Planar fiducial markers: a comparative study

Article *in* Virtual Reality · February 2023

DOI: 10.1007/s10055-023-00772-5

CITATIONS
8

READS
1,995

4 authors, including:

David Jurado

12 PUBLICATIONS **57** CITATIONS

SEE PROFILE

Rafael Muñoz-Salinas

University of Córdoba

118 PUBLICATIONS **5,602** CITATIONS

SEE PROFILE

Rafael Medina-Carnicer

University of Córdoba

108 PUBLICATIONS **3,201** CITATIONS

SEE PROFILE

Planar Fiducial Markers: A Comparative Study

David Jurado-Rodriguez[1,2], Rafael Muñoz-Salinas[1,3,*], Sergio Garrido-Jurado[2], Rafael Medina-Carnicer[1,3]

Abstract

Fiducial markers are a cost-effective solution for solving labeling and monocular localization problems, making them valuable tools for Augmented Reality (AR), robot navigation, and 3D modeling applications. However, with the development of many marker detection systems in the last decade, it has become challenging for new users to determine which is best suited for their needs. This paper presents a qualitative and quantitative evaluation of the most relevant marker systems. We analyze the available alternatives in the literature, describe their differences and limitations, and conduct detailed experiments to compare them in terms of sensitivity, specificity, accuracy, computational cost, and performance under occlusion. To our knowledge, this study provides the most comprehensive and updated comparison of fiducial markers. In the Conclusion section, we offer recommendations on which method to use based on the application requirements.

Keywords: Augmented Reality, Fiducial Planar Fiducial Markers, Pose Estimation, Marker System Comparison,

1. Introduction

Planar fiducial markers are a cost-effective solution for addressing labeling and localization challenges. These markers consist of a two-dimensional pattern with an external border and an internal colored code, enabling unique identification. There are three main advantages of using planar fiducial markers. Firstly, the pose of the camera can be accurately computed based on the marker's shape alone. Secondly, the markers can be detected quickly with low CPU usage [54]. Finally, their detection is robust to changes in lighting and perspective. Because of these benefits, planar fiducial markers have become a widely used tool in fields such as autonomous robots,[32, 62, 23], indoor navigation [16, 42], unmanned vehicles [65, 9, 46], and medicine [36, 52, 25, 58] and even in Augmented Reality (AR) applications [6, 8, 11, 26].

In recent years, several techniques have been proposed for developing AR applications by utilizing SLAM algorithms to create a map of the environment based on natural features [33, 41, 17]. Mobile development frameworks like ARCore [5] and ARKit [6] combine camera detection with inertial information to support these efforts. Although these techniques can be utilized in AR applications, they have several limitations. Firstly, they require a substantial amount of texture, which may not be present in certain indoor environments, such as laboratories and corridors. Secondly, the methods used for relocalization, such as *bag-of-words* (BoW) [20], have limited performance under viewpoint changes, repetitive patterns [64], a lack of texture [72], and changes over time [37]. Thirdly, the scale remains unknown unless multiple cameras or other sensors, such as inertial sensors, are employed.

As a solution to these issues, several authors have suggested the use of fiducial markers to enhance the pose estimation process [12, 71, 34, 61, 48]. The SPM-SLAM method [43] addresses some of the aforementioned limitations by utilizing squared fiducial markers instead of natural features. These markers can be positioned anywhere in the environment, and SPM-SLAM can create a 3D map of them. Later, the study in [45] demonstrated that combining fiducial markers with key points leads to higher accuracy than prior visual-SLAM techniques. This research confirms that fiducial markers are a useful tool that can assist in various aspects of pose estimation.

*Corresponding author

Email addresses: jrdavidrj@gmail.com (David Jurado-Rodriguez), in1musar@uco.es (Rafael Muñoz-Salinas), sgj@seaberyat.com (Sergio Garrido-Jurado), rmedina@uco.es (Rafael Medina-Carnicer)

[1]Departamento de Informática y Análisis Numérico, Edificio Einstein. Campus de Rabanales, Universidad de Cofdoba, 14071, Córdoba, Spain, Tlfn:(+34)957212289

[2]Seabery R&D, Doctor Emilio Haya Prats 13, 21005 Huelva, Spain Tlfn:(+34)959807473

[3]Instituto Maimónides de Investigación en Biomedicina (IMIBIC). Avenida Menéndez Pidal s/n, 14004, Córdoba, Spain, Tlfn:(+34)957213861

[5]https://developers.google.com/ar [last access 09/07/2022]

[6]https://developer.apple.com/augmented-reality/ [last access 09/07/2022]

System	Year	Cites	Cites/Year	Availability	Last Update	Languages	Pose	Examples	Comments
QR Codes[63]	1994	> 10K	-	Code	2022	`C++, Python`	Yes	Yes	It is integrated into OpenCV
ARToolkit [30]	1999	3635	158.0	Code	2017	`C++`	Yes	Yes	Old and not maintained code
Intersense [47]	2002	403	0.8	Not available	-	-	No	No	Needs four markers for pose estimation
Visual Code [53]	2004	269	14.9	Not available	-	-	-	-	-
ARTag [18]	2005	1155	67.9	Not available	-	-	-	-	-
Colored Tags [2]	2005	13	0.8	Not available	-	-	-	-	-
ARToolkit+ [66]	2007	729	48.9	Code	2017	`C++`	Yes	Yes	Halted project
ReacTIVision [29]	2007	605	40.3	Code	2016	`C++, C#`	No	No	Implements TUIO protocol
FourierTag [59]	2007	81	5.4	Code	2015	-	No	No	Needs two markers for pose estimation
CALTag [2]	2010	146	12.2	Not available	-	-	-	-	-
Rune-Tag [4]	2011	135	12.3	Code	2015	`C++`	Yes	No	Crashes in Ubuntu 20.04
BlurTag [51]	2012	9	0.9	Not available	-	-	-	-	-
ArUco [21]	2014	1693	211.6	Code	2022	`C++`	Yes	Yes	Integrated in the OpenCV-Contrib repository
ChromaTag [13]	2015	59	11.8	Code	2017	`C, C++`	Yes	No	Sensitive to lighting changes
AprilTag2 [67]	2016	493	82.1	Code	2018	`C++, Java`	Yes	Yes	Adequate and updated documentation
AprilTag3 [67]	2016	493	82.1	Code	2022	`C`	Yes	Yes	Well documented and wide variety of markers
CCTag [7]	2016	63	10.5	Code	2022	`C++, Python`	No	No	Needs two markers for pose estimation
Vumark [4]	2016	-	-	Binaries	2022	`C++, C#`	Yes	Yes	Good documentation and large number of examples
ArUco3 [54]	2018	495	123.7	Code	2022	`C++`	Yes	Yes	Accessible and documented
TopoTag [73]	2020	20	10.0	Binaries	2021	`C++`	Yes	No	Only for Windows
STag [3]	2019	25	8.3	Code	2020	`C++`	Yes	No	Provides a ROS implementation
DeepTag [74]	2021	2	1.0	Code	2022	`Python`	Yes	No	Based on CNNs

Table 1: **The fiducial marker systems found in the literature**. The table indicates each system the following information. i) The name of the system; ii) Release date; iii) number of cites, obtained from `https://scholar.google.com/`; iv) impact factor (Cites/Year); v) link to its source code or binaries; vi) last time it was updated; vii) programming language; viii) whether the system can estimate the camera position; ix) if the implementation includes examples, projects, demos, or template code; x) a comment about the meaningful features of each method.

However, despite the plethora of fiducial marker systems proposed in recent years, it is challenging for newcomers to choose the best option for their specific use case due to differences in accuracy, speed, and customization between the available alternatives. Additionally, the few comparative studies [28, 57] performed have evaluated a limited set of metrics for a small subset of systems.

Despite the wide variety of alternatives, only a few have become popular and widely adopted by the community. We believe several conditions must be met for a system to be widely used. Firstly, its source code must be available and periodically maintained. Most technologies do not include source code; when available, it is not updated to work with the latest compiler and operating systems updates. It is also important to point out that minimizing the number of external dependencies of a project is necessary to extend its lifetime. Secondly, the source code must work. It seems obvious, but sometimes you download a system and get it set up, only to find that it crashes. As a consequence, the community quickly discards it. Finally, the source code must be designed in such a way that it can adapt to a wide variety of use cases. Many authors release their source code, but it is difficult or impossible to access basic information, like the detected markers or their pose. On other occasions, the system presents some difficulty to be edited, which limits its implementation to very restricted environments.(e.g., ROS [50]). Indeed, such design approaches do not allow their widespread adoption.

This article presents a comprehensive comparative evaluation of 13 fiducial marker systems to help practitioners select the best alternative for their needs.

Table 1 shows a summary of the different alternatives that the authors of this paper have found in the literature. It may not be an exhaustive list but it provides the most relevant works. The table shows relevant information about each method to analyze its real impacts on the community, such as the number of cites, the availability of source code, or the last time it was updated. In the case of the QR Code, it has been impossible to accurately estimate the number of citations, as it is an old method with no single publication to refer to.

Among all the markers detection systems shown in Table 1, only systems that satisfy the following non-functional requirements (NFRs) have been chosen. i) Availability and Reliability: source code and binaries must be available without errors during execution or compilation, ii) Compatibility and Maintainability: the executable should not crash when tested, iii) Usability: it must be able to compute the camera pose.

Based on the above guidelines, the following marker detection systems have been selected for evaluation in this research: QR Codes [63], ARToolkit [30], ARToolkit+ [66], ArUco [21], ArUco3 [54], AprilTag2 [67], AprilTag3, STag [3], TopoTag [73], DeepTag [74], ChromaTag [14], Jumarker[27] and VuMark. Figure 1 shows the marker design of each technology. Note that DeepTag does not propose a different marker design. This technology is compatible with some of the systems cited above.

