that RUNE-Tag is not prone to false positive detection (see Table 3), but fails to detect markers under suboptimal conditions, we can conclude that RUNE-Tag has high precision, but poor recall.

See Figure 14 for the rotation and translation stability across viewing angles. The proposed marker system is both stable and detectable across viewing angles. However, its stability suffers under more acute viewing angles both due to localization difficulties and pose ambiguity. While RUNE-Tag achieves high stability due to many correspondence points, these fine dots also degrade easily, resulting in unreliable detection. ARToolkit+ localizes the marker corners coarsely, which causes its stability to depend on where the corner localizations fall on the quantization grid. ArUco has considerable jitter both due to unstable localization (seen on Figure 14b) and high pose ambiguity. In fact, standard



(a) Rotation



(b) Translation

Figure 14: Pose rotation and translation stability with different viewing angles. Standard deviation of pose rotation for ArUco was 11.70° on average, which could not be shown in this scale.

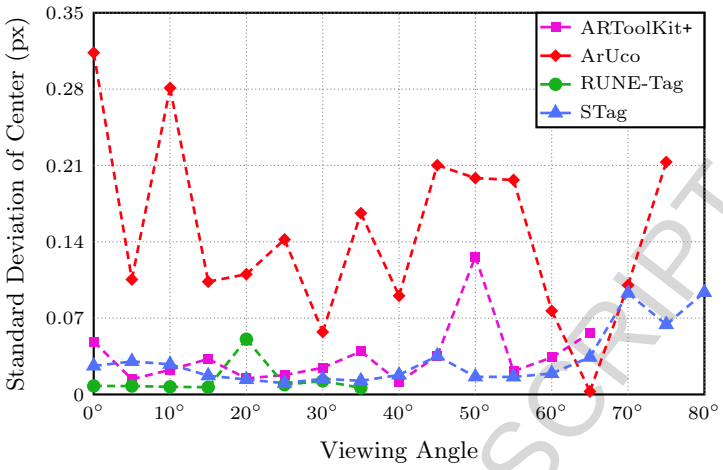
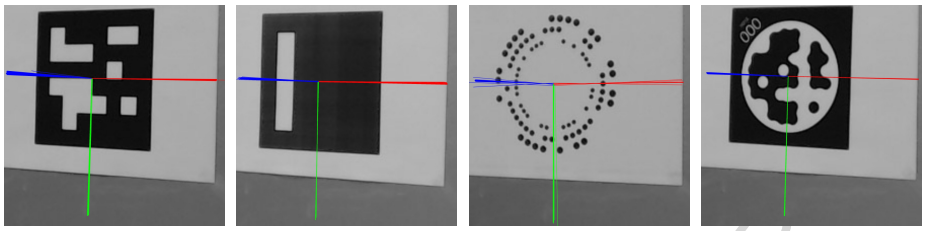


Figure 15: Localization stability with different viewing angles.

deviation of pose rotation for ArUco was very high (11.70° on average), which is why we were not able to illustrate it in Figure 14a. This is to be expected, as the rotation difference between the two ambiguous poses tends to be a lot more than the jitter caused by localization instability.

See Figure 15 for the localization jitter given in pixel units. We can see that these results are somewhat similar to the translation jitter results, as both are not affected by the pose ambiguity. Again, the proposed system can be detected robustly, and its center can be localized stably.

Finally, let us illustrate the difference in stability qualitatively. See Figure 16 for 500 pose axes drawn on markers seen from 30°. Note that we have eliminated incorrect poses due to pose ambiguity, so this figure only illustrates the jitter caused by localization instability. When the estimated pose is stable, the axes will overlap, resulting in a single solid axis. In contrast, an unstable pose will result in scattered axes. The axes from ARToolKit+ and ArUco seem to be scattered fairly uniformly. The axes from RUNE-Tag seem quite solid, yet there are a few outliers. On the other hand, the axes drawn on the proposed system appear to be overlapping, meaning that the proposed pose is stable.



(a) ARToolKit+ (b) ArUco (c) RUNE-Tag (d) STag

Figure 16: 500 pose axes are plotted on markers seen from 30°. The instability in the estimated pose causes the axes to appear scattered. The incorrect pose estimations due to the ambiguity have been eliminated manually.

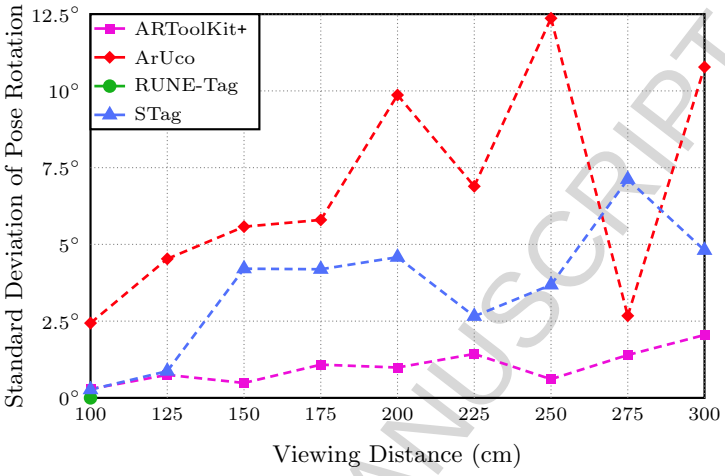
Table 6: Number of false negative detections out of 1000 frames.

Viewing Distance (cm)	100	125	150	175	200	225	250	275	300
ARToolKit+	0	0	0	0	0	0	0	0	0
ArUco	0	0	0	0	0	0	0	0	0
RUNE-Tag	0	1000	1000	1000	1000	1000	1000	1000	1000
STag	0	0	0	0	0	0	0	0	0

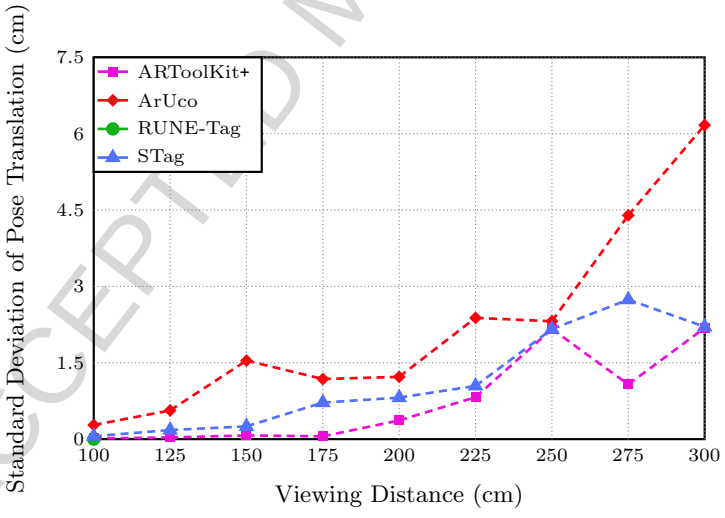
5.5.2. Different Viewing Distances

We have repeated the experiment in Section 5.5.1 with a constant 0° viewing angle and different viewing distances. See Table 6 for detection performances. Similar to the case with large viewing angles, RUNE-Tag cannot be detected when viewed from afar, as its coding circles appear too small to be detected. Since the other systems are use a thick outer border for detection, they can be detected from a distance.

See Figure 17 for pose stability across viewing distances. Unlike changing viewing distance, pose ambiguity becomes the dominant factor in pose stability, which is apparent from the rather large standard deviations in Figure 17a. ARToolkit+ and STag are affected less because they use [26] to estimate the pose. ARToolkit+ is affected even less than STag, as its corners are localized in a more coarse manner, which seems to be an advantage in the case where the marker appears smaller. Compared to pose rotation standard deviations, pose



(a) Rotation



(b) Translation

Figure 17: Pose rotation and translation stability with different viewing distances.

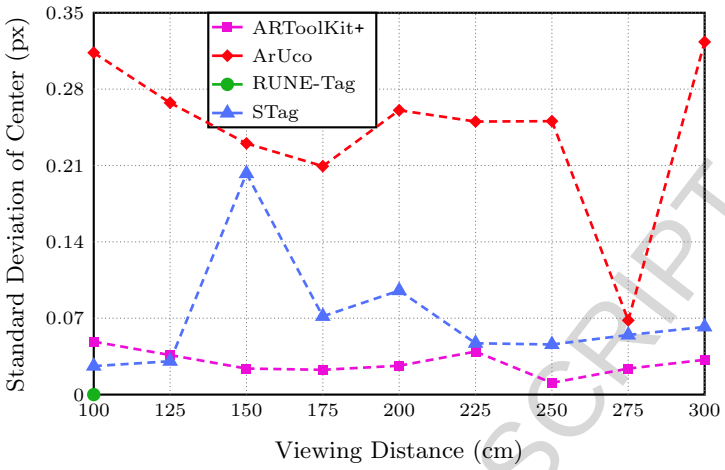


Figure 18: Localization stability with different viewing distances.

Table 7: Running times of each algorithm with 1280×720 images with a single marker on the scene.

	ARToolKit+	ArUco	RUNE-Tag	STag
Running time (ms)	2.6	10.0	579.9	18.1

translation standard deviations are much more comparable for all methods (see Figure 17b).

Finally, we localized the marker centers using $[R|\mathbf{t}]$. It is interesting to see that the marker center stability metrics in Figure 15 and Figure 18 are more similar compared to stabilities in pose. This means that using the markers from a large distance does not degrade their localization stability as much as it degrades their pose stability.

5.6. Running Times

Marker detection algorithms tend to be embarrassingly parallelizable. To nullify the effects of different parallel implementations, we used a single core of a 3.70 GHz Intel Xeon processor to run the experiments. See Table 7 for average running times per image for the experiment in Section 5.5. We can see that ARToolKit+ is very fast, both ArUco and STag perform in real-time, while

RUNE-Tag is considerably slower.

6. Conclusion

The main contribution of the proposed marker system is improving stability with a homography refinement step. This is achieved by estimating the homography with the outer square border and refining it with the inner circular border. To our knowledge, this problem and the respective solution is unique in the literature in the way that a single conic correspondence is used, without additional projective constraints such as a point [15].

