DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

AGV (Autonomous Guided Vehicle) Positioning With Fiducials

Johannes Löbbecke

DEPARTMENT OF INFORMATICS

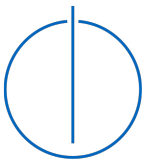
TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

AGV (Autonomous Guided Vehicle) Positioning With Fiducials

FTS (Fahrerloses Transportsysteme) Positionierung mit Markern

Author:	Johannes Löbbecke
Supervisor:	Prof. Dr. Stefanie Rinderle-Ma
Advisor:	Juergen Mangler
Submission Date:	15.09.2021



I confirm that this bachelor's thesis in information systems is my own work and I have documented all sources and material used.

Munich, 15.09.2021 Johannes Löbbecke

Acknowledgments

First of all, I would like to thank Prof. Dr. Stefanie Rinderle-Ma for giving me the opportunity to write this thesis at the chair of Information Systems and Business Process Management. I also wish to extend my sincere thanks to Juergen Mangler, my advisor, for all the hints and tips he provided. In particular, the help in formulating research questions and helping out with experimental setup was invaluable. Furthermore, Id like to acknowledge the assistance of Karolin Winter in the application process and deciding on a research topic. Finally, I am very grateful to my family and friends for all their help and support. Special thanks go to my father, my two brothers and my mother.

Abstract

Positional markers, also known as fiducial markers, have found widespread use for camera pose estimation. The camera's pose can be estimated at low cost and high speed by detecting fiducial markers while dealing with occlusion and distortions. The estimated pose can then be combined with other sensor data or directly be used for AGV navigation. The markers only have to be printed out, placed in the desired environment, and then detected.

However, a large variety of different marker types and different libraries for their detection exist. Choosing the correct marker type for the proper application can therefore be a time-consuming process. This thesis presents an overview of the different marker types and testing results using a generated test set of ArUco, Apriltag, and ARTag markers. Through the construction of prototype implementations, the project validated the effectiveness of the ArUco and the AprilTags libraries. Both libraries were tested and analyzed regarding detection time, accuracy and robustness.

Finally, the thesis attempted to apply the advantages of the 2D bar code Aztec Code to fiducial markers. For this purpose, multiple styles of combined markers were constructed and tested. This style of combined marker can be used to encode information and execute pose estimation using a single marker, however, the high robustness towards occlusion of Aztec Code could not be preserved through modification. All results were evaluated with a focus on applicability to AGV Navigation.

Contents

1 Introduction

1.1 Motivation

Optimization of the interface between autonomous transport vehicles and handling/loading robots, using both static programs, visual navigation, and machine learning solutions, has been the motivation behind countless research contributions [BOO08]. However, training machine learning solutions is complex and computationally expensive, while static programs are restricted in their applications. Meanwhile, Vision-based solutions rely on accurate localization in the environment, which is why it is imperative to design solutions, that can accurately determine the *pose (position and orientation)* of an *Autonomous Guided Vehicle (AGV)* as well as its different parts. Furthermore, to ensure that these solutions are applicable in low-volume, custom-built-to-order production scenarios, they need to be robust and straightforward enough to be usable by workers without a significant IT background.

Positional markers, also known as fiducial markers, are a cost-effective method for pose detection and have, therefore, found widespread use. However, various types of markers exist, and applying the correct type for the needed application requires significant knowledge.

As part of this Bachelors's Thesis, software prototypes were developed, which test the pose detection functionality of fiducial markers. In addition, the thesis presents the results of testing different types of markers regarding their accuracy, computational difficulty, and other aspects to give recommendations on when and how to use which type of marker. Finally, it was attempted to combine the advantages of matrix barcodes with fiducial markers and the approach was evaluated using the example of a combined marker consisting of Aztec Codes and Apriltags as well as slightly modified Aztec Code bar codes.

1.2 Research Questions

Positional markers have seen use in several different applications and their accurate detection is central to visual navigation. Several types of fiducial markers have been

proposed and should be compared. The goals of the thesis can therefore be summarized using the following research questions:

RQ1: Many scenarios can be identified, regarding the interface between robots and AGVs, ranging from full visual identification and dynamic handling of parts, to fully static static programming and gripping of parts. The feasibility of the scenario depends on (a) how exact an AGV can be positioned and (b) how consistent it can be positioned. If inconsistent positioning is considered the norm, which potential alternative solutions exist, and how can they be compared in terms of (1) implementation effort and (2) evolution effort when dealing with fast-changing product lines and small lot sizes?

RQ2: For existing AGV scenarios, what is a minimal set of properties (tilt, distortion, lighting, contrast, ...) defining these scenarios, and how do these properties affect the number, size, and fiducial required for successfully identifying the position of an AGV?

RQ3: How do different markers and libraries perform compared to each other? Can the advantages of Aztec Code (i.e., storing data in the marker) be used in fiducial marker-based applications?

Through answering these three research questions, the thesis aims to contribute in the following ways.

1.3 Contribution

Answering the first research questions results in three solution strategies for implementing the interface between a AGV navigation system and robot handling, by using fiducial markers. Fiducial Markers provide a basis for answering the other research questions.

The second research question contributes by establishing a set of properties as the basis for reviewing and evaluating related literature and tools. Furthermore, these properties can serve as a basis for evaluating new or existing real-world scenarios.

Finally, as part of the thesis, an extensive evaluation was conducted, to compare different types of tags and software implementations. The evaluation was carefully designed to automate the tests in order to achieve reproducible results for a wide ranging set of environmental parameters (lighting, occlusion, ...). This test set can be extended in the future to validate the effectiveness of, for example, the ArUco and Apriltags libraries.

While the results of research questions one and two help in making structured decisions, the research question three deals with a practical problem [Wie+06]. Therefore the *Design Science Research (DSR)* Method was applied throughout the research project.

1.4 Methodology

DSR is inherently a problem-solving process. A purposeful prototype or artifact is created to address a relevant problem [Hev+04; Wie10].

For the problem to be relevant, it has to be a previously unsolved and important business problem, while, for the prototype to be relevant, it has to be rigorously tested. The artifact, the foundations, and the methodologies are then the main research contributions of DSR[Hev+04].

Furthermore, DSR relies on the application of rigorous methods for both artifact construction and evaluation. Mathematical formalism is required to describe the constructed prototype, and the performance metrics must be measurable and comparable[Hev+04].

Finally, DSR needs to be an inherently iterative process with extensive communication of research. The design process can be described as a Generate/Test Cycle [Sim96], and both technology- and management-oriented audiences need to understand the research results[Hev+04].

1.5 Structure of the Thesis

Chapter 2 explains top concepts as they will be used throughout the thesis and introduces the strategies that are evaluated in later chapters. Chapter 3 presents related work while Chapter 4 presents an overview of current major marker types and libraries. Chapter 5 explains the experimental setup and Chapter 6 the results obtained. Finally. Chapter 8 will summarize the conclusions drawn and suggest some directions for future research.

2 Fundamentals

If the following terms are already familiar to the reader, it is suggested to skip to section 2.7.

1. **Pinhole Camera Model:** Widespread model used in Computer Vision
2. **Homogeneous Coordinates:** Alternative coordinate system to Cartesian Coordinates; widespread use in projective geometry and computer vision applications
3. **Camera Intrinsics:** Camera Matrix K, including focal lengths fx and fy and Optical Center coordinates cx and cy as well as the Distortion Coefficients $dst = [p1, p2, k1, k2, k3]$
4. **Camera Extrinsics:** Translation Vector $Tvec = [Tx, Ty, Tz]^T$ and Rotation Vector $Rvec = [Rx, Ry, Rz]^T$
5. **Squared Fiducial Markers:** Squared, white and black alternating markers that serve as points of measurement once captured into the image plane and can therefore be used to determine camera intrinsic and extrinsic parameters. Will be referred to as fiducial markers or markers throughout the thesis.
6. **Ambiguity Problem:** The problem that any pose estimation using just a single fiducial marker is subject to ambiguity. It gets alleviated through the use of Marker Maps
7. **Marker Maps:** Multiple markers placed in set locations. If set up well, their use ensures that more than one marker can always be used for pose estimation, therefore adding robustness through redundancy.

2.1 Pinhole Camera Model

Since navigation using fiducial markers is a visual navigation method, it relies on images captured by a camera. Therefore, it is essential to use a standard model, which allows for a mapping between pixels in the digital image and coordinates in the 3D world viewed by the camera. The model commonly used in computer vision and

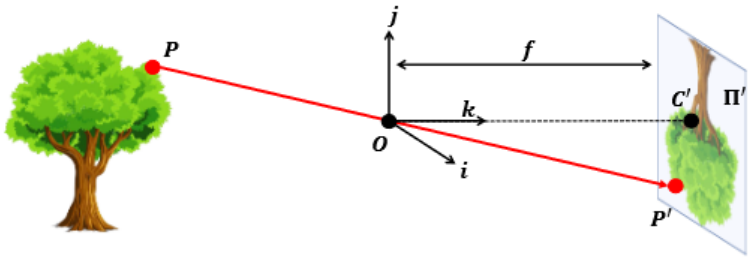


Figure 2.1: Pinhole Camera Model, Graphic by Kenji Hata and Silvio Savarese [HS]

applicable when using fiducial markers is the widespread pinhole camera model. The camera is modeled as a barrier with a small hole or aperture, through which light rays pass through and hit a planar film. In the ideal model case, the aperture is a single point. Therefore, a single ray will pass through the aperture for every point of the 3D objects facing the camera. Since only a single ray passes through for every point, it will project onto a single point on the film, resulting in the desired 1 to 1 mapping.

Formalizing this model, the film is referred to as the image or retinal plane, while the aperture is called the pinhole O or the center of the camera. Meanwhile, the focal length f describes the distance between the image plane and the pinhole O. If a point $P = [x, y, z]^T$ is visible to the camera, then it will project or map to a Point $P' = [x, y]^T$ in the image plane. Therefore we can build a coordinate system $[x, y, z]$ or $[j, i, k]$ in Figure 2.1 with its origin at the pinhole O and the z-axis being perpendicular to the image plane, therefore arriving at point C' or the projection of the Pinhole O in the image plane. The z-axis is also referred to as the optical axis, principal axis, or principal ray. This coordinate system is commonly referred to as the camera reference system or camera coordinate system. The relationship between a point in the image plane and the 3D point in the camera reference system can then be defined as:

$$P' = [x', y']^T = [f * x/z, f * y/z]^T$$

The same relationship can be explained with homogeneous coordinates with the same arbitrary points $P = [x, y, z, s]^T$ and $P' = [x', y', z']^T$, since with homogeneous coordinates, any point in the Euclidean plane $P' = [x', y']^T$ can be described by the set of homogeneous points $P' = [x, y, z]^T$ with z being a common, non zero factor. This allows for the use of matrix multiplication.

However, this simplified model makes several assumptions that do not hold for practical scenarios. The first assumption is that the aperture is only a single point that only allows a single light ray through it. However, in practice, multiple light rays will

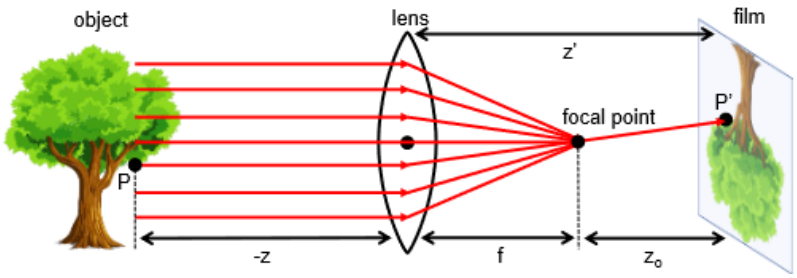


Figure 2.2: Pinhole Camera Model, Graphic by Kenji Hata and Silvio Savarese [HS]

pass from one 3D point through the aperture to different points in the image plane. While this increases the brightness of the image, it also causes the image to appear blurred. This problem gets mitigated through the use of lenses.

In an ideal scenario, the lens placed at the camera center or aperture causes all light rays emitted by a single 3D point to get refracted and then converge onto a single point in the image plane, reestablishing the desired 1 to 1 mapping. However, this property will not hold for every point in the captured 3D space. In particular, the lens will always refract all points P from a plane A, which is at a certain distance from the camera on to precisely one point. In contrast, points that are further away or closer to the camera than A will be mapped onto multiple points in the Image plane and appear more blurred the further away they are from A. This property is usually regarded as the focus or the depth of field at which a camera with a single lens will take clear images. However, modern cameras will usually have multiple lenses and include autofocus functionality that manipulates this distance.

Lenses add a critical property in the form of the focal point, therefore redefining the focal length f. This is because all light rays that travel parallel to the optical axis are converged by a lens onto a single focal point. Therefore, when lenses are added into the model, the focal length f now describes the distance between the focal point and the center of the lens.

However, the current model still disregards several assumptions:

1. Points in the image plane are usually in a different reference system from the actual pixels we capture in digital images, i.e., the origin of the pixel coordinate system is in a different corner than the origin of the model image plane.

2. Digital images consist of discrete pixels, while points in the image plane of the used model are continuous.

3. Cameras will usually add distortion due to imperfect sensors or other reasons determined during production.

In addition, a central aspect for computer vision applications is to be able to project points from reference systems with an arbitrary origin, rather than just the camera reference system. Therefore, more transformations are needed to map any 3D point to pixel coordinates in the image plane. The intrinsic and extrinsic camera parameters are commonly used to describe these.

2.2 Intrinsic Camera Parameters

The intrinsic camera parameters can be summarized using the camera matrix model. This camera matrix consists of:

Focal Lengths fx and fy: The focal lengths f of the camera lens. Normally expressed in pixels and are usually nearly equal.

Optical Center cx,cy: The optical center of the sensor. Also usually expressed in pixels.

With an ideal camera, a 3D point would then using these parameters project onto a point in the digital image plane according to the extended formula:

$$x' = (Xfx/Z) + cx; y' = (Yfy/Z) + cy.$$

To simplify the calculation when using homogeneous coordinates the intrinsic parameters are converted into a matrix form. A complete camera matrix K then has the following form:

$$K = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix}$$

However, as mentioned at the end of the previous section, camera lenses will usually add distortion to images, which can be seen through lines in images appearing bent. To counteract this distortion the intrinsic camera parameters also include a set of distortion coefficients $dst = [p1, p2, k1, k2, k3]$.

Camera matrix and distortion coefficients combined are known as the intrinsic camera parameters because they are unique and inherent to every camera. They depend on factors decided during manufacturing like the lens used. Determining these parameters for a given camera is commonly called camera calibration, and various methods for camera calibration exist. The thesis will present two different methods that were tested

in Chapter 5. It is important to note that while these intrinsic parameters are the same for a camera, regardless of the camera's position, they are unique for every lens used. Therefore, when using a modern camera that uses multiple lenses, commonly with an auto-focus function, the focus must either be locked or the camera re-calibrated for every lens if exact results are needed. That being said, ignoring recalibration for different camera foci can still lead to acceptable results for various applications, like *Augmented Reality (AR)* applications for end-user devices.

The above formula using the intrinsic parameters relies on knowing the 3D coordinates of a point in reference to the camera coordinate system. However, the goal is to project points from an arbitrary reference system, which is key to AGV navigation. Therefore, the model needs to be extended using what are called extrinsic camera parameters.

2.3 Extrinsic Camera Parameters

These extrinsic parameters are given through:

Translation Vector $Tvec = [Tx, Ty, Tz]^T$**:** Describes the 3D translation required to translate from the camera coordinate system into an arbitrary one

Rotation Vector $Rvec = [Rx, Ry, Rz]^T$**:** Describes the 3D rotation required to translate from the camera coordinate system into an arbitrary one

Therefore, the translation vector originates from a point in the camera's field of view and points directly at the camera, while the rotation vector describes the rotation of the point. Accordingly, calibration of extrinsic parameters is also often called camera pose estimation or pose estimation. Since the camera's extrinsic parameters describe 3D translation and rotation, they are commonly given in real-world units like millimeters or meters. This also means that they have to be re-calibrated every time the camera or the object that contains the origin point are re-positioned.