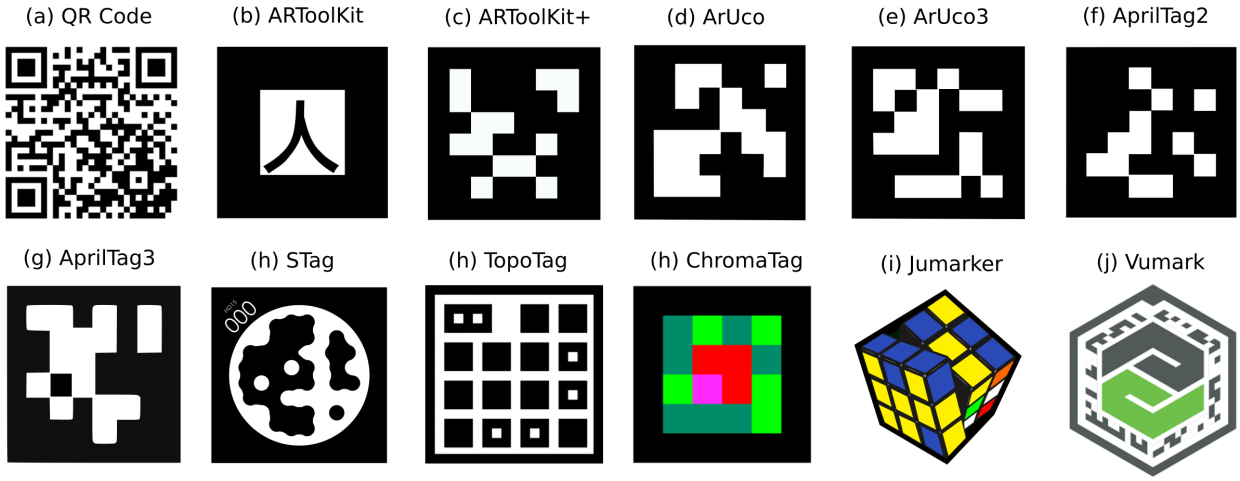


Figure 1: **Fiducial markers** selected for evaluation in this work.

Although it is expected that a newer version of a system will perform better than its predecessor, evaluating the degree of improvement is interesting. In this manner, the reader can assess the extent of the method's evolution. It's noteworthy that the maintenance and updating teams for ArUco and ArUCo3 are different.

The first part of this paper (section 2) introduces the list of fiducial marker systems employed in this research. We also provide a chronological exposition of the alternatives proposed and how they improved upon each other until the most recent works.

The second part of this paper (section 3) compares the selected systems under the same conditions. In particular, we have evaluated their sensitivity, specificity, accuracy, robustness, and speed. To our knowledge, this is the most comprehensive evaluation of fiducial marker systems to date. Finally, section 4 draws the main conclusions.

2. Fiducial markers systems

This section explains the main features of the different fiducial marker systems employed in this comparison. An individual analysis is provided, highlighting their strengths and weaknesses.

2.1. QR Codes

QR Codes (Quick Response Codes) are two-dimensional barcodes invented in 1994 by the Japanese automotive company Denso Wave [24]. They have since become widespread in commercial and industrial environments, often appearing on product packaging to direct users to web links. However, due to their design, they are not typically used in camera tracking or 3D pose estimation applications, as they offer limited robustness against distortions and rotations compared to other systems discussed in this research.

The visual representation of a QR Code consists of black squares arranged in a grid pattern on a white background, facilitating efficient information encoding. Decoding QR Codes is widely supported on common hardware devices, such as smartphones, and numerous tools are available for generating these codes. The smallest units of a QR Code are referred to as "modules," and the appearance and resolution of the code vary based on the amount of data it contains. The patterns in the three corners of the code are always seven modules wide, with the number of modules between them increasing as the amount of data increases.

Despite their widespread popularity, QR Codes are rarely used in applications that require camera positionings, such as robotics, AR, and medical surgery. This is because they store a large amount of information, requiring high-resolution images to be decoded, making it difficult to do so from distances over 50 cm [7]. Additionally, solving the minimal form of the camera positioning problem [68] with just three points can result in up to four possible solutions, which can only be unambiguously determined for low noise levels.

Multiple open-source libraries have implemented additional modules to detect this type of marker in images in real-time, such as VISP[40] or OpenCV [8]. In this work, we have used the latest.

2.2. ARToolKit

ARToolKit [31] is considered one of the first AR software development kits (SDKs). The system uses black-bordered

[7] `https://docs.opencv.org/4.x/de/dc3/classcv_1_1QRCodeDetector.html` [last access 09/07/2022]

[8] `https://docs.opencv.org/4.x/de/dc3/classcv_1_1QRCodeDetector.html` [last access 09/07/2022]

square markers with a user-defined image inside for unique identification. Figure 2 shows three markers of the ARToolKit system, with the default internal image containing the icon "Hiro", named after its creator Hirokazu Kato from the Nara Institute of Science and Technology [30].

The first stage of the ARToolKit detection method involves extracting the borders of the markers through morphological operations on a binary thresholded image. ARToolKit locates the borders by finding connected groups of pixels in the thresholded image, forming square shapes. Once the border is extracted, the homography matrix is calculated from its four corners, yielding the canonical marker image, which is a frontal view of the marker without perspective distortion. Finally, the canonical image is compared to a library of known markers to identify the valid ones.

While ARToolKit allows using visually attractive custom markers, its detection method is prone to errors and not very robust in varying lighting conditions, as demonstrated in later studies [66].

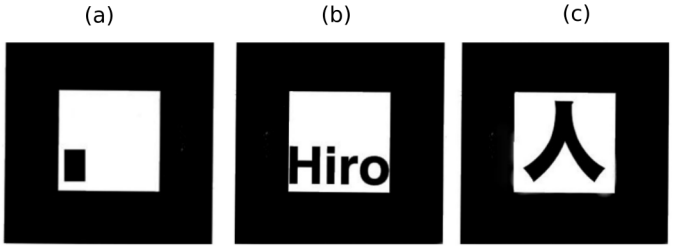


Figure 2: **ARToolKit markers**. Each marker has a unique identification, which is its internal picture.

2.3. ARToolKit+

ARToolKit+ [66] is an evolution of the ARToolKit SDK [31], which is free and open-source. In chronological order, ARToolKit was developed first, followed by ARTag [19] and then by ARToolKit+. Each system is inspired by the previous one. However, only ARToolKit and ARToolKit+ are available for download and use in commercial products.

ARToolKit+ adopts the marker design of ARTag, i.e., a binary pattern of 6×6 bits encoding a unique ID (Figure 1c). However, a novel aspect of ARToolKit+ is using an automatic image thresholding algorithm to detect borders. The thresholding value is dynamically computed using information from the previous frame if a marker was detected. However, a random thresholding value is employed when no markers are found. This approach frees the user from manually setting a threshold, as in ARTag.

The ARToolKit+ source was completely rewritten in C++ with a cleaner and more intuitive design. The ARToolKit+ memory management improves its predecessor, allowing the simultaneous detection of more than one marker per image without affecting its computational cost. Finally, as a novel aspect, ARToolKit+ offers a public and fully accessible plugin for Unity [9]. This is useful for programmers to integrate ARToolKit into their AR applications in a 3D world.

2.4. ArUco

ArUco [21] is probably the most reliable open-source library for real-time fiducial marker detection. By providing the camera calibration and the marker size, the ArUco system can detect markers in digital images and return the position of its corners, the ID of each marker, and the camera location. The ArUco library is a part of the OpenCV as an additional module [10]. In this work, we have used the latest.

Markers must be composed of an external black border and an inner region to encode the binary pattern, which is unique and identifies each marker. Notably, the ArUco library is versatile, allowing the detection of other markers such as ARToolKit+, ChiliTags, and AprilTag. However, in [21], it is recommended to use the ARUCO_MIP_36h12 dictionary, an optimal dictionary generated using Mixed Integer Linear Programming algorithms [22].

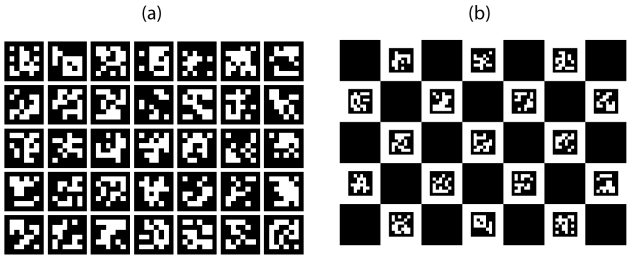


Figure 3: **ArUco features**. (a) *ArUco Boards*. (b) *ChArUco*.

ArUco Boards and ChArUco are two main features that make the library popular. Firstly, an ArUco Board (as seen in Figure 3(a)) consists of a set of markers located in known relative positions to each other, providing a unique reference system. This is useful in certain situations because even if several markers are occluded, their positions can still be estimated based on the visible markers. This leads to a more precise and stable camera pose estimation relative to the center of the board.

[9] `http://www.arreverie.com/blogs/getting-started-with-artoolkit-unity-plugin/` [last access 09/07/2022]

[10] `https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html` [last access 03/01/2023]

Secondly, ChArUco (as seen in Figure 3(b)) combines traditional chessboards with ArUco Boards. The ArUco markers are used to interpolate the position of the chessboard corners, making it more versatile than marker boards by allowing for occlusions or partial views. Furthermore, the interpolated corners belong to a chessboard, providing more accurate subpixel accuracy.

2.5. ArUco3

ArUco3 is the latest version of the ArUco library and is maintained and updated by its core developers. It includes several improvements over the original ArUco.

One of the key improvements in ArUco3 is the ability to detect fiducial markers, which are designed to maximize speed while maintaining accuracy and robustness [54]. This method takes advantage of the increasing camera resolution to detect markers in a smaller version of the image, resulting in a faster detection process. The system also uses multi-scale image rendering to estimate marker corner positions with sub-pixel accuracy in the original image, providing dynamic adaptation for maximum performance."

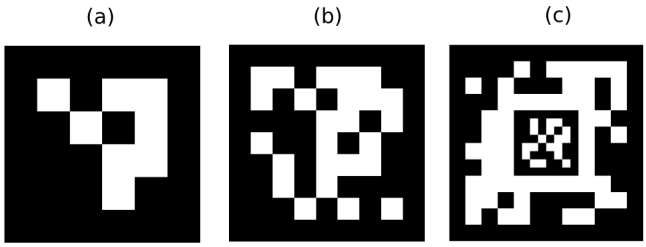


Figure 4: **ArUco3 markers**. (a) Marker from dictionary `DICT_4X4_50`. (b) Marker from dictionary `DICT_7X7_1000`. (c) Fractal Marker

.

ArUco addresses common problems with marker detectors, such as sensitivity to motion blur and partial occlusion. To improve the detection speed, ArUco3 employs a method of detecting fiducial markers designed to maximize speed while preserving accuracy and robustness. The system utilizes multi-scale image rendering to estimate marker corner positions with sub-pixel accuracy. ArUco3 also introduces the Fractal Marker to alleviate the partial occlusion problem and increase the range of distances at which a single marker can be detected.