The proposed solution is significantly more stable than ArUco [13]. Since it does not depend on a large number of fine details, it is significantly more robust across viewing conditions and runs an order of magnitude faster than RUNE-Tag [18]. However, we have seen that its localization characteristics are somewhat more prone to pose ambiguity compared to ARToolkit+ [10]. Considering this pose ambiguity is an important factor in pose estimation stability, future work should focus on marker designs that takes this issue into account.

For the multi-marker case, the inner circular borders can be used to estimate the pose with a multiple conic correspondence approach [36]. The alternative we have proposed is using the ellipse centers as stable, yet inaccurate correspondences of marker centers. This may cause some inaccuracy, but the stability of the point correspondences will yield an even more stable pose estimation.

References

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

[2] J. Rekimoto, Matrix: A realtime object identification and registration method for augmented reality, in: Proc. Asia Pacific Conf. Comput. Human Interaction, 1998, pp. 63–68.

[3] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, K. Tachibana, Virtual object manipulation on a table-top AR environment, in: Proc. Int. Symp. Augmented Reality,, 2000, pp. 111–119.

[4] X. Zhang, S. Fronz, N. Navab, Visual marker detection and decoding in AR systems: A comparative study, in: Proc. IEEE/ACM Int. Symp. Mixed and Augmented Reality, 2002, pp. 97–106.

[5] Make Magazine, See the Secret Prototypes We Found in Valve's VR Lab, http://makezine.com/2016/06/21/exclusive-see-the-secret-prototypes-we-found-in-valves-vr-lab/ (2016).

[6] E. Olson, AprilTag: A robust and flexible visual fiducial system, in: Proc. IEEE Int. Conf. Robotics and Automation, 2011, pp. 3400–3407.

[7] Amazon, Amazon Prime Air's First Customer Delivery, https://youtu.be/vNySOrI2Ny8?t=1m13s (2016).

[8] Boston Dynamics, Atlas, The Next Generation, https://youtu.be/rVlhMGQgDkY?t=1m23s (2016).

[9] H. Kato, M. Billinghurst, Marker tracking and HMD calibration for a video-based augmented reality conferencing system, in: Proc. IEEE/ACM Int. Workshop Augmented Reality, 1999, pp. 85–94.

[10] D. Wagner, D. Schmalstieg, ARToolKitPlus for pose tracking on mobile devices, in: Comput. Vision Winter Workshop, 2007.

[11] M. Fiala, ARTag, a fiducial marker system using digital techniques, in: Proc. IEEE Conf. Comput. Vision and Pattern Recognition, 2005, pp. 590–596.

[12] F. Bergamasco, A. Albarelli, E. Rodola, A. Torsello, RUNE-Tag: A high accuracy fiducial marker with strong occlusion resilience, in: Proc. IEEE Conf. Comput. Vision and Pattern Recognition, 2011, pp. 113–120.

[13] 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.

[14] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, R. Medina-Carnicer, Generation of fiducial marker dictionaries using mixed integer linear programming, Pattern Recognition 51 (2016) 481–491.

[15] D. López de Ipiña, P. R. S. Mendonça, A. Hopper, TRIP: A low-cost vision-based location system for ubiquitous computing, Personal and Ubiquitous Computing 6 (3) (2002) 206–219.

[16] L. Naimark, E. Foxlin, Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker, in: Proc. IEEE/ACM Int. Symp. Mixed and Augmented Reality, 2002.

[17] A. Xu, G. Dudek, Fourier Tag: A smoothly degradable fiducial marker system with configurable payload capacity, in: Proc. Canadian Conf. Comput. and Robot Vision, 2011, pp. 40–47.

[18] F. Bergamasco, A. Albarelli, L. Cosmo, E. Rodola, A. Torsello, An accurate and robust artificial marker based on cyclic codes, IEEE Trans. Pattern Anal. Mach. Intell. 38 (12) (2016) 2359–2373.

[19] J. DeGol, T. Bretl, D. Hoiem, ChromaTag: A Colored Marker and Fast Detection Algorithm, in: Proc. IEEE Int. Conf. Comput. Vision, 2017.

[20] L. Calvet, P. Gurdjos, C. Griwodz, S. Gasparini, Detection and accurate localization of circular fiducials under highly challenging conditions, in: Proc. IEEE Conf. Comput. Vision and Pattern Recognition, 2016, pp. 562–570.

[21] J. Sattar, E. Bourque, P. Giguere, G. Dudek, Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction, in: Proc. Conf. Comput. and Robot Vision, 2007, pp. 165–174.

[22] H. Uchiyama, H. Saito, Random dot markers, in: Proc. Virtual Reality Conf., 2011, pp. 35–38.

[23] A. Fitzgibbon, M. Pilu, R. B. Fisher, Direct least square fitting of ellipses, IEEE Trans. Pattern Anal. Mach. Intell. 21 (5) (1999) 476–480.

[24] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simmulated annealing, Science 220 (4598) (1983) 671–680.

[25] J. Conway, N. Sloane, Lexicographic codes: Error-correcting codes from game theory, IEEE Trans. on Inf. Theory 32 (3) (1986) 337–348.

[26] G. Schweighofer, A. Pinz, Robust pose estimation from a planar target, IEEE Trans. Pattern Anal. Mach. Intell. 28 (12) (2006) 2024–2030.

[27] C. Akinlar, C. Topal, EDPF: A real-time parameter-free edge segment detector with a false detection control, Int. J. of Pattern Recognition and Artificial Intell. 26 (1) (2012) 862872.

[28] C. Akinlar, C. Topal, EDLines: A real-time line segment detector with a false detection control, Pattern Recognition Letters 32 (13) (2011) 1633–1642.

[29] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2003.

[30] G. Sparr, Depth computations from polyhedral images, Image and Vision Computing 10 (10) (1992) 683–688.

[31] J. A. Nelder, R. Mead, A simplex method for function minimization, The Computer Journal 7 (4) (1965) 308–313.

[32] O. Ozsen, C. Topal, C. Akinlar, Parallelizing edge drawing algorithm on CUDA, in: Proc. IEEE Int. Conf. Emerging Signal Process. Applicat., 2012, pp. 79–82.

[33] A. Quattoni, A. Torralba, Recognizing indoor scenes, in: Proc. IEEE Conf. Comput. Vision and Pattern Recognition, 2009, pp. 413–420.

[34] Z. Zhang, A flexible new technique for camera calibration, IEEE Trans. Pattern Anal. Mach. Intell. 22 (11) (2000) 1330–1334.

[35] R. I. Hartley, Chirality, Int. J. of Comput. Vision 26 (1) (1998) 41–61.

[36] J. Kannala, M. Salo, J. Heikkilä, Algorithms for computing a planar homography from conics in correspondence, in: Proc. British Mach. Vision Conf., 2006, pp. 77–86.

Highlights

- STag is a novel fiducial marker system designed for stable pose estimation.
- In addition to the traditional square border, STag has an inner circular border.
- Circular borders can be localized more stably compared to square borders.
- STag uses the stable circular border to refine the estimated homography.
- The refined homography is more stable and can be used for stable pose estimation.