Therefore, the extrinsic parameters are determined by detecting objects with known ground truth information in the image plane. This is where fiducial markers are helpful since fiducial markers provide ground truth information about their structure.

Throughout the thesis, the term marker is often used as an abbreviation of fiducial marker.

2.4 Fiducial Markers

Fiducial markers are objects placed in the 3D space captured by a camera. Once captured in the camera's field of view, they serve as points of reference and measure. Since

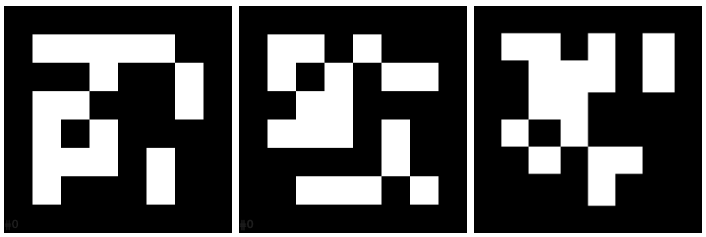


Figure 2.3: Example Markers with Id 0, left to right: ArTag, ArUco, Apriltag

the measurements of the fiducial markers are known, the extrinsic parameters can be estimated by comparing the pixel values with the given real-world measurements of the marker and accounting for the intrinsic camera parameters.

A fiducial marker can be any object that has known ground truth information. However, the use of planar, squared fiducial markers, as can be seen in Figure 2.3, has found widespread success for accurate estimation of extrinsic parameters at low cost. Planar, squared fiducial markers usually consist of a detection border around a unique pattern. The border is either black if the marker is placed in white or bright background or white if the marker is placed in black or very dark background. Since printing markers onto white background is more intuitive, black-bordered markers are used far more commonly. The pattern inside the marker ensures that markers can be differentiated and are unique towards rotation. The advantages of squared fiducial markers, here-forth referred to as simply fiducial markers or markers, are:

Reliable Detection: The square, white and black design means that markers can be detected through simple thresholding on grayscaled input images.

Dictionaries: By varying the pattern enclosed by the detection border as well as the markers size in bits, sets of unique markers, called dictionaries, can be generated. These dictionaries allow for various markers to be placed inside an environment without having to manually differentiate between the different markers.

Camera Calibration and Pose Estimation: Only the size of the printed markers and the camera's intrinsic parameters have to be known for markers to be usable for the determination of extrinsic parameters. In addition, they can be used for camera calibration by positioning markers in patterns like a chessboard pattern.

Theoretical Ease of Use: Markers only have to be printed out, measured, and placed into an environment. Thus, applications can be developed and used without significant computer vision background.

Cost: The use of fiducial markers is inexpensive since markers can be printed onto paper, and no specialized equipment is required to detect markers and accurately estimate the camera extrinsics.

However, an essential remark regarding pose estimation using fiducial markers is that pose estimation or estimation of the extrinsic parameters using a single marker is subject to ambiguity. This problem is commonly regarded as the ambiguity problem.

2.5 Ambiguity Problem and Marker Maps

The ambiguity problem is the result of a reflection of the plane about the z-axis of the camera. Therefore, it can be observed as a swap in the z value of the translation vector of a detected marker. While other approaches to counteract this ambiguity problem have been proposed [SP06; CB14], the thesis recommends using marker maps, which are composed of several markers. By determining the camera's pose using multiple markers, the ambiguity can be resolved through redundancy.

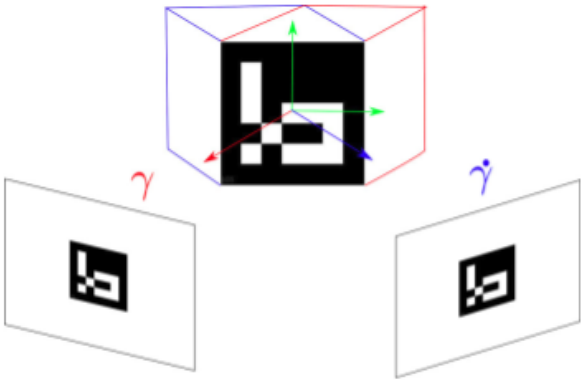


Figure 2.4: Ambiguity Problem: one projection for two camera poses, Graphic from ArUco Documentation[Gar+14; RMM18; Gar+15]

Using intrinsic and extrinsic camera parameters, a pipeline to project an arbitrary point from an arbitrary reference system in the 3D space into the digital image plane can be created.

2.6 Point Projection

The following pipeline is used similarly in different libraries, including *OpenCV* and *Robot Operating System (ROS)*:

Create Homogeneous Transformation Matrix M**:** Create 4x4 Matrix for translation using homogeneous coordinates out of translation and rotation vectors. A method is not presented in this thesis but can be done with all major computer vision libraries, and descriptions can be found in most books on computer vision and online [Dep].

Reference System Transformation: Transform the given but arbitrary point $P^g = (gx, gy, gz, 1)$ from the global reference system into the camera reference system by using the transformation matrix M as:

$$P^c = (Cx, Cy, Cz, 1) = M * (Gx, Gy, Gz, 1)$$

Project the point: Since the point is now in the camera reference system, it can be projected into the image plane using the equation presented before:

$$x' = (Cx * fx/Cz) + cx; y' = (Cy * fy/Cz) + cy.$$

As a reminder cx and fx as well as cy and fy can be found in the camera matrix.

Apply Distortion: As explained in the previous sections, the image plane is distorted due to the camera's intrinsic parameters, so the projected point coordinates still have to be distorted using the distortion coefficients $dst = [p1, p2, k1, k2, k3]$. The result will be the distorted pixel coordinates (x', y').

Using this functionality, several strategies for AGV navigation are considered.

2.7 AGV Navigation Strategies

AGV navigation can be modeled using three steps, as can be seen in the above figure. First, the AGV interacts with objects at position A, then it moves from position A to position B, and finally, it interacts with objects at position B. Object interaction is the same sub-process at both arbitrary positions A and B and can include processes like picking up objects, placing down objects, and manipulating object properties. All three steps can be navigated dynamically or statically. A fully static strategy would interact with objects in exact predetermined positions, follow an exact predetermined path, and again interact in exact predetermined positions. Such a fully static strategy, modeled below, has no advantage of using fiducial markers at any steps.

Therefore, the thesis considered three different AGV navigation strategies. The first is navigation between positions A and B using currently widespread static navigation while navigating object interaction dynamically using fiducial markers. The second is dynamic navigation for all three steps by detecting fiducials using an AGV mounted camera. Finally, the third strategy is dynamic navigation for all three steps by tracking an AGV *covered* in fiducials using multiple cameras. All three strategies have advantages depending on their application fields, which are summarized in the following sections.

2.7.1 Mixed Navigation: AGV Mounted Camera

Static navigation of movement between A and B and dynamic navigation of object interaction is the strategy where fiducial markers can be applied most easily and very effectively. A camera is placed on an AGV that is otherwise moving on predetermined paths or stationary. The AGV mounted camera is used to detect fiducial markers in the object interaction environments. The camera pose does not have to be repeatedly recalculated, but the position of objects in object interaction areas can be arbitrary since their pose can be estimated using the fiducial markers. A typical example would be a warehouse AGV that only moves among static paths but has to pick up objects placed arbitrarily in areas marked with fiducials and position them in other marked positions. Information, like the destination position, can also be encoded in markers through the use of combined markers, as is shown in Chapters 5 and 6.

2.7.2 Dynamic Navigation: AGV Mounted Camera

Dynamic navigation of all three steps using a single AGV mounted camera is the ideal navigation strategy, which in practice is difficult to implement for a real-time application and susceptible to errors. Markers need to be placed at set positions all over the environment, such that the AGV mounted camera can always detect at least one marker. Furthermore, the extrinsic parameters must be repeatedly determined in real-time since the camera's pose is constantly changing. Therefore, the resulting

system would be less robust and rely on a high-resolution camera and significant computing power. As a reminder, a camera can only always be intrinsically calibrated for one lens setting, so auto-focus functionalities can not be trivially applied.

A practical implementation of this strategy can be done similarly as work by Yibo Liu et al. [LSS21]. However this strategy needs to extensively consider problems like occlusion, the ambiguity problem, and marker placement. A more robust strategy is the Multi-View AGV tracking strategy.

2.7.3 Dynamic Navigation: Multi-View AGV Tracking

This strategy is heavily inspired by work by Rafael Munoz Salinas et al. [Sar+21b]. For this strategy, markers must be placed on the AGV while multiple cameras capture the environment. The cameras can be positioned arbitrarily as long as every camera share a portion of their field of view with at least one other camera. The cameras for this setup do not have to be performant since they only have to detect the markers once they enter their field of view. Therefore, to navigate in a large environment, only the amount of cameras has to be increased. In the paper mentioned above, 640/480 resolution cameras were used to track a cube and compare acquired results with the ground truth obtained through an extensive motion capture system. They were able to show that their method was able to track an object with millimeter accuracy in real-time.

Conclusively, the strategy can be used for dynamic AGV navigation. Of course, all strategies can be extended with other sensors.

3 Related Work

There exist countless contributions towards robot navigation, and for an overview, it is advised to refer to survey work as done by Bonin-Font et al. [BOO08]. This survey classifies research contributions under those belonging to computer vision and those belonging to computer control. Furthermore, it further divides proposed navigation strategies under map-based navigation and mapless navigation. Map-based navigation includes both metric map-based navigation and topological map-based navigation. AGV navigation using squared fiducial markers falls into the category of visual navigation and can, using the above classifications, be helpful for both map-based and mapless navigation approaches.

The most critical fiduciual marker dictionaries and libraries were subject of significant research contributions:

- **ArToolKit:** ArToolKit was the most popular library for using squared fiducial markers from roughly 1999 to 2010. It is based on multiple research contributions with work by Kato et al [KB99] being the first research contribution. Most of the original applications of ArToolKit were primarily focused on Augmented Reality. However, other application ideas were proposed and developed as well.

- **ArTag:** ArTags were originally proposed in a paper by Fiala [Fia05] in 2005 and further improved in 2010 [Fia04]. These contributions presented improvements on the markers and detection strategies proposed as part of ArToolKit.

- **Apriltag:** Apriltags are a popular fiducial marker system developed and published by the APRIL Robotics Laboratory at the University of Michigan. The system has been the subject of multiple research papers and improved existing marker types in robustness, efficiency, and flexibility [WO16; KHO19; Ols11]. Its advantages have been validated by several independent research contributions, some of which are introduced below.

- **ArUco:** The ArUco library was developed by Rafael Munoz Salinas et al. as the subject of multiple research papers [Gar+14; RMM18; Gar+15]. These contributions presented several improvements regarding performance aspects like robustness and detection time. The ArUco library is an especially useful contribution since, in addition to its own proposed dictionaries, it can be used with all

major marker types and has an extensive documentation. It was central to the success of this thesis, and is recommended as a starting point for any desired fiducial marker application.

Several other less popular marker types have also been proposed like RUNE-Tags [Ber+11], TopoTag [YHD20], CAL-Tag [AHH10], and more [Pea+21; Zha+21].

Several contributions have tested the performance of markers in different conditions and for different application types.

Sagitov et al. compare Apriltags, ArTags and CALtags in regards to resistance to occlusion and rotation [Sha+20]. Interestingly, they do not compare with the performance of the dictionaries proposed by ArUco, while using the ArUco library to detect the ArTags. For the detection of the Apriltags and CALtags, they use their respective libraries. Similar to the results of this thesis, they were able to validate that their chosen dictionaries have low robustness to edge occlusion, I.e., the blocking of a markers detection border.

Wang et al. tested Apriltags for AGV navigation [Wan+19] by building an application using the ROS platform. They achieved accurate navigation using four different core functionalities in AGV map construction, autonomous positioning, path planning, and path tracking. Their work was primarily focused around a single particular AGV, and they mainly considered Apriltags, while this thesis provides more an overview and comparison over how fiducial markers can be used for AGV navigation, while testing out the functionality of combined markers.

Yibo Liu et al. developed a navigation strategy for a small self-driving vehicle using a single marker [LSS21]. Unlike the second navigation strategy discussed previously, they use multiple vehicle-mounted cameras to create a 360°view and only uses a single marker in an environment without any potential occlusion. Since their approach does not use redundant markers, the contribution focuses heavily strategies to address the ambiguity problem and their experiments are conducted without occlusion.

Fiducial markers were also tested successfully underwater while analyzing the performance of different libraries regarding robustness to different properties (angles, water conditions, ...)[Ces+15]. Rafael Munoz Salinas et al. also propose strategies to detect markers in generic difficult conditions like camera defocus or motion blur [Mon+17].

No related work that directly uses combined markers, as they are presented in later chapters, was found. Wolf et al. propose a framework for robotic laboratory automation, for which they use both fiducial and barcode markers, however, they do not combine said markers [Wol+21].

Finally, a important contribution by Rafael Munoz Salinas et al. was published in March 2021 [Sar+21b]. It describes a new strategy to track an object through 3D space

using fiducial markers. This research contribution was a significant step forward since it allows tracking any shape with planar surfaces without requiring re-calibration of the camera extrinsic parameters. Furthermore, since it only uses fiducial markers that do not have to be placed into exact positions on the object, it is a very affordable and widely applicable technique compared to other current solutions. Other found solutions require re-calibration of extrinsic camera parameters or rely on specific equipment and lighting to achieve similar results [SMP05; WWh07]. .

4 Fiducial Markers

This chapter presents an overview over major fiducial marker libraries and dictionaries.

4.1 ArUco

The ArUco library is a popular open source library for the detection of fiducial markers developed by Rafael Munoz et al. [RMM18; Gar+14; Gar+15]. The library can be used to detect fiducial markers of most popular dictionaries including ArToolKit(4.2), ArTag(4.3), AprilTags(4.5), and of course their own proposed type ArUco markers. ArUco also allows for the generation of a personal dictionary if none of the existing dictionary types match the intended application requirements. This thesis tests the markers of the dictionary ARUCO_MIP_36h12, since the ArUco library recommends them as a marker type that can be detected fast and reliably. The ArUco library is also the only library found that includes additional functionality for marker maps.

4.2 ARToolKit

ArToolKit is a Open Source C/C++ library for AR applications developed by Hirokazu Kato et al. in 1999 [KB99]. However, according to the ArToolKitX website, [ArTb], it was sold in 2015 and has since 2017 been abandoned. Furthermore, any official documentation, as can be found in [ArTa] is also severely outdated. For these reasons, it was decided not to include ArToolKit markers in the test set.

4.3 ARTag

ArTag is a fiducial marker system developed by Mark Fiala [Fia05; Fia04] to improve upon the markers and marker detection processes of the ArToolKit library. ArTag is a single dictionary consisting of 2046 unique markers that use checksum and forward error correction techniques to improve robustness and detection speed.

ArTags are included in the test set of this thesis. In particular, the performance of ArTag markers was tested to evaluate how well the ArUco library works with different dictionaries.

4.4 ARToolKitX

ArToolKitX is a *Software Development Kit (SDK)* developed by Ben Vaughan and Phil Lamb as a spiritual successor of ArToolKit and can be found at [ArTb]. It consists of multiple libraries and utilities to help developers create AR applications running on Linux, Windows, macOS, iOS, and Android. Like ArUco, ArToolKitX offers methods for generating predefined dictionaries as well as generating personal dictionaries. However, the SDK focuses more on direct practical implementation compared to the ArUco. Therefore, ArToolKitX markers are not part of the test set, since the purpose of the thesis is to find fiducial markers fitting for AGV positioning and only developing prototype applications for detection, testing, and verification calculations.

The ArToolKitX SDK and the predefined dictionaries lack documentation and can be quite hard to use. In particular, the markers can not simply be pulled from a GitHub and printed but have to be generated. Then according to their generation parameters, different steps have to be taken during installation. This could prove difficult for maintenance and extension of a system built for AGV positioning and is not fitting for the prototypical implementations that were the goal of this thesis.