ArUco3 also presents a tracking method based on Discriminative Correlation Filters, which allows for tracking markers under strong motion blur conditions. Additionally, a marker map, which consists of a set of markers placed in any 3D position with respect to each other, can be obtained through a method that solves the Shape from Motion problem applied to markers. This allows for printing a set of markers, placing them in an environment, and obtaining a 3D map indicating their position. A calibrated camera can be positioned in the environment by looking at any marker.

2.6. AprilTag

The AprilTag marker system [49] allows the detection of binary square markers similar to those of ARTag, ARToolKit+, or ArUco.

Figure 5: **AprilTag markers**. (a) Marker from *Tag36h11* dictionary. (b) Marker from *Tag16h5* dictionary.

The detection process involves several stages: the search for linear segments, detection of squares, calculation of the position and orientation of the tag, and decoding the internal code. Square detection uses a recursive 4-level depth search, where the tree adds a side of the square at each level. At the identification stage, the validity of the code within the discovered marker is verified. To encode an internal picture, AprilTag employs a lexicode system, characterized by two parameters: the number of codewords (internal pattern) bits and the minimal Hamming distance between any two codes. The lexicode generates tags' codes, enabling the detection and correction of bit errors. AprilTag has several marker families that differ in two parameters: the number of bits used for encoding and the minimal Hamming distance. For instance, Tag16h5 Figure 5 (a) represents a 16-bit marker (4×4 array) with a minimum Hamming distance of 5 bits between any two codes; Tag36h11 Figure 5 (b) refers to a 36-bit marker (6×6 array) with a minimum Hamming value of 11 bits between any two codes.

Although we have outlined the AprilTag method, it has not been tested in our experiments as it is currently considered an abandoned project. However, it serves as the basis for the development of AprilTag2 and AprilTag3.

2.7. AprilTag2

The AprilTag 2 method improves its previous version by reducing false positives, using adaptive thresholding, continuous boundary segmentation, faster decoding, and edge refinement. Additionally, it achieves better performance on small images and allows for decimated input

images, resulting in significant speed gains. The detection process involves the following steps: searching for quad shapes in the image, filtering to select squares, estimating the position and orientation of the tag, and decoding the identification code. Once the marker is identified, the camera's location in the environment is calculated.

The design of AprilTag 2 markers is similar to ArUco markers, with black and white squares and an inner region that encodes unique identifiers. The most commonly used marker detection dictionary is called Tag36h10. Unlike the original AprilTag, AprilTag 2 introduces a more robust method of generating markers, guaranteeing a minimum Hamming distance between tags under all possible rotations. This marker generation process, which uses a lexicode-based method with minimum complexity heuristics, has been shown to reduce false-positive rates [67].

2.8. AprilTag3

AprilTag3 is an improved system based on the previous version that includes a faster detector, improved detection rate on small tags, and flexible marker design. The improvements aim to satisfy the most demanding needs of the users.

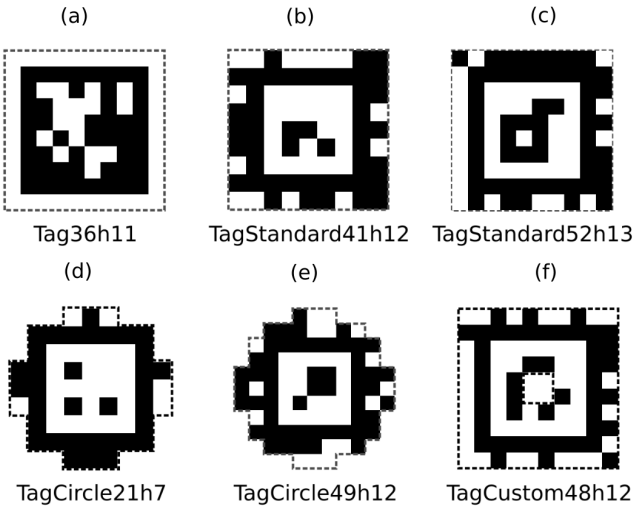


Figure 6: **AprilTag3 markers**. AprilTag3 supports all pregenerated families. Depending on the use case, it is recommended to use a particular family.

The most novel feature of AprilTag3 is the flexible layout markers [35] in contrast to the classic layout supported in AprilTag2. The marker's data bits can now go outside the marker border, increasing data density and the number of possible markers at the cost of decreasing detection distance. It is also possible to define layouts with "holes" inside the marker border. For example, this can be used for drone applications by placing different-sized tags inside each other to allow detection over a wide range of distances.

Given the large number of marker dictionaries supported by AprilTag3 (see Figure 6), it is advisable to use the `TagStandard41h12` family in most cases (Figure 6(b)). However, depending on the specific needs of each application, the developers recommend the following marker families. If a high number of uniquely identified markers are needed, use `TagStandard52h13` Figure 6 (c). If it is needed to maximize space usage in a small circular object, use `TagCircle21h7` or `TagCircle49h12` Figure 6 (d,e). If it is required to make a recursive marker, it uses `TagCustom48h12` Figure 6(f). For compatibility with the ArUco detector, use `Tag36h11` Figure 6(a).

2.9. STag

STag [3] employs squared markers with a circular pattern inside (as shown in Figure 7). The encoding consists of a total of 48 black-and-white circular bits. The main contribution of the STag method is the precision obtained in the homography calculation, which increases the accuracy of camera pose estimation. The detection process starts by identifying squares in the image using the EDPF algorithm [1]. The candidate list is then validated. Once the marker has been validated and recognized in the image, a refining step is performed to fit the inner circular edge as an ellipse, from which the homography is calculated.

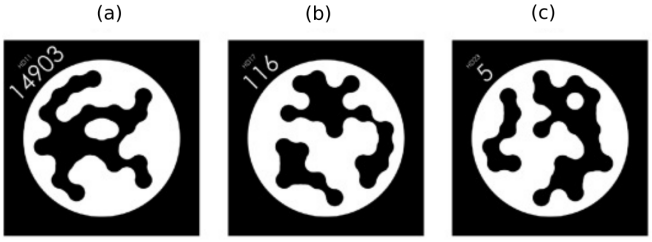


Figure 7: **STag markers**. Each marker has a unique identification, which is its internal circular codification.

2.10. TopoTag

TopoTag [73] introduces a fiducial marker system based on topological and geometrical patterns by constructing binary trees. Unlike previous systems that search for the marker's outline, TopoTag offers an alternative approach to locating binary topological patterns after performing robust threshold filtering on the image. Geometrical information is employed to decode and identify markers. Regarding the marker designs (as seen in Figure 8), there is no shape constraint, meaning both the internal and external regions can be freely customized as long as the desired internal topological structure is maintained [10].

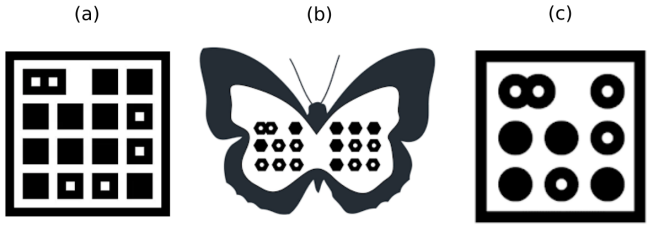


Figure 8: **TopoTag markers**. TopoTag supports customized marker designs with different shapes, e.g., squares, circles, and hexagons.

As a unique feature, in contrast to traditional marker systems that only use four points (the marker corners) for camera pose estimation, TopoTag utilizes a total of 16 feature points (the nodes of the binary tree) to enhance the precision of the method.

2.11. DeepTag

DeepTag [74] is a deep learning-based general framework that supports the detection of multiple marker families. Unlike other systems, DeepTag does not propose a specific marker design. Instead of relying on edge detection and image binarization, it uses Convolutional Neural Networks (CNNs) [60] to regress key points and digital symbols from local shapes directly.

The method consists of three stages. Firstly, regions that contain markers are identified. Secondly, another network refines the region to locate the marker corners precisely. Finally, the marker's internal region is analyzed to determine if it is indeed a marker.

One advantage of this technology over previous works is its ability to detect a wide range of marker types (e.g., ArUco, AprilTag, TopoTag). However, a drawback of CNNs is that they require a prior training step with at least 70,000 images containing markers placed at different distances and angles of view [74]. Additionally, they have high computational demands.

2.12. ChromaTag

ChromaTag [14] proposes a design based on the AprilTag markers created by Edwin Olson. As a novel aspect, the creators of ChromaTag encode data using multiple colors in the LAB color space. The L channel represents the illumination level in the image, while the A and B channels store colors ranging from green to red and blue to yellow, respectively. It is worth noting that the LAB color space is the closest to human vision.

In 2015, ChromaTag introduced its first marker dictionary, called *36H11*. It was based on the AprilTag marker template and used a combination of red and yellow colors for internal coding and blue and orange colors for the external part. However, the markers with high-intensity colors led to poor detection quality, leading to a design change. In 2017, ChromaTag proposed a new design, also based on AprilTag markers, with green color gradients on the outer grid and red color gradients on the inside. This resulted in a new marker dictionary named *16H5*.

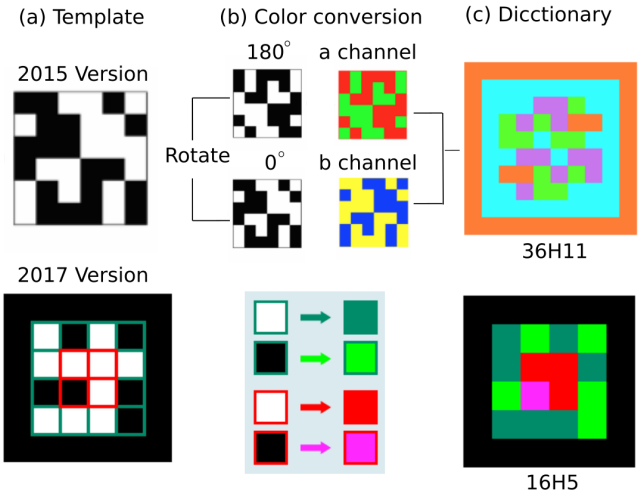


Figure 9: **ChromaTag markers**. It uses colored markers designs as an alternative to black and white traditional markers. Two marker families are displayed: 16H5 and 36H11.

