Dynamic fiducial markers for camera-based pose estimation

Dem Fachbereich
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades
eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von

M. Sc. Raul Acuña

geboren am 30. September 1983 in Caracas, Venezuela

Referent:	Prof. Dr.-Ing. J. Adamy
Korreferent 1:	Prof. Dr.-Ing V. Willert
Korreferent 2:	Prof. Dr. G. Fernandez
Tag der Einreichung:	19. Oktober 2020
Tag der mündlichen Prüfung:	01. Februar 2021

D17
Darmstadt 2021

Acuña Godoy, Raúl Eduardo: Dynamic fiducial markers for camera-based pose estimation
Darmstadt, Technische Universität Darmstadt,
Year thesis published in TUprints: 2021
URN: urn:nbn:de:tuda-tuprints-176507
Date of the disputation: 01.02.2021

Published under CC BY-SA 4.0 International - Creative Commons, Attribution ShareAlike
https://creativecommons.org/licenses/

Erklärungen laut Promotionsordnung

§ 8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§ 8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§ 9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§ 9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, den 18. August 2021 Raúl Acuña

A mis papás, mi pasado, la fuente inagotable de amor, herramientas y sabiduría que me han ayudado a llegar hasta aquí.

A mi esposa Nohe, mi presente, la roca donde ato mi barco cuando la mar de mi mente está en temporal y con quien rio mientras recorremos juntos el camino de la vida.

Al bebé que viene, Roland, mi futuro, la razón para convertirme en la mejor versión de mi mismo para formar parte de su vida y sus memorias.

Contents

Abbreviations and Basic Notation

Abbreviations

UGV Unmanned Ground Vehicle 53

HDMI High-Definition Multimedia Interface 98

IBVS Image-Based Visual Servoing 89

PBVS Position-Based Visual Servoing 89

LM Levenberg–Marquardt 121

ViSP Visual Servoing Platform 129

IoT Internet of Things 88

Basic Notation

x	Scalar
$\boldsymbol{x}$	Vector
$\mathbf{M}$	Matrix
${}^{a}\mathbf{T}_{b}$	Homogeneous transform matrix that converts coordinates from frame a to frame b
$\boldsymbol{O}_{i}$	Origin of coordinate system
$\tilde{\square}$	Noisy estimate of the true value of $\square$
$\hat{\square}$	Optimal estimate of the true value of $\square$
$d\left(\boldsymbol{x}, \boldsymbol{y}\right)$	Geometric distance (euclidean) between two vectors $\boldsymbol{x}, \boldsymbol{y}$

Abstract

This dissertation introduces new techniques that increase the accuracy of camera-based pose estimation using fiducial markers. The problem of camera-based pose estimation involves finding a camera's pose relative to some coordinate system by detecting some known features in the environment; when the visual appearance of these features is known beforehand, they are called fiducials.

The visual-based pose estimation process is highly complex since the estimated pose accuracy depends on many interconnected factors that have to be considered simultaneously; this thesis aims to identify the most influential factors and proposes solutions that mitigate the effect of the sources of error, hence increasing the estimated pose's accuracy and robustness. We base our solutions on exploiting an interaction between the camera and what the camera is measuring; this, in essence, means that the features change and adapt to better suit the measurement by either moving in space to better locations or changing their shape dynamically.

Kurzfassung

In dieser Dissertation werden neue Techniken vorgestellt, die die Genauigkeit der kamerabasierten Posenschätzung mit Hilfe von visuellen Referenzmarken erhöht. Das Problem der kamerabasierten Posenschätzung besteht darin, die Pose einer Kamera relativ zu einem Koordinatensystem zu bestimmen, indem einige bekannte Merkmale in der Umgebung erkannt werden. Wenn die visuelle Erscheinung dieser Merkmale im Voraus bekannt ist, werden diese als Referenzpunkte bezeichnet.

Der visuell-basierte Prozess der Posenschätzung ist sehr komplex, da die Genauigkeit der geschätzten Pose von vielen miteinander verbundenen Faktoren abhängt, die gleichzeitig berücksichtigt werden müssen. Diese Arbeit zielt darauf ab, die einflussreichsten Faktoren zu identifizieren und schlägt Lösungen vor, die die Auswirkungen der Fehlerquellen mildern und somit die Genauigkeit und Robustheit der geschätzten Pose erhöhen. Die in dieser Arbeit vorgestellte Lösung basiert auf der Ausnutzung einer Interaktion zwischen der Kamera und dem, was die Kamera erfasst. Dies bedeutet im Wesentlichen, dass sich die Merkmale ändern und anpassen, um der Erfassung besser zu entsprechen, indem sie sich entweder im Raum an bessere Orte bewegen oder ihre Form dynamisch verändern.

1 Introduction

> What is "real"? How do you define "real"? If real is what you can feel, smell, taste and see, then 'real' is simply electrical signals interpreted by your brain.
>
> *Morpheus, The Matrix*

This thesis, in general terms, is about perception, specifically, artificial vision-based perception for localization. The initial quote of this chapter summarizes perception as mainly a set of measurements that are interpreted by a cognitive mechanism. From a human's point of view, perception is about becoming aware or conscious of something, but it can also be the ability to see, hear, or realize something through the senses. From a more general or biological point of view, perception comprises the neurophysiological processes, including memory, by which an organism becomes aware of and interprets external stimuli [88]. Replace the word "neurophysiological" with "electromechanical" and the word "organism" with "system" and realize that the previous sentence is valid for any intelligent system, e.g., robots.

In essence, there are two factors at play in perception, and they are difficult to separate: the measurement and the insight, which is the interpretation of the measurement, or as a human would say: "becoming aware"; how much of each aspect is a purely reactive process in the brain and how much is a more profound thought process is still a matter of discussion [16]. Connecting again with the initial quote, our reality is defined by the **quality** of the electrical signals received by the brain; moreover, the brain filters the reality for us. We keep what is necessary and cheaper to survive. If an organism wants to increase its perception capabilities, it will need extra receptors and additional cognitive resources to process them.

From all the receptors, perhaps the most expensive in terms of cognitive resources is the perception of light or vision. The human brain is quite efficient at vision; it can perceive conceptual information from an image

in less than 13 ms [15]; in the blink of an eye, the brain can extract an enormous amount of high-level structural information from the world with low energy requirements (compared to other organisms or computer-based vision systems). Vision is also arguably the most useful perception tool for living organisms. It is assumed that one of the reasons for the Cambrian evolutionary explosion is that the evolution of advanced eyes started an arms race that accelerated evolution ("Light Switch" theory of Andrew Parker [89]), which is not a surprise, vision allows for fast locomotion and navigation, object detection and recognition, threat avoidance and more.

The perception process is tied to an anticipation of what is being observed. A cognitive model of what is being perceived is, in general, necessary, either by being pre-coded in the brain structure (what we call instincts) or learned by experience. Anticipation implies a prediction based on the current measurement and the available model of the object of perception. The perception is not passive; it is an active process connected to actions in the world and how they change in time. Moreover, perception can spur action tendencies, e.g., in pianists, the thought of the perception of a particular note triggers the motor centers of the brain associated with the production of that note with the finger [74].

To better illustrate the coupling of action and perception, let us describe a different kind of "Light Switch". In the scenario of Fig. 1.1, there is a robot that can perform only two possible actions, move forward or move backward, and only one sensor for localization, a camera. It is clear for us that if the robot moves forward, it will push the button, turning light off, and will not be able to locate itself again with the camera. We can infer this only because we have a high-level concept of what a light switch does, and additionally, we have a cognitive model of the consequences of specific actions and its influence on perception.

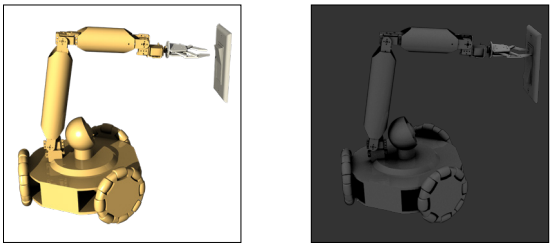


Figure 1.1: Coupling of action and perception.

In autonomous systems, the coupling between action, perception, cognition, and a goal is called the action-perception cycle. The action-perception cycle is the circular flow of information between the organism and its environment in the course of a sensory-guided sequence of behavior towards a goal; it involves cognitive processing of the information followed by a decision, an action. The actions change the state of the organism in the world (or the world itself), and then a new perception is repeated in the new state.

The light switch example brings us closer to the topic of this dissertation, vision-based localization. For now, we are going to leave by the side topics related to basic image processing (object detection, face detection, etc.) and focus on vision while moving (similar to a living organism), e.g., a robot with a camera, because it is when actions and time matter. In the literature, the problem of mobile vision can be grouped into two categories: "Visual Servoing" and "Active vision".

In Visual Servoing, a system uses feedback information extracted from a vision sensor to control the motion of a robot. In Active Vision, a system manipulates the viewpoint of the camera to get better information from the environment. At first, both of them seem very similar. The main difference is that for Visual Servoing, the perception is a tool (input) for the action; meanwhile, for Active Vision, the action (control) is a tool for the perception; this is, in essence, a task-driven differentiation. However, the action-perception coupling is always there; they are not independent of each other. In both Visual Servoing and Active Vision fields, there is little information about how the current state of the system affects the quality of the perception and how a bad perception can lead to wrong actions, which will detriment the perception in the future.

For humans, this causality is clear, if one wants to see something better, one has to change the point of view, change the angle, come closer, open the eyes to get more light, or cover them to protect from sun glare. In essence, humans already know a sequence of actions that will allow them to obtain the best measurements from the scene, translating these behaviors to artificial vision systems is a daunting task; how to know which actions are beneficial? How to filter out non-relevant information? Is there a minimum amount of information for the task? What is redundant? Is there information better suited to a particular sensor? Can it be modified somehow?

Solutions to the previous questions in Active Vision are mainly based on performing actions that increase the **amount** of information. For example, take observations of an object from different points of view, combine

multiple observations to counter sensor noise, focus on only relevant parts of the image to save computation time, or move towards places that have more information (e.g., rich in diverse textures) [4, 22, 30]. However, there has been little emphasis on the **quality** of the information, or about what information is **minimal** for the task at hand [15] and even less about cooperation between the observer and what is being observed.

The accuracy of the localization is a crucial aspect of mobile robotics since it allows the robot to navigate an environment and perform high-level tasks. Vision-based localization is highly convenient for robotics since it only requires one sensor, the camera, which does not interfere with the environment (no signals are emitted). However, in order to improve the performance of the localization, we need a full picture of the perception process, which should include the coupling between actions and perception, and this need brings us to the formal definition of the main problem.

1.1 Problem statement

The main focus of this dissertation is vision-based pose estimation for robotics, which we can summarize as the process of taking images of some known features in the environment and calculate the camera pose (position and rotation) relative to the features; this allows the camera (and by extension the robot) to locate itself in the world. The main goal of this thesis is to obtain the most accurate pose estimation possible by exploiting concepts from active vision, interaction, and robotic cooperation.

The process of camera pose estimation from available features is shown in Fig. 1.2. The first step is to define what is a feature and how to measure it both in the world and in the image, then we need to calibrate the camera in order to match image measurements to world units. Once these two essential elements are configured, we can initialize the action-perception cycle, which starts with the capture of an image by the camera sensor, followed by a group of algorithms which extract the position of the features in image coordinates from the information contained in the pixels, and finally, another set of algorithms will use this measurements plus the definition of a feature to calculate a pose. Once the pose is calculated, the cognition system can decide the best action to perform next to fulfill the goal, and the cycle is repeated.

The problem of visual-based pose estimation is highly complex since its performance depends on many interconnected factors. The only way of obtaining meaningful and reliable results is by having a birds-eye view of

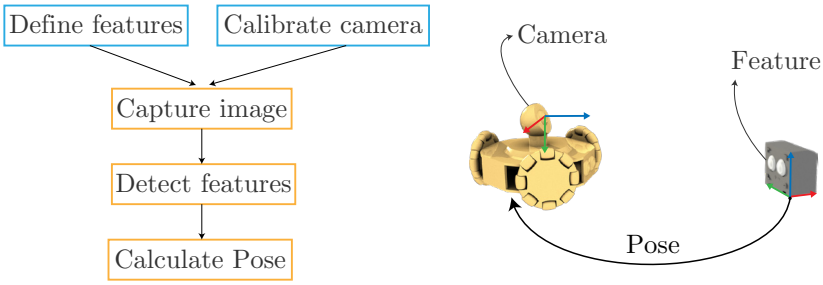


Figure 1.2: Basic elements of camera-based pose estimation.

the whole process and identifying the potential pitfalls. What follows is a short description of the necessary steps required for visual-based pose estimation and the potential error sources.