4.5 AprilTag3

AprilTag, or the most recent version AprilTag 3, is an open-source fiducial marker system developed and maintained by the April Robotics Laboratory. It has been the subject of multiple papers describing its development process and advantages [WO16; KHO19; Ols11]. Apriltags are both easy to use and powerful. They can be used with a C library, bindings in other languages, and the ArUco library.

Apriltag3 offers both predefined dictionaries, whose markers can be downloaded from their GitHub page, and the option of generating personal marker types. Both options are well documented and easy to navigate, as can be seen on the aforementioned GitHub page [Rob]. Therefore it was decided to test AprilTag3 Markers both using the Apriltag python bindings [duc] and the ArUco library. The tagStandard41h12 dicitonary was chosen, since it is recommended as the best dictionary for general purpose applications.

The following table summarizes the overview over major fiducial marker libraries (Figure 4.1.

	ArUco	Apriltag	ArToolKitX
OS:	Linux recommended, but other OS can be used		
Generation:	trivial	trivial	complicated
Options:	Predefined types + Personal type Generation		
Supports:	All Major types	Only its own types	Only its own types
Language:	C++	Python, C(++), Matlab, Julia	C(++), Java, Unity

Table 4.1: Fiducial Marker Libraries Overview; Note: Language Support does not mean, the library can not be used in other languages, but rather lists the currently well supported languages, C(++) means both C and C++

5 Experimental Setup

In line with the research questions, the purpose of the experiments was to (RQ1/RQ2) find a minimal set of properties, which define the different AGV navigation scenarios, (RQ2) evaluate them using test results and (RQ3) use the results to compare different dictionaries and libraries. The chosen properties are (1) lighting conditions, (2) occlusion, and (3) angles. Therefore the testing environment focused on simulating these properties while keeping other factors as controlled as possible. Furthermore, for the evaluation it was essential to have access to accurate ground-truth information.

5.1 Camera

The camera chosen for the generation of the test set was the Ipevo VZ-R document camera. This camera was chosen because its pose can be controlled quite freely and reliably. This means that any results acquired using the pose-estimation functionalities given by the used libraries could be compared with manually acquired data, and the markers could be tested in regards to their resistance to angles. Furthermore, since the camera on the Ipevo VZ-R is mounted on a durable stand, the camera's pose is more robust regarding minor effects that could influence results like vibration. The camera also support a wide range of different resolutions ranging from 3264/1840 to 640/480 pixels. Finally, the Ipevo VZ-R offers several useful functionalities, like locking the camera's focus, and has an included light that can be used for lighting condition testing.

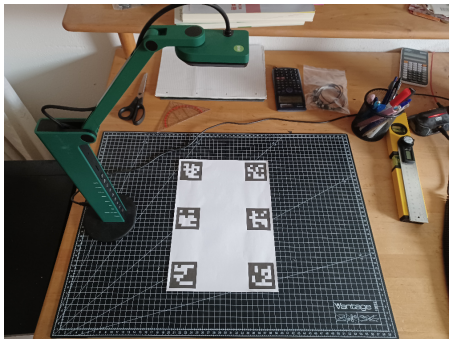


Figure 5.1: Testing Environment

5.2 Marker Map Design

As stated in Chapter 4, the dictionaries chosen were the ARUCO_MIP_36h11 dictionary proposed by the ArUco library, the tagStandard41h12 proposed by the AprilTags Libary,

and the ArTag type proposed by [Fia05]. All these dictionaries contain square markers with black borders since printing markers onto white paper is more natural than printing white-bordered markers onto a black background.

Since RQ3 aims at evaluating whether the advantages of Aztec Code Markers can be applied to fiducial markers, they were also included in the test set. Aztec Code are unlike fiducial markers, a type of 2D-Barcode. They are further introduced in Section 5.7.

Six markers, with the IDs 0-5, of all dictionaries were printed onto a white DIN A4 paper each as shown in 5.2. This implements a simple marker map. The design of the marker map, including ground-truth positional information, can be found in Figure 5.2. This ground truth information is reliable since it was decided during the creation of the marker map using vector graphics with pixels as measurement. Every marker has a side length of 50 pixels, which equals to 50 mm when printed. This ground-truth information is used to test the accuracy of the camera pose estimation up to the mm level.

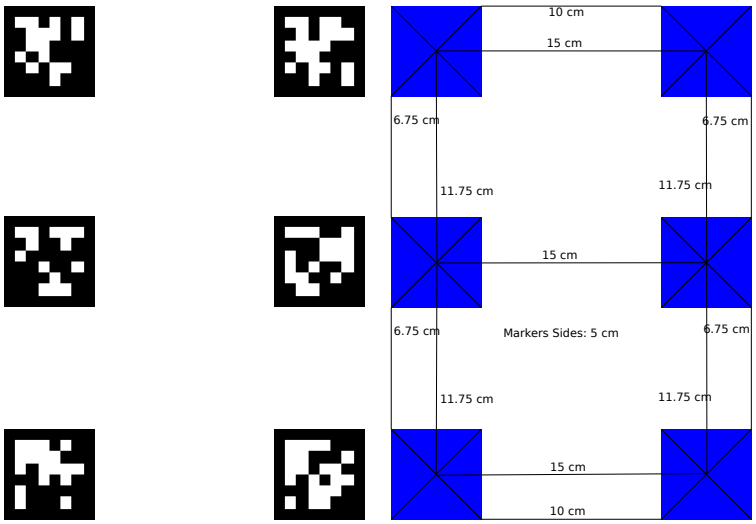


Figure 5.2: Marker Map Design with ground truth information

The test set was created within a single hour to ensure that all markers would be subject to the intended light conditions for each specific test. The test set was generated in the resolutions 3264/1840 and 640/480 to test out the effects of lower and higher camera resolutions.

Table 5.1: Camera Matrix OpenCV

5,82E+02	0	3,20E+02
0	5,82E+02	2,42E+02
0	0	1

Table 5.2: Distortion OpenCV

1,62E-01
5,22E-01
-1,50E-03
-3,49E-04
-4,32E+00

Table 5.3: Camera Matrix ArUco

5,66E+022	0	3,20E+02
0	5,66E+02	2,43E+02
0	0	1

Table 5.4: Distortion ArUco

1,73E-01
-5,12E-01
-1,44E-03
-1,51E-03
3,77E-01

5.3 Camera Calibration

As described in Chapter 1, the intrinsic parameters are needed for pose estimation and point projection. This thesis used both the approach proposed by OpenCV and the approach presented by the ArUco library to ensure accurate calibration and evaluate their advantages.

5.3.1 Chessboard: OpenCV

OpenCV uses the widespread chessboard calibration method. A black and white chessboard pattern is printed onto a white background. The sides of the squares and the entire chessboard are measured. By using multiple pictures of the chessboard from various angles and distances, the camera matrix and distortion coefficients of the camera used to capture the images can be calculated. The results of the calibration on the used camera can be found in Table 5.1 and Table 5.2.

5.3.2 Markerboard: ArUco Library

ArUco proposes its own calibration board consisting of ArUco markers arranged in a chessboard style pattern, as can be seen in Figure 5.3. The advantage of this board is that it does not have to be entirely visible in the pictures taken for calibration. Otherwise, the process is entirely the same. In both cases, the squares were measured in cm. Therefore the values in the resulting matrices 5.1 and 5.3 are also in cm. All values are rounded up. The exact values are in the complete tables in Chapter 9. Both calibration

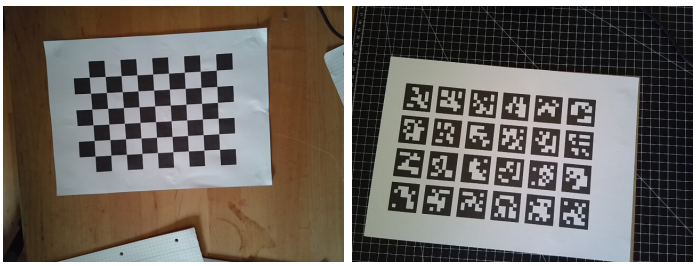


Figure 5.3: Examples of Calibration Images: left to right: Chessboard and ArUco Markerboard

methods resulted in very similar camera matrices, while the distortion coefficients are small regardless of the method used.

5.4 Testing Setup: Lighting Conditions

Four images for each chosen dictionary test for different lighting conditions. Fiducial markers become unreliable in reduced lighting conditions since the originally white spaces around and inside the markers appear black, making detection difficult. The four different lighting conditions simulate reasonable lighting conditions that an AGV might encounter and worse. Due to lack of specialized equipment, these tests are qualitative and only decided on worsening conditions by instinct. However, since all markers were tested with the same worsening lighting conditions, they can still be compared.

5.5 Testing Setup: Systematic Occlusion

Occlusion is a significant problem when using fiducial markers. Since the information is encoded in black and white squares and the marker is detected through thresholding, once the marker is partly obscured, detection can quickly become erroneous or fail. In particular, blocking even a tiny part of the detection border can lead to no marker being detected. In contrast, objects that block the bits inside a marker can lead to false marker ids or false negatives. To compare different dictionaries regarding robustness to occlusion, a series of test cases were constructed on the designed marker maps and tested under different lighting conditions.

Initial Tests: Initial, qualitative tests, using household objects (pen, triangle rulers, etc)

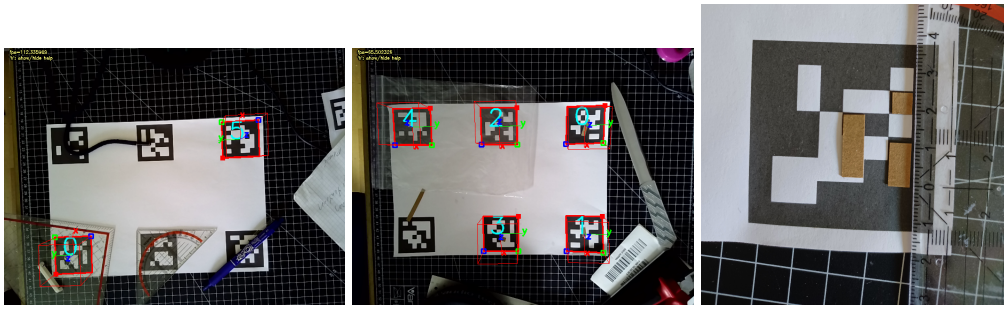


Figure 5.4: Examples of Occlusion Testing: Initial tests using household objects with very negative results on the left, final testing using bit-blockers and foil in the middle, closeup of bit-blockers on the right

Detection Border: A small bit size block that blocks the detection border (bottom left)

Thin, see through foil: Two layers of thin see through foil that test for markers protected from water through foil as well as other potential occlusion conditions (top left and top middle)

1 bit blocker: One bit blocker placed inside the marker (bottom middle)

2 bit blocker: Two bit blocker placed inside the marker (bottom right)

3 bit blocker: Three bit blocker placed inside the marker (top right)

Bit-blockers were used after initial tests using household objects, resulted in incomparable results. In particular, any object going directly over a marker, like, a pen, always stopped detection. Other ideas of simulating environments through blocking markers using transparent triangle rulers also resulted in no detection. Examples can be seen in Figure 5.4. These examples show how initial occlusion testing using larger objects cannot be used to compare different marker types since no markers, regardless of type, can be detected using these tests. Meanwhile, blocking markers with bit-blockers, leads to comparable results.

However, placing bit-blockers on the same positions with respect to the marker center for each dictionary is complex and does not equal fair treatment. Therefore, it was decided that all bit markers are placed in positions with similar detection difficulty for each dictionary. In particular, the 3-bit blocker was always placed such that it passes from a black square into a white square and back into a black square at least once for each dictionary. This method was deemed adequate due to the significant amount of test cases. Furthermore, two markers are blocked with foil since the results

of blocking with foil can simulate visual distortions, while initial tests using foil were more comparable than blocking with triangle rulers. Furthermore, these tests also determine if a marker printed on paper could be protected from fluids using foil wrapping.

5.6 Testing Setup: Angles

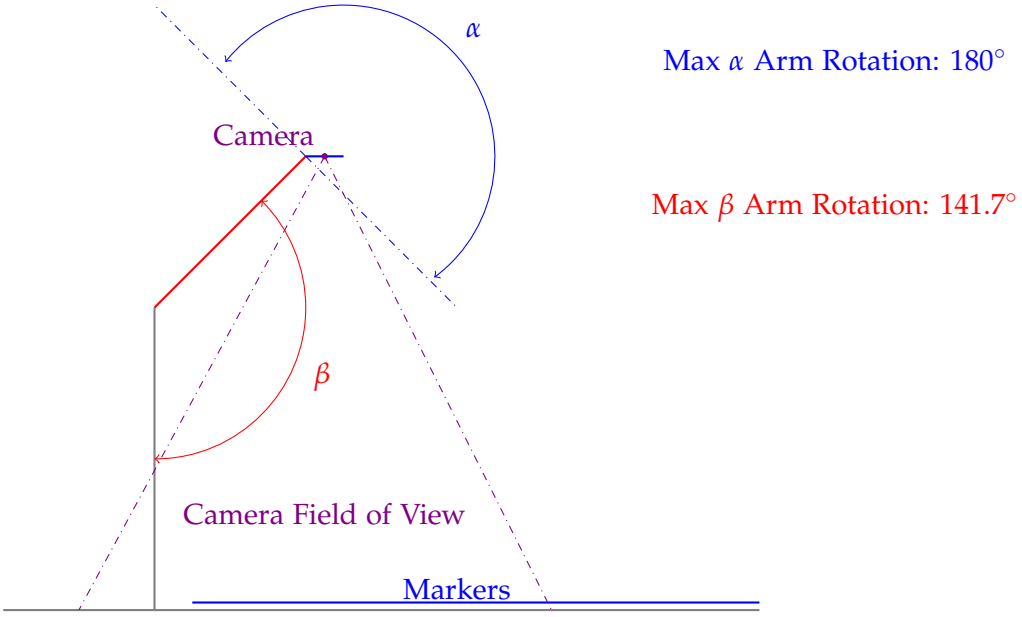


Figure 5.5: Angle Tests Configuration

Two series of test cases were created to test the robustness of fiducial markers to angles. Three images per dictionary test purely for successful detection at angles without analysing accuracy, while another three images per dictionary test for accuracy at smaller angles. The angles were constructed by moving the camera's pose and adjusting the position of the marker map to still be in the camera's field of view. For a sketch of the angle setup, refer to 5.5. For the first test set, both the α and β angles were manipulated. Accordingly, the marker map was positioned inside the blue area such that it is always in the camera's field of view. Only the α angle was manipulated for the second test set, while the β angle was kept at a constant 141.7°.

The angles for the second set are therefore described with reference to the horizontal surface on which the markers are placed. Therefore, a 20°angle equals raising the blue arm, whose rotation is described by α, by 20°above the horizontal line. According to

this system, the angles chosen for the second set of three markers per dictionary were 17°, 20°, and 30°.

5.7 Experimental Setup: Aztec Code

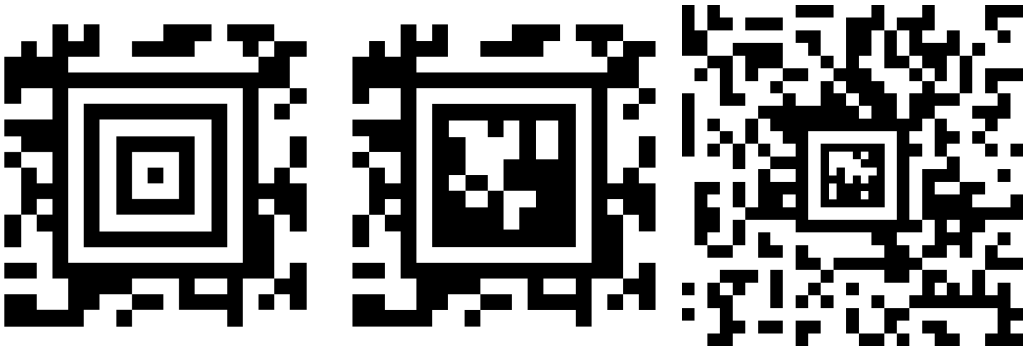


Figure 5.6: left to right: Aztec Code, small Aztec Code with Apriltag center, larger Aztec Code with Apriltag center

RQ3 aims at evaluating whether the advantages of Aztec Code markers can be applied to fiducial markers. Aztec Code was chosen because of the advantages stemming from its design. As can be seen in Figure 5.6, Aztec Code uses a bull's eye style pattern consisting of concentric squares in alternating black and white as its detection pattern. The encoded information is build around the detection pattern. This design allows the encoded information to freely grow around the pattern while the pattern stays the same size, no matter how much information is encoded. Since the detection pattern is inside the marker, it also does not require a blank white zone around the marker and can save space. Furthermore, the concentric squares allow for robust and performant detection since they add redundant detection patterns. Meanwhile, the robustness of the encoded information is ensured through Reed-Solomon error correction and can be increased freely with 23% being recommended.