Figure 9 shows the generation process of the two families of ChromaTag markers mentioned above (36H11 and 16H5). As a starting point, both families use an AprilTag tag template Figure 9 (a). Afterward, a color remapping Figure 9 (b) is performed where the black and white markers are transformed into multichromatic markers. Note that the 36H11 marker family is not based solely on the RGB color scheme, but uses the LAB color space for high color gradients. Thus, Figure 9 (c), shows the final design of the two families of ChromaTag markers.

The main advantage of ChromaTag is its ability to encode more information than traditional markers. It enables the creation of dictionaries with a larger number of markers or smaller dictionaries with a substantial Hamming distance between them.

2.13. Jumarker

Jumarker [27] presents a method for designing, detecting, and tracking customized markers as an alternative to the traditional black-and-white square marker design. The main contribution of this work is to offer designers the possibility of creating visually appealing and customized markers, making them suitable for use in commercial environments where appearance is important. Figure 10(a-c) shows some custom marker designs created by Jumarker authors.

Jumarker offers the freedom to design marker templates and automatically generate a set of uniquely identified markers while maintaining a common visual appearance. Templates can be created using any vector graphics tool, as long as a series of rules are followed. Each marker encodes a unique identifier and a Cyclic Redundancy Check code (CRC) [39], using two different colors, to avoid false positives. The designer chooses the position and colors for these bits.

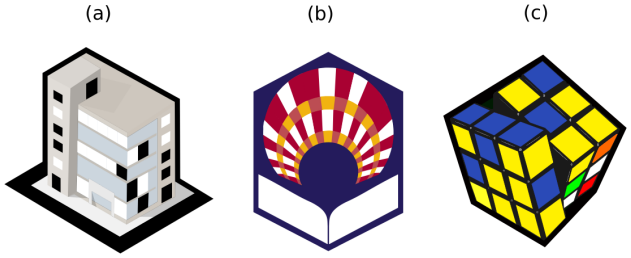


Figure 10: **Jumarker markers**. Jumarker presents customized marker designs as an alternative to squared black and white traditional markers.

The detection steps are similar to most of the marker systems explained above. The external border is first detected, and then the candidates are evaluated by analyzing the colors of the inner pixels. The detection robustness of such markers varies depending on the design. The marker outline's shape and the size designed for the inner region containing the binary bit combination are the most important aspects to be taken into account. The system can estimate the pose of a calibrated camera using the border corners.

In contrast to the performance of black and white square markers, custom markers present some difficulty in being detected at long distances. The marker contour design is limited to polygons, not supporting curved contours.

2.14. VuMark

VuMark is a commercial AR platform developed by the Vuforia company [11]. It allows for detecting, locating, and identifying real objects using only a normal camera. As an alternative to traditional markers, it proposes using customized markers that follow basic rules to ensure uniqueness, detectability, and data encoding capabilities. The software development kit (SDK) supports integration into users' applications, and the SDK is available on the Vuforia website. Note: The method for customized marker detection by VuMark is not publicly disclosed.

Vuforia offers an Adobe Illustrator plugin for designing markers. The markers encode unique IDs or data into the image, icon, or logo, which serves as a common identifier across a product range and represents the company's brand identity. Figure 11(a-c) displays some customized markers designed by VuMark authors. VuMark main contribution is storing the marker templates in an online database. Users can integrate the VuMark marker detection into their own applications through their public SDK.

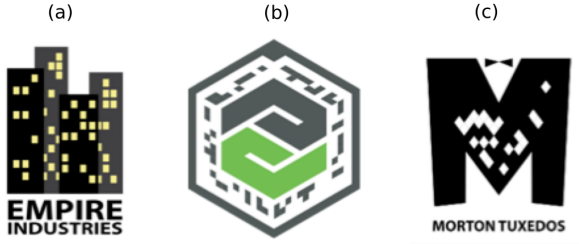


Figure 11: **VuMark markers**. VuMark markers can be customized to reflect a company's brand identity closely.

However, the methodology for customized marker detection in VuMark is not publicly disclosed as it is a private project with no technical information. Nevertheless, we have used the public SDK available on their website to evaluate the performance of VuMark.

3. Experiments

This section presents the experiments conducted to evaluate the performance of each fiducial marker system considered in this work (as depicted in Figure 1). The experiments were computed on a single CPU, an Intel Core™ i7-10510U 2.30GHz, with 8 GB RAM, running the Ubuntu 20.04.1 operating system. The following sections provide a structure for the presentation of results. Section 3.1 presents the experimental setup in which all experiments were conducted. Section 3.2 analyzes the sensitivity and specificity of the evaluated systems, while Section 3.3 analyzes the accuracy of each system in estimating the camera position with respect to the marker, given different viewing angles and distances. Section 3.4 analyzes the vertex jitter, which refers to the noise in each system's estimation of the marker corners' location. Section 3.5 evaluates the ability of each system to detect markers under different levels of occlusion. Finally, Section 3.6 presents the computing speed of the systems and 3.7 discusses the experiments' limitations. It is worth mentioning that all methods were observed to be invariant to marker rotation around the optical axis. As a result, we did not include these experiments in the paper.

[11] `https://developer.vuforia.com/` [last access 09/07/2022]

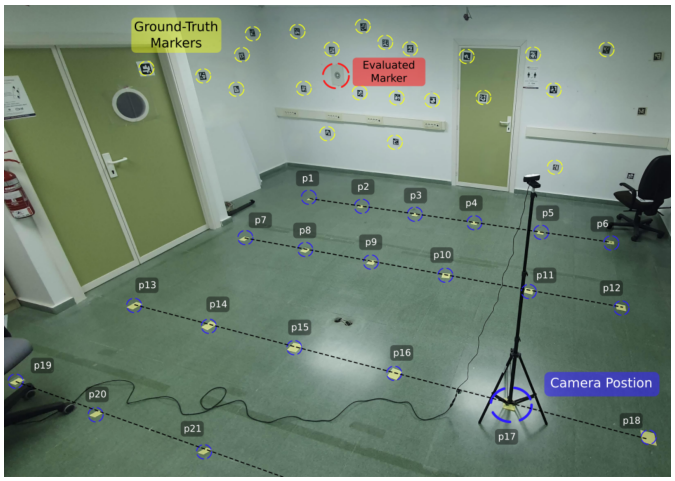


Figure 12: **Laboratory overview**. The marks on the floor ($p1$ to $p24$) show the locations employed for recording. The evaluated marker (in red) was placed in front of the camera location surrounded by 19 markers (yellow) used to estimate the ground-truth camera position using the method [44]

.

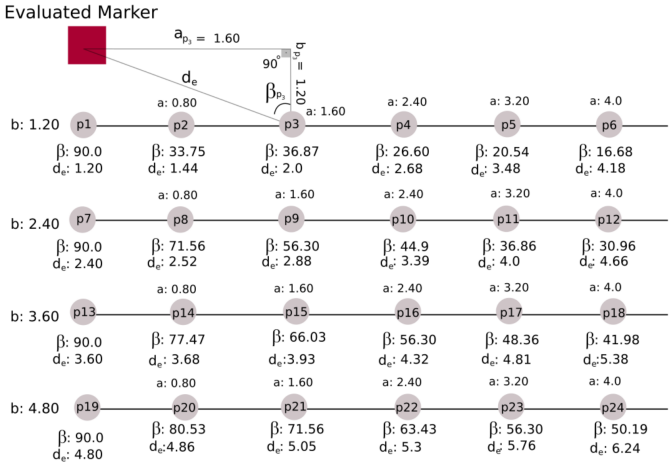


Figure 13: **Distances and viewing angles**. A representation of the distance and viewing angles of each camera location in the experimentation labs.

3.1. Experimental Set Up

Figure 12 shows the laboratory where the experiments were conducted. To ensure a fair comparison, the markers of the evaluated systems (shown in Figure 1) were printed in the same size (10×10 cm). In the laboratory, a total of 24 camera locations (designated as $p1$ to $p24$) were selected, each with a unique distance and viewing angle in relation to the evaluated marker. The distance ranges from 1.2 to 6.2 meters, and the viewing angle ranges from 20^o to 90^o, with a fixed light intensity of 200 lux and a fixed height of 1.5 meters for both the camera and marker.

At each position (from $p1$ to $p24$), we left the camera static on a tripod and recorded a sequence for each marker. In our recordings the camera does not move, instead, we replace the markers making sure they all have their center at the same location. So, for each location, a total of thirteen video sequences were recorded. In total, 312 video sequences with a resolution of 1920×1080 were captured, each sequence having on average 1246 frames. All the video sequences recorded are publicly available [12].

The method proposed in [44] is used as the ground truth to estimate the relative position of the camera with respect to the evaluated marker. This method involves placing ArUco3 markers in the environment and capturing images of them. The method maps the three-dimensional marker locations from these images, which can then be used to estimate the camera's position. As seen in Figure 12, the evaluated marker is surrounded by 19 visible ArUco3 markers (yellow circles) in the camera recordings. The ground truth method provides accurate camera location information based on the ArUco3 markers in the images, but it requires at least four markers to be visible for good accuracy.

3.2. Sensitivity and Specificity analysis

This experiment aims to analyze the true positive rate (TPR) of each system, with a focus on understanding their performance based on the distance and angle with respect to the marker being detected. To do so, we use video sequences $p1$ to $p24$. Additionally, this section also analyzes the false positive rate. For this analysis, we use a total of 968 images from the ImageNet database [15] which depict various scenes from everyday life but contain no markers, and we run the systems on these images to determine the number of false positives detected.