- Features definition. Here we separate into two categories. Naturally occurring features and artificial features (fiducial markers). We have no control over the environmental features; some scenarios are rich in good separable features; others can be completely absent from features for localization. If natural features are bad or absent, it would be possible to use fiducial markers instead, but this requires an environmental intervention, which is not always possible. Additionally, predefined features have to be measured beforehand, and errors may be introduced. For example, when using printed planar markers, the printer may be wrongly calibrated, or the paper can bend and not be completely planar due to humidity.

- Camera intrinsics. The parameters of a real camera have to be found through a calibration process; hence these parameters are just an approximation of the real value; the estimated parameters will introduce errors in the measurements made with the camera.

- Image noise. In digital cameras, electrical sensors are used to capture the protons and convert them into electrical signals subject to electrical noise propagating to the measurements. The sensor gain amplifies this noise when the gain increases to capture low light scenarios. Usually, the image noise is modeled as Gaussian noise, but this is only a wishful assumption. Additionally, we have to add the error produced by pixel quantization.

- Feature detection algorithms. The algorithms used to detect the features of an image will also introduce errors in the detection. The errors come both from the intrinsical limitations of the methods and problems in their programmatic implementation.

- Homography estimation and Non-linear pose optimization. The algorithms used for homography and pose estimation produce the true result only when the input data has no errors. Assuming all the previous errors are introduced in the data, the problem is transformed into a minimization of algebraic or geometrical errors for an optimal pose estimate. Due to the highly non-linear characteristics of the underlying mathematical problem, the pose estimate can be a local minimum far away from the correct value.

- Camera-to-marker relative pose. The scale plays an essential role; if the marker covers a significant portion of the camera image, the critical parts of the marker shape will be less susceptible to sensor noise, and it will be easier for the detectors to recognize its features. Moreover, the pose estimation algorithms work better in some relative poses than others; for example, fronto-parallel is a bad configuration for rotation estimation.

We tackle some of these problems in order to obtain the most accurate pose estimate possible. Our solutions are related to the action-perception cycle in what we call now an interaction-perception cycle. The idea is that both the observer and the one being observed work together to improve the pose estimation; it is similar to someone waving the arms in the air so another person can separate it from a group of people. The primary assumption is that we will have some control over the shape of the features to detect; this means that those shapes are considered as fiducials and are designed in a way that facilitates the detection process. The properties of the fiducials are hence assumed to be known beforehand by the detection system.

The first factor that we consider is the intelligent use of artificial features by introducing collaboration; next, we focus on the optimal spatial configuration of features that produces better results on the pose estimation algorithms, and finally, we present a smart fiducial marker that uses feedback from the pose estimation process to display an optimized fiducial.

1.2 Contributions

- A comprehensive comparative study on the state of the art of artificial features (fiducial markers) for the last 30 years.
- Mobile Marker Odometry. A cooperative odometry scheme for accurate pose estimation of mobile robots without using any environment features.
- An extension to Mobile Marker Odometry where environmental features are fused with marker features for improved accuracy when the environment is rich in good features while maintaining the accuracy when the environment lacks features.
- A methodology for finding optimal spatial configuration of features that improve the robustness of pose estimation algorithms versus noise. This methodology will allow designers of fiducial markers to test quality of their designs and will allow the creation of stable feature data-sets for pose algorithms comparison.
- A new fiducial marker design: the dynamic fiducial marker, based on the interaction between the observer and the marker. We use screens to display the shapes that define the fiducial, and these shapes can change over time to better suit the particular state of the observer. We design the shapes following the findings on optimal features spatial configuration. The dynamic marker has higher accuracy than traditional fiducials and can be used for camera calibration and pose estimation problems.

1.3 Dissertation outline

In Fig. 1.3, we present the general outline of this dissertation. In Chapter 2, we introduce the fundamentals of camera-based pose estimation problem and the mathematical notation used in the rest of the dissertation. We also provide a quick overview of the current state of the art in camera-based pose estimation methods. In Chapter 3, we then connect the field of pose estimation with the fiducial marker design field since fiducials are used to obtain correspondences between world coordinates and image coordinates to obtain the position and orientation of a camera. We will as well present an in-depth analysis of the fiducial marker designs throughout history. Chapter 4 is about our first contribution, a new cooperative odometry

scheme for robotics that uses only artificial features but does not require environmental intervention. In Chapter 5, we analyze the influence of the relative position of control points in the robustness of traditional pose estimation algorithms and propose a new optimization methodology to find optimal control points. Chapter 6 presents our final contribution, the dynamic marker, which is a consolidation of all the results obtained in the previous chapter applied to a new intelligent fiducial design that can change its shape over time to facilitate the process of pose estimation, increasing accuracy and robustness. Finally, in Chapter 7, we present the conclusions obtained in this dissertation.

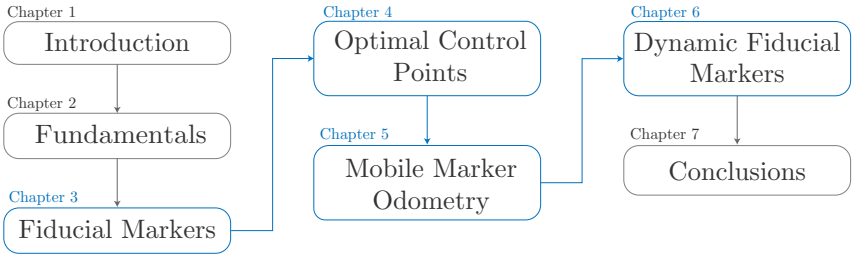


Figure 1.3: Outline. Chapters with contributions are highlighted in blue.

2 Fundamentals of camera-based pose estimation

In this chapter, we introduce the camera-based pose estimation problem. We first elaborate on the mathematical notation that connects the field of robotics and computer vision and then explain the essential elements of the camera-based pose estimation problem by studying the state of the art of pose estimation methods.

2.1 Position and orientation representation

We will use the conventions of the robotics community to represent orientations and positions, which may differ in some cases to the ones used traditionally on computer vision. The conventions and notations used in this work are based on Chapter I of the Handbook of Robotics [115]. An extract of the relevant parts is presented here for the sake of clarity.

2.1.1 Coordinate frames and position vectors

A coordinate reference frame i consists of an origin $\boldsymbol{O}_i$ and three mutually orthogonal basis vectors, denoted $[\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i]$ which are fixed to a particular body.

The position of the origin of a coordinate frame i relative to another coordinate frame j will be denoted by a 3×1 vector

$$ {}^{j}\boldsymbol{p}_i = \begin{bmatrix} {}^{j}p_i^x \\ {}^{j}p_i^y \\ {}^{j}p_i^z \end{bmatrix}. \tag{2.1} $$

Each component of the vector ($ {}^{j}p_i^x $, $ {}^{j}p_i^y $ and $ {}^{j}p_i^z $) are the cartesian coordinates of $\boldsymbol{O}_i$ in the j frame. For convenience we can define a static

coordinate frame on the world (world coordinate frame) with origin $\boldsymbol{O}_w$ and basis vectors $[\hat{\boldsymbol{x}}_w, \hat{\boldsymbol{y}}_w, \hat{\boldsymbol{z}}_w]$, which will serve as reference for all our transformations.

A vector that represents point r on coordinate frame i will be ${}^i\boldsymbol{r}$ and the same point represented in another coordinate frame j will be ${}^j\boldsymbol{r}$. Given that the position of the origin of a coordinate frame i relative to another coordinate frame j is ${}^j\boldsymbol{p}_i$, and there is no relative rotation between the frames, a point ${}^i\boldsymbol{r}$ can be transformed into coordinate frame j by doing the following operation:

$$ {}^j\boldsymbol{r} = {}^j\boldsymbol{p}_i + {}^i\boldsymbol{r}, \tag{2.2} $$

the ordering of the elements on this equation allows the reader to follow the transformation from right to left. Notice how the subscripts and superscripts cancel each other, which helps the readability of the transformation chain. This notation is summarized in Fig. 2.1.

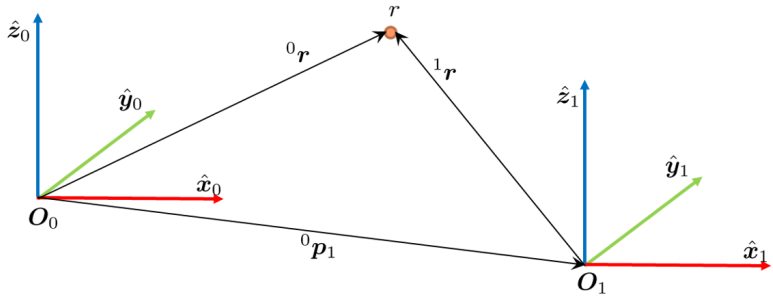


Figure 2.1: Position vectors representation.

2.1.2 Rotation matrices

The orientation of a coordinate frame i relative to a frame j can be denoted by expressing the basis vectors $[\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i]$[1] in terms of the basis vectors $[\hat{\boldsymbol{x}}_j, \hat{\boldsymbol{y}}_j, \hat{\boldsymbol{z}}_j]$. This can be written as a 3×3 matrix ${}^j\mathbf{R}_i$ called the rotation matrix, where its components are the dot products of basis vectors of the two coordinate frames with the following form:

[1] We defined in the Basic Notation section that $\hat{\square}$ is the optimal estimate of the true value of $\square$ but we make an exception in the case of basis vectors since it is a notation common in the robotics community.

$$ {}^{j}\mathbf{R}_{i} = \begin{bmatrix} \hat{\boldsymbol{x}}_i \cdot \hat{\boldsymbol{x}}_j & \hat{\boldsymbol{y}}_i \cdot \hat{\boldsymbol{x}}_j & \hat{\boldsymbol{z}}_i \cdot \hat{\boldsymbol{x}}_j \\ \hat{\boldsymbol{x}}_i \cdot \hat{\boldsymbol{y}}_j & \hat{\boldsymbol{y}}_i \cdot \hat{\boldsymbol{y}}_j & \hat{\boldsymbol{z}}_i \cdot \hat{\boldsymbol{y}}_j \\ \hat{\boldsymbol{x}}_i \cdot \hat{\boldsymbol{z}}_j & \hat{\boldsymbol{y}}_i \cdot \hat{\boldsymbol{z}}_j & \hat{\boldsymbol{z}}_i \cdot \hat{\boldsymbol{z}}_j \end{bmatrix}. \tag{2.3} $$

The components of the rotation matrix are generally called direction cosines (due to the dot product of two unit vectors), and it has the property of being orthogonal; the columns and rows are orthogonal unit vectors (orthonormal); hence its inverse is the transpose $\mathbf{R}^{\top} = \mathbf{R}^{-1}$.

With this formulation, a simple rotation of frame i around the p_j^z axis with an angle θ will be

$$ \mathbf{R}_Z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4} $$

around the p_j^y axis

$$ \mathbf{R}_Y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{2.5} $$

and finally around the p_j^x axis

$$ \mathbf{R}_X(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}. \tag{2.6} $$

A set of rotations matrices can be combined (chained) together by simple matrix multiplication. For example, if we know the rotation matrix of frame k relative to frame j, denoted as ${}^{k}\mathbf{R}_{j}$, and the rotation matrix of frame j relative to frame i, denoted as ${}^{j}\mathbf{R}_{i}$, then we can find the total rotation of frame i relative to frame k as follows:

$$ {}^{k}\mathbf{R}_{i} = {}^{k}\mathbf{R}_{j}\,{}^{j}\mathbf{R}_{i}. \tag{2.7} $$

The rotation matrix contains nine elements; however, only three parameters are required to define a body's orientation in space. As a result, there is no unique representation. A typical minimal representation of rotations

can be defined by the so-called euler angles $[\alpha, \beta, \gamma]$, where each angle is a rotation around an axis of a moving coordinate frame. When combining rotation matrices using matrix multiplication, the ordering matters; this means that there are different conventions for the euler angle representation, depending on how the axis rotations are chained together. The most common representation is the Z-Y-X (or 3-2-1 notation), which defines a rotation around the Z-axis by an angle α, then a rotation around the Y-axis of the rotated coordinate frame by an angle β, and finally, a rotation around the X-axis of the two-times rotated coordinate by an angle γ. Other common representations for the rotation matrix are the axis-angle and quaternions-based representation, which are discussed in detail in [115].