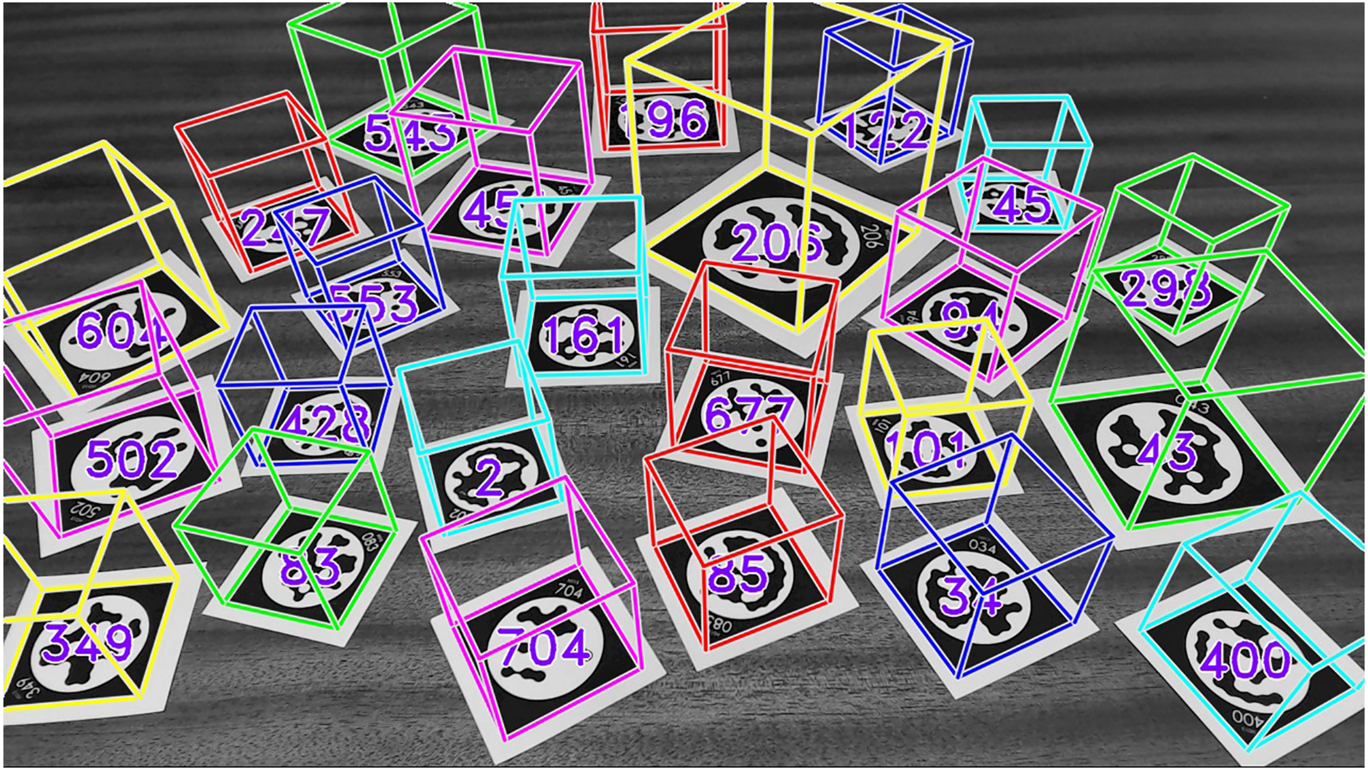
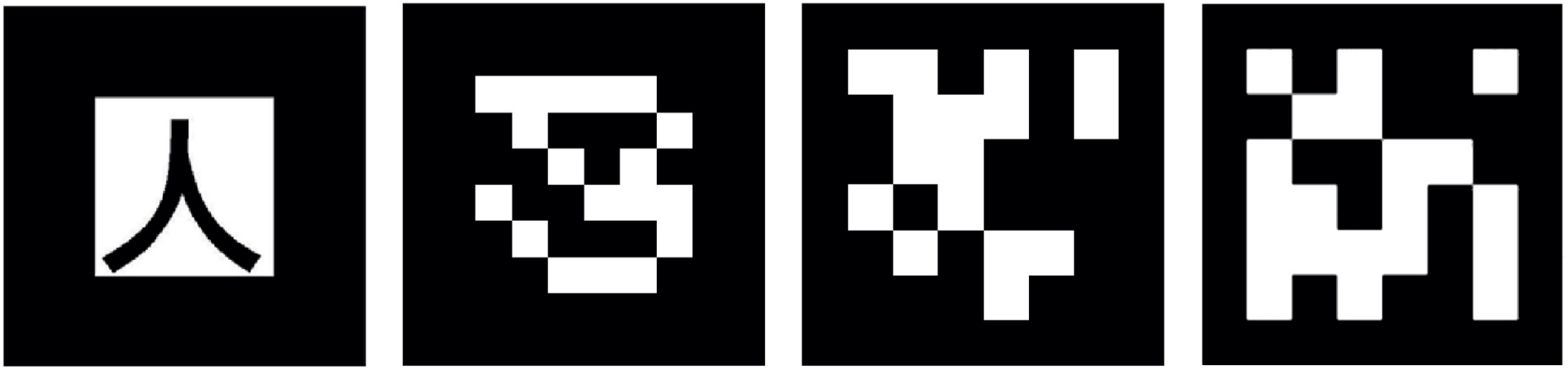
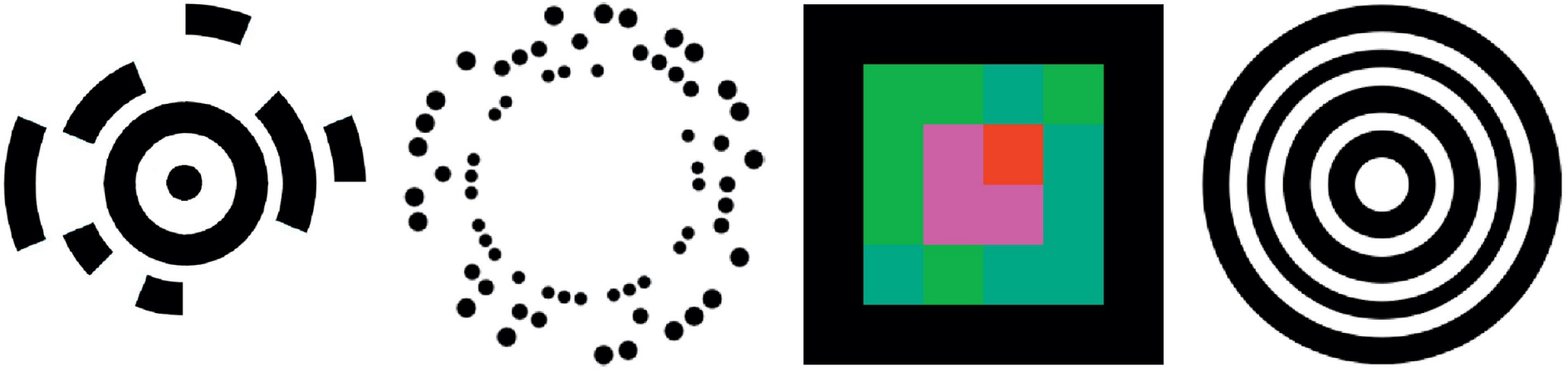

543
196
122
45
206
553
161
604
677
428
101
43
502
2
83
85
704
349
400

Figure 1

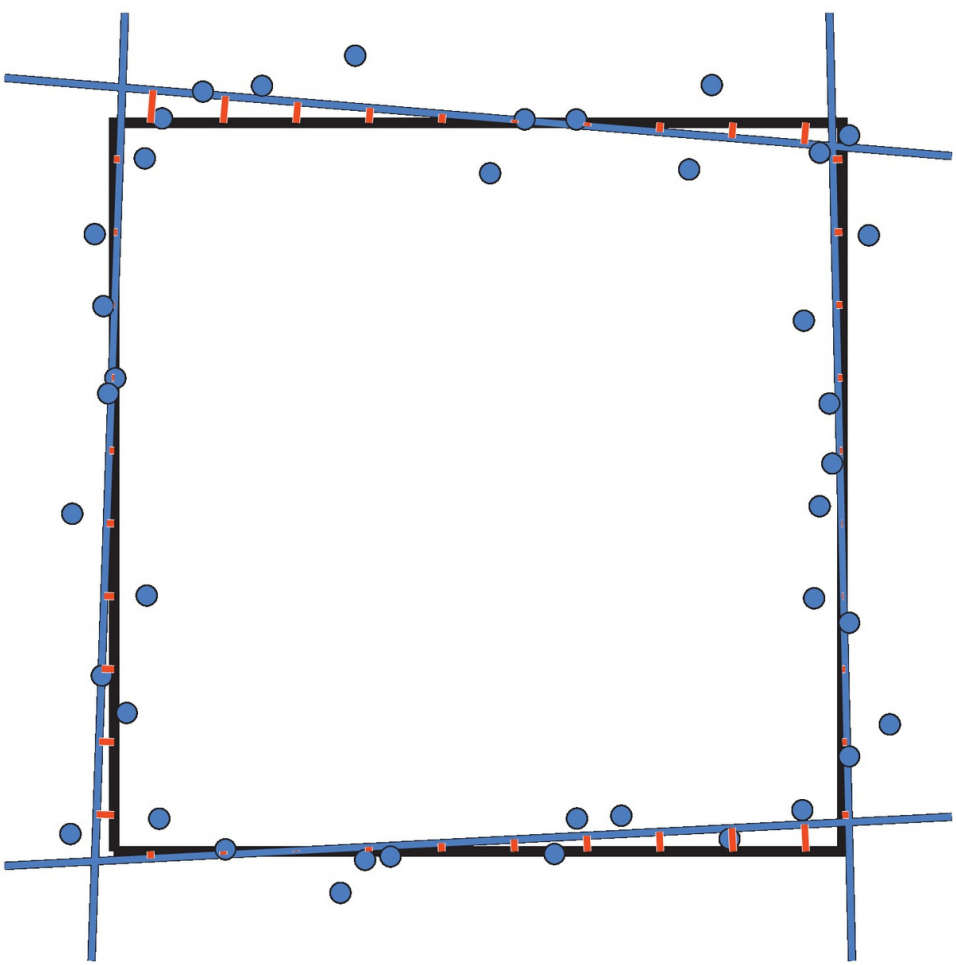


(a) ARToolkit [9] (b) ARTag [11] (c) AprilTag [6] (d) ArUco [13]

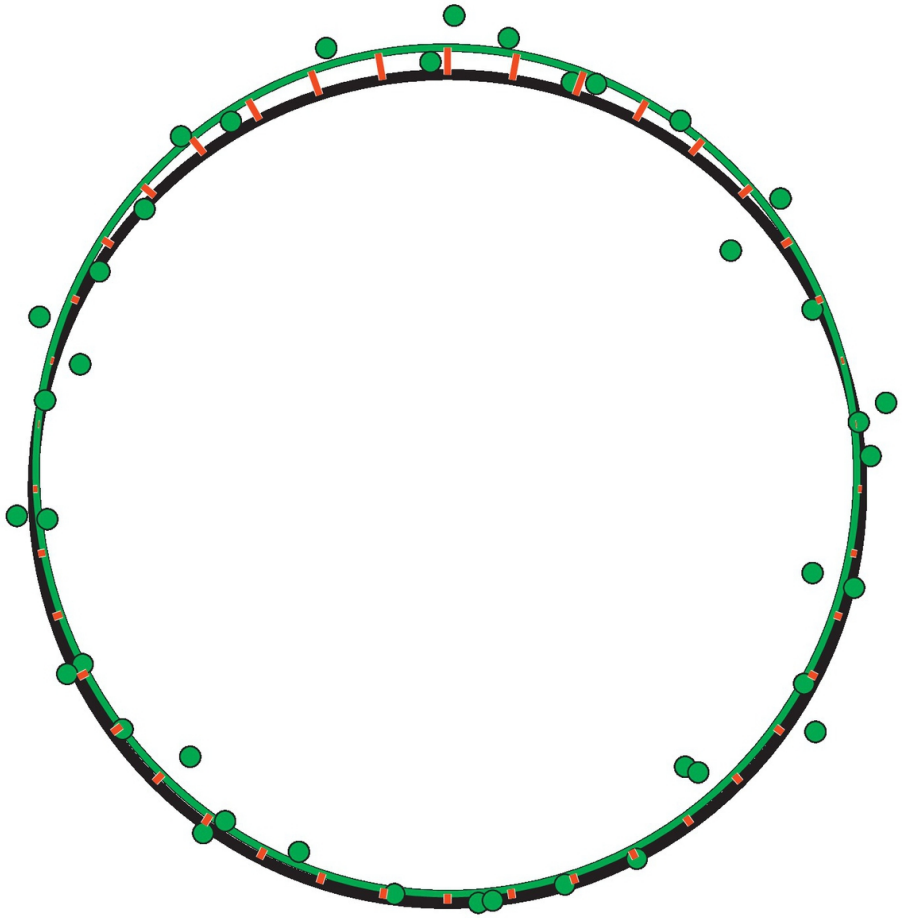


(e) TRIP [15] (f) RUNE-Tag [18] (g) ChromaTag [19] (h) CCTag [20]