Figure 14 (a) shows the TPR of each system as a function of the distance between the camera and the marker. As expected, the TPR decreases with increasing distance. The results indicate that QR Code and ChromaTag are the worst-performing systems. For distances greater than 2 meters, they are unable to detect markers. They are followed by VuMark and Jumarker, which experience a substantial reduction in TPR (about 30%) for distances greater than 2 meters. The maximum distance at which both systems can identify markers is 3 meters, as shown in our experiments. The systems ARToolKit, TopoTag, ARToolKit+, and DeepTag obtain very similar results, maintaining stable detection (greater than 95%) for distances less than 1.5 meters. Beyond this distance, the TPR progressively decreases to 65.34%, 68.21%, 73.98%, and 75.62%, respectively, in the worst case (3 meters). AprilTag2, AprilTag3, and STag maintain a TPR of around 75% up to 2 meters. The maximum distance at which

[12] `https://tinyurl.com/2nm5z4b3` [last access 09/07/2022]

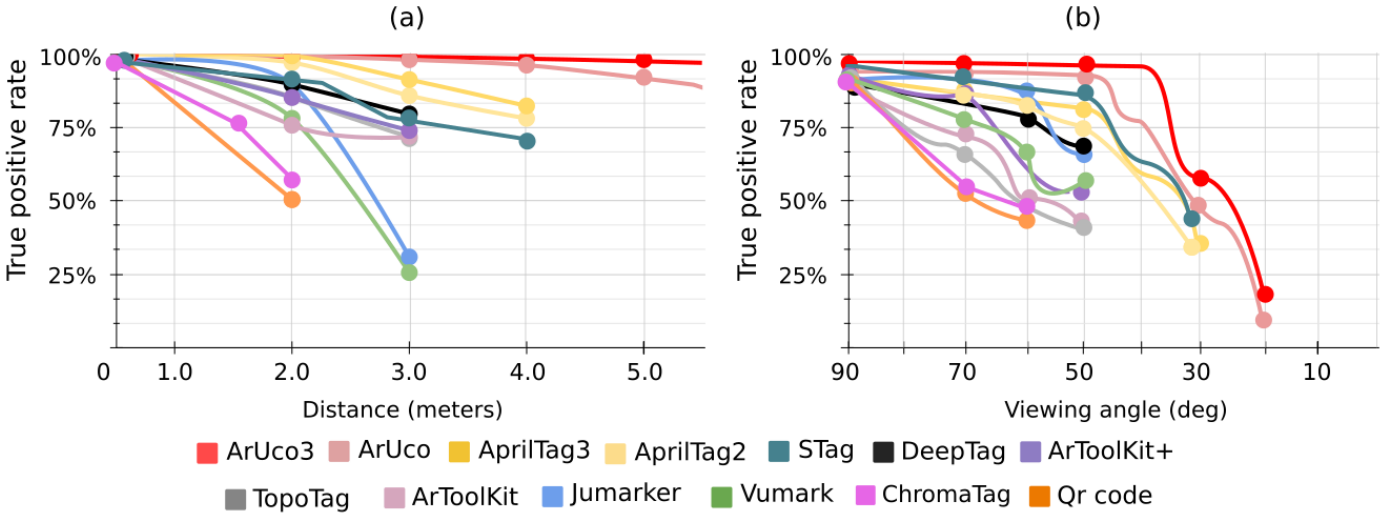
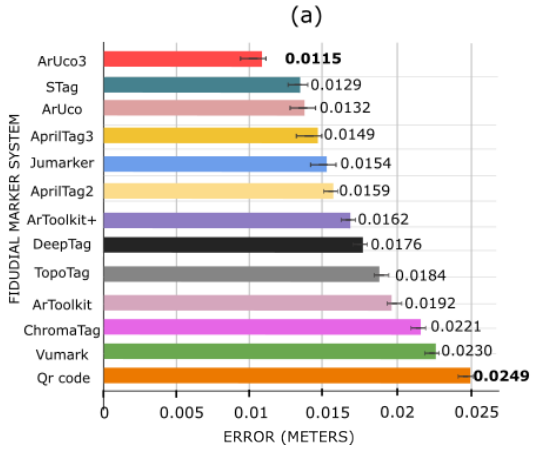


Figure 14: **True Positive rate.** (a) TPR as a function of the distance and (b) TPR as a function of camera angle w.r.t the marker.



(b)

	ArUco3	STag	ArUco	AprilTag3	Jumarker	AprilTag2	ArToolKit+	DeepTag	TopoTag	ArToolKit	Vumark	ChromaTag	Qr Code
ArUco3	X	12.5	20.5	18.9	4.5	9.2	2.1	4.8	7.5	2.3	1.2	1.4	1.2
STag	12.5	X	**70.4**	22.3	15.6	23.5	4.5	12.6	8.9	1.3	2.6	4.1	5.3
ArUco	20.5	**70.4**	X	**74.7**	25.4	12.3	7.1	8.2	12.7	4.2	5.4	6.9	5.2
AprilTag3	18.9	22.3	**74.7**	X	**82.1**	34.2	8.6	12.5	16.4	2.1	8.2	5.9	4.1
Jumarker	4.5	15.6	25.4	**82.1**	X	50.4	7.1	14.25	22.5	4.1	12.5	2.4	7.2
AprilTag2	9.2	23.5	12.3	34.2	50.4	X	6.8	12.3	21.2	5.3	6.1	4.8	6.1
ArToolKit+	2.1	4.5	7.1	8.6	7.1	6.8	X	**80.5**	30.5	37.9	22.5	9.4	7.2
DeepTag	4.8	12.6	8.2	12.5	14.25	12.3	**80.5**	X	41.8	35.8	24.8	12.5	8.5
TopoTag	7.5	8.9	12.7	16.4	22.5	21.2	30.5	41.8	X	**73.8**	29.5	11.6	6.2
ArToolKit	2.3	1.3	4.2	2.1	4.1	5.3	37.9	35.8	**73.8**	X	16.8	39.8	12.6
Vumark	1.2	2.6	5.4	8.2	12.5	6.1	22.5	24.8	29.5	22.3	X	25.6	16.8
ChromaTag	1.4	4.1	6.9	5.9	2.4	4.8	9.4	12.5	11.6	39.8	25.6	X	14.6
Qr Code	3.8	5.3	5.2	4.1	7.2	6.1	7.2	8.5	6.2	12.6	16.8	14.6	X

Figure 15: **Average error in camera pose estimation.** (a) Average error of each evaluated system. (b) *W-values* of the non-parametric Wilcoxon test. [69], highlighting the non-significant results.

these systems can detect markers is 4 meters, with TPRs of 78.11%, 82.35%, and 78.124% for AprilTag2, AprilTag3, and STag, respectively. Lastly, ArUco3 and ArUco are capable of detecting markers up to a distance of 5.5 meters while maintaining a rate of over 95% in all tests.

Figure 14 (b) shows the TPR of each system as a function of the viewing angle of the camera w.r.t the marker. Note that for angles near 90^o, all systems obtain their highest TPR. As the viewing angle decreases, each method progressively reduces its TPR. In this case, QR Code and ChromaTag are again the worst-performing systems, with a TPR that decreases to 50% for angles close to 65^o. When the angle is less than, 60^o this system cannot identify the marker. Regarding TopoTag, ARToolKit, VuMark, and ARToolKit+, we can see that the results are similar. They all reduce their TPR by about 15% for every 10^o, unable to identify the marker for angles greater than 50^o. DeepTag, Jumarker, AprilTag2, AprilTag3, and STag maintain a high TPR (greater than 75%) up to angles greater than 60^o. However, while DeepTag and Jumarker are unable to detect the marker for angles less than 50^o, AprilTag2, AprilTag3, and STag can identify markers up to angles close to 30^o. Finally, ArUco and ArUco3 are the best-performing systems. Both can detect, although with a low true positive rate (less than 25%) markers with angles close to 20^o. Note that when the angle is greater than, 50^o the rate is higher than 95% at all times.

Finally, it is worth mentioning that no false detection has been obtained after using 968 images from the ImageNet dataset as input for the detection methods. Thus, the false positive rate of the methods is zero in our tests.

3.3. Markers systems accuracy

This experiment evaluates the precision of each system in estimating the camera pose. As previously stated, 19 markers (represented as yellow circles in Figure 12) were

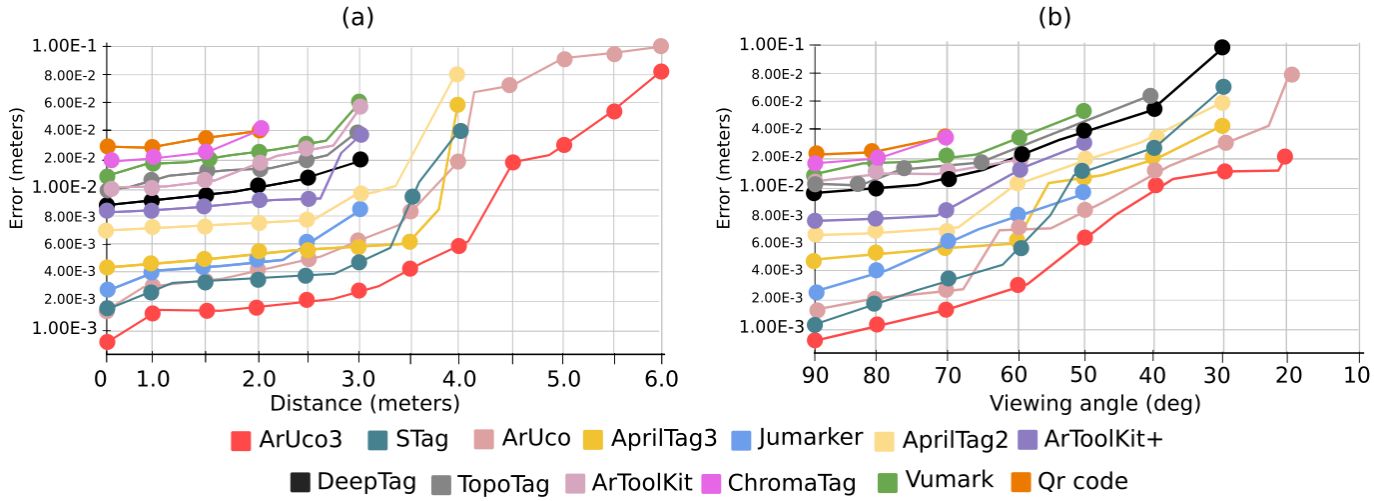


Figure 16: **Errors in camera pose estimation.** (a) Error as a function of the distance between the marker and the camera, (b) Error as a function of the camera viewing angle.

used to estimate the ground-truth camera pose, following the method described in [44]. Then, the error of each system was calculated as the difference between the ground truth and the system's estimation.

Figure 15 (a) displays the average error of each system across all analyzed frames. ArUco3 is the best method, while QR Codes perform the worst. A non-parametric Wilcoxon test [69], with a confidence level of 0.99, was conducted to analyze the differences between the methods. 24 video sequences were captured for each method, so the critical value for W at $N = 24$ with $p < 0.01$ is 61. Figure 15 (b) shows the *W-values*, highlighting the non-statistically significant results (*W-values* greater than 61). As observed, the differences between ArUco3 and the rest of the methods are statistically significant. Hence, we can conclude that ArUco3 is the best method for camera pose estimation in these experiments.