2.1.3 Transformation matrix and homogeneous coordinates

A transformation is a displacement plus a rotation of a coordinate frame relative to another. The order in which the rotation and the displacement are applied matters. In our convention, we apply first the rotation and then the displacement; for example, if we want to apply a rotation and a translation to the vector ${}^{i}\boldsymbol{r}$ to represent it in the coordinate frame j, we will do it in the following order:

$$ {}^{j}\boldsymbol{r} = {}^{j}\mathbf{R}_{i} \cdot {}^{i}\boldsymbol{r} + {}^{j}\boldsymbol{p}_{i} \,, \tag{2.8} $$

but we would like to have a more convenient representation of the transformation by defining a transformation matrix ${}^{j}\mathbf{T}_{i}$ that rotates and translates coordinates from frame i to frame j. To create the transformation matrix, we need to augment euclidean coordinates into homogeneous coordinates. Homogeneous coordinates or projective coordinates are a system of coordinates used in projective geometry introduced in 1827 by August Ferdinand Möbius. They allow the representation of points at infinite using finite coordinates and simplify the calculations in affine transformations and projective geometry.

Given a 3D point with euclidean coordinates (x, y, z), the representation $(\lambda x, \lambda y, \lambda z, \lambda)$ is called the set of homogeneous coordinates for the point, which is valid for any real number λ different to zero. The set of vectors $[\lambda x, \lambda y, \lambda z, \lambda]^{\top}$ for varying values of λ is then a representation of the point in $\mathbb{R}^3$. To transform homogeneous coordinates into euclidean coordinates,

we divide the first three homogeneous coordinates by λ; this is valid when $\lambda \neq 0$. If $\lambda = 0$ the point is said to be defined in the infinite, which is a conventional definition used in projective geometry.

We can define the transformation matrix from frame i to frame j denoted by ${}^{j}\mathbf{T}_{i}$ as a composition of the rotation matrix ${}^{j}\mathbf{R}_{i}$ and the translation ${}^{j}\boldsymbol{p}_{i}$, which will transform vectors in homogeneous coordinates representing 3D vector in euclidean coordinates. If the elements of the rotation matrix and translation vector are denoted as

$$ {}^{j}\mathbf{R}_{i} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \qquad {}^{j}\boldsymbol{p}_{i} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}, \tag{2.9} $$

then the transformation matrix ${}^{j}\mathbf{T}_{i}$ is defined as a 4×4 matrix with the following form:

$$ {}^{j}\mathbf{T}_{i} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{21} & r_{22} & r_{23} & p_2 \\ r_{31} & r_{32} & r_{33} & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.10} $$

Finally, if we define a point in coordinate frame i, but now in homogeneous coordinates ${}^{i}\boldsymbol{r}$, we can write the following:

$$ {}^{j}\boldsymbol{r} = {}^{j}\mathbf{T}_{i} \cdot {}^{i}\boldsymbol{r}\,, \tag{2.11} $$

which is a more convenient representation for coordinate transformations.

2.2 Image formation process

Now that 3D coordinate frames and transformations between them are defined, it is possible to introduce another kind of transformation, the projective transform. This transform is related to the image formation process in cameras, which is a two-dimensional representation of a three-dimensional world (we lose a dimension) [49].

The traditional way of modeling this process is based on the central projection where each point in three-dimensional space creates a ray that passes through a fixed point in space called the center of projection; this

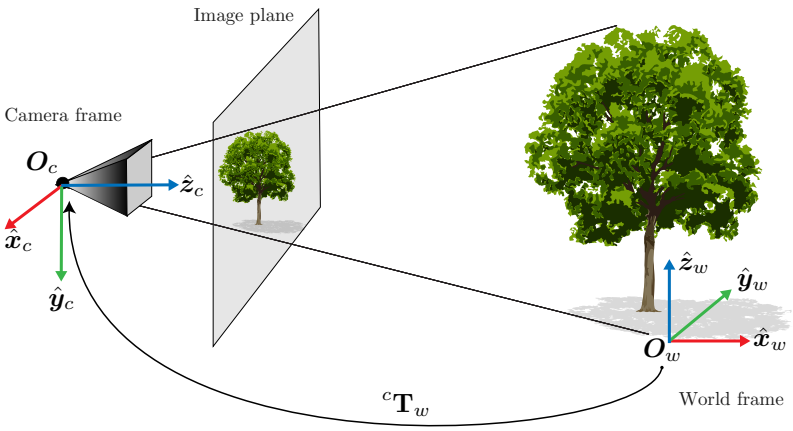


Figure 2.2: Example of the image formation and definition of world and camera coordinate systems. Note the direction of the arrow in the transform ${}^c\mathbf{T}_w$; this will be the convened direction for all transformation matrices.

ray then intersects a specific plane called the image plane. The intersection point of the ray with the image plane is the projection of the 3D point.

As a side note, this model is only one of the possible ways of modeling how a camera captures images, however, camera lenses usually do not focus the rays in precisely one point, and some distortion is introduced in the final image. More advanced projection models can be used to minimize these problems, but they are out of the scope of this research.

We will define a coordinate frame for the camera and some conventions on its orientation related to the world coordinate system, as in Fig. 2.2. Note that the Z-Axis of the camera coordinate system is perpendicular to the image plane; this axis will be called the optical axis. The intersection of the optical axis and the image plane is the principal point or image center. Finally, we define the transform between the world and the camera coordinate system as ${}^c\mathbf{T}_w$.

The distance between the center of the camera coordinate frame $\boldsymbol{O}_c$ and the principal point on the image plane is called the camera constant or focal length f. We can then define a central projection transformation based on a pinhole camera model which converts 3D homogeneous coordinates on the camera coordinate system into 2D homogeneous coordinates on the image plane:

$$\lambda \begin{bmatrix} {}^{I}r^{x} \\ {}^{I}r^{y} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}_c} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{\Pi}_0} \begin{bmatrix} {}^{c}r^{x} \\ {}^{c}r^{y} \\ {}^{c}r^{z} \\ 1 \end{bmatrix}. \tag{2.12}$$

Where ${}^{c}r$ is a point in the camera coordinate system in 3D homogeneous coordinates, ${}^{I}r$ is its projection on the image plane and f is the focal length.

$\mathbf{K}_c \in R^{3\times3}$ is defined as the central projection matrix and $\mathbf{\Pi}_0$ is the canonical projection matrix. The origin of the image coordinate system lays on the principal point, as can be seen in Fig. 2.3.

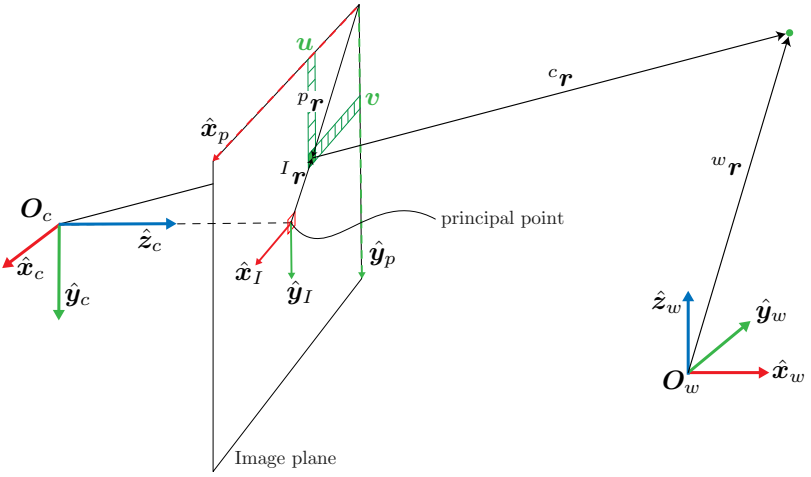


Figure 2.3: Definition of world coordinate system and the main camera related coordinate systems.

Camera images are generally defined with the origin in the top left corner, and the points are also scaled and sheared. The following transformation accounts for this with a scale factor in each axis (s_x, s_y), a principal point offset (c_x, c_y), and a shear distortion factor (s_θ):

$$\lambda \begin{bmatrix} {}^{p}r^{x} \\ {}^{p}r^{y} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} s_x & s_\theta & c_x \\ 0 & s_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}_s} \begin{bmatrix} {}^{I}r^{x} \\ {}^{I}r^{y} \\ 1 \end{bmatrix}, \tag{2.13}$$

converting then from image coordinates to pixel coordinates through the matrix $\mathbf{K}_s$. The two matrices $\mathbf{K}_c$, $\mathbf{K}_s$ and can be consolidated into one, the camera matrix $\mathbf{K}$[2], and the whole process of transforming points from world coordinates to pixel coordinates will be

$$\lambda\, {}^{p}\boldsymbol{r} = \underbrace{\mathbf{K}_s\mathbf{K}_c}_{\mathbf{K}} \boldsymbol{\Pi}_0\, {}^{c}\mathbf{T}_w\, {}^{w}\boldsymbol{r} \tag{2.14}$$

with $\mathbf{K}$ as:

$$\mathbf{K} = \begin{bmatrix} fs_x & fs_\theta & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.15}$$

We can simplify this by denoting $f_x = fs_x$, $f_y = fs_x$ and $fs_\theta = 0$ (which is a fair assumption) obtaining then the final camera matrix

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.16}$$

One final coordinate frame relevant for us is the normalized image frame η, which is equivalent to a projection in the camera plane in world units (no influence of internal camera parameters). Assuming the calibration matrix $\mathbf{K}$ to be known, the homogeneous pixel coordinates ${}^{p}\boldsymbol{r} = [{}^{p}r^{x}, {}^{p}r^{y}, 1]^\top$ can be transformed to homogeneous normalized image coordinates ${}^{\eta}\boldsymbol{r}$ in world units (usually metric units) by

$${}^{\eta}\boldsymbol{r} = \mathbf{K}^{-1}\, {}^{p}\boldsymbol{r}. \tag{2.17}$$

[2]Specifically, it is a transformation from normalized image coordinates to pixel coordinates: ${}^{p}\mathbf{K}_\eta$.

2.3 Space resection and the PnP problem

Camera or space resection is a term used in photogrammetry in which the relative spatial position and orientation of a camera are obtained by using image measurements of some defined control points. A control point is a distinct location in 3D space with known coordinates in a given frame, e.g., in the world coordinate frame. The problem of camera resection is also known in the computer vision community as the Perspective-n-Point (PnP) problem.

The Perspective-n-Point problem consists in obtaining the relative camera transformation ${}^{c}\mathbf{T}_{w}$ given a set of n control points $\{({}^{w}\mathbf{r}^{i})|i = 1, \ldots, n\}$ in 3D space (world coordinates), a corresponding set of projected 2D points $\{({}^{p}\mathbf{r}^{i})|i = 1, \ldots, n\}$ in the camera image (pixel coordinates) and a known camera matrix $\mathbf{K}$; the essence of this problem is summarized in Fig. 2.4.

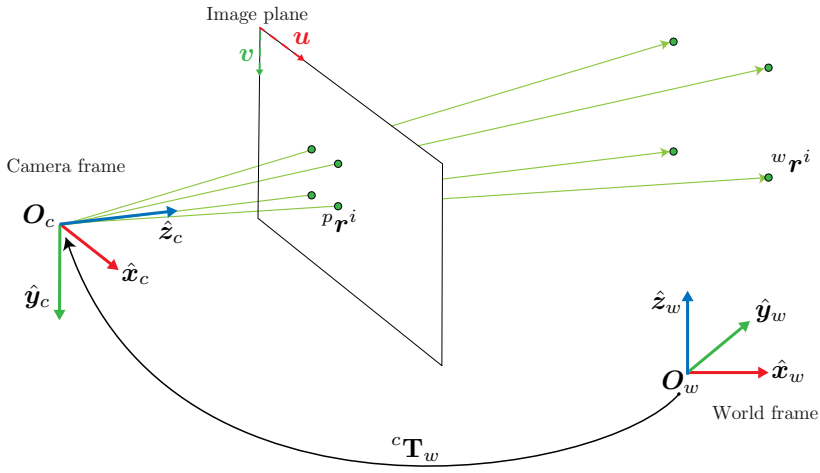


Figure 2.4: The Perspective-n-Point problem.

PnP can be considered an over-constrained (only for $n \geq 3$) and generic solution to the pose estimation problem from point correspondences. PnP methods can be classified into those who solve for a small and predefined number n of points, and those who can handle the general case. Several solutions have been presented in the literature [77]. The minimal case (when $n = 3$) in general provides four solutions for non-collinear points. Thus, prior knowledge has to be included to choose the correct solution.

Since it has been proven that pose accuracy usually increases with the number of points [77], PnP approaches that use more points ($n > 3$) are usually preferred. The general PnP methods can be broadly divided into whether they are iterative or non-iterative.