To verify the advantages of Aztec Code compared to fiducial markers, generated Aztec Codes were compared to the chosen fiducial marker dictionaries regarding robustness to occlusion, lighting conditions, and angles. The lighting conditions and angle tests were executed in the same manner as for the fiducial markers. For the occlusion tests, only four Aztec Codes were printed onto a DIN A4 paper, since even very small Aztec Codes are already larger than fiducial markers. By only printing four Aztec Codes onto the DIN A4 paper, the size of the Aztec Code can be adjusted, such

that every square of the Aztec Code is just as large as every square of every tested fiducial marker. Therefore, the same bit-blockers could be used for occlusion testing.

To evaluate whether the advantages of Aztec Codes can be applied to a fiducial marker-based navigation system, different strategies were tested.

5.7.1 Aztec Code: Combined Markers

A initial idea of using just the center squares of the detection pattern as a fiducial marker could not be implemented since concentric squares are not unique regarding rotation and mirroring. Therefore positional information can not be uniquely determined through it. Including the orientation bits, marked blue in Figure 5.7 on the right, makes the detection pattern unique towards rotation and mirroring. However, using the entire detection pattern as a fiducial marker still has several problems. It only offers a single type of fiducial marker instead of a dictionary, and therefore additional information has to be encoded. Furthermore, standard fiducial marker detection strategies can not be applied to the Aztec Codes detection center since these strategies rely on borders around the actual marker. Conclusively, this strategy can be useful for certain applications, but requires existing libraries to be extended. Therefore, it was instead decided to replace the center squares of the detection pattern with fiducial markers from existing dictionaries. The resulting type of combined marker ensures that the pose detection can be executed accurately. In addition, there are a sufficient amount of ids available for any potential application without having to encode additional data. An example of two of these Aztec Codes, with the detection center replaced with Apriltags of the dictionary 36h11, can be found in Figure 5.6. Of course, a combined marker can be created with any fiducial marker dictionary. However, the 36h11 dictionary was chosen since it can be detected with both the ArUco and the Apriltags library and was tested as part of the thesis. For a practical implementation, a smaller dictionary might be more useful, since additional information can be encoded in the Aztec Code and the bits of a smaller dictionary would be described by larger squares, therefore increasing the robustness to occlusion.

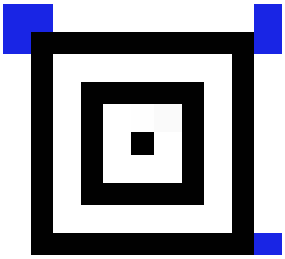


Figure 5.7: Detection Pattern

Since the actual pose detection is done on Apriltags, combined markers are not included in the test set but are instead validated using a series of unique tests. In particular, it was tested whether it would still be relatively simple to decode the Aztec Code outside of the AprilTag marker and whether increasing the size of the encoded information would have adverse effects, outside of simply requiring larger markers or

a higher resolution image. Both tests were successful and Section 6.8 includes code examples, which show how to utilize the created combined markers.

While the created combined markers, can both encode information and be used for pose estimation, the main advantage of the Aztec Codes resistance to occlusion gets lost, since the detection pattern is replaced with a fiducial marker. The thesis therefore attempted to find improved combined markers, by only making slight modifications to the detection pattern instead of entirely replacing it with existing fiducial markers. The goal of these modifications was, to create Aztec Codes that can be directly used with both current fiducial marker libraries and current barcode reader applications and libraries, since they only make slight modifications.

5.7.2 Aztec Code: Slight Modifications to Detection Pattern

Figure 5.8: Slightly Modified Aztec Codes

As can be seen in Figure 5.8, several different modifications were tested. The strategy behind the different modifications was to change as little of the detection pattern as possible, while making it unique towards rotation.

However, while all the created modified markers can be used as fiducial markers (tested using ArUco as a custom dictionary), they either were not detected at all by barcode reader libraries and applications or they lost their robustness advantage and became unreliable.

Conclusively, slightly modifying Aztec Code detection centers removes the robustness advantage of Aztec Code and combined markers are the better solution for markers that can both encode information and be used for pose detection.

6 Experimental Results

The following sections will summarize the testing results acquired using the ArUco library and the Apriltags Python bindings. The contents of the tables are average values to focus on the most relevant parts, and all positional information is displayed in mm, rounded up to the 100-micrometer level. More complete testing results tables can be found in Chapter 9. Aztec Codes were tested using the Zxing Libary [Zxi] as well as the QR Reader Android Application, and the Aspose Web App [Asp].

6.1 Detection Time

All chosen markers were tested using the ArUco library both while passing the dictionary and without passing the dictionary. The results are summarized in Table 6.1.

All markers were detected with over 100 fps as long as the dictionary was passed to the detection method. Therefore time performance of the detection is not a problem towards implementing AGV navigation using fiducial markers.

	ArUco	ArTag	Apriltag
Without Dictionary	12.8061 ms/78fps	Not Detected	10.8996 ms/90fps
With Dictionary	8.27623 ms/120fps	8.21145ms/130fps	9.652ms/103fps

Table 6.1: Detection times with ArUco

As can be seen above, the best performing marker was the ArUco marker of the dictionary type 36h11. However, whether this is due to the ArUco library or the dictionaries was not further analyzed, since all markers are already shown to be performant enough for an AGV navigation system.

Unfortunately, while the ArUco library is supposed to try to find markers of any dictionary whenever no dictionary was passed, it was unable to detect the ArTags without the passed dictionary. The reason for this error is unknown and was not further researched since it is not relevant to the research questions.

6.2 Accuracy of Pose Detection

The accuracy of the pose detection is a central aspect of evaluating the performance of different markers and libraries. To verify the accuracy of the pose estimation, the estimated camera extrinsic parameters are compared with ground truth information. Due to lack of specialised equipment, potentially erroneous information is collected during measurement. However, these errors can be circumvented by comparing the inter-distance between markers. Since the inter-distances between markers are determined during printing using vector graphics as explained in Chapter 5, they present ground truth information accurate up to at least the millimeter level.

In particular, the inter-marker-distance approach can be trivially applied whenever the markers are parallel to the camera. As seen in Figure 5.2 in Chapter 5, the distances between all markers are known. Therefore, by adding up the x and y values of the translation vectors of markers A and B and accounting for signs, the resulting inter distance δ between the center points of A and B can be calculated. Comparing this δ with the determined ground truth information then shows the accuracy regarding x and y. In the ideal case, the values would be 150mm for δy and 117.5mm for δx.

In addition, this approach is less error-prone since even if the testing paper's position shifts slightly with respect to the camera, it will not affect the inter distance between markers. The disadvantage of this approach is that it is only applicable if two directly adjacent markers in the marker map are detected and the markers are parallel to the camera. Therefore, this approach can not be applied for angled markers, so a different approach to verify accurate pose detection on the angled images was used. This approach is presented in Section 6.5. However, due to the amount of markers in one marker map and the size of the test set, a sufficient amount of results to verify the reliability of the pose detection are produced.

	δx-error	δy-error
Apriltags1	0.95	0.65
Apriltags2	0.75	0.7
ArUco	0.39	0.28
ArTag	0.63	0.87

Table 6.2: Average δ in mm, high resolution

Table 6.2 and 6.3 present the average δ errors. All the translation vectors, as well as the δ values, can be found in Chapter 9. Apriltags1 refers to the results acquired using the ArUco library, while Apriltags2 to the results acquired using the Apriltags library.

	δx-error	δy-error
Apriltags1	0.72	0.38
Apriltags2	0.58	0.79
ArUco	0.4	0.53
ArTag	0.79	0.45

Table 6.3: Average δ in mm, low resolution

These results show that the average error is always below 1 mm, therefore verifying the desired accuracy at mm level for the x and y errors.

According to these results, the ArUco dictionary 36h11 outperforms the other types regarding accuracy. In particular when increasing the resolution, the ArUco library was consistently able to estimate the cameras pose with x and y errors smaller than half a mm.

The thesis also found a accuracy advantage when increasing the camera resolution. While this advantage was small in the experimental environment, it is reasonable to assume that this was due to the short distance between markers and the camera. Therefore, when increasing this distance, a more significant correlation between camera resolution and pose detection accuracy is expected.

6.3 Camera Calibration

Since the calibration was done with both the strategies proposed by ArUco and OpenCV, it was decided to compare which method leads to more accurate camera intrinsic parameters. For this comparison, only test images with six detected markers were used. Finally, it was attempted to account for as many other factors that can influence pose estimation accuracy as possible. Hence, the images used to compare different calibration strategies were without occlusion, at ideal lighting conditions, and the camera parallel to the marker. This also means that the inter-marker-distance can be used to compare the accuracy of the x and y values of the translation vectors. Furthermore, only results acquired using the ArUco library were compared.

These results indicate that the calibration using ArUco results in slightly more accurate camera intrinsic parameters. However, the difference is insignificant, and when comparing just the average results for high-resolution images, the errors using the ArUco calibration are not strictly smaller than the errors using the OpenCV calibration. This can be attributed to the fact that the ArUco high-resolution calibration was done later in the thesis project, and therefore the calibration was done more finely.

Calibration Method:→		OpenCV		ArUco		Row
Average Delta Values:		δx	δy	δx	δy	Avg
ArUco Markers	640/480p	1.32	0.85	1.28	0.69	1.04
	3264/1680p	0.57	0.07	0.59	0.12	0.34
Apriltag Markers	640/480p	0.37	0.95	0.2	0.85	0.60
	3264/1680p	0.09	0.24	0.14	0.05	0.13
ArTag Markers	640/480p	0.47	0.84	0.49	0.68	0.62
	3264/1680p	1.72	1.48	1.66	1.45	1.58
Column Averages:		0.76	0.74	0.73	0.64	

Table 6.4: Average Errors of both calibration approaches in mm

One aspect to note is that, except for the ArTags, a sizeable increase in accuracy can be observed when increasing the camera resolution for these results. A likely reason is that other influencing factors were controlled for these results, whereas they were included in the average calculations for the previous accuracy tests.

It is important to note that these results can not be directly attributed to the respective calibration strategies since the camera calibration was done by hand and did therefore not control all factors.

6.4 Lighting Conditions

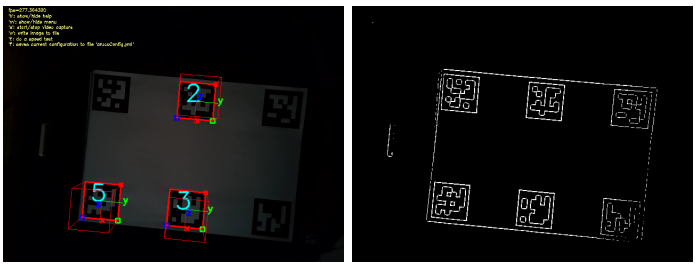


Figure 6.1: Lighting Conditions Example: original image with detected markers on the left, thresholding on the right

As can be seen in Figure 6.1, fiducial markers become unreliable in very poor lighting. It shows how thresholding is unable to differentiate between white and black spaces in a poorly lit environment. Due to lack of equipment, the thesis did not measure the different lighting conditions but rather conducted qualitative tests. However, since an AGV will usually work in an environment with constant, artificial light, the tests

validate that fiducial markers are detected reliably even in worse lighting conditions than expected from an AGV environment. Only without any light source will the detection stop working. As can be seen in Figure 6.1, even with reduced lighting, as long as markers were detected successfully, pose detection was also successful.

Table 6.5 summarizes some results in regards to lighting condition tests. The lighting conditions are worse with every row, and row two is already at a level extremely unlikely for an AGV environment. As before, Apriltags1, in the context of Table 6.5, refers to Apriltags tested with the ArUco library, while Apriltags2 refers to the results acquired using the Apriltags library. Table 6.5 shows the results with high-resolution images, but the results with low resolution are very similar.

ArUco	ArTag	AprilTags1	Apriltags2	AztecCode
0-5	0-5	0-5	0-5	0-5
0-5	0-5	0-5	0-5	0-5
0,2,3,5	1,2,3,5	2,3,4,5	0-5	0-5
∅	∅	∅	0,2	∅

Table 6.5: Detected Marker IDs with worsening lighting conditions, high resolution

From these results, it can be suspected that the Apriltags library is more reliable at detecting Apriltags markers at reduced lighting conditions. In particular, the Apriltags library successfully detected markers on the same input images that could not be detected with the ArUco library.

Aztec Code performed slightly better than ArUco and ArTag, which can be attributed to their high resistance to occlusion. It has to be noted, that the reliability of the Aztec Code detection varied significantly with different applications and libraries. In particular, a phone application called QR Scanner in the Google Play Store, was able to detect markers in significantly darker environments, however these differences could not be quantified and are therefore not further documented. Conclusively, all tested fiducials and both libraries are robust against lighting conditions.

6.5 Angles

As explained in the previous Chapter, for the first set of angle tests, the detected camera extrinsic parameters were not analyzed, and it was instead tested whether detection of markers at different camera angles is possible. These tests were successful, and markers can be detected at 20°, 30°, and 60°angles as can be seen in Figure 6.2. To analyze the accuracy of the pose detection on angled images, the chosen angles for the second set were 17°, 20°, and 30°. For these images, the camera focus was locked.

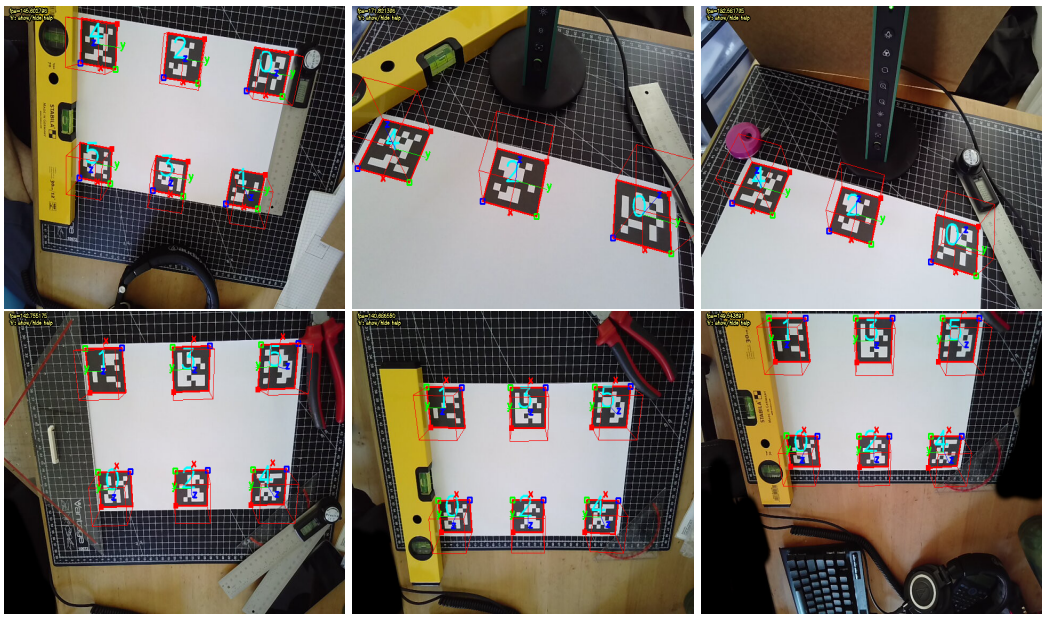


Figure 6.2: Examples of Angled tests, First Test Set above, Second Test Set below

For this second set, the inter-marker distance approach for testing accuracy, as used earlier, can not be trivially applied since the camera is not parallel to the markers. Therefore the approach chosen was to measure the distance between the camera and the marker centers and compare it with the norm of the translation vector. Since the translation vectors point from the center of the marker into the camera, the norm of the translation vector calculated using $\sqrt{x^2 + y^2 + z^2}$ should in the ideal case equal the measured distance. However, measuring the distance by hand using simple rulers is error-prone. Therefore, a qualitative way to validate the accuracy of the results is used.