To analyze the errors in each method for estimating the camera pose, Figure 16 considers the distance and viewing angle. As shown in Figure 16 (a), the error is plotted as a function of the distance between the camera and marker, ranging from 1.0 to 6.0 meters. ArUco3 and QR Code exhibit the lowest and highest errors, respectively. ArUco3 has the highest error of 0.08 meters at a distance of 6 meters. It is noteworthy that only ArUco3 and ArUco are able to detect the marker and estimate the camera pose when the marker is located beyond 4 meters. The other systems can only estimate the pose up to a maximum distance of 3 to 4 meters, with the exception of QR Code which requires the marker to be within 2 meters.

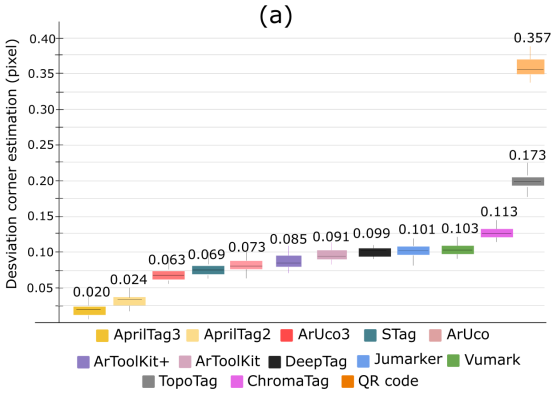
Figure 16 (b) depicts the error as a function of the camera's viewing angle with respect to the marker, within a range of 90^o to 10^o. The error increases as the viewing angle decrease. Note that none of the systems can estimate the camera pose for angles less than 20^o. As in the previous case, ArUco3 and QR Code show the lowest and highest error, respectively, with values of 0.15 meters and 0.04 meters. ArUco3 is followed by ArUco, with an error of less than 0.08 meters in its worst-case scenario (at 20^o)

3.4. Vertex jitter analysis

Vertex jitter refers to the inaccuracy in estimating the position of the marker corners. Since the corners are employed to estimate the camera pose, its precision is relevant, especially in AR applications. Small changes in the corner locations in a video sequence lead to an unpleasant "shaking" effect on 3D objects rendered on top of the scene. Although this is not a significant problem in most robotic applications, it is for some AR applications. For that reason, most marker systems implement an algorithm to estimate the corner's locations with sub-pixel accuracy.

Figure 17(a) displays each system's average, minimum, and maximum errors. The ground-truth corner location was estimated as the average observed position along the whole video sequence, and the standard deviation provides the measured error (as in [21]). As observed, AprilTag3 and QR Code are the systems with the lowest and highest error, with 0.020 and 0.357 pixels respectively. Except for TopoTag and QR Code, all systems have an error of less than 0.11 pixels.

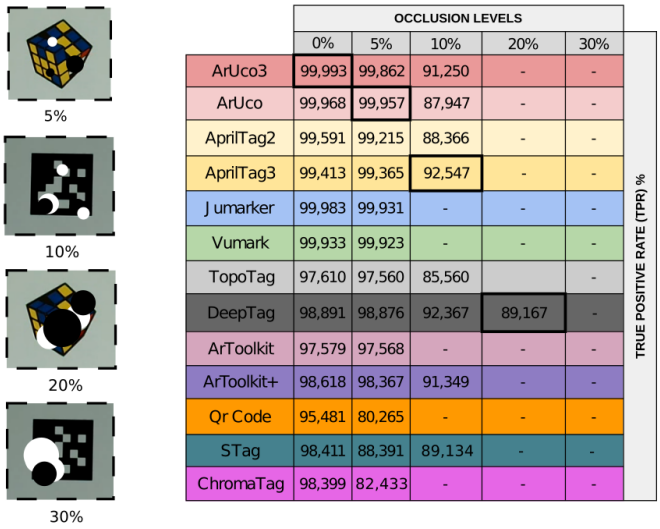
Figure 17(b) shows the results of the Wilcoxon [69] test using a confidence level of 0.99. The results show that while the differences between AprilTag3 and AprilTag2 are not statistically significant, the differences with the rest of the methods are. We conclude then that the AprilTags methods are the best in this experiment.



(b)

	AprilTag3	AprilTag2	ArUco3	STag	ArUco	ArToolKit+	ArToolKit	DeepTag	Jumarker	Vumark	ChromaTag	TopoTag	Qr Code
AprilTag3	X	72.3	25.6	16.5	13.6	9.5	5.6	5.3	4.2	2.1	1.8	1.5	0.8
AprilTag2	72.3	X	22.5	13.2	11.4	8.2	4.5	4.3	3.2	2.0	1.5	1.2	1.2
ArUco3	25.6	22.5	X	62.2	65.6	32.5	12.3	9.8	7.6	5.2	4.8	3.5	2.1
STag	16.5	13.2	62.2	X	65.6	45.2	32.5	12.5	8.6	6.5	5.3	4.2	2.6
ArUco	13.6	11.4	65.6	65.6	X	75.6	42.3	26.8	12.6	8.6	7.1	5.6	4.2
ArToolKit+	9.5	8.2	32.5	45.2	75.6	X	78.5	41.2	33.4	25.5	12.5	5.1	4.8
ArToolKit	5.6	4.5	12.3	32.5	42.3	78.5	X	63.5	64.8	36.5	22.5	8.9	5.6
DeepTag	5.3	4.3	9.8	12.5	26.8	41.2	63.5	X	62.6	46.5	42.4	9.4	5.8
Jumarker	4.2	3.2	7.6	8.6	12.6	33.4	64.8	62.6	X	55.6	43.9	11.5	6.2
Vumark	2.1	2.0	5.2	6.5	8.6	25.5	36.5	46.5	55.6	X	56.2	12.5	6.4
ChromaTag	1.8	1.5	4.8	5.3	7.1	12.5	22.5	42.4	25.8	56.2	X	13.5	6.7
TopoTag	1.5	1.2	3.5	4.2	5.6	9.8	16.7	9.4	11.5	12.5	13.5	X	6.8
Qr Code	1.2	0.8	2.1	2.6	4.2	6.2	5.6	2.5	6.2	6.4	6.7	6.8	X

Figure 17: **Jittering errors.** (a) Average, minimum, and maximum deviations in marker corner estimation for each evaluated system. (b) *W-values* of the non-parametric Wilcoxon test [69].



(b)

	OCCLUSION LEVELS 0%	5%	10%	20%	30%
ArUco3	99,993	99,862	91,250	-	-
ArUco	99,968	99,957	87,947	-	-
AprilTag2	99,591	99,215	88,366	-	-
AprilTag3	99,413	99,365	92,547	-	-
Jumarker	99,983	99,931	-	-	-
Vumark	99,933	99,923	-	-	-
TopoTag	97,610	97,560	85,560		-
DeepTag	98,891	98,876	92,367	89,167	-
ArToolkit	97,579	97,568	-	-	-
ArToolkit+	98,618	98,367	91,349	-	-
Qr Code	95,481	80,265	-	-	-
STag	98,411	88,391	89,134	-	-
ChromaTag	98,399	82,433	-	-	-

TRUE POSITIVE RATE (TPR) %

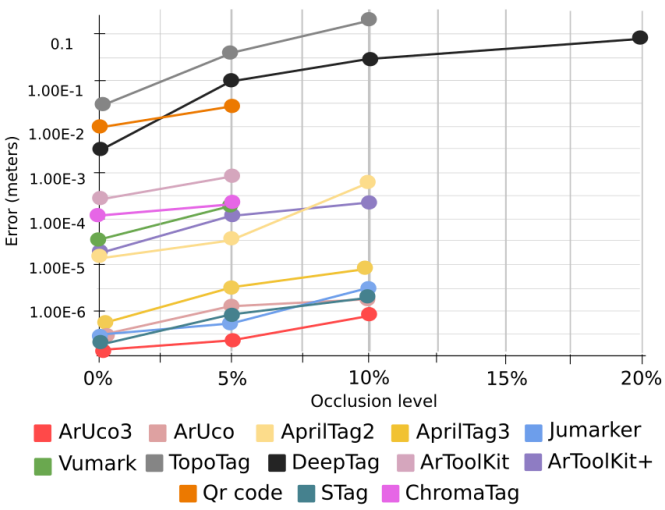


Figure 18: **Performance under occlusion.** (a) Representative examples of the images employed in the experiment. (b) True Positive Rate under occlusion. (c) The camera poses error estimation under different occlusion levels.

3.5. Markers detection under occlusion

This experiment aims to analyze each system's capability of detecting markers under occlusion. In this case, only the video sequences captured at p_1 (as shown in Figure 12) have been used. The other recording locations are too far from the cameras to be employed in this experiment.

The method used to test occlusion is identical to the one described in [54]. It involves placing black and white circles on the marker area, providing precise measurements of the amount of occlusion applied to each image. In our study, occlusion levels ranging from 0% to 30% of the total marker area were applied, resulting in a total of 125 synthetic images per marker. Figure 18(a) displays some examples of the images used in this experiment

Figure 18(b) presents the True Positive Rate (TPR) of each system in detecting markers under different levels of occlusion. The best method for each level is highlighted. The systems can be divided into three groups based on the results obtained. The first group consists of QR Code, ARToolKit, VuMark, and ChromaTag, which are unable to detect markers when the occlusion level exceeds 5%, with a detection rate of 80.265%, 99.023%, 99.923%, and 82.123%, respectively. The second group comprises ArUco3, ArUco, AprilTag2, AprilTag3, Jumarker, TopoTag, and ARToolKit+. These systems can detect markers under 10% occlusion levels, except for ArUco, TopoTag, and STag, which have TPRs of 87.94%, 85.560%, and 82.433%, respectively. The rest of the systems in this group have TPRs above 90%. Finally, in the third group, DeepTag is found to be the system with the highest resistance to occlusions, with a TPR of 89.167% even when the occlusion rate is 20%

Figure 18(c) shows the camera pose errors of the systems under different levels of occlusions. Each system estimates

the camera position using the ground truth and synthetic image, and the translational error between the two poses is used as the error measure. TopoTag has the highest error among the systems and ArUco3 has the lowest error. When the occlusion level is 10%, TopoTag reaches an error of more than 0.1 meters, while ArUco3 always has an error of fewer than 0.1 millimeters in the worst case.

In conclusion, if detecting the marker under occlusion percentages higher than 10% is essential, DeepTag is the only system capable of doing so. On the other hand, if a minimum error in camera position estimation under low occlusion levels (less than 10%) is necessary, ArUco3 would be the most advisable system to use.