2.3.1 Iterative PnP methods

Iterative approaches formulate the problem as a non-linear least-squares problem. They differ in the choice of the cost function to minimize, which is usually associated to an algebraic or geometric error. Some of the most important iterative methods in chronological order are the POSIT algorithm [85], the LHM [73], the Procrustes PnP method or PPnP [40], and the global optimization method SDP [106].

Most iterative methods have the disadvantage that they return only a single pose solution, which might not be the true one. Most of them can only guarantee a local minimum, and the ones that find a global minimum remain computational intensive. The major limitation of iterative methods is that they are relatively slow, neither convergence nor optimality can be guaranteed, and a good initial guess is usually needed to converge to the right solution.

2.3.2 Non-iterative PnP methods

Non-iterative methods try to reformulate the problem so that a potentially large equation system may solve it. However, early non-iterative solvers were also computational demanding and produced worse results for a larger number of points. The first efficient and non-iterative $O(n)$ solution was EPnP [65], which was later improved using an iterative method to increase accuracy.

More recent approaches are based on polynomial solvers trying to achieve linear performance without the problems of EPnP and with higher accuracy. The DLS [51] method introduced in 2011 was the first successful $O(n)$ method by using a Cayley parameterization of the rotation, unfortunately with some degenerate cases. Later on, the Robust PnP or RPnP [67] was the first method that is more accurate when a low number of points is used $n \leq 5$, with similar accuracy to LHM but faster. In 2013 the OPnP [124] was presented, which is similar to DLS but uses a non-unit quaternion representation of the rotation. The following year, the UPnP [58] method was published, which is a linear non-iterative method that generalizes the solution into the non-perspective n-point problem; in

contrast to OPnP, the authors used normalized unit quaternions to represent rotations. More recently, in optDLS [82], a return to the Cayley rotation parameterization used on DLS is proposed, mentioning a simple trick to avoid the singularities; the method is three times faster than OPnP.

When six or more control points are available, it is possible to apply the linear one-step approach named the Direct Linear Transform (DLT) [49] followed by a non-linear optimization based on the reprojection error to obtain the pose. This method consists of solving a homogeneous linear system of equations for the 12 unknown parameters of the transformation matrix ${}^{c}\mathbf{T}_{w}$. Each point correspondence provides two equations; therefore, a minimum of 6 control points are needed. However, this solution is over-parameterized and very sensitive to noise and outliers. Nonetheless, the posed obtained through the DLT serves as an excellent fast initialization for non-linear optimization.

2.3.3 Planar pose estimation

A particular case of PnP is the Planar Pose Estimation (PPE), which is a space resection problem that involves the process of recovering the relative pose of a plane with respect to a camera's coordinate frame from a single image measurement. A PPE problem can be solved by calculating the object-plane to image-plane homography transformation and then extracting the pose from the homography matrix (this is known as homography decomposition [109, 123]) or by using a set of points in the plane as the measurement, with the PnP methods (planar PnP) as a particular case. Some of the most important planar PnP methods are the iterative RPP-SP [105] and the more recent direct method IPPE [21]. In general, planar PnP methods outperform the best homography decomposition methods when noise is present. Additionally, homography decomposition methods only provide a single solution in contrast to modern planar-PnP methods.

The basic linear algorithm for homography estimation is again the DLT [49], which, in contrast to the general 3D case, needs only four control points to find the homography since the homography matrix has nine unknown parameters. In essence, this method finds a Total Least Squares (TLS) solution of a linear system of equations, but the problem is that the error structure of the linear equations is not appropriate for TLS. To overcome this issue, Harker and O'Leary in [46] used an orthogonalization step that has the advantage of being non-iterative and 5-6 times faster than the DLT; even though the actual accuracy of this new method

also depends highly on the input data, its performance is similar or better than the DLT, which makes it arguably the best algorithm for homography estimation.

For both homography estimation methods, the previous normalization of the measurements is a crucial step to improve the quality of the estimated homography [49]. However, the normalization has some disadvantages [93]: First, the normalization matrices are calculated from noisy measurements and are sensitive to outliers. Second, for a given measurement, the noise affecting each point is independent of the others; in normalized measurements, this independence is removed [17]. A method is proposed in [93] to overcome this problem by avoiding the normalization and using a Taubin estimator instead, obtaining similar results as the normalized one but with increased complexity.

2.3.4 Point degeneracy and pose ambiguity problem

One common problem of PnP methods is that they will work properly under some conditions but will fail under others, and these conditions are not the same for all the solvers. Each method has its singularities; this means that the performance of the solver will depend on the kind of data provided to it as an input. Even more, for the same input, different methods may produce different solutions.

The first factor is the number of control points needed. For example, the DLT method for general 3D points requires six or more non-coplanar points; however, when using planar points, the DLT needs only 4, and the P3P method requires only 3 points.

In the presence of noise, the higher the amount of control points available, the higher the accuracy of the methods, i.e., more data is better. However, if the data contains outliers, some methods are susceptible to it. When the data is unreliable, it will require the use of an algorithm like the RANdom SAmple Consensus (RANSAC) [32] to remove outliers before applying a PnP algorithm to it.

For some pose estimation algorithms, there may exist special cases where the algorithm may fail to output the correct pose. These exceptional cases are defined by the configuration of the control points in the space and the relative pose of the camera to these points. For example, a general degeneracy in PnP is when the control points are co-linear [121]; in this configuration, the number of solutions for the transformation ${}^{c}\mathbf{T}_{w}$ are infinite (although the distance from the optical center to the control points can be uniquely determined) and no algorithm can produce the unique

solution. Some PnP algorithms will fail if the model points are co-planar (e.g., general DLT or some EPnP implementations). Harder to identify are some corner cases which appear only in some configurations for some algorithms, and even worse, in only some implementations of a given algorithm. For example, a known problem is the OpenCV original implementation of Zhang's method for solving PnP with co-planar points [123], which will fail when the control points are precisely the four corners of a square. Many of these special and corner cases are not correctly documented in the implementations, and there are little comparative studies that focus on the corner cases of each algorithm.

When the model points are co-planar, a known issue for PnP is the pose ambiguity; it occurs when the planar model is either small or viewed far to the camera (weak perspective). In these cases, there exists a rotation ambiguity that corresponds to an unknown reflection of the plane about the camera's z-axis. Methods based on the homography decomposition will return only one solution, which may be the wrong one (i.e., a reflection of the correct solution); more modern algorithms like IPPE [21] will return both solutions and let the user decide based on other metrics like the reprojection error. However, in some cases, the reprojection error is very similar between these two solutions, and additional external information would be needed, for example, to mention a few: more control points, analysis of the scene brightness (orientation influences the brightness of the planes), temporal filtering, or discarding wrong solutions by applying sensor fusion techniques [54].

2.3.5 The golden standard

All of the above make selecting a PnP method a daunting task. The user needs specific knowledge of the limitations of the algorithms and some insight on the characteristics of the control points and their measurements, which is not always available.

In general, the golden standard for pose estimation has been to use any of the above one-shot solutions to calculate an initial pose estimate followed by a non-linear optimization. For the planar case, the typical pipeline uses homography decomposition or IPPE for the initial guess, and for the non-planar case, the DLT method; this initial guess is then optimized with a non-linear optimization based on the minimization of the reprojection error to obtain the final solution.

We have to remember that, in essence, solving the problem of PnP is only to find the solution of space resection given a certain number of

control points. Nevertheless, this is only half of the story. The PnP methods are generally agnostic to how the control points are defined in the world, how they are detected in an image, and how correspondences between control points and image points are determined. Each one of these elements influences the final outcome of the pose estimation process since, in each one of these steps, errors are introduced. When one is trying to obtain the pose of the camera, it is vital to understand the whole process of image capture and detection and each of the algorithms involved and not only the PnP method selected. By looking at the problem as a whole and not as independent algorithms, it will be possible to identify the fundamental problems and find solutions to them.

3 Control points and fiducial markers

In this section, we expand on some of the most used ways of measuring and identifying control points using cameras, emphasizing pose estimation. Afterward, we focus on fiducial markers, which are artificial shapes designed to be detected by cameras, going through a comparative analysis of the most relevant planar fiducial designs in the last 30 years.

3.1 Features and control points

A control point is a defined point in 3D space represented in the world (or main) coordinate system. We assume that the coordinates of a control point are known, or they can be measured with another method (e.g., manually) with high accuracy. Control points are usually defined in certain relevant places in the 3D scene, for example, in the corners of a particular object or on the center of a manufactured marker.

On the other hand, a feature is a piece of information that can be obtained from an image. In the context of camera-based pose estimation, a feature is some part of the image that can be paired to a shape in the 3D scene with low uncertainty or tracked in successive captures of the same scene in different camera poses.

Features may be used to pair control points and their corresponding image points; this requires information on the 3D shapes that must be detected in the environment. However, it is hard to define a 3D object's feature invariant to different camera poses. Features are more commonly used when little information is provided about the 3D scene structure, and similar shapes or points of information want to be tracked in consecutive images. If the pose of the camera does not change that much in consecutive images, then it is easier to define features that remain invariant between camera shots; this can be used, for example, to perform odometry (measuring the change of camera position over time) in what is called Visual Odometry (VO) or to reconstruct the 3D shape of an object or the

scene from several image captures in what is denominated Structure from Motion (SFM).

3.1.1 Sift, Surf and ORB features

The features represent some information available on the image; this information, of course, has some relationship with the 3D scene captured by the camera. A feature detector algorithm detects features in an image by performing some local mathematical operations in the image's pixel values, e.g., a gradient. The representation of this local information is what is called a feature descriptor. For example, an edge can be described by its position and length, but it also may include information about its slope.

An ideal feature should be unique in the image and invariant to different camera poses so it can be uniquely identified and matched for all the images, and the mathematical operation should be simple enough to be processed in real-time. For example, corners are easily identifiable features in images. A standard algorithm to find corners is the Harris Corner Detector algorithm [47], which uses image gradients and finds the junctions of them. However, the real problem is to find features that are invariant to transformations.

There have been many feature detector algorithms and feature descriptions proposed to solve the invariance issue. The most commonly used are the Scale Invariant Feature Transform (SIFT) [23], the Speed up Robust Feature (SURF) [5] and the Oriented FAST and Rotated BRIEF (ORB) [98], which rely on scale-space theory [77]. Of these approaches, SIFT is the more accurate in matching keypoints but also the slowest and not suitable for real-time, while SURF and ORB can be used in real-time with similar performance [56]. At the moment, ORB is arguably the best feature detector with a good compromise in speed and accuracy.

An example of the ORB feature detection algorithm is presented in Fig. 3.1; the algorithm can match features even with rotations and scale variations; however, a small subgroup of the matches are wrong. If this feature matches are used for homography or pose estimation, it may produce a wrong solution if an outlier rejection scheme is not used. Even rejecting outliers, if the environment has other similar shapes or if the same kind of shapes are repeated many times, it will be harder and harder to find unique correspondences. Ideally, unique shapes should be selected to avoid this problem; even more, if it is possible to control the environment, it would be better to design the shapes that are easier to recognize in the image. These designed shapes (also known as fiducials) have artifi-

cial features that will be easily detected by an image processing algorithm uniquely without introducing outliers.

Figure 3.1: Example of the ORB feature detection algorithm used to automatically match features from a known object (a book) presented on the left side of the image to a photo of the book in a real setting. The first match (from top to bottom) is matched incorrectly.

3.1.2 Fiducial markers

A fiducial marker is an artificial object that serves as a reference for camera-based measurements. The shape and colors of a fiducial are artificially designed, so computer vision algorithms can easily detect the fiducial among all the possible shapes which appear in an image [27]. Single fiducials or a constellation of them are placed on points of interest within the camera's field of view, and then, the fiducials are detected either manually or with a computer vision algorithm in the camera's images see Fig. 3.2.

The main uses of a fiducial in computer vision applications are: 1) Position and orientation reference, also called landmarks. 2) Reference of scale for objects on the scene; this is particularly relevant for monocular cameras since the scale is lost in the projection process. 3) Encode information, i.e., identification tags, used, for example, to single out a particular fiducial among a constellation of them.

There are many fields where fiducial markers are used. In circuit manufacturing, circular plated fiducials are placed on the surface of PCB boards to align pick-and-place assembly equipment; this orients the tool to know where to place the different chips and components (see Fig.

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

Figure 3.2: Fiducial markers used in different fields. a) Circular fiducials used for PCB alignment (marked in red). b) Self-adhesive skin markers, which provide visibility in body imaging procedures such as Computerized Axial Tomography (CT) and Magnetic Resonance Imaging (MRI). c) Reflective spherical markers for full body motion capture with a multi-camera system. d) Planar fiducial markers for pose control of a quadcopter.