Figure 2



(a) Quad localization



(b) Ellipse localization

Figure 3

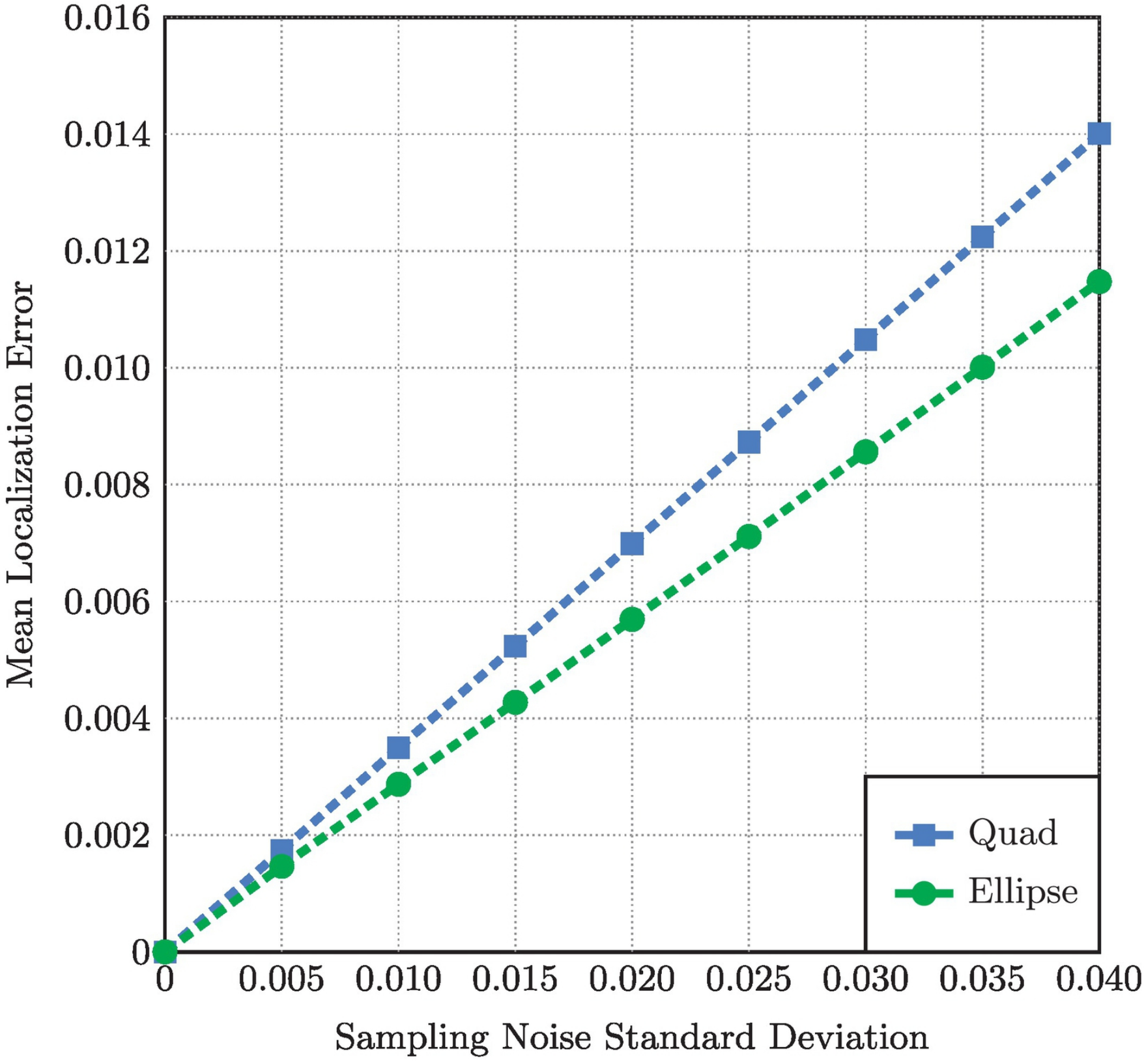
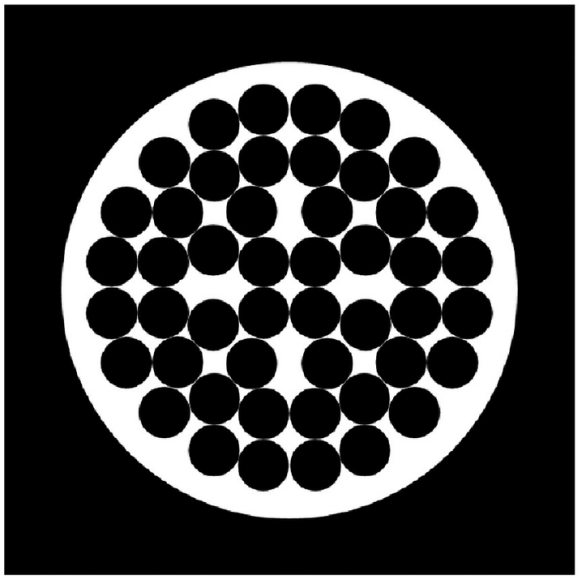
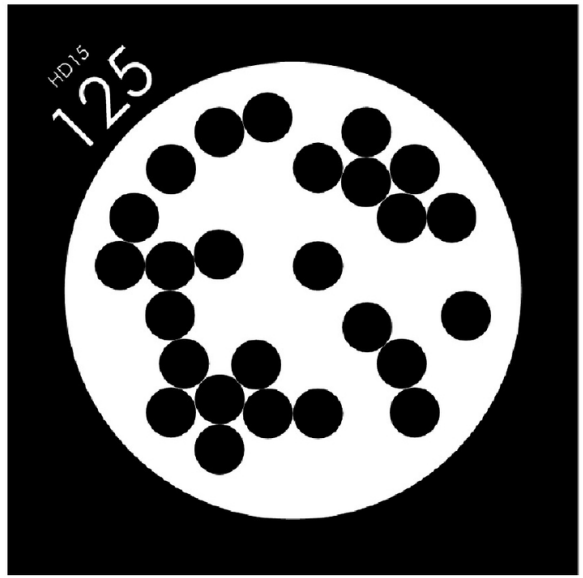

0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0
Mean Localization Error
0
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
Sampling Noise Standard Deviation
Quad
Ellipse

Figure 4

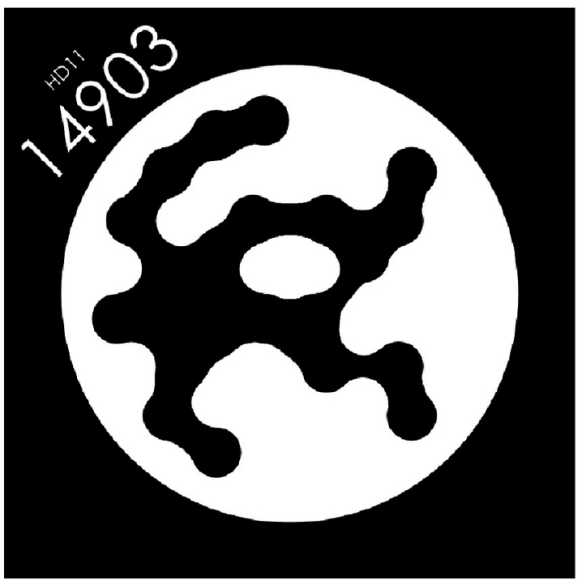


(a)

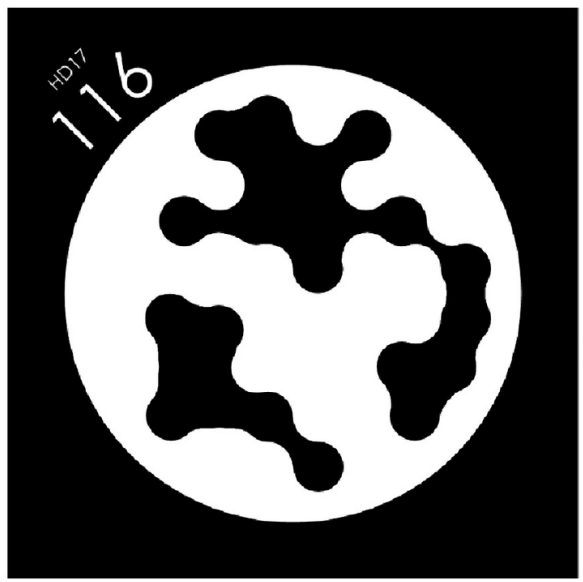


(b)

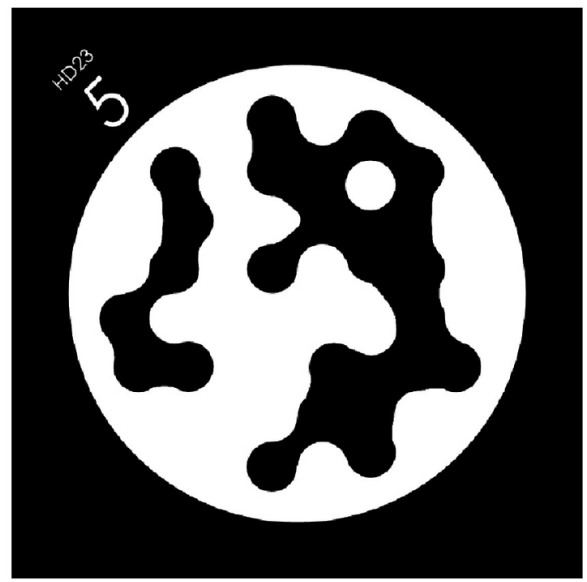
(c)