3.6. Computing speed

Detecting markers in real time is crucial in both commercial and industrial environments. The aim of these experiments is to analyze the computing speed of each system. Figure 19 shows the average speed when employing images with a resolution of 1920×1080 pixels.

As shown, ArUco3, ArUco, Jumarker, and STag have the highest frame rates among the systems. In contrast, DeepTag, QR Code, and AprilTag2 have the lowest frame rate. However, it is noteworthy that the ArUco family almost doubles the speed of the other methods.

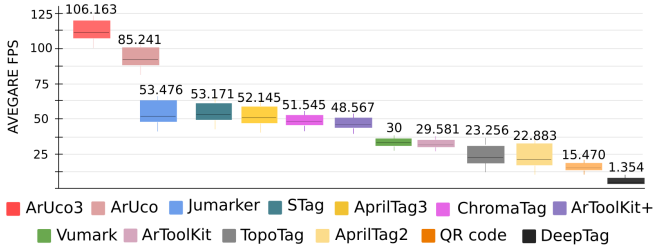


Figure 19: **Computing speed**. Average FPS of each system in processing a frame of 1920×1080 pixels.

Based on the results, it can be concluded that marker design does not significantly impact computing speed. Despite using similar black-and-white markers by ArUco, AprilTag2, AprilTag3, and ARToolKit+, there is significant variation in the detection speed. This variation is attributed to the differences in each system's detection algorithms and internal code optimizations.

3.7. Threats to validity

This section highlights the limitations of the experiments in this paper. The primary constraint is that we only evaluated the performance using static images. Our experiments did not consider the effect of motion blur on the performance of the methods, due to the difficulty in conducting controlled and reproducible experiments with identical testing conditions for all systems. To address this aspect, it would have been necessary to use a device similar to a robotic arm, capable of repeating programmed movements while holding the camera or marker. Instead, our tests used both static cameras and markers to ensure identical experimental conditions. Despite this, we acknowledge that comparing the performance of the systems under movement is a worthy aspect to investigate in future work.

Nevertheless, fiducial marker systems are widely used in numerous applications where the camera is in continuous motion, such as robotics, UAVs, augmented reality, and autonomous driving [56, 5, 70, 38]. In these cases, various techniques can be employed to reduce or eliminate image blur, such as reducing camera exposure time, using global shutter cameras, smoothing camera movement with a stabilizer, or utilizing software methods to track the marker under blurred conditions [55]. However, in instances where these techniques cannot be employed, blur may pose a problem that would prevent the use of markers, and therefore, the results of this work may not be applicable.

Finally, with regard to the occlusion experiment, it is acknowledged that synthetic occlusion may not always reflect some real-world scenarios. However, to our knowledge, it is the only way to consistently replicate experiments and obtain the maximum levels of occlusion that each method can handle. Through this approach, the ability of each method to detect the marker under varying levels of occlusion is demonstrated. In the subsequent section, we will summarize the performance of each method and provide recommendations on the most suitable approach for different use cases.

4. Conclusions

This paper has evaluated the sensitivity, specificity, accuracy, jittering, occlusion resiliency, and speed of thirteen fiducial marker systems. Figure 20 shows a summary of the results obtained. In each table, we present the rank of each method in the different experiments carried out. In light of the results obtained, we can draw the following conclusions.

Firstly, ArUco3 ranks highest in accuracy, detection distance, detection angle, and speed, followed by ArUco. However, the main drawback of ArUco3 is the jittering error, for which AprilTag3 performs best. As a result, we recommend AprilTag3 for Augmented Reality applications due to its lower shaking effect. It is important to note that AprilTag3 estimates camera position using only a single marker, whereas ArUco3 uses multiple markers (boards, fractal markers, and marker mapping) to extend

(a)

Pose estimation accuracy		
Rank	Method	Error (m)
1º	ArUco3	0,0115
2º	STag	0,0129
3º	ArUco	0,0132
4º	AprilTag3	0,0149
5º	Jumarker	0,0154
6º	AprilTag2	0,0159
7º	ArToolKit+	0,0162
8º	DeepTag	0,0176
9º	TopoTag	0,0184
10º	ArToolKit	0,0192
11º	ChromaTag	0,0221
12º	Vumark	0,023
13º	Qr Code	0,0249

(b)

Marker detection distances			
Rank	Method	Max (m)	TPR (%)
1º	ArUco3	6.0	99.8
2º	ArUco	5.5	98.9
3º	AprilTag3	4.0	82.3
4º	AprilTag2	4.0	80.1
5º	STag	4.0	78.1
6º	DeepTag	3.0	75.6
7º	ArToolKit+	3.0	73.9
8º	TopoTag	3.0	68.2
9º	ArToolKit	3.0	65.3
10º	Jumarker	3.0	26.4
11º	Vumark	3.0	25.0
12º	ChromaTag	2.0	52.2
13º	Qr Code	2.0	50.0

(c)

Marker detection angles			
Rank	Method	Min (deg)	TPR (%)
1º	ArUco3	20	22.2
2º	ArUco	20	18.6
3º	STag	30	45.6
4º	AprilTag3	30	42.7
5º	AprilTag2	30	38.4
6º	Jumarker	50	70.6
7º	DeepTag	50	71.5
8º	ArToolKit+	50	55.4
9º	Vumark	50	57.7
10º	ArToolKit	50	43.7
11º	TopoTag	50	42.8
12º	ChromaTag	60	51.2
13º	Qr Code	60	45.2

(d)

Jittering errors		
Rank	Method	Error (m)
1º	AprilTag3	0,020
2º	AprilTag2	0,024
3º	ArUco3	0,063
4º	STag	0,069
5º	ArUco	0,073
6º	ArToolKit+	0,085
7º	ArToolKit	0,091
8º	DeepTag	0,099
10º	Jumarker	0,101
11º	Vumark	0,103
12º	ChromaTag	0,113
11º	TopoTag	0,173
13º	Qr Code	0,357

(e)

Markers detection under occlusions			
Rank	Method	Max level (%)	Error (m)
1	DeepTag	20	8,00E-02
2º	ArUco3	10	5,00E-07
3º	STag	10	1,20E-06
4º	ArUco	10	1,31E-06
5º	Jumarker	10	2,35E-06
6º	AprilTag3	10	5,65E-06
7º	ArToolKit+	10	5,65E-04
8º	AprilTag2	10	8,65E-04
9º	TopoTag	10	1,65E-04
10º	Vumark	5	1,65E-04
11º	ChromaTag	5	2,65E-04
12º	ArToolKit	5	8,65E-04
13º	Qr Code	5	7,65E-02

(f)

Computing Speed		
Rank	Method	FPS
1º	ArUco3	106.1
2º	ArUco	85.2
2º	Jumarker	53.4
3º	STag	53.1
4º	AprilTag3	52.1
5º	ChromaTag	51.4
6º	ArToolKit+	48.5
7º	Vumark	30.0
8º	ArToolKit	29.5
9º	TopoTag	23.5
10º	AprilTag2	22.8
11º	Qr Code	15.4
12º	DeepTag	1.3

Figure 20: **Summary of the experiments**. Each table shows the rank of the different methods tested.

the tracked area beyond a single marker. If the application requires tracking in larger areas than a single marker, ArUco3 is a better option. Secondly, the results indicate that, in general, non-customized markers perform better than customized ones. However, if custom markers are necessary for industrial and commercial applications with a focus on aesthetics, we recommend using JuMarker over Vumark as it performed better in all tests. If resistance to occlusion is a key requirement and high speed is not necessary, we recommend DeepTag, which can detect markers with occlusion levels up to 20%. It is important to note that QR codes were not designed for camera pose estimation, but rather for encoding a large amount of information such as URLs. They require high image resolution for detection and lack an external black border to aid in detection and improve pose estimation, which contributes to their relatively poor performance in our experiments.

Declarations

Funding and/or Conflicts of interests/Competing interests

This project has been funded under the Industrial Ph.D. Program of Córdoba University with Seabery R&D, Project 1380047-F UCOFEDER-2021 of Andalusia and Project PID2019-103871GB-I00 of Spanish Ministry of Economy, Industry and Competitiveness, and FEDER.

Some authors of this work are also the authors of the ArUco, ArUco3, and Jumarker libraries. Nevertheless, this

has not affected the impartiality of the tests conducted. All systems and datasets employed in this paper are public so that other researchers can reproduce our results.

References

[1] Cuneyt Akinlar and Cihan Topal. Edpf: A real-time parameter-free edge segment detector with a false detection control. *International Journal of Pattern Recognition and Artificial Intelligence*, 26(01):1255002, 2012.

[2] Bradley Atcheson, Felix Heide, and Wolfgang Heidrich. CALTag: High Precision Fiducial Markers for Camera Calibration. In Reinhard Koch, Andreas Kolb, and Christof Rezk-Salama, editors, *Vision, Modeling, and Visualization (2010)*, 2010.

[3] Burak Benligiray, Cihan Topal, and Cuneyt Akinlar. Stag: A stable fiducial marker system. *Image and Vision Computing*, 89:158–169, 2019.

[4] Filippo Bergamasco, Andrea Albarelli, Emanuele Rodolà, and Andrea Torsello. Rune-tag: A high accuracy fiducial marker with strong occlusion resilience. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 113–120, 2011.

[5] Mahathi Bhargavapuri, Animesh Kumar Shastry, Harsh Sinha, Soumya Ranjan Sahoo, and Mangal Kothari. Vision-based autonomous tracking and landing of a fully-actuated rotorcraft. *Control Engineering Practice*, 89:113–129, 2019.

[6] Su Cai, Xu Wang, and Feng-Kuang Chiang. A case study of augmented reality simulation system application in a chemistry course. *Computers in Human Behavior*, 37:31–40, 2014.

[7] Lilian Calvet, Pierre Gurdjos, Carsten Griwodz, and Simone Gasparini. Detection and accurate localization of circular fiducials under highly challenging conditions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[8] Jan Čejka, Fabio Bruno, Dimitrios Skarlatos, and Fotis Liarokapis. Detecting square markers in underwater environments. *Remote Sensing*, 11(4):459, 2019.

[9] Juan Chen, Caiming Sun, and Aidong Zhang. Autonomous navigation for adaptive unmanned underwater vehicles using fiducial markers. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9298–9304, 2021.

[10] Enrico Costanza and John Robinson. A region adjacency tree approach to the detection and design of fiducials. In *1st International Conference on Vision, Video, and Graphics (VVG)*, pages 63–69, 2003.