3.2a). In medicine, fiducials help track points of interest on the person's body when images are taken from different points of view or by several different imaging devices during medical examinations (Fig. 3.2b); they are also used to track the endpoint of surgical tools for laparoscopic procedures. In the film and game industries, infrared-reflective spheres are used for motion capture; this allows, for example, the recording of body movement sequences or facial expressions, which are then used to animate 3D virtual models (Fig. 3.2c). In augmented reality and robotic applications, fiducials are used for tracking objects in the environment or for defining the origin of coordinate systems; this allows to localize the camera on the environment and to measure the relative position of tracked objects (Fig. 3.2d) [87].

3.1.3 Criteria for a good fiducial

Shape

The shape of a fiducial should be easy to recognize for an image processing algorithm. Ideally, the selected shape should occupy a minimum of space on the display surface, and it should not occur naturally in the environment. The more simple the shape, the easier it would be for the detection algorithm to recognize it in real-time. The detection should be fast but maintaining accuracy. [57, 87].

Color

The presence of color in a fiducial marker is usually a compromise. Color can be a rich form of information, especially for identification and segmentation. However, the representation of colors in images is not unique. The imaging sensors can have different sensitivities to the light spectrum; this means that each camera produces different color representations of the same scene, and for consistent measurements, they require color calibration. For this reason, most fiducials are designed using black and white shapes without considering the color for simplicity.

Orientation

The detection of a fiducial should be independent of the orientation of the camera that captures it. The image should not favor some orientations over others. Thus, the fiducial design should have non-symmetrical elements that will allow the detection algorithm to orient the marker in the image properly. Additionally, the non-symmetrical elements are a crucial part of unambiguous full 6DOF pose estimation.

Size

The ideal size of a fiducial in world coordinates is a compromise between many factors. First, to be detected, the chosen shape should cover a minimum quantity of pixels in the image. The minimum amount of pixels increases with the complexity of the chosen shape and the amount of information that has to be transmitted within it (e.g., the ID). Second, the size is also dependent on the camera properties (e.g., pixel density, resolution, lenses) and the minimum and maximum distance at which the marker has to be detected. For example, a large marker can be better for long ranges, but it may be out of the field of view when the camera is too close.

Identification

A marker should be unique. It must be easily separated from all the other shapes present in the environment, and it should be separable from other markers of the same family. In many applications, the users place several markers in different environment locations, and it should be easy for the processing algorithm to discern one marker from the others in these situations. The solution is implementing a coding scheme inside

the marker, which defines a unique ID for each one. The code inside the marker requires a certain amount of display area, and the amount of area required increases with the complexity of the code.

Robustness

The robustness of a fiducial is related mainly to the identification and the pose estimation processes. The identification should work on all conditions and be consistent even in different environmental conditions (e.g., lighting conditions). Once identified, the second most important task of a fiducial is the process of pose estimation, which is highly susceptible to errors in the position of the points. An ideal fiducial should provide almost noise-free point coordinates to avoid pose errors or, at the very least, provide the characteristics of the error based on the pose.

Construction

The fiducial can be any shape in the space, either flat or three-dimensional, and theoretically, there are no constraints on the shapes and sizes. However, in practice, a fiducial should be reliable and easy to manufacture. Inaccuracies in the construction of the fiducial will translate into measurement errors; hence, the user should precisely measure the position of the control points in the fiducial after manufacturing. The design also has to consider that the usage of materials that expand or contract at different temperatures or deteriorate with time may introduce errors in the measurements, which are hard for the user to consider.

3.2 Planar fiducial markers

To fulfill all the fiducial requirements is not easy, mainly because a fiducial should be easy to manufacture and measure. For high accuracy 3D pose estimation, the best solution has been fixed multi-camera systems that emit Infrared Light (IR), the IR bounces on small sphere-shaped fiducials and reflect back to the camera system, a software then computes a pose of a predefined sphere configuration from the images of all the cameras to a common coordinate system. The usage of IR simplifies detection since the only shapes that will appear in the cameras are the fiducial spheres. However, this system requires much space, mainly a room dedicated to this pre-calibrated multi-camera system, and that is why other alternatives are preferred.

Planar fiducial markers may be the best alternative when it is impossible to use an IR based camera system. They can be printed on paper using a regular desktop printer, making them convenient, cheap, and easy to manufacture. Consumer printers can consistently produce markers consistently with always the same dimensions and generate complex shapes with or without color. In contrast, custom non-planar markers are much more complicated to manufacture, and their usage is limited to specific research scenarios.

The advantages of planar fiducials are evidenced by the available amount of research in this field; there have been more than thirty years of history in planar fiducial designs, and in this section, we will focus on describing their evolution and the current state of the art.

3.2.1 Planar fiducial design

The surface area available on a given plane is the key factor that drives the design of planar fiducials. Ideally, one would want a marker as small as possible that does not interfere with the environment and occupies less space on the camera's field of view. However, an ideal marker should also provide the necessary amount of information required for detection, identification, and control points definition. For example, a marker should provide at least four control points for 6DOF pose estimation, and additional control points are preferred since they provide more redundancy, stability, and robustness to noise, but more control points will require more display surface.

The fiducial has to display the necessary information through minimal 2D shapes in the most efficient way possible; the used shapes should be simple enough to be detected with simple and fast image processing algorithms. Simultaneously, the selected shapes should be ideally invariant to projections, so their structure can be recovered from images taken from different points of view. It is no surprise that the most common geometric shapes used in fiducial markers throughout history are also the easiest ones to recognize with simple operations on image pixels, which are mainly blobs (circles) and corners (squares).

We are interested in the evolution of the fiducial designs from their first instances to the most commonly used nowadays. This with the intention of finding which are the best available fiducials and what are the current problems in the field. However, there are no studies available that track and compare fiducials' whole development up to this day. Most papers that propose a new fiducial limit themselves to mention some designs that

may be relevant only to their new proposal. It is a daunting task for a new researcher to go across the literature and find all the references and relationships between the existing fiducials.

In the following parts, we are going to study the history of fiducial designs by grouping and finding the existing relationships between the different designs; at the end of this chapter, we will be able to create a timeline of fiducials, which will allow us to identify the critical problems that will be tackled in this dissertation.

We can group fiducial markers based on the kind of basic geometrical shape they use (circles or corners), we can further separate them by the identification method they use, and finally, we can group the ones that have mixed (hybrid) designs. Based on shape, we can separate them into corner markers and circular markers. The corner markers can be separated into squared markers with a square-like shape with internal pre-registered code and QR markers that do not require pre-registration. Finally, we have the hybrid/special category, where we placed all the mixed designs.

For each group, we select the most representative fiducials of that category, and we are going to order them by the first date of publication, highlighting the relationships between them.

3.2.2 Squared markers

Square markers are planar fiducials whose design relies on the detection of corners. We are not going to focus on describing the different detection methods for corners; it will suffice to say that they are based on the detection of gradients on the borders of the square and then finding intersections of these edges (corners); the corner position can then be optimized with sub-pixel techniques [11]. The corners themselves are not enough to define a fiducial since the squared shape is symmetrical (cannot be used to extract orientation), and identification is required. The usual design of this kind of fiducials is a black square with some black and white squares inside for identification; refer to Fig. 3.3 for examples of the essential planar fiducials of this group.

The identification is perhaps the most important part that has driven the different designs over time. With proper internal coding, a square marker can be simultaneously identified and oriented for pose estimation. Traditionally, the identification has been located on the inside of the square, but this design has changed in new proposals. Since the identification is based on a code, it requires the definition of bits of information in the fiducial; these bits are usually represented by internal smaller black and

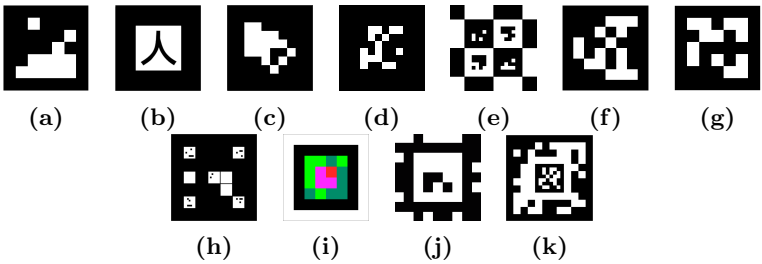


Figure 3.3: Most relevant squared fiducial markers in chronological order. a) Matrix [94], b) ARToolkit [90], c) ARTag [31], d) ARToolkit Plus [114], e) Caltag [3], f) AprilTag [86], g) ArUco [38], h) HArCO [117], i) ChromaTag [24], j) AprilTag Uramaki [61], k) Fractal Aruco [97] .

white squares (white for 1 and black for 0), which are organized usually by a snake-like sequence starting by one of the corners. Some fiducials rely on other kinds of shapes, even textures, and some of them also include color. However, the snake-like coding has been the dominant design; the reason for this dominance is that the internal code can include error correction strategies, making the identification more robust.

The typical pipeline for detection is first to detect the corners, then use the four corner points to define a homography, find the proper orientation based on the internal code, and finally extract the pose from the homography matrix. Now we are going to go across the fiducials of Fig. 3.3 highlighting with bullet points the central claims or contributions of each design:

- 1998 Matrix [94]. The first squared fiducial marker to our knowledge. A black square with internal binary code for identification. No robustness to occlusion.
- 2000 ARToolkit [90]. Use of any image as an internal code, the image is detected by correlation with a template. High false detection rate and inter maker confusion. Open source.
- 2005 ARTag [31]. Introduction of binary defined codes with error correction to replace ARToolkit's pattern recognition and identification step.
- 2007 ARToolkit Plus [114]. Introduced binary marker patterns (similar to those of ARTag). Performance improvements. Open source.

- 2010 Caltag [3]. Basic shape similar to ARTag. New strategies for the definition of pattern layouts used for calibration (repetition of tags in a board). Introduction of saddle points for additional accuracy in corner detection.

- 2011 AprilTag [86]. Improvements on the detection side. Robust to lighting variation and occlusion with a focus on accuracy. It proposed a modified lexicode that allows the definition of code families that reduce the rate of false positives in identification. Open source and used by NASA.

- 2014 ArUco [38, 39]. Similar to AprilTag. It introduced code dictionaries to maximize the inter-marker distance and the number of bit transitions. Markerboards are robust to occlusion. Good and well-maintained codebase. Open source, but new versions have code obfuscation in critical parts.

- 2015 HArCO [116, 117]. A design suitable for UAV's autonomous landing, increasing range of detection by adding a hierarchical structure to the markers; this is the first instance of a marker inside a marker in the literature.

- 2017 ChromaTag [24]. First fiducial based on colors that allows faster detection than other markers. ChromaTag uses two shades of red and two shades of green for the binary code. The shades are only differentiable in the B channel of LAB colorspace, so detection is done in the A channel (where the difference between red and green is maximal), and the decoding is done in the B channel. The marker's general configuration is still a black square over white background with a color-based coding on the inside. Open source.

- 2019 AprilTag Uramaki [61]. Flexibility on the identification positioning, the code can be placed on the outside of the square (as the Uramaki kind of Sushi), leaving space in the center for recursive markers like HArCO. Open source.

- 2019 Fractal Aruco [97].A modification of the original Aruco that moves the code to the inner border of the square, leaving space in the center for recursive markers like HArCO and AprilTag Uramaki.

When one compares the design of Aruco or AprilTag, which are one of the most recent square marker designs, and the first design made by

Rekimoto in 1998 [94], it is interesting to notice that the shape of the fiducials has not changed fundamentally; most of the improvements along time has been on the software side and the encoding of the identification inside the marker. The basic structure has remained the same: a black and white square with a binary codification represented by smaller squares. Only recently, new and more interesting designs started to appear, for example, ChromaTag for real-time detection and the sequence of recursive markers designs that try to increase the detection range, such as HArCO, AprilTag Uramaki, and Fractal Aruco. The need to increase the range has been driven mainly by the robotics community, especially for autonomous Unmanned Areal Vehicle (UAV) landing. AprilTag and Aruco markers have become the standard fiducial markers for most applications due to their excellent performance and clean code that the authors provide freely.

3.2.3 QR markers

QR markers are a 2D version of barcodes that have been used in the industry for a long time. They represent information in a sequence of black and white squares representing the binary data. Not all the QR codes are designed to provide pose estimation since their typical functionality is to transmit information. However, it can be said that QR codes have inspired the identification methodology of squared fiducial markers. The problem with QR codes for pose estimation is that since they do not present a bounding square, the detection takes more time. Additionally, they require to be parallel to the camera (since there is no way for a priori projective correction with a homography), and they usually have to fill a significant portion of the image. The most relevant QR code based markers are shown in Fig. 3.4.