(d)



(e)



(f)

Figure 5

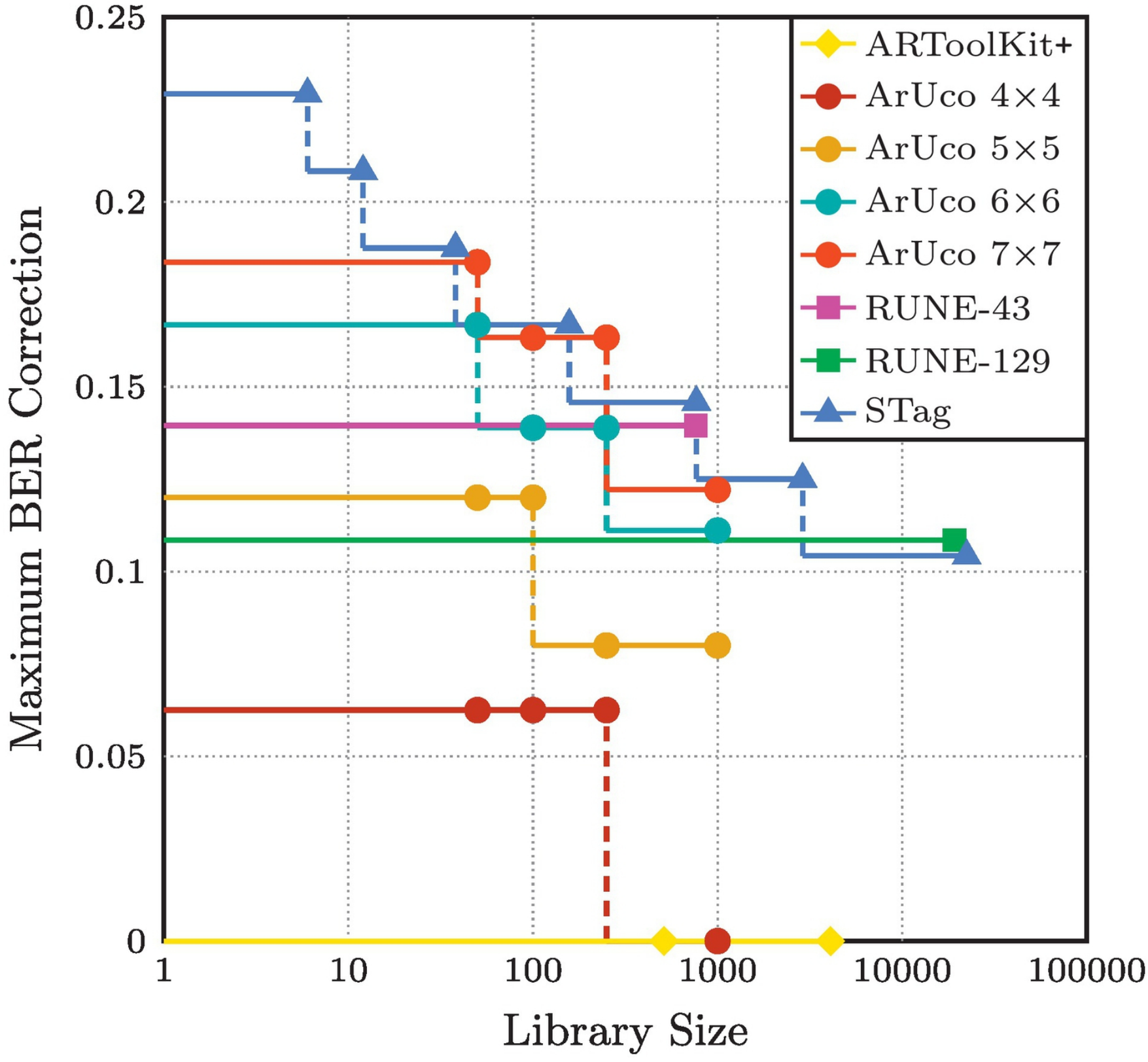

Maximum BER Correction
0.25
0.2
0.15
0.1
0.05
0
1
10
100
1000
10000
100000
Library Size
ARToolKit+
ArUco 4×4
ArUco 5×5
ArUco 6×6
ArUco 7×7
RUNE-43
RUNE-129
STag

Figure 6

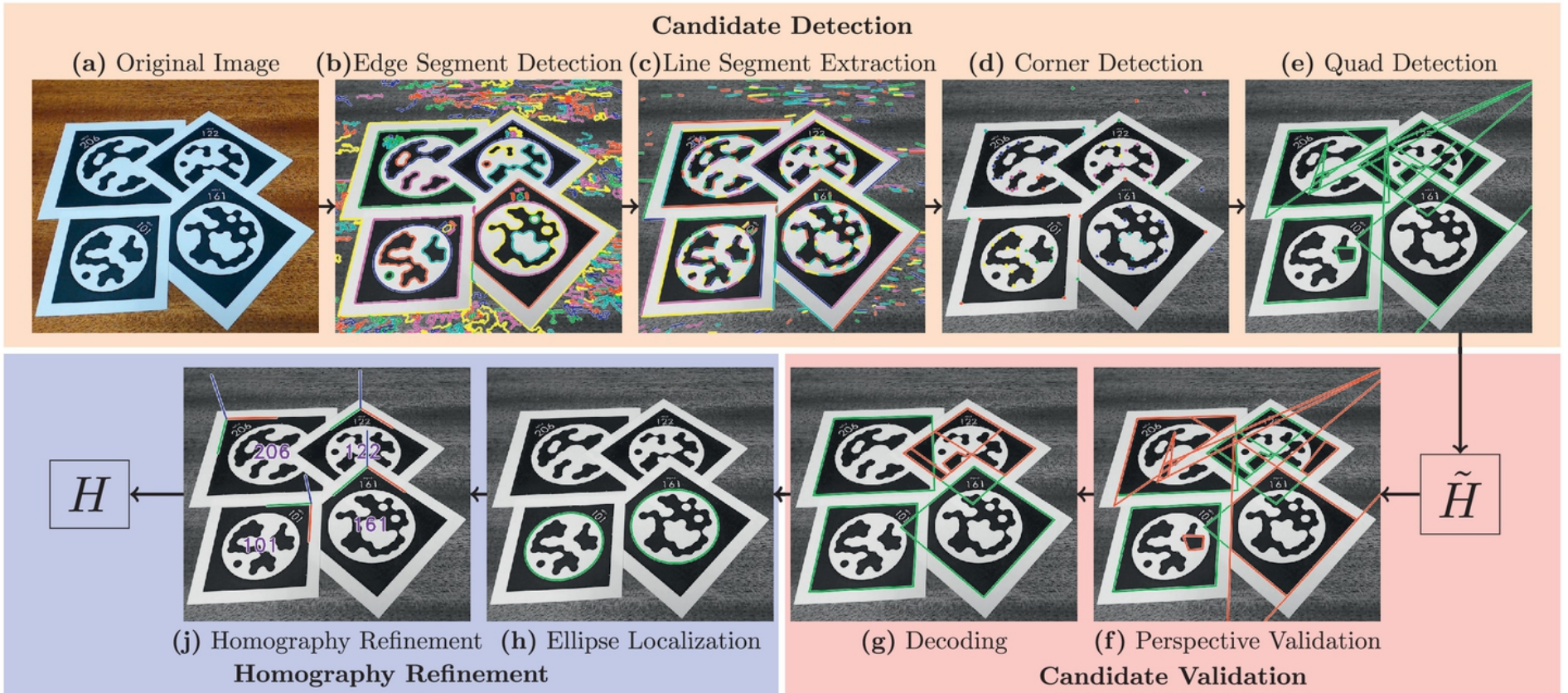

Candidate Detection
(a) Original Image
(b)Edge Segment Detection
(c)Line Segment Extraction
(d) Corner Detection
(e) Quad Detection
H
206
122
161
101
(j) Homography Refinement
(h) Ellipse Localization
Homography Refinement
(g) Decoding
(f) Perspective Validation
Candidate Validation
$\tilde{H}$