In particular, it was analyzed whether all calculated norms follow the same order as the measured distances. This means, if a measured distance between marker A of the marker map and the camera is shorter than the measured distance between marker B and the camera, then the norm of the translation vector of A needs to be smaller than the norm of the translation vector of B.

This test was successful for all tested angles at 17, 20 and 30 degrees. Results can be verified, using the complete tables in Chapter 9.

Finally, the thesis tested whether Aztec Codes could be detected at the same minor angles. Overall, the thesis found that Aztec Codes are robust to minor angles. However, the robustness varies heavily from application to application. In particular, the Aspose web app had difficulties detecting all markers at minor angles, while Zxing is reliable

as long as only a single Aztec Code is found per image. However, these problems are implementation dependant and can be fixed in a custom library.

Conclusively, the thesis is able to verify that all tested fiducial markers and Aztec Codes are robust to minor angles. Other contributions support this observation and present more exhaustive results[Ces+15]. [Ces+15].

6.6 Systematic Occlusion

This section present the results of testing Aztec Code and fiducial markers regarding robustness to occlusion. First the results regarding Aztec Code are summarized.

6.6.1 Systematic Occlusion: Aztec Code

As expected, due to the Solomon-Reed error correction, Aztec Codes are very robust to occlusion in the encoding area. The generated Aztec Codes can always be decoded when bit-blockers with sizes 1-5 bits block the detection area. This robustness can be further increased depending on the encoding used, with 23% maximum erroneous bits being the standard.

However, the robustness to occlusion in the detection center is worse than expected. While rectangle bit-blockers blocking 1-5 bits of the detection center do not block detection, the slightly modified markers can not be detected. This is unexpected since the modified markers only modify 4-6 bits. A potential reason is that the modified markers need to change bits in at least three corners of one of the concentric squares, while rectangle bit blockers will block a maximum of two corners.

Nevertheless, fiducial markers show a even lower robustness to occlusion.

6.6.2 Systematic Occlusion: Fiducial Markers

As explained in Section 5.5, the systematic occlusion tests are the property where it was expected that fiducial markers would perform the worst, and the thesis validates this expectation. As can be seen in Figure 6.3, even minor occlusion leads to no detection of the marker. In particular, even blocking out small parts of the detection border entirely stops detection.

As explained, the thesis first tested for detection while blocking with various household objects like pens or see-through triangle rules. However, since no markers were detected, the results were incomparable, so to acquire more valuable results, the bit blockers were created. Furthermore, the thesis tested for occlusion using a thin see-through foil, which leads to very unreliable results, where even slight changes in light

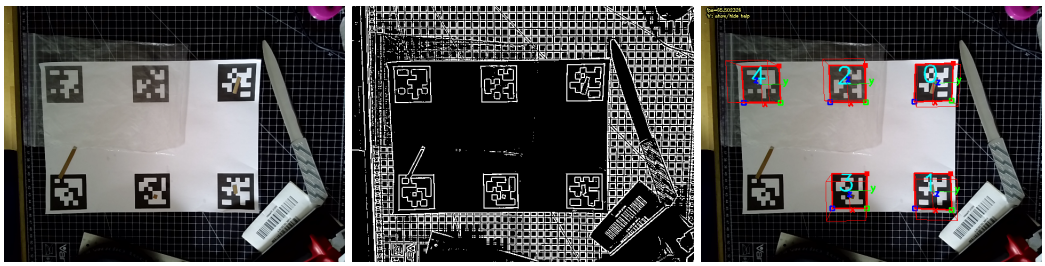


Figure 6.3: Occlusion Example, left to right: Original Image, grayscaled and applied threshholding, Original Image with red cubes projected on top of detected markers

conditions lead to detection errors, as can be seen when looking at the threshold images in Figure 6.4 below.

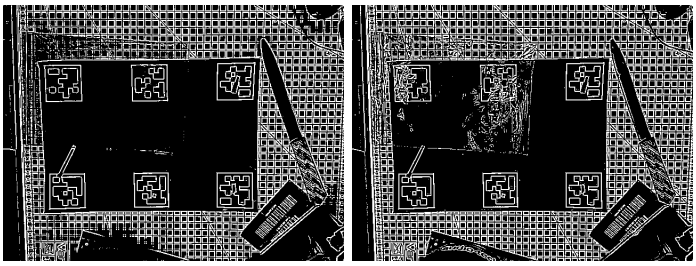


Figure 6.4: Occlusion Example: Threshold on the left is able to detect the Markers blocked by foil; In the image on the right, the light source added causes the foil to block the detection of Markers

All Occlusion testing results are summarized in the following Table 6.6.

Therefore, the thesis concludes that the most important property for AGV scenarios is the amount of occlusion that can be expected. In any environment where occlusion can be anticipated, redundant markers are essential to ensure accurate pose estimation.

6.7 Code Examples

This section presents several code examples on how to use fiducial markers for point projection and drawing an object into the image according to 3D coordinates.

		Aztec Code	ArTag	ArUco	AprilTag
Foil	with Light	X	X	X	X
	without Light	✓	✓	✓	✓
BitBlockers	1	✓	✓	✓	✓
	2	✓	✓	✓	✓
	3	✓	✓	✓	✓
Center/Border	slightly	✓	X	X	X
	1-5 Bit	✓	X	X	X
	6,7 Bit	rarely	X	X	X
	8+ Bit	X	X	X	X
Household Objects	Large Ruler	✓	✓	✓	✓
	Small Ruler	X	X	X	X
	Pen	X	X	X	X

Table 6.6: Occlusion Summary: BitBlockers are placed inside fiducials and in the encoding area, while Center/Border block parts of the detection center and fiducial marker border respectively

6.7.1 Middle Point Projection

The advantages of marker maps are largely of a practical nature and were therefore tested using test scripts. The following python code example will show how to use the OpenCV method *projectPoints* to project the middle point of the marker map into the image plane, given the translation and rotation vectors as well as ground truth information that is known about the marker map. The method can be used analogously in C++, and other computer vision libraries have similar functions.

```
projection_points = np.array([0.075,0,0])
imagepoints, _ = cv2.projectPoints( projection_points,
                                    Rxyz,Txyz,
                                    camera_matrix,
                                    dist_matrix)
```

The above code assumes that the detected marker was the bottom middle marker. Therefore the projected middle point is on the same height and the same x coordinate as the detected marker and 7.5 cms or 0.075 m along the y axis with respect to the marker. The units that need to be used, depend on the units that were used during camera calibration. In this case the camera was calibrated using meters, so the used value is 0.075. Rxyz and Txyz are the rotation and translation vectors, which were previously determined with respect to the marker, so in most languages and libraries,

they would be called similar to *marker.Rxyz* and *marker.Txyz* respectively. The camera matrix and the distortion coefficients are needed to account for the camera intrinsics as explained in chapter 1.

The above example projected the point using one particular marker, but to project the middle point using any detected marker, the projection_points need to be adjusted dynamically. The following pseudo-code example will show an example using pattern matching.

```
match tags[0].id:
    case [0]
        projection_points = np.array([0.075,0.1175,0])
    case [1]
        projection_points = np.array([-0.075,0.1175,0])
    case [2]
        projection_points = np.array([0.075,0,0])
    case [3]
        projection_points = np.array([-0.075,0,0])
    case [4]
        projection_points = np.array([0.075,-0.1175,0])
    case [5]
        projection_points = np.array([-0.075,-0.1175,0])
```

This code snippet takes the first detected Marker and, according to its ID returns the correct projection points to project the middle point when used with the *projectPoints* method as shown above. A practical implementation could store the ground truth position of markers in a key-value store and use those to project positions into the two-dimensional image plane as well as increase the accuracy of the projection and account for the ambiguity problem as explained in Chapter 1. For example, in python a implementation using the running marker map example would look as follows:

```
Dictionary = {
    0: ("bottom left", np.array([0.075,0.1175,0])),
    1: ("top left", np.array([-0.075,0.1175,0])),
    2: ("bottom middle", np.array([0.075,0,0])),
    3: ("top middle", np.array([-0.075,0,0])),
    4: ("bottom right", np.array([0.075,-0.1175,0])),
    5: ("top left", np.array([-0.075,-0.1175,0]))
}
## Tags are Detected and saved in tags
## ...
## Iterate over tags, to project all the middle points
```

```
for tag in tags:
    ## Some Libraries (like aruco) will already convert the rotation
    ## matrix into a rotation vector, however for completion sake,
    ## it was included in this example
    Trvec, _ = cv.Rodrigues(tag.pose_R)
    ## Use Key Value store to access stored information
    projection_point = Dictionary[tag.tag_id][1]
    ## project Points
    middle_point, _ = cv.projectPoints(projection_point,
                                       Trvec,tag.pose_t,
                                       cameraMatrix,dist_matrix)
    px=px+middle_point[0]
    py=py+middle_point[1]
px=px/len(tags)
py=py/len(tags)
projected_multimarker_middle = [px,py]
```

In this example, the key-value store only contains a tuple with basic information and the array that declares the middle point in ground truth information based on the position of the marker. The key-value store is then utilized when iterating over all detected tags to pass the correct projected points for every detected tag. After all projected coordinates are added up, they are then divided by the number of detected tags to calculate the average value of all projected points. To account for the ambiguity problem, the implementation just has to search all values for outliers before calculating the average since the ambiguity problem can be found by watching for a significant change in the z value of the transformation vector. The thesis did not implement a method to improve the accuracy of the point projection using multiple detected markers.

A practical example could store more helpful information inside of a key-value store associated with every tag. Here combined markers can be used to encode information directly in the marker, instead of having to access a potentially large key-value store.

6.7.2 Cube Projection

A typical test task on fiducial markers is to project an object, commonly a cube, on top of a fiducial marker. The following code will again use OpenCV and NumPy in python and can still be used analogously in other languages.

The first step towards drawing a cube above every detected marker is to project the points that need to be drawn using the projectPoints method. To clarify, the marker itself is the bottom side of the cube while the translation vector still originates from the

center of the marker. Therefore, the cube corners are *markersize*/2 into both x and y direction and the four points creating the top side of the cube above the marker are *markersize* into the z-direction. The code below stores the position of every point of the cube in reference to the center of the marker in a NumPy array.

```
size_div_2 = markersize/2
cube_project_points =np.float32([[-size_div_2, -size_div_2, 0],
                            [-size_div_2, size_div_2, 0],
                            [size_div_2, size_div_2, 0],
                            [size_div_2, -size_div_2, 0],
                            [-size_div_2, -size_div_2, -markersize],
                            [-size_div_2, size_div_2, -markersize],
                            [size_div_2, size_div_2, -markersize],
                            [size_div_2, -size_div_2, -markersize]])
```

The created matrix is passed to the OpenCV projectPoints method for every detected marker stored in tags:

```
for tag in tags:
    cube_pts, jac = cv.projectPoints(cube_project_points,tag.pose_r,
                                tag.pose_t,cameraMatrix,dist_matrix)
    cube_pts = np.int32(cube_pts).reshape(-1, 2)
```

Since cube_pts is a CV mat, it gets transformed back into a NumPy array with a fitting form for further use.

The thesis tested several ways of using the projected points to draw the cube and presents the most recommended one as a python method:

```
def draw_cube(cube_pts,img):
    for i in range(4):
        cv.line(img,cube_pts[i], cube_pts[(i+1) % 4],(0,0,255),5)
    for i in range(4):
        cv.line(img,cube_pts[i+4], cube_pts[4+(i+1) % 4],(0,0,255),5)
    for i in range(4):
        cv.line(img,cube_pts[i], cube_pts[(i+4)],(0,0,255),5)
    return img
```

The method gets passed the projected cube points as well as the image and then draws the 12 lines needed to draw the cube. It returns the image with the cube drawn.

6.8 Aztec Code

This section analyses the practicality of combined markers as they were created according to Section 5.7.

Since the combined markers use the 36h11 dictionary, the performance of the combined markers in regards to accuracy and robustness is equal to the results of the Apriltags as explained in the previous sections. Furthermore, combined markers can be created with every squared fiducial marker dictionary. However, to evaluate the viability of using combined tags in a practical implementation, an approach that can both detect the fiducial inside of the Aztec code and then still detect and decode the information outside of the detection pattern had to be developed. For this purpose, a series of different strategies were considered:

Extension Strategy: Utilize a existing library (like ArUco) to detect the Apriltag, and extend that library with additional functionality to decode the Aztec Code

New Development Strategy: Develop a completely new library that utilizes the algorithms used in existing libraries for marker detection and then uses the positional information acquired from those libraries to decode the Aztec code

Prototype Testing Strategy: Use existing libraries for pose detection of the Apriltag as well as the decoding of the Aztec code and build an adapter style script, that edits input images by masking the Apriltag inside the Aztec code with the original Aztec Code detection pattern.

Ultimately, since an entirely new library went beyond the scope of the thesis project and it is questionable whether extending a library with the intended functionality is a useful contribution, the prototype testing strategy was used.

The enclosed Apriltags were therefore detected using the Apriltags library, while the ZXing library [Zxi] was used for decoding. Finally, the following python code was used to mask the enclosed Apriltags and generate an image to decode the Aztec code from. As in the previous examples, tags still contains all detected tag objects.

```
img_mask = cv.imread(mask_path)
for tag in tags
    pts_dst = tag.corners
    size = img_mask.shape
    pts_src = np.array(
                      [
                       [0,0],
                       [size[1] - 1, 0],
```

```
                    [size[1] - 1, size[0] -1],
                    [0, size[0] - 1 ]
                    ],dtype=float
                   )
h, status = cv.findHomography(pts_src, pts_dst)
temp = cv.warpPerspective(img_mask, h,
                        (img_dst.shape[1],img_dst.shape[0]))
cv.fillConvexPoly(img_dst,pts_dst.astype(int), 0, 16)
img_dst = img_dst + temp
cv.imshow("output",img_dst)
k = cv.waitKey(0)
cv.imwrite("./"+tag.tag_id+".jpg", img_dst)
```

An implementation that is part of a more extensive system would have to adjust the mask read into img_mask using *cv.imread(mask_path)* dynamically since the size of the detection pattern depends on the size of encoded information. However, for this testing prototype, it is sufficient to manually adjust the mask_path depending on the size of the input Aztec code. The rest of the code uses methods given by OpenCV to find the homography between the mask points and the destination points, warp the mask into the perspective of the camera, and then fill in the area with the warped mask. The result is then added to the original image. The original image with the mask fit on top of the Apriltag can then be passed to a decoding method and is in this example saved into the current directory. Decoding the Aztec Code from the masked images was successful using the ZXing library.

Conclusively, detection of combined markers is trivial with both small and sizeable Aztec codes and in both high and low resolution. Thus, the intended functionality of a marker that can freely encode information and still be used for pose estimation is achieved. However, the high robustness to occlusion of Aztec Code is lost.

7 Discussion

This section reflects on the results of this thesis, regarding the research questions and the contributions.