(a) **(b)** **(c)** **(d)** **(e)**

Figure 3.4: Most relevant QR code based fiducial markers in chronological order. a) Maxicode [37], b) Quick Reponse QR [37], c) Cyber-Code [95], d) VisualCode [96], e) Visual QR Codes [100].

The path of QR code designs has been as follows:

- 1992 Maxicode [37]. Public domain system created by the United Parcel Service (the US Post), mainly for tracking and managing packages. The dots inside follow the style of a 2D barcode but organized in an hexagonal grid. No pose estimation.
- 1994 Quick Reponse QR [37]. Also an instance of a 2D barcode with four standardized encoding modes. It has become arguably the most-used type of 2D code. No pose estimation.
- 2000 Cyber-Code [95]. From the same creator of the Matrix square markers. First 2D barcode marker that can estimate camera pose.
- 2004 VisualCode [96]. Based on Cyber-Code but including a code coordinate system independent of the point of view that uses orientation features (a set of guiding bars on the bottom and side).
- 2015 Visual QR Codes [100]. A modern QR design that contains more empty space than traditional QR codes, the empty space can be used to display images that almost hide the dots used for codification, which is useful, for example, to place a company logo inside the QR code. No pose estimation.

3.2.4 Circular markers

Circles are the second basic geometrical shape that can be used to represent fiducial control points. Circular marker designs are based either on detecting circle centers, circle borders (conic), or both. Circles can be used similar to the corners of a square; in this case, the pose is calculated by detecting the center of a minimum of 4 circles. Additionally, the circle border detection can be used for conic-based pose estimation, where the projective properties of the circle, which is a conic, are exploited to extract the pose. In contrast to square fiducials, there is no consensus on the methodology to represent the identification and the necessary asymmetry for 6DOF pose estimation. The most representative circular-based fiducials are shown in Fig. 3.5.

- 1992 CCC [41]. Black ring on a white background that produces concentric contrasting circles. Identification based on the radius ratio of the concentric circles. Four detected markers are required for pose estimation using PnP.

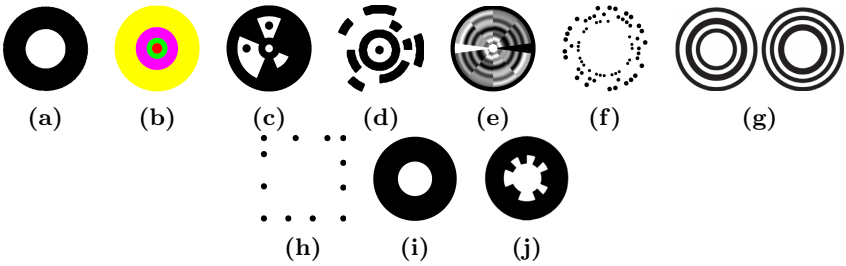


Figure 3.5: Most relevant circular shaped fiducial markers in chronological order. a) CCC [41], b) CCC Color [18], c) Intersense [81], d) 2002 TRIP [72], e) Fourier Tag [103], f) RUNE-Tag [7], g) CCTag [13], h) Pi-Tag [8], i) Whycon [59], j) Whycode [70], .

- 1998 CCC Color [18]. An extension of the CCC concept. They introduce multi-ring color fiducials with different number of rings at different size levels. The identification is based on a color code. Pose estimates require the detection of at least three of these fiducials.
- 2002 Intersense [81]. A combination of the inner code of the Matrix squared fiducials with the CCC circular fiducials. The result is a black circle with an inner barcode. Four detected markers required for pose estimation using PnP.
- 2002 TRIP [72]. Concentric black arcs are used to define the identification code. The arcs are detected by fitting ellipses to image contours and detecting the concentric ones. The pose estimation is based on conics, so in contrast to previous designs, only one fiducial is needed.
- 2007 Fourier Tag [103]. Circular marker that encodes the inner data in the frequency spectrum. It allows part of the information to be extracted even when the marker is small or when the image is blurry. They use the same pose estimation method as TRIP.
- 2011 RUNE-Tag [7]. Consists of rings of dots with robustness to large occlusions due to the use of error-correcting codes. The identification is based on cyclic codes defined by the dots. The detection process starts by detecting all the ellipses on the scene and clusters them based on their rotation, finding the ones that are coplanar and with the same size. They use both the conic information of the internal

dots and rings to do point correspondence of the dots and finally apply PnP with the dot's centers.

- 2012 CCTag [13, 14]. The design is a plane with two concentric CCC markers. The method uses conics for pose estimation. The new version of 2016 is highly robust to blur and challenging conditions but slow (4 fps) and with a limited identification dictionary.
- 2013 Pi-Tag [8]. Dots aligned in the edges of a square with positions that exploit the co-linearity and cross-ratio projective invariant for orientation and identification. The pose is solved by PnP of the dot's centers. The method is slow when there are many ellipses in the environment due to the cross ratio's necessary comparison between all the ellipses.
- 2013 Whycon [59, 60, 84]. A re-emerge of the CCC markers but with an improved detector based on connected regions. Pose estimation based on conics but unable to provide 6DOF with a single marker due to the symmetry of the marker.
- 2017 Whycode [70] A modification of the original Whycode with the inclusion of a radial based coding in the inner ring of the marker. The internal code allows full 6DOF at the expense of detection range.

Like the squared markers, the first and latest proposed fiducial share almost the same design in this category. If we compare the CCC marker design from 1992 and the Whycode marker design of 2017, we notice that the shape is the same, only with the addition of an internal code. Some more creative markers like Fourier Tags, RUNE-TAG, and CCTag did not hold enough traction to be widely spread, and eventually, the fundamental design came back to its roots. Most of the improvement has been on the detector's performance and in the usage of conics for pose estimation. Circular markers are arguably more precise than their square-based counterparts, especially those from the last decade, but they lack identification flexibility. In terms of range, markers like Whycon have a greater maximum detection distance than AprilTags or Aruco due to the design's simplicity.

3.2.5 Hybrid/special markers

Some markers are based on combinations of different techniques and cannot be classified into purely circle or corner based; we call these markers

hybrids or special markers. In this category, we include markers based on topological configurations that have produced some of the most modern innovative designs, although not widespread in their usage.

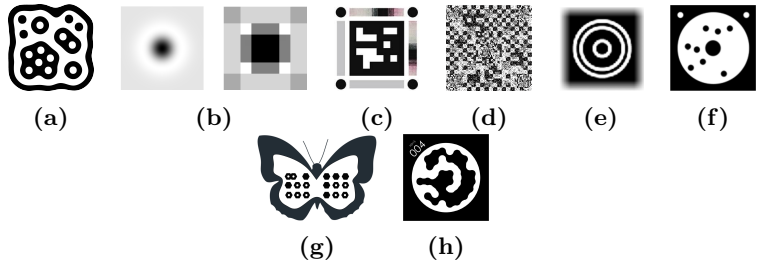


Figure 3.6: Most relevant hybrid fiducial markers in chronological order. a) ReacTIVision [55], b) SiftSurfTag [104], c) Lentimarkers [110], d) Fractal Marker Field [50], e) Prasad [91], f) X-Tags [9], g) TopoTag [122], h) STag [6] .

- 2007 ReacTIVision [55]. Started as an alternative to squared fiducial markers to be used in the Reactable system (an interactive music tool), its shape was optimized using genetic algorithms. The idea was to have a fiducial with more compact symbol sizes as well as improved processing. The fiducials are based on a topology tree. For detection, the image is segmented into connected alternating white and black regions, creating a graph; this graph is searched for the fiducial's encoded tree structure, which is matched to a dictionary of IDs. This marker family does not provide a pose estimation.

- 2009 SiftSurfTag [104]. The authors propose markers optimally suited for SIFT and SURF feature detectors. The blob/circle like marker is adapted to trigger maximum response with a SIFT detector while the squared one triggers maximum response with a SURF detector. No direct pose can be estimated from the markers, but they can be used as a source for high-quality features in the environment for VO.

- 2012 Lentimarkers [110]. A hybrid marker based on a traditional square fiducial with additional strips on the borders. The strips have a microlens array foil on top of a black and white pattern that allows the detection of rotation without calculating a homography, this with the hope of avoiding the inaccuracies of homography estimation in fronto-parallel configurations. The novel technique has

high accuracy, but it is hard to manufacture the markers with the required precision.

- 2012 Fractal Marker Field [50]. This is the first instance of a fractal marker to overcome the problems of range in fiducial detection. The design is based on a fractal structure of markers inspired by chessboard matrix codes (like QR codes). The fractal structure allows a virtually infinite number of nested markers. However, it requires high image resolution, and the detector has low detection rates compared to traditional fiducials.

- 2015 Prasad [91]. This fiducial is designed to be robust to blur in a similar way to modern implementations of CCTag. The design is binary coded fiducial with concentric white rings of equal widths over a black background with blurred borders.

- 2016 X-Tags [9]. Another square and circle hybrid. The design consists of a random arrangement of dots on a white circle background distributed around a bigger black central dot. The white circle is embedded in a black square. The detection is based on defining a set of cross-ratio invariants using the dots plus the additional constraints of line intersections. The pose estimation is more accurate than Aruco markers because X-Tags uses the centers of the dots (more accurate than corners) plus more than four points. However, there is no information about the detection rates, which can be problematic since X-Tags uses a voting mechanism for identification.

- 2019 TopoTag [122]. Similar to ReacTIVision, TopoTag utilizes topological structure information. To avoid rotational ambiguities, the design defines a baseline node that is different to other nodes in the topology tree. The authors claim that their solution offers more accurate pose estimation than Aruco because they can use several nodes as control points. However, this is a very recent not peer-reviewed paper, so the results have to be taken with a grain of salt; to the date of this writing it is only available in pdf format published in arXiv and no open code was found.

- 2019 STag [6]. Hybrid design, a black outer square used for homography estimation and the inner inside circle for refining the homography using conics. The inner encoding is novel; it uses circular black and white regions inside the white circle, which are then morphologically dilated and eroded repeatedly, resulting in the final encoded

pattern. The code has error correction capabilities, which, coupled to an advanced square detector, allows for a successful detection even with occluded corners. STag is perhaps the most advanced modern design which exploits both circles and corners. However, to this writing date, it has not been peer-reviewed and can only be found on arXiv.

The hybrid markers are the most interesting from all the categories. Some designs focus on increasing the range (e.g., the Fractal Marker Field), others on being robust to blur (e.g., Prasad), and others, in perspective invariance (e.g., SiftSurfTag). In this category, some designs implement the identification in a novel and creative way using a topology (e.g., ReacTIVision, TopoTag). We believe that a topology based identification could also be applied to other marker families in the future, for example, squared fiducials. The most common characteristic of this bunch is the use of a combination of squares and circles to increase accuracy, which may be true in theory, but there is a lack of comparisons between marker families and a significant dependency on how the software is implemented. We believe that many of this category's markers show better promise than AprilTags and Aruco markers from the theoretical point of view, but the latter (AprilTags and Aruco) have a proven and reliable codebase.

3.3 Discussion

All the markers categorized above are organized in a timeline in Fig. 3.7 and Fig. 3.8, this timeline summarizes almost 30 years of fiducial development, and it is in itself an excellent tool to visualize the relationship between different marker families and their evolution through time. We see that the shape that appeared first as a fiducial is the circle, closely followed by the square. The principal identification methodology originated from the QR code family and was later fused inside the squared and circle markers.

We see some convergent designs. For example, as QR-based markers try to offer pose estimation capabilities, they start to look similar to squared fiducials. As mentioned before, squared markers and circle-based markers look almost the same after 30 years of development, but now the software is faster, robust, and accurate. The most revolutionary designs come from the hybrid category, where we start to see the merging of circles and squares and new ways of identifying the fiducials. Besides some punctual

exceptions, color is not used; it may be due to the difficulty of identifying and representing color uniquely since there is much variation depending on the light and the camera sensor's specifics. Without a color-calibrated camera, the representation of color is, in general, ambiguous.

The remaining question is: Which is the best fiducial? The answer is relative, there is no definitive one. The selection of a fiducial will always be based on sacrificing a feature in favor of another. For example, if one wants more range, it is necessary to use a simpler marker, which translates into less varied IDs and maybe not a full 6DOF pose. The selection is about what the shape in itself can offer, how good the associated software is, and how the developers implemented the pose estimation step, which, as we already saw, can be done with a lot of different methodologies with varying results. In general terms if the task requires long range detection then use Whycon, for high accuracy choose X-Tags and for general applications select Aruco since it has the best software.