[11] Ajaya Kumar Dash, Santosh Kumar Behera, Debi Prosad Dogra, and Partha Pratim Roy. Designing of marker-based augmented reality learning environment for kids using convolutional neural network architecture. *Displays*, 55:46–54, 2018. Advances in Smart Content-Oriented Display Technology.

[12] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.

[13] Joseph DeGol, Timothy Bretl, and Derek Hoiem. Chromatag: A colored marker and fast detection algorithm. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1481–1490, 2017.

[14] Joseph DeGol, Timothy Bretl, and Derek Hoiem. Chromatag: A colored marker and fast detection algorithm. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1472–1481, 2017.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[16] Naser El-Sheimy and You Li. Indoor navigation: state of the art and future trends. *Satellite Navigation*, 2(1), may 2021.

[17] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

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

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

[20] D. Galvez-López and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, Oct 2012.

[21] 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*, 47(6):2280–2292, 2014.

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

[23] Lionel Heng, Benjamin Choi, Zhaopeng Cui, Marcel Geppert, Sixing Hu, Benson Kuan, Peidong Liu, Rang Nguyen, Ye Chuan Yeo, Andreas Geiger, Gim Hee Lee, Marc Pollefeys, and Torsten Sattler. Project autovision: Localization and 3d scene perception for an autonomous vehicle with a multi-camera system. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4695–4702, 2019.

[24] Denso Corp Toyota Central R&D Labs Inc. Two-dimensional code. In *JP Patent JP2938338B2*.

[25] Michelle Iocolano, Seth Blacksburg, Todd Carpenter, Michael Repka, Susan Carbone, Gizem Demircioglu, Maryann Miccio, Aaron Katz, and Jonathan Haas. Prostate fiducial marker placement in patients on anticoagulation: Feasibility prior to prostate sbrt. *Frontiers in Oncology*, 10, 2020.

[26] David Jurado, Juan M. Jurado, Lidia Ortega, and Francisco R. Feito. Geuinf: Real-time visualization of indoor facilities using mixed reality. *Sensors*, 21(4), 2021.

[27] David Jurado-Rodríguez, Rafael Muñoz-Salinas, Sergio Garrido-Jurado, and Rafael Medina-Carnicer. Design, detection, and tracking of customized fiducial markers. *IEEE Access*, 9:140066–140078, 2021.

[28] Michail Kalaitzakis, Brennan Cain, Sabrina Carroll, Anand AAmbrosi, Camden Whitehead, and Nikolaos Vitzilaios. Fiducial markers for pose estimation. *Journal of Intelligent & Robotic Systems*, 101, 2021.

[29] Martin Kaltenbrunner and Ross Bencina. Reactivision: A computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, page 69–74, 2007.

[30] 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)*, pages 85–94, 1999.

[31] I Poupyrev H Kato, Mark Billinghurst, and Ivan Poupyrev. Artoolkit user manual, version 2.33. *Human Interface Technology Lab, University of Washington*, 2, 2000.

[32] Shehryar Khattak, Christos Papachristos, and Kostas Alexis. Marker based thermal-inertial localization for aerial robots in obscurant filled environments. In *Advances in Visual Computing*, pages 565–575. 2018.

[33] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

[34] Manfred Klopschitz and Dieter Schmalstieg. Automatic reconstruction of widearea fiducial marker models. In *In ISMAR*, pages 1–4. IEEE Computer Society, 2007.

[35] Maximilian Krogius, Acshi Haggenmiller, and Edwin Olson. Flexible layouts for fiducial tags. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2019.

[36] Christian Kunz, Vera Genten, Pascal Meißner, and Björn Hein. Metric-based evaluation of fiducial markers for medical procedures. In Baowei Fei and Cristian A. Linte, editors, *Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling*, volume 10951, pages 690 – 703. International Society for Optics and Photonics, 2019.

[37] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník. Artificial intelligence for long-term robot autonomy: A survey. *IEEE Robotics and Automation Letters*, 3(4):4023–4030, Oct 2018.

[38] Vincent Lepetit, Pascal Fua, et al. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends® in Computer Graphics and Vision*, 1(1):1–89, 2005.

[39] Bin Li, Hui Shen, and David Tse. An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check. *IEEE communications letters*, 16(12):2044–2047, 2012.

[40] Éric Marchand, Fabien Spindler, and François Chaumette. Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics & Automation Magazine*, 12(4):40–52, 2005.

[41] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[42] Rafael Muñoz-Salinas, Manuel J. Marín-Jimenez, and R. Medina-Carnicer. Spm-slam: Simultaneous localization and mapping with squared planar markers. *Pattern Recognition*, 86:156–171, 2019.

[43] Rafael Muñoz-Salinas, Manuel J. Marín-Jimenez, and R. Medina-Carnicer. SPM-SLAM: Simultaneous localization and mapping with squared planar markers. *Pattern Recognition*, 86:156 – 171, 2019.

[44] Rafael Muñoz-Salinas, Manuel J. Marín-Jimenez, Enrique Yeguas-Bolivar, and R. Medina-Carnicer. Mapping and localization from planar markers. *Pattern Recognition*, 73:158–171, 2018.

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

[46] Mohammad Nahangi, Adam Heins, Brenda McCabe, and Angela Schoellig. Automated localization of uavs in gps-denied indoor construction environments using fiducial markers. In Jochen Teizer, editor, *Proceedings of the 35th International Symposium on Automation and Robotics in Construction (ISARC)*, pages 88–94, Taipei, Taiwan, July 2018. International Association for Automation and Robotics in Construction (IAARC).

[47] Leonid Naimark and Eric Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings. International Symposium on Mixed and Augmented Reality*, pages 27–36. IEEE, 2002.

[48] Michael Neunert, Michael Blösch, and Jonas Buchli. An open source, fiducial based, visual-inertial state estimation system. *arXiv preprint arXiv:1507.02081*, 2015.

[49] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407, May 2011.

[50] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[51] Alexander Reuter, Hans-Peter Seidel, and Ivo Ihrke. Blurtags: spatially varying psf estimation with out-of-focus patterns. In *20th International Conference on Computer Graphics, Visualization and Computer Vision 2012, WSCG'2012*, pages 239–247, 2012.

[52] Lisanne S Rigter, Eva C Rijkmans, Akin Inderson, Roy PJ van den Ende, Ellen M Kerkhof, Martijn Ketelaars, Jolanda van Dieren, Roeland A Veenendaal, Baukelien van Triest, Corrie AM Marijnen, et al. Eus-guided fiducial marker placement for radiotherapy in rectal cancer: feasibility of two placement strategies and four fiducial types. *Endoscopy International Open*, 7(11):E1357–E1364, 2019.

[53] Michael Rohs and Beat Gfeller. Using camera-equipped mobile phones for interacting with real-world objects. In *Advances in Pervasive Computing*, pages 265–271, 2004.

[54] Francisco J. Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76:38–47, 2018.

[55] Francisco J. Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Tracking fiducial markers with discriminative correlation filters. *Image and Vision Computing*, 107:104094, 2021.

[56] Eric Royer, Maxime Lhuillier, Michel Dhome, and Jean-Marc Lavest. Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision*, 74(3):237–260, 2007.

[57] Artur Sagitov, Ksenia Shabalina, Roman Lavrenov, and Evgeni Magid. Comparing fiducial marker systems in the presence of occlusion. In *2017 International Conference on Mechanical, System and Control Engineering (ICMSC)*, pages 377–382, 2017.

[58] Hamid Sarmadi, Rafael Muñoz-Salinas, M. Álvaro Berbís, Antonio Luna, and R. Medina-Carnicer. 3d reconstruction and alignment by consumer rgb-d sensors and fiducial planar markers for patient positioning in radiation therapy. *Computer Methods and Programs in Biomedicine*, 180:105004, 2019.

[59] Junaed Sattar, Eric Bourque, Philippe Giguere, and Gregory Dudek. Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction. In *Fourth Canadian Conference on Computer and Robot Vision (CRV'07)*, pages 165–174. IEEE, 2007.

[60] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[61] Karam Shaya, Aaron Mavrinac, Jose Luis Alarcon Herrera, and Xiang Chen. A self-localization system with global error reduction and online map-building capabilities. In *Intelligent Robotics and Applications*, pages 13–22. Springer, 2012.

[62] Garrett Thomas, Melissa Chien, Aviv Tamar, Juan Aparicio Ojea, and Pieter Abbeel. Learning robotic assembly from cad. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3524–3531, 2018.

[63] Sumit Tiwari. An introduction to qr code technology. In *2016 international conference on information technology (ICIT)*, pages 39–44. IEEE, 2016.

[64] A. Torii, J. Sivic, M. Okutomi, and T. Pajdla. Visual place

recognition with repetitive structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(11):2346–2359, Nov 2015.

[65] Athanasios Tsoukalas, Anthony Tzes, and Farshad Khorrami. Relative pose estimation of unmanned aerial systems. In *2018 26th Mediterranean Conference on Control and Automation (MED)*, pages 155–160, 2018.

[66] Daniel Wagner and Dieter Schmalstieg. Artoolkitplus for pose tracking on mobile devices. In *IEEE International Workshop on Haptic Audio Visual Environments and Their Applications*, pages 147–152, 2005.

[67] John Wang and Edwin Olson. AprilTag 2: Efficient and robust fiducial detection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

[68] Ping Wang, Guili Xu, Zhengsheng Wang, and Yuehua Cheng. An efficient solution to the perspective-three-point pose problem. *Computer Vision and Image Understanding*, 166:81–87, 2018.

[69] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. 1992.

[70] Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid, and Juan Tardós. A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems*, 57(12):1188–1197, 2009.

[71] Tatsuya Yamada, Takehisa Yairi, Suay Halit Bener, and Kazuo Machida. A study on slam for indoor blimp with visual markers. In *ICCAS-SICE, 2009*, pages 647–652. IEEE, 2009.

[72] Shichao Yang, Yu Song, Michael Kaess, and Sebastian. Scherer. Pop-up SLAM: Semantic monocular plane slam for low-texture environments. In *International Conference on Intelligent Robots and Systems (IROS) 2016*, pages 1222–1229, 2016.

[73] Guoxing Yu, Yongtao Hu, and Jingwen Dai. TopoTag: A Robust and Scalable Topological Fiducial Marker System. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 27(9):3769–3780, 2021.

[74] Zhuming Zhang, Yongtao Hu, Guoxing Yu, and Jingwen Dai. DeepTag: A General Framework for Fiducial Marker Design and Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.