> **RQ1:** Many scenarios can be identified, regarding the interface between robots and AGVs, ranging from full visual identification and dynamic handling of parts, to fully static static programming and gripping of parts. The feasibility of the scenario depends on (a) how exact an AGV can be positioned and (b) how consistent it can be positioned. If inconsistent positioning is considered the norm, which potential alternative solutions exist, and how can they be compared in terms of (1) implementation effort and (2) evolution effort when dealing with fast-changing product lines and small lot sizes?

The thesis identified and elaborated three different scenario solution strategies.

Mixed Navigation: AGV Mounted Camera: A camera is mounted on an AGV following static paths, and fiducials are used for navigation during object interaction. Therefore the markers are only placed at the object interaction environments. The thesis validates that fiducial markers are cost-effective and reliable for this approach due to their high robustness to lighting conditions and angles as well as their high accuracy. Furthermore, combined markers can encode destination information for an object that the AGV is supposed to transport.

Dynamic Navigation: AGV Mounted Camera: A camera is mounted on an AGV for dynamic navigation, and fiducial markers are positioned in the environment. The environment must be filled with enough markers such that the AGV mounted camera can always accurately detect a marker, no matter its current position. Furthermore, the camera pose has to be repeatedly re-estimated with every movement. Due to this strategies complex setup, computational complexity, and low robustness, the thesis recommends using strategy one or three instead. In particular, the thesis validates that fiducial markers have low robustness to occlusion, which is quite likely in this approach.

Dynamic Navigation: Multi View AGV Tracking: The thesis analyzed the method for Multi-View Object tracking presented by Salinas et al.[Sar+21b] and evaluated it towards AGV navigation. It was decided that the approach is well suited but requires

significant development work to implement in practice, which can be addressed in future work.

> **RQ2:** For existing AGV scenarios, what is a minimal set of properties (tilt, distortion, lighting, contrast, ...) defining these scenarios, and how do these properties affect the number, size, and fiducial required for successfully identifying the position of an AGV?

The most important property is the amount of occlusion that can be expected. Fiducial markers are resistant to tilt, distortion, and lighting conditions, while their robustness to occlusion is low. Furthermore, the pose of an AGV can be successfully identified using a single markers. However, this pose estimation is subject to ambiguity. Adding redundant markers to the system allows for consideration of the ambiguity problem and higher robustness to occlusion. Therefore, this thesis concludes, the only relevant criteria for positioning a camera and thus successfully implementing a scenario is, that as many as possible markers are visible. Of course, updating the AGV position of a moving AGV requires some computational effort, depending on update frequency.

> **RQ3:** How do different markers and libraries perform compared to each other? Can the advantages of Aztec Code (i.e., storing data in the marker) be used in fiducial marker-based applications?

For this thesis three dictionaries were tested. (1) the ArTag dictionary, (2) the ARUCO_MIP_36h11 dictionary of the ArUco library, as well as (3) the tagStandard41h12 dictionary of the AprilTags library. All three dictionaries were tested using the ArUco library, while the tagStandard41h12 was additionally tested using the AprilTags library. All dictionaries performed similarly regarding accuracy and robustness. ARUCO_MIP_36h11 was slightly more accurate on average compared to the other dictionaries. The AprilTags library was slightly more resistant to lighting conditions. However, the thesis could not find a dictionary that consistently outperforms other dictionaries using its generated test set. This validates the effectiveness of the ArUco library in detecting all three dictionaries.

Furthermore, the thesis found that Aztec Code is significantly more robust towards occlusion in both the encoding and the detection pattern compared to fiducial markers. But Aztec Codes lack a library that efficiently calculates positional information.

Finally, combined markers using Aztec Code and fiducial markers were created and tested successfully using a prototypical implementation. The resulting combined markers can be used to determine extrinsic camera parameters while encoding information around the actual marker, therefore saving space and allowing for more straightforward implementation and setup.

However, the thesis was unable to apply all of the advantages of Aztec Codes to custom fiducial marker dictionaries, and in particular, the robustness to occlusion of the detection center could not be preserved through minor modifications.

8 Conclusions and Future Work

8.1 Conclusions

The thesis found that fiducial markers can at low cost be effectively applied to AGV Navigation systems. They are performant and accurate while robust to lighting conditions and angles. The thesis also validates the expected low resistance to occlusion, in particular edge occlusion. Therefore it recommends adding redundancy to any fiducial marker system in an environment where occlusion can be expected.

Slightly modifying Aztec Codes to be usable as fiducials does not solve this low robustness issue. However, it allows for the creation of combined markers, which can be used to encode information and estimate the camera's pose using a single marker.

The thesis did not find consistent accuracy advantages for any of the chosen dictionaries or libraries. However, it recommends the ArUco library as the basis for future development of any fiducial marker-based system due to its extensive documentation and high flexibility.

8.2 Future Work

Based on the findings in this thesis, future work could evaluate the use of combined markers by extending the ArUco library to support Aztec Codes. The thesis presented Aztec Codes as the basis for creating combined markers. However, other bar codes could be more fitting, although Aztec Codes have been especially selected because of their minimal size and high robustness properties.

A weakness of this thesis is the lack of accurate measurement equipment, which is why the evaluation is mainly comparing inter-marker distances. Further research could use more sophisticated measurement equipment, like an infrared-based motion capturing system, to acquire more precise ground truth information and, therefore, more accurately determine detection errors. The thesis also tested markers in a realistic environment (low lighting, dirt, ...). However, future research could test the performance of fiducial markers in an even more challenging environment (factory floor; vibrations, ...). The ArUco library can also serve as an excellent starting point for further development. It would be helpful to extend it with further functionality and

create effective bindings into other languages. Work in this area can be done without licensing concerns due to the copyleft strategy of the GNU GPL licensing used for the ArUco library. As shown in this thesis, fiducial markers are excellent for AGV applications. Similarly, fiducial markers can be used in medical applications as proposed by [Sar+21a; Sar+20]. Further research could use the Design Science Methodology to explore these practical applications.

Finally, while fiducial markers systems like ArUco and Apriltags are already very sophisticated, their performance could still be improved, especially considering moving AGVs. Some areas that could be of additional interest are:

- **Enclosed markers:** Markers that add 4 smaller squares at the corners of the detection border. ArUco and OpenCv claim, that due to the higher amount of corner points, positional information can be obtained more accurately.
- **Markers optimised for Marker-Maps:** As shown in this thesis, marker maps have several advantages in regards to occlusion robustness, results validation and more. However several dictionaries were originally developed for single marker detection, which implies that optimization of libraries and dictionaries towards marker maps could still be possible.

Conclusively, most future work can focus on practical implementations of the methods proposed and tested by the literature and extending current libraries.

9 Tables and Resources

9.1 Camera Calibration

The following tables contain the complete camera intrinsics as they were obtained using the OpenCV and ArUco calibration methods.

Table 9.1: Camera Matrix OpenCV

581,541128540775	0	320,422667597193
0	581,517860082219	242,241448860773
0	0	1

Table 9.2: Distortion Coefficients OpenCV

0,162122828365548
0,522180865685194
-0,00149567796552038
-0,00034946935243907
-4,32208263744623

Table 9.3: Camera Matrix ArUco

566,009272879317	0	320,404549268637
0	565,974681786813	243,100316589768
0	0	1

Table 9.4: Distortion Coefficients ArUco

0,172885304434444
-0,511949868131412
-0,00143915984385438
-0,00150696093849475
0,377441880643545

9.2 Accuracy Test Results

The following tables contains Translation Vectors and δ values obtained as part of the accuracy tests. Results marked with * are less accurate, since they compare two markers that were not directly adjacent to each other, since the directly adjacent marker was not detected. These less accurate results were excluded from the average error calculation as explain in Chapter 6. Rotation vectors were also detected, but not listed here, since they are not relevant for the δ values. Image IDs 0-3 are without occlusion and worsening lighting conditions. Image IDs 4-6 are with occlusion and worsening lighting conditions.

ID	Translation Vector			δ x	δ y
0	x	y	z	δ x	δ y
0	-10.692	4.67392	43.0428	15.00442	11.76328
1	-10.308	-10.3305	41.7971	15.00442	11.74333
2	1.07128	4.95819	42.771	14.97729	11.87812
3	1.43533	-10.0191	41.5229	14.97729	11.85317
4	12.9494	5.28178	42.5199	14.98689	11.87812
5	13.2885	-9.70511	41.1829	14.98689	11.85317
1	x	y	z	δ x	δ y
0	-10.8866	4.91056	43.2349	14.96166	11.722962
1	-10.8426	-10.0511	41.9642	14.96166	11.698929
2	0.836362	4.91758	43.0489	14.96978	11.849338
3	0.856329	-10.0522	41.8859	14.96978	11.871671
4	12.6857	4.95452	42.7771	14.97652	11.849338
5	12.728	-10.022	41.6058	14.97652	11.871671
2	x	y	z	δ x	δ y
2	0.827498	6.02728	43.0549	14.9528	11.878202
3	0.869909	-8.92552	41.7806	14.9528	11.852191
4	12.7057	6.09493	42.7146	14.95635	11.878202
5	12.7221	-8.86142	41.4371	14.95635	11.852191
3	x	y	z	δ x	δ y
No Markers Detected, too dark					
4	x	y	z	δ x	δ y
0	-13.2723	4.97267	45.3638	15.32007	12.598412
1	-11.7685	-10.3474	41.5508	15.32007	11.6949058
2	-0.673888	5.10814	42.1558	14.97154	11.857888
3	-0.0735942	-9.8634	41.244	14.97154	11.6949058
4	11.184	5.61295	42.0086	NA	11.857888

5	x	y	z	δ x	δ y
2	-0.674688	5.13601	42.227	15.05819	NA
3	-0.015975	-9.92218	41.531	15.05819	11.829225
5	11.8452	-9.36787	41.144	14.50388	11.829225
6	x	y	z	δ x	δ y
2	-0.668788	5.13035	42.2304	NA	NA
5	11.8227	-9.34568	41.0794	NA	NA

Table 9.10: Apriltag Pose Detection, high resolution, Test Set 0-6, values with * are less accurate

ID	Translation Vector			δ x	δ y
0	x	y	z	δ x	δ y
0	-9.97436381	5.30690696	43.88017924	14.989098	11.8106507
1	-9.53052313	-9.68219104	41.97297333	15.26523218	11.69081275
2	1.83628689	5.58304114	43.86869729	14.95551041	11.81486901
3	2.16028962	-9.37246927	42.02404992	15.22848541	11.83227901
4	13.6511559	5.85601614	43.04460544	14.96579355	11.81486901
5	13.99256863	-9.10977741	41.64974219	14.96579355	11.83227901
1	x	y	z	δ x	δ y
0	-10.08382684	5.48023273	43.75149749	14.88627192	11.67958907
1	-10.06716091	-9.40603919	42.1765546	14.92955155	11.64985092
2	1.59576223	5.52351236	44.04294284	14.90551514	11.80061014
3	1.58269001	-9.38200278	42.31785176	14.9292381	11.77743908
4	13.39637237	5.54723532	43.33047841	14.92339932	11.80061014
5	13.36012909	-9.376164	41.86160358	14.92339932	11.77743908
2	x	y	z	δ x	δ y
0	-10.09000307	6.57457467	43.8389588	14.87802405	11.68363815
1	-10.027213	-8.30344938	42.21303978	14.93155102	11.61932085
2	1.59363508	6.62810164	43.93147631	14.90041758	11.78536659
3	1.59210785	-8.27231594	42.31207766	14.94061571	11.73019392
4	13.37900167	6.66829977	43.16156899	14.88625435	11.78536659
5	13.32230177	-8.21795458	41.65261614	14.88625435	11.73019392
3	x	y	z	δ x	δ y
0	-10.04233168	5.57451649	43.54120623	NA	11.66090273
2	1.61857105	5.62480574	43.75249921	NA	11.66090273
4	x	y	z	δ x	δ y

1	-11.01353272	-9.72618465	41.72514459	15.44673309	11.66195248
2	0.07777716	5.72054844	43.20336612	14.96597431	11.85544392
3	0.64841976	-9.24542587	41.864575	15.45733705	11.66195248
4	11.93322108	6.21191118	42.74649222	15.45733705	11.85544392
5	x	y	z	δ x	δ y
1	-10.95270499	-9.7314573	41.63268376	15.46150563	11.65056391
2	0.08038733	5.73004833	43.12538136	14.97001218	11.75308377
3	0.69785892	-9.23996385	41.87607362	15.45370306	11.65056391
4	11.8334711	6.21373921	42.3784308	15.45370306	11.75308377
5	12.4370476	-8.70512077	41.28390887	14.91885998	11.73918868
6	x	y	z	δ x	δ y
1	-10.92403574	-9.70027543	41.51198383	15.41808906	11.62339629
2	0.08170115	5.71781363	43.02093684	14.91464409	NA
3	0.69936055	-9.19683046	41.69875062	14.91464409	11.62339629
5	12.39812443	-8.67613872	41.14833586	NA	12.31642328

Table 9.11: Apriltag Pose Detection, high resolution, Test Set 0-6, with Apriltag Library, values with * are less accurate

ID	Translation Vector			δ x	δ y
0	x	y	z	δ x	δ y
0	-11.94842497	9.63871178	42.66961441	15.03775533	11.7089955
1	-11.63980608	-5.39904355	42.4470419	15.31113497	11.71354628
2	-0.23942947	9.91209142	42.72815078	15.05125565	11.81353726
3	0.0737402	-5.13916423	42.56748811	15.2656829	11.80438347
4	11.57410779	10.12651867	42.16824895	15.01028762	11.81353726
5	11.87812367	-4.88376895	41.98029431	15.01028762	11.80438347
1	x	y	z	δ x	δ y
0	-11.92074127	9.61894862	42.74371944	14.99193377	11.68223879
1	-11.6071453	-5.37298515	42.49310558	15.24555969	11.67066487
2	-0.23850248	9.87257454	42.72215416	14.98429621	11.75917375
3	0.06351957	-5.11172167	42.53767899	15.19960112	11.74498957
4	11.52067127	10.08787945	42.16417131	14.95094794	11.75917375
5	11.80850914	-4.86306849	41.92801565	14.95094794	11.74498957
2	x	y	z	δ x	δ y
0	-11.91311989	9.52865583	42.96637186	15.03852187	11.77869672
1	-11.56311279	-5.50986604	42.3420453	15.17712912	11.66857478
2	-0.13442317	9.66726388	42.84089532	14.99399101	11.84067683

3	0.10546199	-5.32672793	42.70192812	15.22723844	11.71782893
4	11.70625366	9.90051051	42.4612081	14.98941343	11.84067683
5	11.82329092	-5.08890292	41.76133321	14.98941343	11.71782893
3	x	y	z	δ x	δ y
0	-12.43587638	9.13081604	42.27170686	14.92423471	11.6261766
1	-11.83016789	-5.79341867	41.93099463	15.38882122	11.59914407
2	-0.80969978	9.59540255	42.37945926	14.93933326	11.7472636
3	-0.23102382	-5.34393071	42.22097346	15.38377461	11.72556766
4	10.93756382	10.0398439	41.9039298	14.92905821	11.7472636
5	11.49454384	-4.88921431	41.66031633	14.92905821	11.72556766
4	x	y	z	δ x	δ y
1	-11.28633997	-9.92350787	42.08089764	15.44785671	11.64996099
2	-0.25924503	5.52434884	43.97988334	14.94688379	11.81174403
3	0.36362102	-9.42253495	42.25432596	15.43319112	11.64996099
4	11.552499	6.01065617	43.29962793	15.43319112	11.81174403
5	x	y	z	δ x	δ y
1	-11.58289785	-5.3816019	42.43559367	15.25117159	11.67022147
2	-0.23002167	9.86956969	42.81510067	14.99047952	NA
3	0.08732362	-5.12090983	42.57844983	14.99047952	11.67022147
5	11.87679553	-4.88573396	42.14935623	NA	11.78947191
6	x	y	z	δ x	δ y
1	-11.56444873	-5.52333253	42.29054792	15.23905295	11.63635911
2	-0.23862262	9.71572042	42.91967322	14.99623596	11.8248797
3	0.07191038	-5.28051554	42.55349161	15.2511634	11.63635911
4	11.58625708	9.97064786	42.39686751	15.2511634	11.8248797
5	11.80076903	-5.01079617	41.82413605	14.98144403	11.72885865