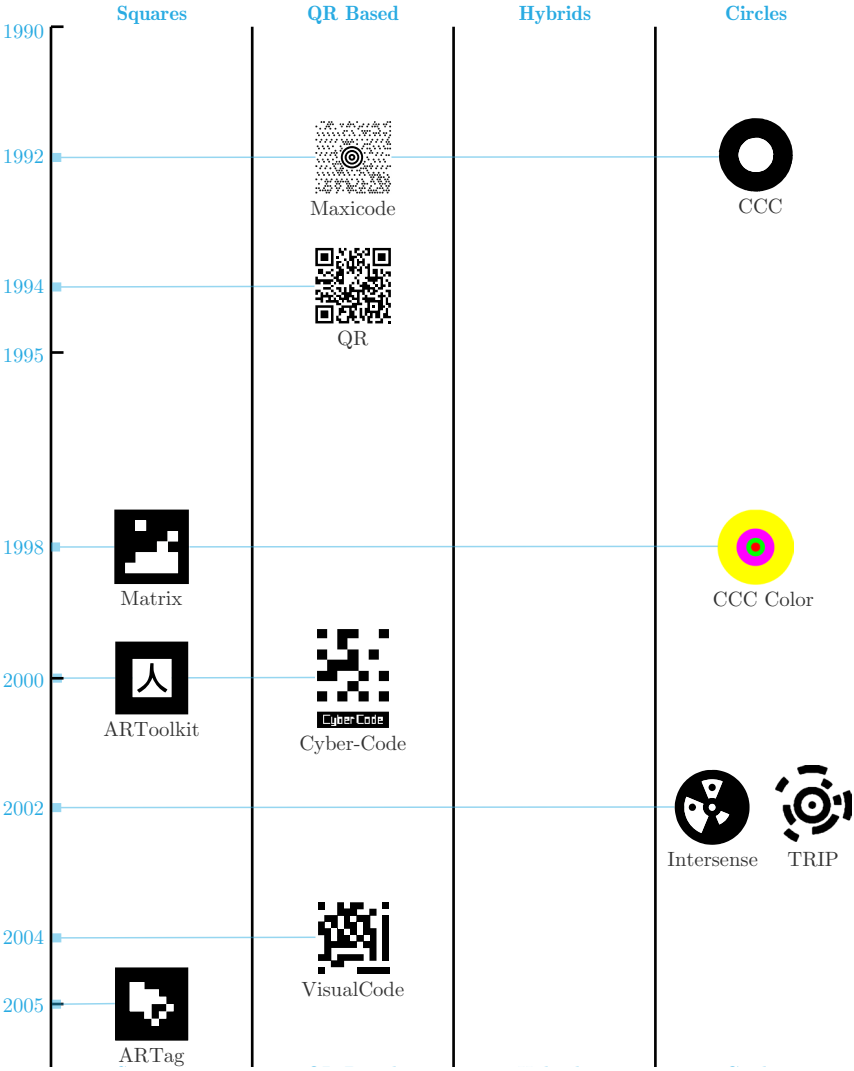


Figure 3.7: 30 years of fiducials part I.

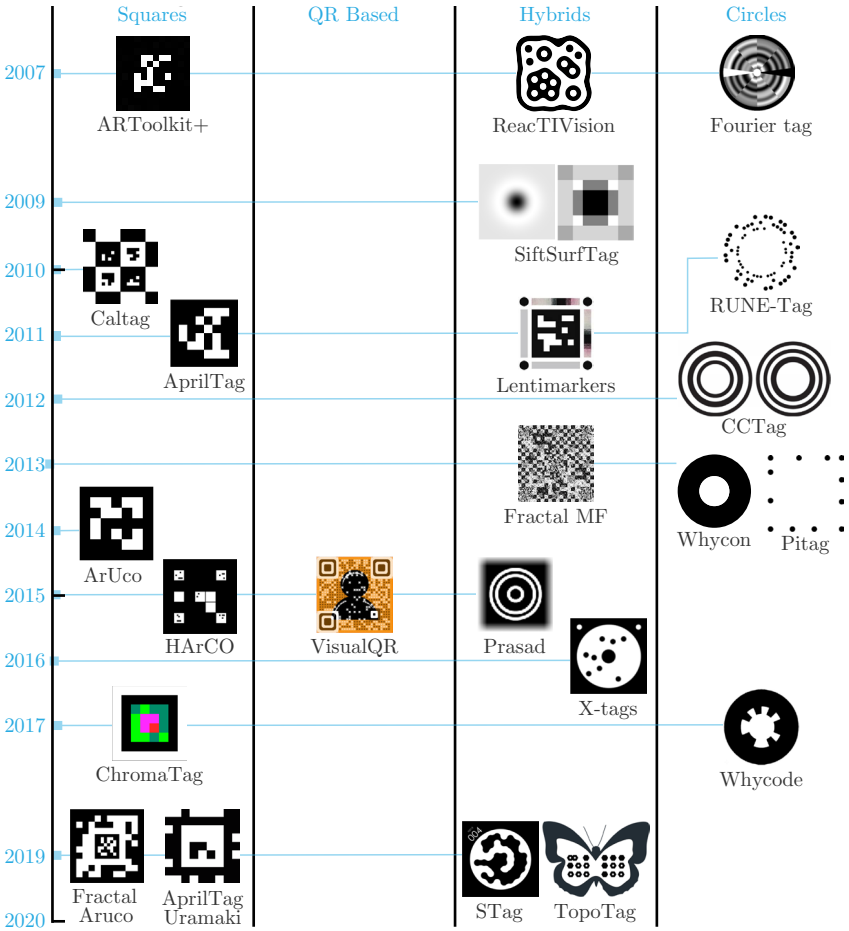


Figure 3.8: 30 years of fiducials part II.

4 Mobile Marker Odometry

In this chapter, we will present our first approach to improve the accuracy of the pose estimation process. We are going to focus specifically on the error introduced by the features.

If we want to estimate a camera pose from a sequence of camera images, we must first detect some features in the environment. We can detect features automatically by using algorithms such as SIFT, SURF or ORB, but these detectors can present some problems, for example, miss some of the features, produce false positives, or in the best case, detect features consistently from image to image but with errors. The problem is that we have no control over what is going to be available in the environment as a feature, and we have to rely on pre-coded algorithms to try to obtain some generic features. Moreover, there are exceptional cases where there may be no features at all, for example, white and uniform hallways, and there can be places where there are repetitive textures that look very similar in the image, so the detector will confuse features from image to image. One solution would be to place artificial features in the environment instead (aka fiducial markers); however, it is not practical to place fiducial markers in all the places where a pose is needed since it would require too much environmental intervention. Therefore, we propose and validate a robot cooperation scheme that will allow the use of accurate features coming from fiducial markers but without needing to intervene in the environment; this solution will offer robust and accurate visual-based pose estimation for robots, even in challenging environments.

This chapter's content is organized as follows: in Section 4.1, we introduce the topic of visual pose estimation applied to robotics while connecting the field of Visual Odometry (VO) to fiducial marker pose estimation. In Section 4.2 we describe our solution based on mobile markers, which we then expand in Section 4.3 by describing all the possible robotic architectures, and finally, in Section 4.4, we present the experiments and their analysis in Section 7.2.

4.1 Introduction

Visual pose estimation and localization is a problem of interest in many fields, from robotics to augmented reality and autonomous cars. Possible solutions are dependent on the camera(s) configuration available to the task (monocular, stereoscopic, or multi-camera) and the amount of knowledge about the structure and geometry of the environment.

Visual pose estimation can be classified into two different categories: The first one, based on Marker-Based (MB), relies on some detectable visual landmarks like fiducial markers or 3D scene models with known feature coordinates [38, 77]; The second category works Markerless (ML), without any 3D scene knowledge [36, 77]. The difference between MB and ML methods is illustrated in Fig. 4.1.

MB methods estimate the relative camera pose to a marker with known absolute coordinates in the scene. Therefore, these methods are driftless, need only a monocular camera system, and the accuracy of the pose estimation depends on the accuracy of the measurement of 2D image coordinates, which are projections of known 3D marker coordinates, and the kind of algorithm used to realize spatial resection [45, 119].

ML methods estimate relative poses between camera frames based on static scene features with unknown absolute coordinates in the scene and apply dead reckoning to reach the absolute pose within the scene w.r.t a known initial pose. Due to this incremental estimation, errors are introduced and are accumulated by each new frame-to-frame motion estimation, which causes unavoidable drift.

These methods can be further divided into pure visual VO [36] and more elaborate Visual Simultaneous Localization and Mapping (V-SLAM) approaches [64] including the new developments on Semi-Dense VO [29]. Basic VO approaches estimate frame-to-frame pose changes of a camera-based on some 2D feature coordinates, their optical flow estimates [120], and their 3D reconstruction using epipolar geometry in conjunction with an outlier rejection scheme to verify static features [12]. Even if some additional temporal filtering like Extended Kalman Filtering (EKF) or Local Bundle Adjustment (BA) is applied, drift can be reduced but cannot be avoided [36].

V-SLAM approaches [64] accumulate not only camera poses but also 3D reconstructions of the back-projected extracted 2D features of VO in a global 3D map. Thus, drift can be reduced using additional temporal filtering on the 3D coordinates of the features in the map or global BA and loop closure techniques to relocate already seen features via map match-

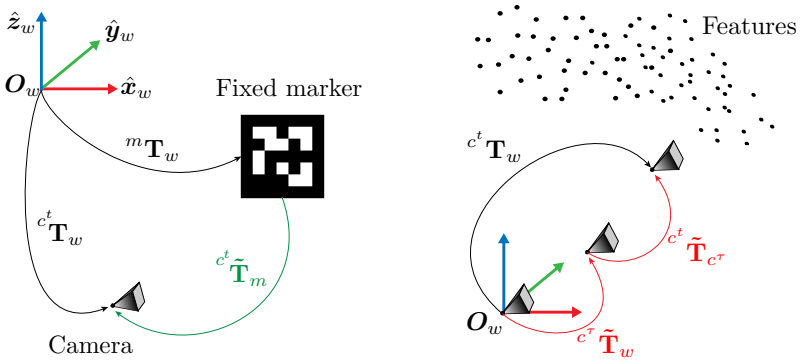


Figure 4.1: Camera-based pose estimation methods. MB pose estimation (a) uses fixed markers with known pose ${}^{m}\mathbf{T}_w$ to obtain the absolute pose of the camera ${}^{c^t}\mathbf{T}_w$ via a measurement ${}^{c^t}\tilde{\mathbf{T}}_m$ (in green). VO (b) detects fixed features along consecutive image frames in a ML environment to estimate relative camera poses ${}^{c^t}\tilde{\mathbf{T}}_{c^\tau}$ (in red) and infers the absolute pose ${}^{c^t}\mathbf{T}_w$ by concatenation.

ing. Both approaches can be realized with a monocular or a stereo vision system, whereas the stereo approach is much less prone to drift because of the superior resolution of scale estimates. Alternatively, additional sensors like Inertial Measurement Unit (IMU) can be integrated to improve the scale/drift problem in monocular systems and apply sensor fusion to increase robustness and reduce the drift as in Visual Inertial Odometry approaches [66].

The main advantage of MB versus ML methods (besides the fact that they do not drift) is the knowledge of error-free 3D coordinates of simple and unambiguously detectable landmarks. Thus, for MB methods, the error of spatial resection reduces to errors in the 2D coordinate estimation of known corresponding 3D coordinates projected onto the image plane [45]. In contrast, ML methods have to deal with additional errors, like 1) outliers (e.g., non-static features), 2) 2D-2D correspondence errors from optical flow estimates and 3) 3D reconstruction errors stemming from inaccurate stereo vision, wrong disparities, or scale estimations [12]. Moreover, ML methods usually require good illumination (enough brightness and contrast) of the environment, scenes rich in texture, and a minimum amount of feature overlap between frames.

To summarize, in terms of accuracy and computational complexity, MB methods clearly outperform ML methods. However, the most significant advantage of ML methods is that only features that are already present in the environment are needed for localization. Hence, ML methods do not require the modification of the environment with artificial markers and/or a topological survey to define landmarks covering the sensor's whole navigation space.