Figure 7

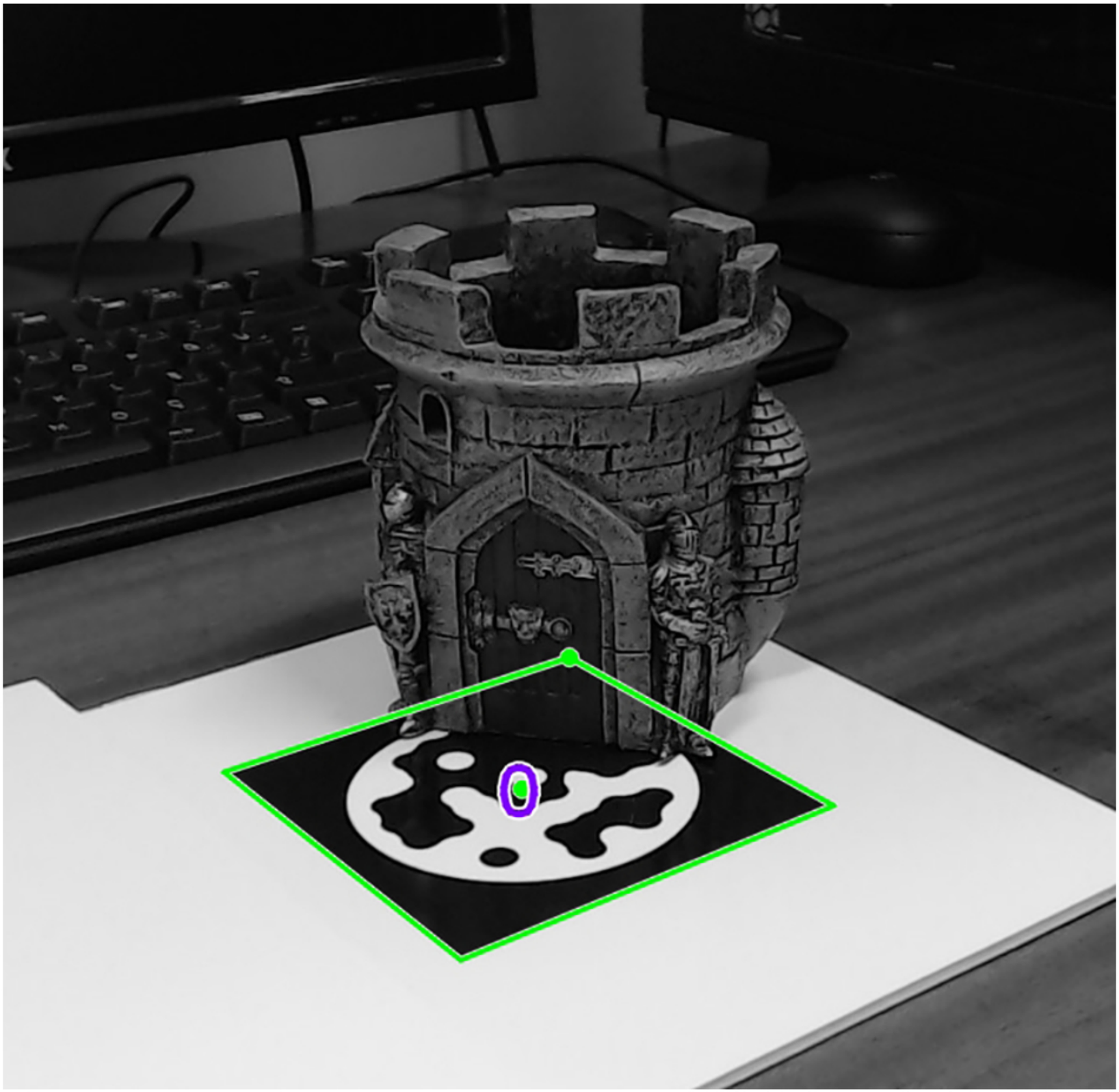


Figure 8

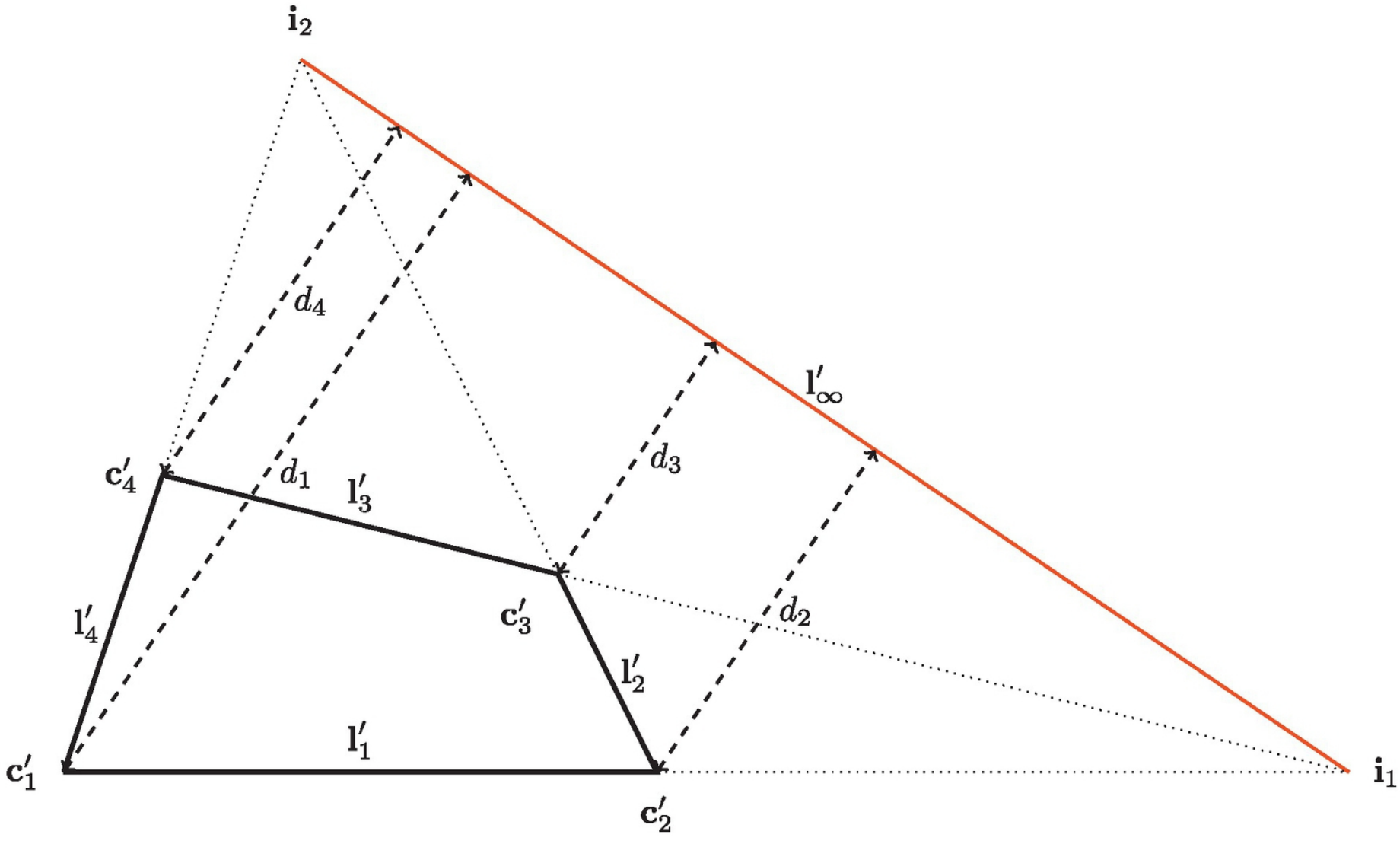
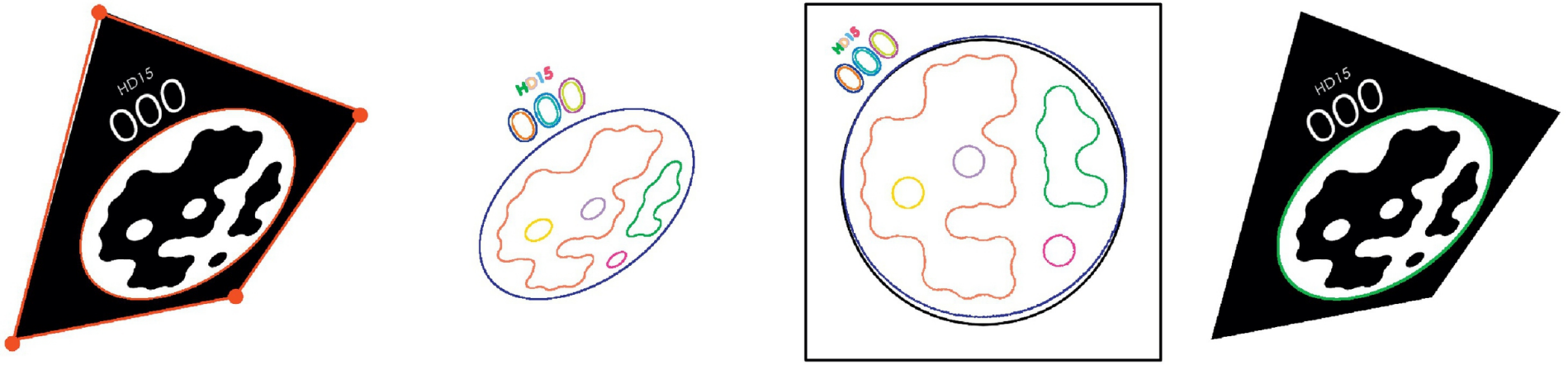

$\mathbf{i}_2$
d_4
$\mathbf{l}'_\infty$
d_3
$\mathbf{c}'_4$
d_1
$\mathbf{l}'_3$
$\mathbf{l}'_4$
$\mathbf{c}'_3$
d_2
$\mathbf{l}'_2$
$\mathbf{l}'_1$
$\mathbf{c}'_1$
$\mathbf{i}_1$
$\mathbf{c}'_2$

Figure 9



(a) (b) (c) (d)