Table 9.12: Apriltag Pose Detection, low resolution, Test Set 0-6, with Apriltag Library, values with * are less accurate

9.3 Calibration Tests

ID	x	y	z	δx	δ y
ArUco,Aruco Calibration,low,res					
0	11.9703	-5.14995	41.6982	11.8774099	15.06807
1	11.6562	9.91812	41.6422	11.798589	15.06807
2	0.0928901	-5.32323	42.2339	11.7870901	15.10366
3	-0.142389	9.78043	42.2339	11.833111	15.10366

4	-11.6942	-5.51376	41.9371	11.7870901	15.08901
5	-11.9755	9.57525	42.3761	11.833111	15.08901
ArUco,OpenCV Calibration, low res				0.1774099	0.06807
0	11.9729	-5.08983	43.0484	11.8817446	15.08401
1	11.6648	9.99418	43.0424	11.810917	15.08401
2	0.0911554	-5.26058	42.9518	11.7924554	15.10539
3	-0.146117	9.84481	43.473	11.849483	15.10539
4	-11.7013	-5.45163	43.2159	11.7924554	15.09987
5	-11.9956	9.64824	43.734	11.849483	15.09987
ArUco, ArUco Calibration, high res				0.1817446	0.08401
0	13.3793	-9.62243	41.5222	11.80845	14.98887
1	12.9804	5.36644	42.6017	11.78858	14.98887
2	1.57085	-9.93038	41.8503	11.77235	14.95806
3	1.19182	5.02768	42.9761	11.77202	14.95806
4	-10.2015	-10.2408	42.0901	11.77235	14.96996
5	-10.5802	4.72916	43.3172	11.77202	14.96996
ArUco, OpenCV Calibration, high res				0.10845	0.01113
0	14.1225	-9.02539	42.8034	11.8069	14.99386
1	13.7542	5.96847	43.9248	11.79764	14.99386
2	2.3156	-9.33425	43.02	11.77196	14.96959
3	1.95656	5.63534	44.1791	11.76011	14.96959
4	-9.45636	-9.64906	43.4505	11.77196	14.98183
5	-9.80355	5.33277	44.5877	11.76011	14.98183
Apriltag, ArUco Calibration, low res				0.1069	0.00614
0	-12.0341	9.58047	42.329	11.76956	15.08427
1	-11.717	-5.5038	42.0114	11.750948	15.08427
2	-0.26454	9.84982	42.0768	11.84524	15.10559
3	0.033948	-5.25577	41.7729	11.912752	15.10559
4	11.5807	10.0004	41.565	11.84524	14.99944
5	11.9467	-4.99904	41.6023	11.912752	14.99944
Apriltag, OpenCV Calibraiton, low res				0.06956	0.08427
0	-12.0542	9.65335	43.6838	11.786075	15.09485
1	-11.7244	-5.4415	43.2917	11.7563289	15.09485
2	-0.268125	9.9152	43.3194	11.858125	15.10853
3	0.0319289	-5.19333	42.9343	11.9150711	15.10853
4	11.59	10.0773	42.9673	11.858125	15.01559
5	11.947	-4.93829	42.9412	11.9150711	15.01559
Apriltag, ArUco Calibration, high res				0.086075	0.09485

0	-10.692	4.67392	43.0428	11.76328	15.00442
1	-10.308	-10.3305	41.7971	11.74333	15.00442
2	1.07128	4.95819	42.771	11.87812	14.97729
3	1.43533	-10.0191	41.5229	11.85317	14.97729
4	12.9494	5.28178	42.5199	11.87812	14.98689
5	13.2885	-9.70511	41.1829	11.85317	14.98689
Apriltag, OpenCV Calibration, high res				0.06328	0.00442
0	-9.92645	5.27777	44.3403	11.75801	15.02392
1	-9.57052	-9.74615	43.1728	11.74426	15.02392
2	1.83156	5.56192	43.9573	11.89204	14.98646
3	2.17374	-9.42454	42.6707	11.85626	14.98646
4	13.7236	5.88203	43.8466	11.89204	14.99733
5	14.03	-9.1153	42.4675	11.85626	14.99733
ArTag, ArUco Calibration, low res				0.05801	0.02392
0	12.0007	-5.11172	41.6053	11.798501	15.06758
1	11.6884	9.95586	41.7081	11.7431718	15.06758
2	0.202199	-5.30741	41.625	11.870799	15.04205
3	-0.0547718	9.73464	41.8959	11.8961282	15.04205
4	-11.6686	-5.54541	42.0393	11.870799	15.09992
5	-11.9509	9.55451	42.3458	11.8961282	15.09992
ArTag, OpenCV Calibration, low res				0.098501	0.06758
0	11.9972	-5.05009	42.9347	11.796943	15.08349
1	11.6985	10.0334	43.1174	11.75726	15.08349
2	0.200257	-5.24515	42.7817	11.876257	15.04267
3	-0.05876	9.79752	43.1218	11.91514	15.04267
4	-11.676	-5.48322	43.3205	11.876257	15.11304
5	-11.9739	9.62982	43.7146	11.91514	15.11304
ArTag, ArUco Calibration, high res				0.096943	0.08349
0	-12.8485	5.87405	42.5034	11.58439	14.85512
1	-10.7742	-8.98107	42.2162	11.57926	14.85512
2	-1.26411	7.50077	42.2181	11.72441	14.8759
3	0.80506	-7.37513	41.9602	11.72984	14.8759
4	10.4603	9.13832	42.0633	11.72441	14.87046
5	12.5349	-5.73214	41.794	11.72984	14.87046
ArTag, OpenCV Calibration, high res				0.11561	0.14488
0	-12.0905	6.46795	43.8844	11.578602	14.85223
1	-10.0246	-8.38428	43.5518	11.57462	14.85223
2	-0.511898	8.10024	43.4457	11.735298	14.87874

3	1.55002	-6.7785	43.1089	11.73598	14.87874
4	11.2234	9.73259	43.3906	11.735298	14.87168
5	13.286	-5.13909	43.0585	11.73598	14.87168
				0.121398	0.14777

Table 9.13: Calibration Test Results, Marker Id, Translation Vector, Inter-marker distances x and y, average error for every test image

ID	x	y	z	Norm	measured	Difference
Apriltags17 = Apriltags, with Aruco, Angle:17°						
0	-9.17594	3.33656	47.4371	48.43148732	46.5	1.931487319
2	2.54478	3.17342	47.3146	47.48913426	45	2.489134261
3	2.22491	-11.0811	42.381	43.86217234	42.5	1.362172344
4	14.4411	2.97866	47.2258	49.47417478	47	2.474174781
5	13.7979	-10.9982	41.0486	44.68030897	44	0.6803089695
Apriltags20						1.637582728
0	12.2905	-6.79977	42.9037	45.14444302	45	0.1444430245
1	12.6377	7.0995	49.388	51.47124348	50.5	0.9712434816
2	0.742449	-6.86727	43.9686	44.50784665	44	0.5078466535
3	0.880549	7.20725	49.8829	50.40856605	48	2.408566053
4	-11.1146	-6.84291	44.0893	45.98071471	45	0.9807147065
5	-11.1372	7.23637	50.3119	52.03546441	50	2.035464409
Apriltags30						1.174713055
0	-11.5486	-1.86162	51.5992	52.90853647	49.5	3.408536472
1	-11.458	-14.7691	43.3482	47.20676352	44.5	2.70676352
2	0.306246	-1.79952	51.2092	51.24172346	48	3.241723463
3	0.338508	-14.7377	43.3662	45.80329346	43	2.803293456
4	12.1837	-1.81176	51.2477	52.70722697	50	2.707226974
5	12.1827	-14.6134	43.0921	47.10518784	44.5	2.605187838
ArUco17						2.912121954
0	-9.2293	3.29661	47.3491	48.35271333	46.5	1.852713334
1	-9.58392	-10.9581	42.4408	44.86817338	44	0.868173384
2	2.58545	3.1197	47.2442	47.41762872	45	2.41762872
3	2.26247	-11.0747	42.0147	43.50865163	42.5	1.00865163
4	14.3769	2.92511	46.8059	49.051440316	47	2.051440315
5	14.1106	-11.3102	41.9541	45.68562316	44	1.685623157
ArUco20						1.647371757
0	-11.0801	7.16705	49.8288	51.54652783	50	1.546527828

1	-11.1275	-6.90692	44.0795	45.9840094	45	0.9840093988
2	0.769758	7.15645	49.721	50.23927891	48	2.239278906
3	0.677554	-6.91512	44.0182	44.56321235	44	0.5632123537
4	12.6906	7.23842	49.8865	51.9818135	50.5	1.4818135
5	12.5144	-6.93732	43.9031	46.17595484	45	1.175954844
ArUco30						1.331799472
0	-11.4379	-1.86436	51.0661	52.36456783	49.5	2.864567828
1	-11.4711	-14.707	43.1187	46.97982837	44.5	2.479828372
2	0.334699	-1.85774	51.2041	51.23888248	48	3.238882483
3	0.323177	-14.7718	43.3212	45.77157293	43.5	2.271572925
4	12.1007	-1.80706	50.8209	52.272902	50	2.272901997
5	12.0806	-14.6423	42.9016	46.91359215	44.5	2.413592148
ArTag17						2.590224292
0	14.0546	-11.118	41.6958	45.38371341	44	1.383713409
1	14.468	3.01688	47.0748	49.34026129	47	2.340261288
2	2.40408	-11.0111	42.2058	43.6847053	42.5	1.184705304
3	2.71619	3.21218	47.4926	47.6785365	45	2.678536505
4	-9.47918	-10.8601	42.4822	44.86127442	44	0.8612744171
5	-9.1294	3.35245	47.3797	48.36755976	46.5	1.867559763
ArTag20						1.719341781
0	12.4101	-7.77127	42.9917	45.41684146	45	0.4168414612
1	12.6505	6.24791	49.2854	51.26521416	50.5	0.7652141591
2	0.754176	-7.81413	43.6019	44.30299192	44	0.302991916
3	0.839685	6.22908	49.2922	49.69132207	48	1.691322073
4	-11.1352	-7.82712	43.8296	45.89433868	45	0.894338678
5	-11.0648	6.2588	49.4699	51.0771317	50	1.077131698
ArTag30						0.8579733309
0	11.9863	-14.4667	42.3434	46.32407927	44.5	1.824079269
1	12.3985	-1.84085	52.0935	53.58026011	50	3.580260108
2	0.404495	-14.7788	43.3297	45.78252339	43.5	2.282523388
3	0.398768	-1.88798	51.6594	51.69542623	48	3.695426227
4	-11.456	-14.7267	51.4541	52.75827212	49.5	3.258272118
5	-11.5013	-1.90551	51.4541	52.75827212	49.5	3.258272118
Apriltag217 = Apriltag, with Apriltag, Angle 17°						2.862517049
0	-9.0645	3.3560	46.6974	47.6873	46.5	1.187310481
1	-9.3812	-10.6918	41.2937	43.6749	44	0.3251036698
2	2.5817	3.2293	47.2313	47.4119	45	2.411900177
3	2.2269	-10.9279	41.4689	42.9424	42.5	0.4423896676

4	14.2680	2.9719	46.0062	48.2596	47	1.259552936
5	14.0451	-11.1608	41.1608	44.9004	44	0.9003639868
Apriltag220						1.087770153
0	12.4253	-6.8246	42.7917	45.0788	45	0.07877012753
1	12.5127	7.0524	48.2617	50.3537	50.5	0.1462568169
2	0.7832	-6.8135	43.6408	44.1764	44	0.1764016756
3	0.9128	7.2208	49.4564	49.9891	48	1.989113556
4	-10.9683	-6.7294	43.0798	44.9607	45	0.03931296602
5	-10.8939	7.1906	48.9338	50.6448	50	0.6448259345
Apriltag230						0.512446846
0	-11.2580	-1.7636	50.0712	51.3515	49.5	1.851480762
1	-11.2968	-14.5636	42.0822	45.9416	44.5	1.441587423
2	0.3472	-1.7532	51.1776	51.2089	48	3.208891492
3	0.3603	-14.5681	42.3792	44.8147	43	1.814757495
4	12.0332	-1.7163	49.9853	51.4420	50	1.442034134
						1.951750261

Table 9.14: Angles Table, Marker Id, Translation Vector, Norm of Translation Vector, Measured Distance to Camera, Difference, Average for every test image below differences

ID	Translation Vector			δ x	δ y
0	x	y	z	δ x	δ y
0	11.9703	-5.14995	41.6982	15.06807	11.8774099
1	-2.16071	-2.21179	0.0991085	15.06807	11.513811
2	0.0928901	-5.32323	41.7921	15.10366	11.6013099
3	-0.142389	9.78043	42.2339	15.10366	11.833111
4	-11.6942	-5.51376	41.9371	15.08901	11.6013099
5	-11.9755	9.57525	42.3761	15.08901	11.833111
1	x	y	z	δ x	δ y
0	-10.9272	9.54004	42.3543	15.02055	11.792229
1	-10.5778	-5.48051	41.9295	15.02055	11.74628
2	0.865029	9.84586	42.3211	15.07614	11.772171
3	1.16848	-5.23028	41.7469	15.07614	11.83512
4	12.6372	10.0573	41.755	15.03835	11.772171
5	13.0036	-4.98105	41.6183	15.03835	11.83512
2	x	y	z	δ x	δ y
2	0.84388	9.77593	42.05	15.03213	N/A
3	1.18467	-5.2562	41.7938	15.03213	N/A
3	x	y	z	δ x	δ y
No Markers Detected, too dark					
4	x	y	z	δ x	δ y
0	-11.6111	5.40239	42.7404	14.97123	11.74563
1	-9.60672	-9.56884	42.2379	14.97123	11.7266
2	0.13453	6.94515	42.3914	14.96452	11.73917
3	2.11988	-8.01937	42.1669	14.96452	11.74622
4	11.8737	8.52261	42.2281	14.97525	11.73917
5	13.8661	-6.45264	41.9265	14.97525	11.74622
5	x	y	z	δ x	δ y
1	-10.6127	-5.49441	42.0351	15.33707*	11.77884
2	0.867051	9.84266	42.3076	15.06678	12.099749*
3	1.16614	-5.22412	41.8086	15.06678	11.77884
5	12.9668	-4.96333	41.5057	14.80599*	11.80066
6	x	y	z	δ x	δ y
2	0.857996	9.81809	42.1977	NA	NA
Very Dark Light Conditions					

Table 9.5: ArUco Pose Detection, low resolution, Test Set 0-6, values with * are less accurate

ID	Translation Vector			δ x	δ y
0	x	y	z	δ x	δ y
0	-9.87094	4.98457	43.2234	14.97046	11.76913
1	-9.90771	-9.98589	42.006	14.97046	11.76477
2	1.89819	4.96048	42.9483	14.95089	11.79491
3	1.85706	-9.99041	41.7375	14.95089	11.81124
4	13.6931	4.97259	42.651	14.98879	11.79491
5	13.6683	-10.0162	41.4389	14.98879	11.81124
1	x	y	z	δ x	δ y
0	-9.8709	4.98457	43.2233	14.97057	11.76897
1	-9.90798	-9.986	42.0068	14.97057	11.76507
2	1.89807	4.96048	42.9473	14.95045	11.79443
3	1.85709	-9.98997	41.7357	14.95045	11.81081
4	13.6925	4.97232	42.6492	14.98832	11.79443
5	13.6679	-10.016	41.4379	14.98832	11.81081
2	x	y	z	δ x	δ y
0	-9.88411	4.99634	43.2957	14.99382	11.74188
2	1.90789	4.97666	43.19	14.97414	11.75771
3	1.85777	-9.99748	41.7684	14.97414	11.74188
5	13.6656	-9.99124	41.3959	14.9679	11.75771
3	x	y	z	δ x	δ y
No Markers Detected, too dark					
4	x	y	z	δ x	δ y
1	11.2897	5.51216	41.6572	NA	11.851962
3	-0.562262	5.05261	41.8393	NA	11.851962
5	x	y	z	δ x	δ y
1	11.1819	5.46994	41.806	15.88904*	NA
4	-11.8012	-10.4191	41.5922	15.88904*	NA
6	x	y	z	δ x	δ y
3	0.400165	5.24288	42.2373	NA	NA