Figure 10

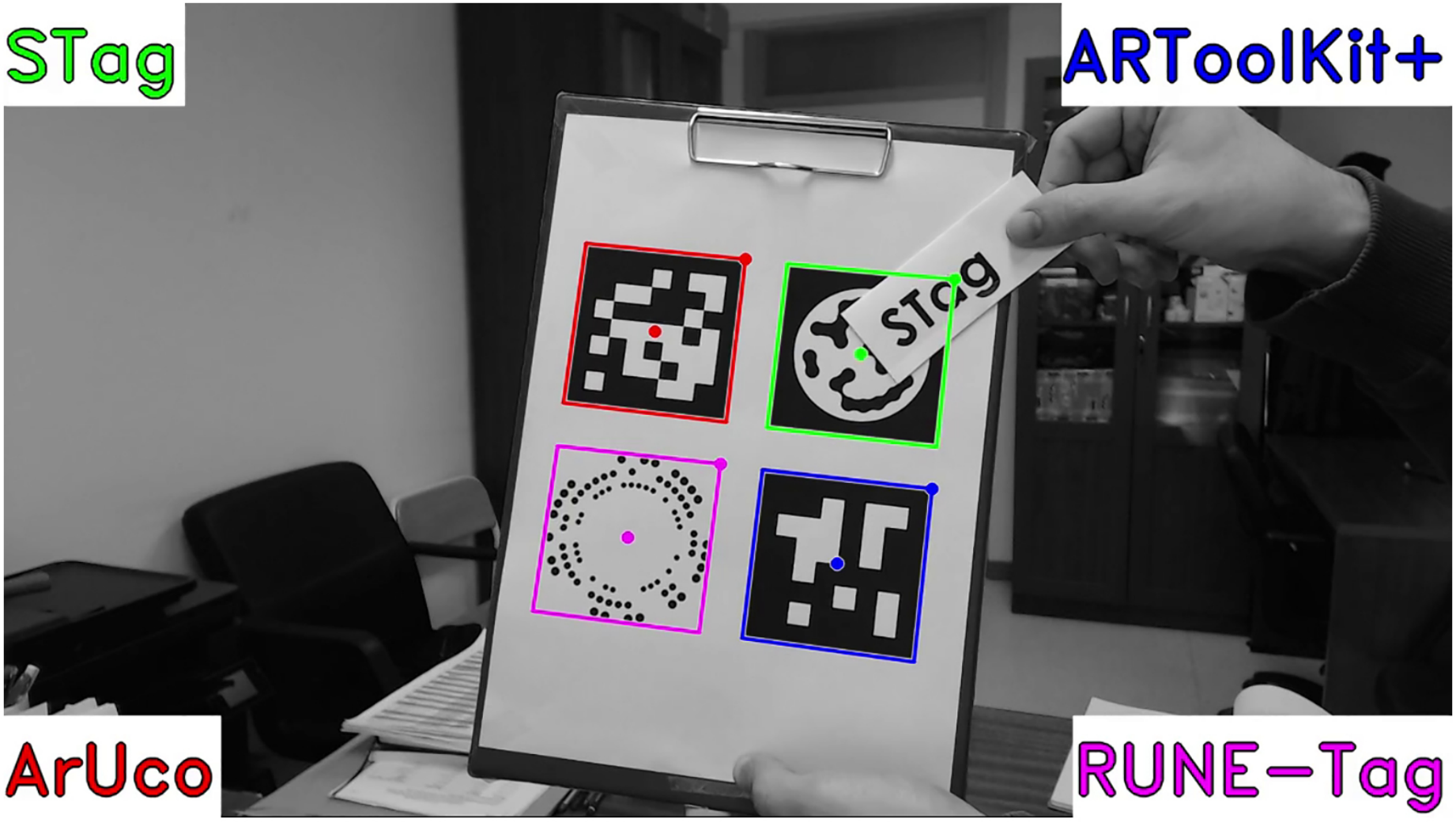


Figure 11

Figure 12

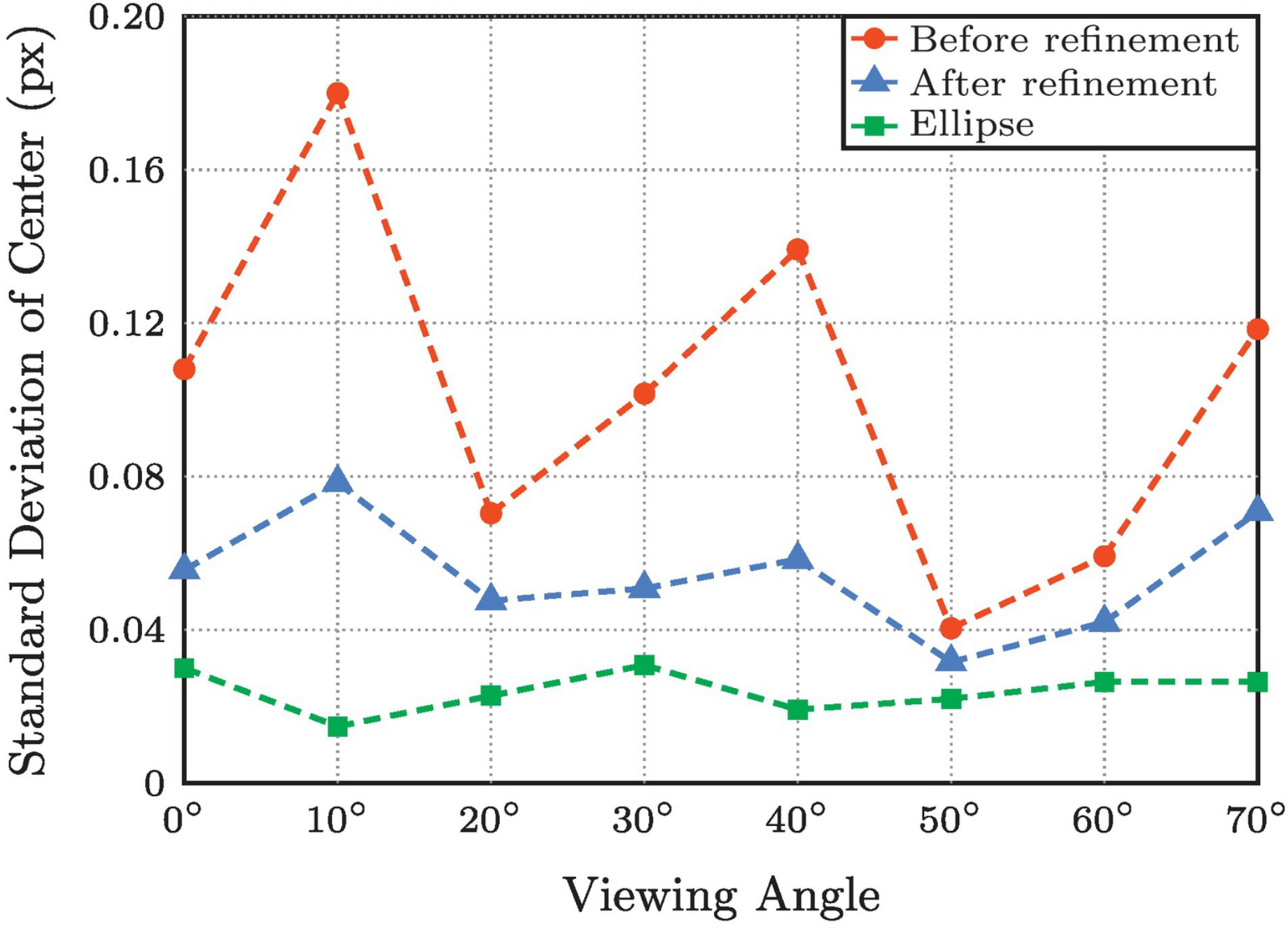
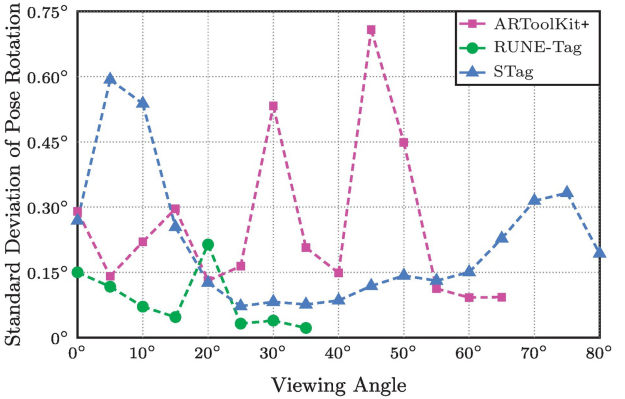
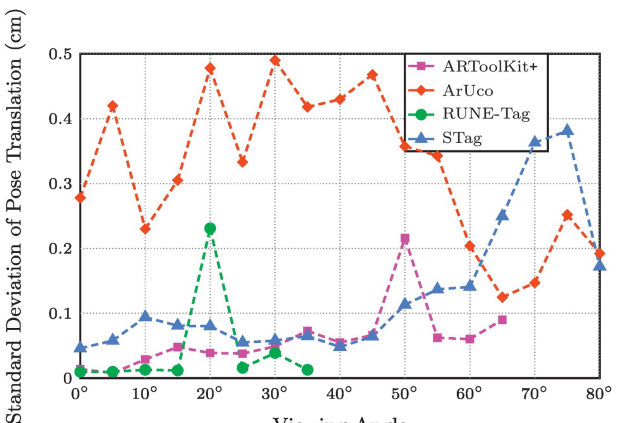

Standard Deviation of Center (px)
0.20
0.16
0.12
0.08
0.04
0
0°
10°
20°
30°
40°
50°
60°
70°
Viewing Angle
Before refinement
After refinement
Ellipse

Figure 13



(a) Rotation



(b) Translation

Figure 14

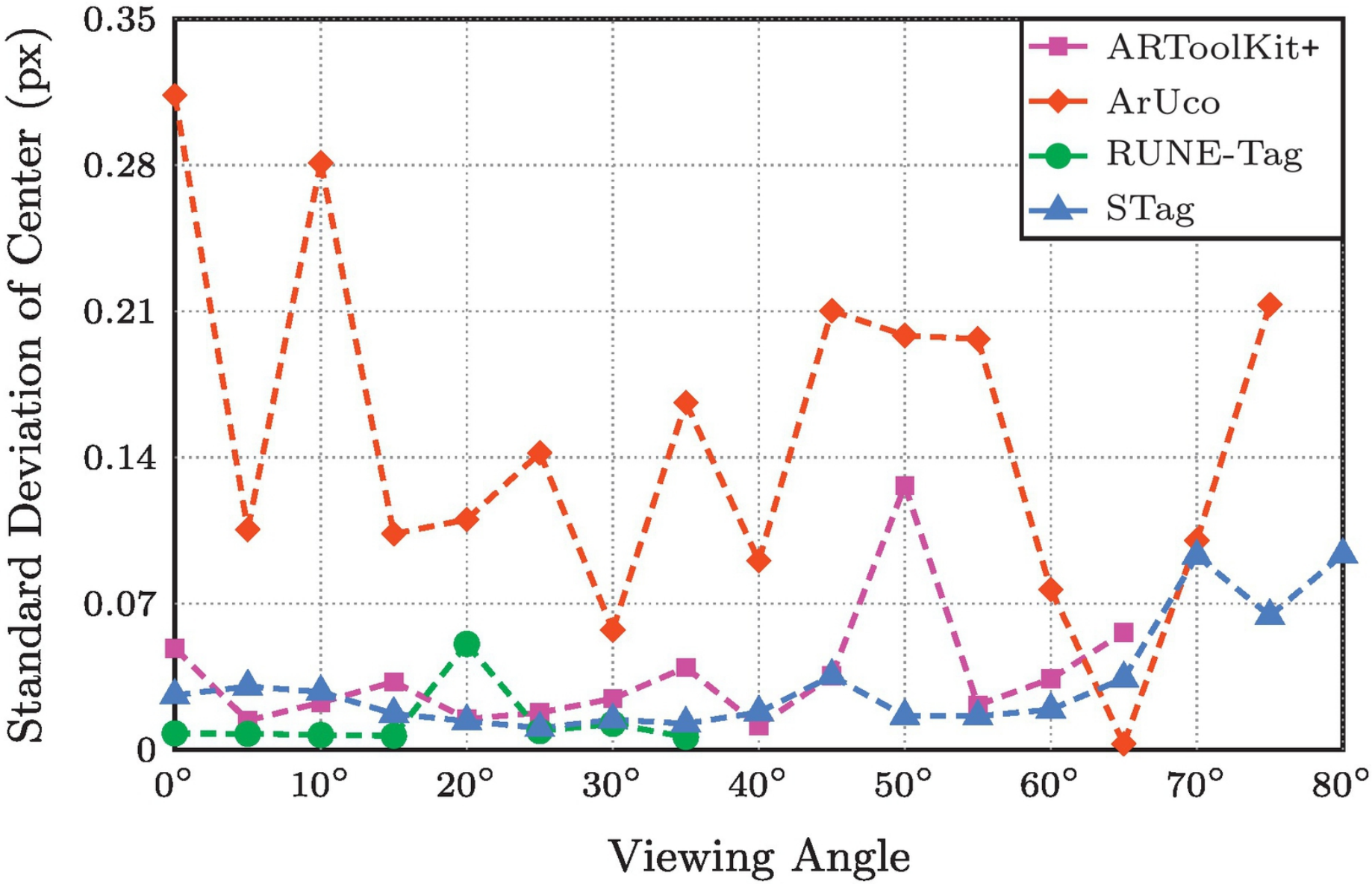
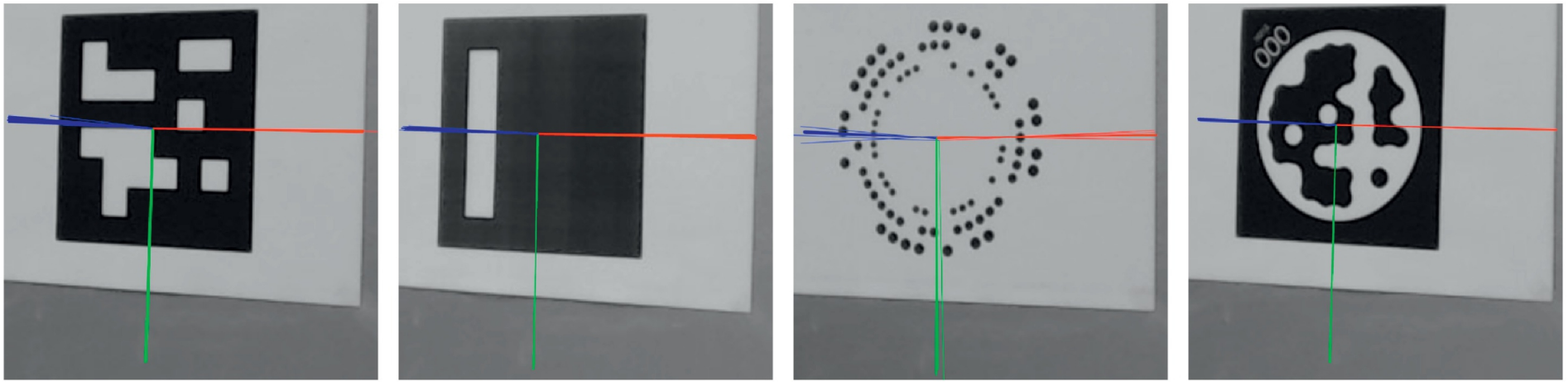
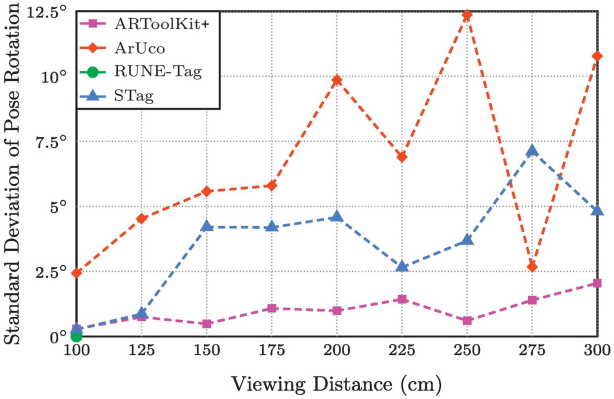

Standard Deviation of Center (px)
0.35
0.28
0.21
0.14
0.07
0
0°
10°
20°
30°
40°
50°
60°
70°
80°
Viewing Angle
ARToolKit+
ArUco
RUNE-Tag
STag

Figure 15

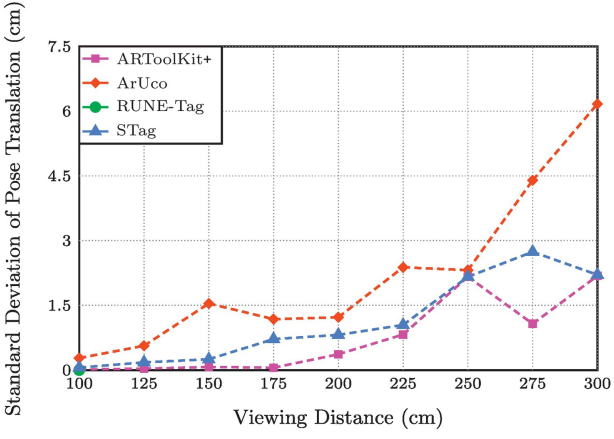


(a) ARToolKit+ (b) ArUco (c) RUNE-Tag (d) STag

Figure 16



(a) Rotation



(b) Translation

Figure 17

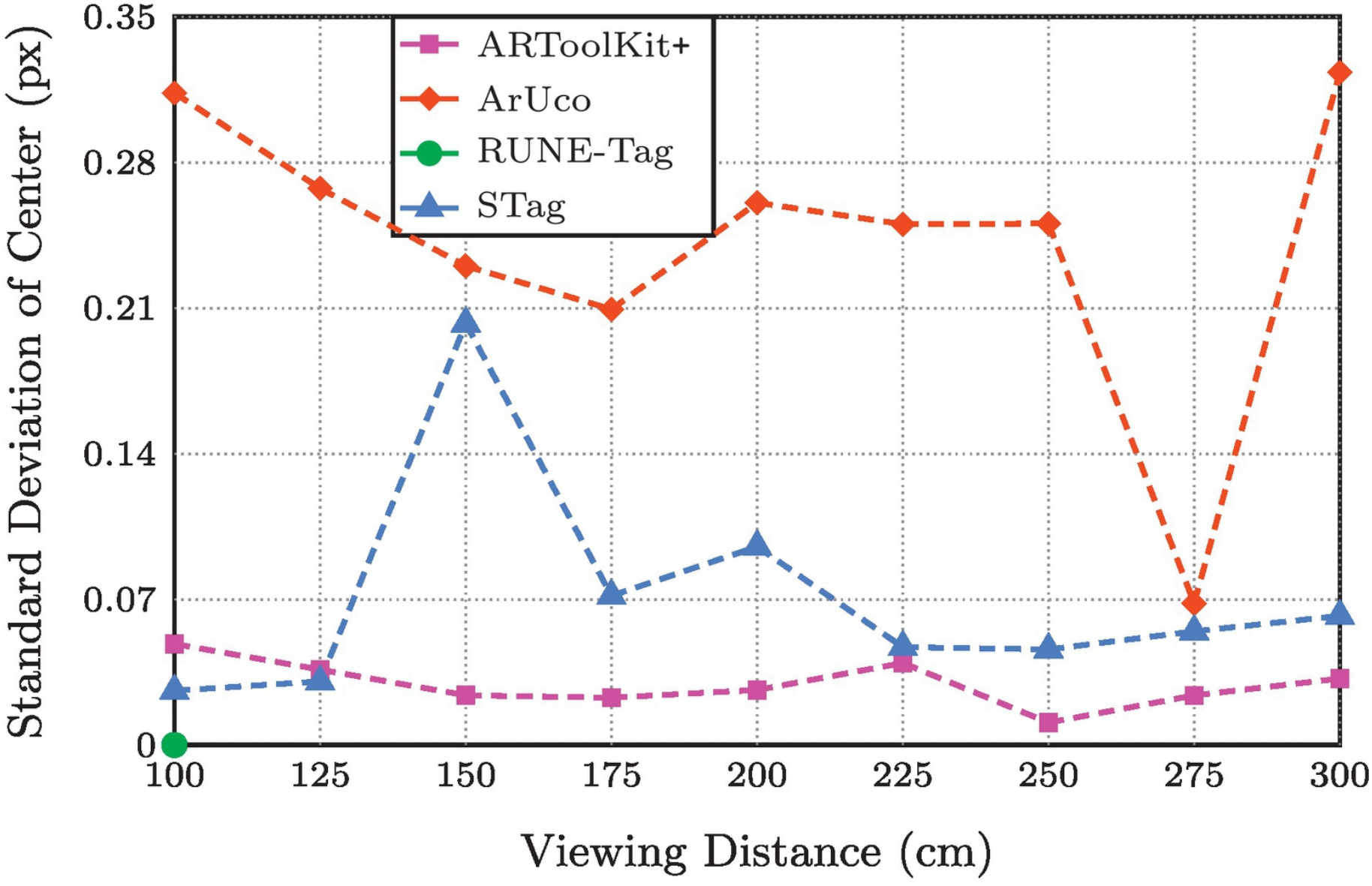

ARToolKit+
ArUco
RUNE-Tag
STag
Standard Deviation of Center (px)
0.35
0.28
0.21
0.14
0.07
0
100
125
150
175
200
225
250
275
300
Viewing Distance (cm)

Figure 18