Table 9.6: ArUco Pose Detection, high resolution, Test Set 0-6, values with * are less accurate

ID	Translation Vector			δ x	δ y
0	x	y	z	δ x	δ y
0	12.0007	-5.11172	41.6053	15.06758	11.798501
1	11.6884	9.95586	41.7081	15.06758	11.6336282
2	0.202199	-5.30741	41.625	15.04205	11.466401
3	-0.0547718	9.73464	41.8959	15.04205	11.8961282
4	-11.6686	-5.54541	42.0393	15.09992	11.8961282
5	-11.9509	9.55451	42.3458	15.09992	11.8961282
1	x	y	z	δ x	δ y
0	-11.9323	9.52853	42.4378	15.03662	11.755657
1	-11.6002	-5.50809	41.8477	15.03662	11.531375
2	-0.176643	9.68746	42.0313	15.01758	11.503357
3	0.068825	-5.33012	41.9097	15.01758	11.882675
4	11.68	9.911	41.9221	15.05341	11.503357
5	11.9515	-5.14241	41.6743	15.05341	11.882675
2	x	y	z	δ x	δ y
2	-0.179785	9.74349	41.9081	15.04183	NA
3	0.0632672	-5.29834	41.3734	15.04183	11.8383328
5	11.9016	-5.10625	41.3145	14.84974	11.8383328
3	x	y	z	δ x	δ y
No Markers Detected, too dark					
4	x	y	z	δ x	δ y
0	12.9992	-10.3864	41.5757	15.04791	11.80267
1	13.2375	4.66151	42.6928	15.04791	11.75347
2	1.19653	-10.1581	41.7065	15.0217	11.85113
3	1.48403	4.8636	43.0892	15.0217	11.75347
4	-10.6546	-9.98008	41.9834	NA	11.85113
5	x	y	z	δ x	δ y
1	-11.6644	-5.52304	42.0776	15.21224	11.611335
2	-0.175145	9.6892	42.056	15.21224	NA
3	0.053065	-5.33557	41.9624	15.02477	11.611335
6	x	y	z	δ x	δ y
2	-0.177776	9.72775	41.8309	NA	NA

Table 9.7: ArTag Pose Detection, low resolution, Test Set 0-6, values with * are less accurate

ID	Translation Vector			δ x	δ y
0	x	y	z	δ x	δ y
0	-12.8485	5.87405	42.5034	14.85512	11.58439
1	-10.7742	-8.98107	42.2162	14.85512	11.57926
2	-1.26411	7.50077	42.2181	14.8759	11.72441
3	0.80506	-7.37513	41.9602	14.8759	11.72984
4	10.4603	9.13832	42.0633	14.87046	11.72441
5	12.5349	-5.73214	41.794	14.87046	11.72984
1	x	y	z	δ x	δ y
0	-12.8455	5.86879	42.5761	14.8565	11.57888
1	-10.7801	-8.98771	42.3089	14.8565	11.578553
2	-1.26662	7.50196	42.3169	14.87734	11.73062
3	0.798453	-7.37538	41.9951	14.87734	11.773347
4	10.464	9.15354	42.2003	14.89623	11.73062
5	12.5718	-5.74269	42.0222	14.89623	11.773347
2	x	y	z	δ x	δ y
1	-10.7882	-9.00059	42.3556	16.50359*	11.591317
2	-1.27122	7.503	42.3262	14.89311	NA
3	0.803117	-7.39011	42.138	14.89311	11.591317
5	12.524	-5.72879	41.8393	13.23179*	11.720883
3	x	y	z	δ x	δ y
No Markers Detected, too dark					
4	x	y	z	δ x	δ y
0	11.6986	-9.55788	41.0575	15.00876	11.7664629
1	11.2111	5.45088	41.931	15.00876	11.740178
2	-0.0678629	-9.90804	41.2742	14.96852	11.8232371
3	-0.529078	5.06048	42.1258	14.96852	11.740178
5	-11.8911	-10.3041	41.6241	15.36458	11.8232371
5	x	y	z	δ x	δ y
1	12.1707	6.74339	42.0369	15.47918	11.743528
2	1.05224	-8.73579	41.2875	15.47918	11.81264
3	0.427172	6.23357	42.1262	15.49448	NA
4	-10.7604	-9.26091	41.5801	15.49448	11.81264
6	x	y	z	δ x	δ y
2	0.160453	-9.82514	41.467	15.01297	NA
3	-0.521023	5.18783	42.1383	15.01297	NA

Table 9.8: ArTag Pose Detection, high resolution, Test Set 0-6, values with * are less accurate

ID	Translation Vector			δ x	δ y
0	x	y	z	δ x	δ y
0	-12.0168	9.55747	42.4286	15.04594	11.747924
1	-11.6939	-5.48847	42.0573	15.04594	11.6498835
2	-0.268876	9.80403	42.0968	15.051	11.752776
3	0.0440165	-5.24697	41.9453	15.051	11.8968835
4	11.4839	9.93134	41.4175	14.93408	11.752776
5	11.9409	-5.00274	41.721	14.93408	11.8968835
1	x	y	z	δ x	δ y
0	-12.0145	9.5557	42.4203	15.04264	11.74582
1	-11.6918	-5.48694	42.0489	15.04264	11.6473223
2	-0.26868	9.80243	42.0895	15.04935	11.75178
3	0.0444777	-5.24692	41.9492	15.04935	11.8955223
4	11.4831	9.92951	41.4148	14.93146	11.75178
5	11.94	-5.00195	41.7177	14.93146	11.8955223
2	x	y	z	δ x	δ y
0	-11.9119	9.4083	42.3024	15.04541	11.742842
1	-11.6619	-5.63711	42.0397	15.04541	11.5972418
2	-0.169058	9.57689	42.0593	15.02856	11.848658
3	0.0646582	-5.45167	41.8442	15.02856	11.8102418
4	11.6796	9.80899	41.8693	15.03551	11.848658
5	11.8749	-5.22652	41.3786	15.03551	11.8102418
3	x	y	z	δ x	δ y
No Markers Detected, too dark					
4	x	y	z	δ x	δ y
0	-12.0543	4.94107	43.1244	15.00347	11.759941
1	-11.3661	-10.0624	41.814	15.00347	11.689329
2	-0.294359	5.41878	42.9923	15.01418	11.911759
3	0.323229	-9.5954	41.8082	15.01418	11.689329
4	11.6174	5.96004	42.9737	NA	11.323041
5	x	y	z	δ x	δ y
1	-11.6392	-5.4935	41.9647	15.29195	11.5850607
2	-0.25288	9.79845	42.1297	15.29195	NA
3	0.0541393	-5.24264	41.8443	NA	11.5850607
6	x	y	z	δ x	δ y
1	-11.6673	-5.65176	42.0018	15.26823	11.6393577
2	-0.262155	9.61647	42.1098	15.26823	NA
3	0.0279423	-5.39698	41.838	NA	11.6393577

Table 9.9: Apriltag Pose Detection, low resolution, Test Set 0-6, values with * are less accurate

List of Figures

List of Tables

Bibliography

[AHH10] B. Atcheson, F. Heide, and W. Heidrich. "CALTag: High Precision Fiducial Markers for Camera Calibration." In: *VMV*. 2010.

[ArTa] ArToolKit. *ArToolKit Official Documentation*. http://www.hitl.washington.edu/artoolkit/documentation/. Accessed: 2021-6-23.

[ArTb] ArToolKit. *ArToolKiteX Official Website*. http://www.artoolkitx.org. Accessed: 2021-6-23.

[Asp] Aspose. *Aspose Barcode Reader*. https://products.aspose.app/barcode/recognize/aztec. Accessed: 2021-8-13.

[Ber+11] F. Bergamasco, A. Albarelli, E. Rodolà, and A. Torsello. "RUNE-Tag: A high accuracy fiducial marker with strong occlusion resilience." In: *CVPR 2011*. 2011, pp. 113–120. DOI: 10.1109/CVPR.2011.5995544.

[BOO08] F. Bonin-Font, A. Ortiz, and G. Oliver. "Visual Navigation for Mobile Robots: A Survey." In: *Journal of Intelligent and Robotic Systems* 53.3 (May 2008), p. 263. ISSN: 1573-0409. DOI: 10.1007/s10846-008-9235-4.

[CB14] T. Collins and A. Bartoli. "Infinitesimal Plane-Based Pose Estimation." In: *International Journal of Computer Vision* 109 (2014), pp. 252–286.

[Ces+15] D. Cesar, C. Gaudig, M. Fritsche, M. Reis, and F. Kirchner. "An evaluation of artificial fiducial markers in underwater environments." In: May 2015. DOI: 10.1109/OCEANS-Genova.2015.7271491.

[Dep] Ø. U. C. Dep. for Information Technology. *Apriltag Github*. http://www.it.hiof.no/~borres/j3d/math/homo/p-homo.html. Accessed: 2021-6-23.

[duc] duckietown. *Apriltag Python Binding Github*. https://github.com/duckietown/lib-dt-apriltags. Accessed: 2021-6-23.

[Fia04] M. Fiala. "ARTag, An Improved Marker System Based on ARToolkit." In: (Jan. 2004). DOI: 10.4224/5763247.

[Fia05] M. Fiala. "ARTag, a fiducial marker system using digital techniques." In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 2. 2005, 590–596 vol. 2. DOI: 10.1109/CVPR.2005.74.

[Gar+14] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez. "Automatic generation and detection of highly reliable fiducial markers under occlusion." In: *Pattern Recognition* 47 (June 2014), pp. 2280–2292. DOI: 10.1016/j.patcog.2014.01.005.

[Gar+15] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer. "Generation of fiducial marker dictionaries using Mixed Integer Linear Programming." In: *Pattern Recognition* 51 (Oct. 2015). DOI: 10.1016/j.patcog.2015.09.023.

[Hev+04] A. Hevner, A. R, S. March, S. T, Park, J. Park, Ram, and Sudha. "Design Science in Information Systems Research." In: *Management Information Systems Quarterly* 28 (Mar. 2004), pp. 75–.

[HS] K. Hata and S. Savarese. *CS231A Course Notes 1: Camera Models*. https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf. Accessed: 2021-8-13.

[KB99] 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)*. 1999, pp. 85–94. DOI: 10.1109/IWAR.1999.803809.

[KHO19] M. Krogius, A. Haggenmiller, and E. Olson. "Flexible Layouts for Fiducial Tags." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2019.

[LSS21] Y. Liu, H. Schofield, and J. Shan. "Navigation of a Self-Driving Vehicle Using One Fiducial Marker." In: *CoRR* abs/2104.12954 (2021). arXiv: 2104.12954.

[Mon+17] V. M. Mondéjar-Guerra, S. Garrido-Jurado, R. Muñoz-Salinas, M. Marín-Jiménez, and R. Medina-Carnicer. "Robust identification of fiducial markers in challenging conditions." In: *Expert Systems with Applications* 93 (Oct. 2017). DOI: 10.1016/j.eswa.2017.10.032.

[Ols11] E. Olson. "AprilTag: A robust and flexible visual fiducial system." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[Pea+21] J. B. Peace, E. Psota, Y. Liu, and L. C. Pérez. *E2ETag: An End-to-End Trainable Method for Generating and Detecting Fiducial Markers*. 2021. arXiv: 2105.14184 [cs.CV].

[RMM18] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer. "Speeded up detection of squared fiducial markers." In: *Image and Vision Computing* 76 (2018), pp. 38–47. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2018.05.004.

[Rob] A. Robotics. *Apriltag Github*. https://github.com/AprilRobotics/apriltag. Accessed: 2021-6-23.

[Sar+20] H. Sarmadi, R. Muñoz-Salinas, M. Á. Berbís, A. Luna, and R. Medina-Carnicer. *Joint Scene and Object Tracking for Cost-Effective Augmented Reality Assisted Patient Positioning in Radiation Therapy*. Oct. 2020.

[Sar+21a] H. Sarmadi, R. Muñoz-Salinas, M. Á. Berbís, A. Luna, and R. Medina-Carnicer. *3D Reconstruction and Alignment by Consumer RGB-D Sensors and Fiducial Planar Markers for Patient Positioning in Radiation Therapy*. Mar. 2021.

[Sar+21b] H. Sarmadi, R. Muñoz-Salinas, M. Á. Berbís, and R. Medina-Carnicer. *Simultaneous Multi-View Camera Pose Estimation and Object Tracking with Square Planar Markers*. Mar. 2021.

[Sha+20] K. Shabalina, A. Sagitov, L. Sabirova, H. Li, and E. Magid. "ARTag, AprilTag and CALTag Fiducial Systems Comparison in a Presence of Partial Rotation: Manual and Automated Approaches." In: Jan. 2020, pp. 536–558. ISBN: 978-3-030-11291-2. DOI: 10.1007/978-3-030-11292-9_27.

[Sim96] H. A. Simon. *The Sciences of the Artificial (3rd Ed.)* Cambridge, MA, USA: MIT Press, 1996. ISBN: 0262691914.

[SMP05] T. Svoboda, D. Martinec, and T. Pajdla. "A Convenient Multi-Camera Self-Calibration for Virtual Environments." In: *Presence* 14 (Aug. 2005), pp. 407–422. DOI: 10.1162/105474605774785325.

[SP06] G. Schweighofer and A. Pinz. "Robust Pose Estimation from a Planar Target." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.12 (2006), pp. 2024–2030. DOI: 10.1109/TPAMI.2006.252.

[Wan+19] Y. Wang, S. Liu, L. Wang, and J. Huang. "AGV Navigation Based on AprilTags2 Auxiliary Positioning." In: *2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. 2019, pp. 1–8. DOI: 10.1109/CISP-BMEI48845.2019.8965978.

[Wie+06] R. Wieringa, N. Maiden, N. Mead, and C. Rolland. "Requirements engineering paper classification and evaluation criteria: A proposal and a discussion." In: *Requir. Eng.* 11 (Mar. 2006), pp. 102–107. DOI: 10.1007/s00766-005-0021-6.

[Wie10] R. Wieringa. "Design Science Methodology: Principles and Practice." In: *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*. ICSE '10. Cape Town, South Africa: Association for Computing Machinery, 2010, pp. 493–494. ISBN: 9781605587196. DOI: 10.1145/1810295.1810446.

[WO16] J. Wang and E. Olson. "AprilTag 2: Efficient and robust fiducial detection." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2016.

[Wol+21] Á. Wolf, D. Wolton, J. Trapl, J. Janda, S. Romeder-Finger, T. Gatternig, J.-B. Farcet, P. Galambos, and K. Széll. *Towards Robotic Laboratory Automation Plug Play: The "LAPP" Framework*. 2021. arXiv: 2106.10129 [cs.RO].

[WWh07] L. Wang, F. Wu, and Z. hu. "Multi-Camera Calibration with One-Dimensional Object under General Motions." In: Jan. 2007, pp. 1–7. DOI: 10.1109/ICCV.2007.4408994.

[YHD20] G. Yu, Y. Hu, and J. Dai. "TopoTag: A Robust and Scalable Topological Fiducial Marker System." In: *IEEE Transactions on Visualization and Computer Graphics* (2020), pp. 1–1. ISSN: 2160-9306. DOI: 10.1109/tvcg.2020.2988466.

[Zha+21] Z. Zhang, Y. Hu, G. Yu, and J. Dai. *DeepTag: A General Framework for Fiducial Marker Design and Detection*. 2021. arXiv: 2105.13731 [cs.CV].

[Zxi] Zxing. *Zxing (Zebra Crossing) Github*. https://github.com/zxing/zxing. Accessed: 2028-6-23.