Even with the disadvantage of needing environmental intervention, MB methods in the form of fiducial markers have been used for relative pose estimation and tracking in the robotics community for quite some time, e.g., as beacons for UAV autonomous landing [68] or as landmarks for the relative pose estimation of a UAV to a group of UGV's [20]. Shared coordinates systems for multi-robot configurations have also been a topic of interest; cooperative localization originally introduced by Kurazume et al. [62] introduces the use of some robots as moving landmarks and others to detect them using lasers. Fox et al. propose to use a sample-based version of Markov localization, which synchronizes each robot's belief to increase accuracy [34]. Wildermuth et al. use a camera system mounted on top of a robot to calculate each surrounding robot's relative position and their transformations in a common coordinate frame [118]. More recently, Dhiman et al. developed a system of mutual localization that uses reciprocal observation of fiducials for relative localization without ego-motion estimates or mutually observable world landmarks [25].

Our work's primary motivation has been developing a real-time visual localization method with the accuracy of MB pose estimation without needing to modify the environment to integrate the fiducials. **Our novel idea is that the robots themselves will serve as the visual markers in the environment**; this can be highly beneficial in cases where traditional odometry schemes fail due to environmental conditions or in environments which are hard for traditional VO algorithms, such as indoor spaces with flat textures, low light conditions, smoke, or underwater environments.

For this purpose, we propose the Mobile Marker Odometry (MOMA) method, a cooperative visual odometry scheme based on mobile visual markers. Our work is inspired by [62] but with a complete realization for the case when the landmarks are visual fiducial markers; this avoids the need to use expensive laser-based sensors and opens the path to integrate this scheme into traditional VO pipelines. Additionally, a study of the propagation of the error was performed based on the particulars of planar fiducial marker detection and its pros and cons compared to other popular

feature based VO and V-SLAM approaches.

4.2 Mobile Marker Odometry (MOMA)

We define the concept of a MOMA as a fiducial marker that has one of two possible configurable states at any given time: **Mobile** if the marker is moving or permitted to move and **Static** otherwise. A MOMA can either be moved by some entity or by itself. We define the *observer* as the entity that performs the detection and pose estimation of the marker, in our case, a camera. The camera also needs to have one of these two states at a given time in order to perform pose estimation.

Marker-based visual localization (MB) A marker is a set of known features with known marker frame coordinates[1]. Visual marker-based pose estimation uses known fixed markers to obtain an estimation of the absolute pose ${}^{c^t}\tilde{\mathbf{T}}_w$ of a camera coordinate frame c at some time t in world coordinates. We assume that the pose of the fixed marker in world coordinates ${}^{m}\mathbf{T}_w$ is known, and also, the structure of the marker is predefined and easy to detect. Once the marker is detected, we can estimate the relative pose ${}^{c^t}\tilde{\mathbf{T}}_m$ of the camera in the marker frame using a PnP method, and by extension, the pose of the camera in world coordinates by concatenation:

$$ {}^{c^t}\tilde{\mathbf{T}}_w = {}^{c^t}\tilde{\mathbf{T}}_m \, {}^{m}\mathbf{T}_w \tag{4.1} $$

The error in global camera pose ${}^{c^t}\tilde{\mathbf{T}}_w$ will be associated only with the relative pose estimation between marker and camera ${}^{c^t}\tilde{\mathbf{T}}_m$. Hence, no drift will be accumulated in contrast to dead reckoning approaches.

The reasons for the robustness and preciseness of a MB pose estimate are twofold. First, the correspondences between 3D world points in marker coordinates $\{({}^{m}\boldsymbol{x}_i)|i = 1, \ldots, n\}$ and 2D camera image points in pixel coordinates $\{({}^{p}\boldsymbol{x}_i)|i = 1, \ldots, n\}$, can be extracted unambiguously using the knowledge about the configuration of each 3D point ${}^{m}\boldsymbol{x}_i$ on the marker. Second, the pose of the marker ${}^{m}\mathbf{T}_w$ itself is known in advance from very precise measurements and does not have to be extracted online. Thus, the only source for errors is the extraction of the coordinates of the 2D

[1] For this section coordinates $\boldsymbol{x} = [x, y, z, 1]^\top$ are assumed to be homogeneous coordinates, as long as not stated otherwise.

projections ${}^{p}\boldsymbol{x}_i$, which depends on the resolution of the camera, and the chosen method to get subpixel accuracy [77]. The relations for MB pose estimations are sketched in Fig. 4.1a.

Markerless pose estimation (visual odometry) Contrary to MB pose estimation, VO is a dead reckoning (coupled navigation) approach. Given some initial known pose of the camera in world coordinates ${}^{c^\tau}\mathbf{T}_w$ at time τ, to get an estimate of the absolute camera pose ${}^{c^t}\tilde{\mathbf{T}}_w$ at a posterior time t, we have to estimate the relative frame poses between consecutive times $\tau = t-1$ and t, denoted ${}^{c^t}\tilde{\mathbf{T}}_{c^\tau}$, via recursive accumulation:

$$ {}^{c^t}\tilde{\mathbf{T}}_w = {}^{c^t}\tilde{\mathbf{T}}_{c^\tau}\, {}^{c^\tau}\mathbf{T}_w \,. \tag{4.2} $$

The relative pose can be extracted from the following homogeneous 3D-3D point correspondences

$$ {}^{c^\tau}\boldsymbol{x} = {}^{c^\tau}\tilde{\mathbf{T}}_{c^t}\, {}^{c^t}\boldsymbol{x}. \tag{4.3} $$

After including the collinearity equation, the reprojection error at time t between reprojected 3D coordinates ${}^{c^t}\boldsymbol{x}$ and homogeneous normalized image coordinates from the previous time ${}^{\eta^\tau}\boldsymbol{x}$:

$$ \epsilon_t = d\left({}^{\eta^\tau}\boldsymbol{x},\, \boldsymbol{\Pi}_0\, {}^{c^\tau}\tilde{\mathbf{T}}_{c^t}\, {}^{c^t}\boldsymbol{x}\right)^2 , \tag{4.4} $$

where $d(a,b)$ is a function that calculates the geometric distance between a and b.

Solving the least squares optimization

$$ {}^{c^\tau}\hat{\mathbf{T}}_{c^t} = \mathrm{argmin}_{{}^{c^\tau}\tilde{\mathbf{T}}_{c^t}} \sum_{{}^{\eta^\tau}\boldsymbol{x},\, {}^{c^t}\boldsymbol{x}} \epsilon_t \,, \tag{4.5} $$

leads to the optimal relative pose estimates ${}^{c^\tau}\hat{\mathbf{T}}_{c^t}$ (see also Fig. 4.1b). The 3D coordinates ${}^{c^t}\boldsymbol{x}$ of the features are not known and their estimation changes over time. Thus, they have to be reconstructed as ${}^{c^t}\boldsymbol{x} = \lambda\, {}^{\eta^t}\boldsymbol{x}$, for example using a stereo vision system that extracts the depth λ of each 2D coordinate ${}^{\eta^t}\boldsymbol{x}$. Also a proper correspondence search to get the 2D-2D correspondences of $\{{}^{\eta^\tau}\boldsymbol{x}, {}^{\eta^t}\boldsymbol{x}\}$ coordinate pairs is needed for a proper reconstruction and a good optimization result from equation (4.5). Unfortunately, a correspondence search in a ML environment is ambiguous and

prone to errors because it is based on some optical flow algorithms [120]. Since this reconstruction is not error-free and accumulates along frames, the ML pose estimation is worse than MB pose estimation and prone to drift due to equation (4.2), which happens at camera framerate.

MOMA To maintain the accuracy of the camera pose estimation while using only one marker to cover the whole environment, the marker and the camera have to move in turns. The instants of time where they switch the roles of moving or static are denominated as $t = \tau_i$.

Let us start by assuming that the marker is static and that we know its location in the world ${}^{m^\tau}\mathbf{T}_w$ at a given instant $t = \tau$; given this information, we can estimate the pose of a moving camera in world coordinates ${}^{c^t}\tilde{\mathbf{T}}_w$ by detecting the marker-to-camera pose ${}^{c^t}\tilde{\mathbf{T}}_{m^\tau}$ and with the known absolute marker pose in world coordinates ${}^{m^\tau}\mathbf{T}_w$ as follows

$$ {}^{c^t}\tilde{\mathbf{T}}_w = {}^{c^t}\tilde{\mathbf{T}}_{m^\tau}\, {}^{m^\tau}\mathbf{T}_w \,. \tag{4.6} $$

Let us say that the camera stops moving at $t = \tau_1$ and from that moment the marker moves (first switch). The pose of the camera at that instant ${}^{c^{\tau_1}}\tilde{\mathbf{T}}_w$ can be calculated with (4.6), and from now on, the pose of the marker while it moves can be calculated in a similar way:

$$ {}^{m^t}\tilde{\mathbf{T}}_w = {}^{m^t}\tilde{\mathbf{T}}_{c^{\tau_1}}\, {}^{c^{\tau_1}}\tilde{\mathbf{T}}_w \,. \tag{4.7} $$

Notice that if we unravel (4.7) by substituting ${}^{c^{\tau_1}}\tilde{\mathbf{T}}_w$ with (4.6) as follows:

$$ {}^{m^t}\tilde{\mathbf{T}}_w = {}^{m^t}\tilde{\mathbf{T}}_{c^{\tau_1}}\, {}^{c^{\tau_1}}\tilde{\mathbf{T}}_{m^\tau}\, {}^{m^\tau}\mathbf{T}_w \,, \tag{4.8} $$

we see how the current pose of the marker in world coordinates is a result of a concatenation of transformations which accumulate through each switch τ in time. At time t in equation (4.8) there are two discrete measurements, ${}^{c^{\tau_1}}\tilde{\mathbf{T}}_{m^\tau}$ and ${}^{m^t}\tilde{\mathbf{T}}_{c^{\tau_1}}$, and each subsequent switch will introduce a new estimate to the chain.

Although there is drift by the accumulation of the relative poses of the marker according to (4.6), as a matter of principle the accumulated error in (4.6) for MOMA is much lower than in (4.2) for visual odometry because no backprojection based on error-prone 3D reconstructions ${}^{c^t}\boldsymbol{x}$ has to be applied. Instead, only error-free 3D marker coordinates ${}^{m}\boldsymbol{x}$ and unambiguous and precise 2D sub-pixel correspondences ${}^{p}\boldsymbol{x}$ from a known

fiducial marker that can be detected very robustly are used. Additionally, the error accumulation for MOMA, according to (4.6), only happens at discrete time instances $t = \tau_i$, which occur on a much lower frequency at specific waypoints rather than on the high frame rate of the camera like in ML-VO. Finally, since the odometry scheme comprises only the camera and the mobile marker, no environment features are required.

In conclusion, the whole MOMA process is only based on applying the least-squares optimization along a specific caterpillar-like (see also Section 4.3) marker-camera motion pattern. The minimal motion pattern and concurrent optimizations are summarized graphically in Fig. 4.2.

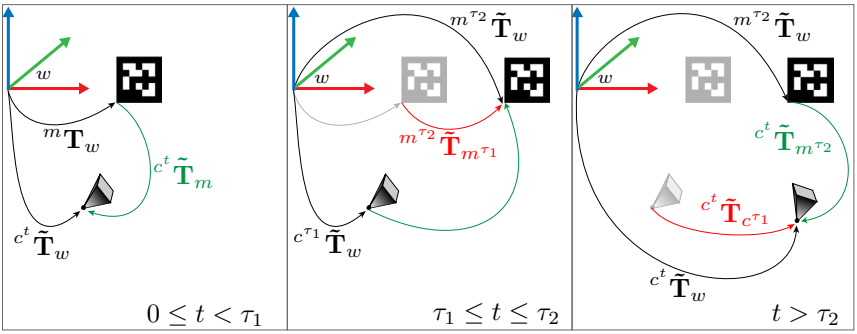


Figure 4.2: The basic MOMA odometry cycle. At $t = 0$ the marker is static and the camera can obtain its pose $^{c}\tilde{\mathbf{T}}_{w}$ knowing the initial marker pose $^{m}\mathbf{T}_{w}$. During $0 \leq t < \tau_1$ the camera moves in relation to the marker and estimates its pose $^{c^t}\tilde{\mathbf{T}}_{w}$ continuously by measuring the relative pose $^{c^t}\tilde{\mathbf{T}}_{m}$ to the marker. During the time interval $\tau_1 \leq t \leq \tau_2$ the camera is static and the marker moves to a new location within the camera's field of view. At time $t = \tau_2$ the marker stops moving and the marker's absolute pose $^{m^{\tau_2}}\tilde{\mathbf{T}}_{w}$ can be estimated via $^{c^{\tau_1}}\tilde{\mathbf{T}}_{w}$ and $^{m^{\tau_2}}\tilde{\mathbf{T}}_{c^{\tau_1}}$. Finally, starting from $t > \tau_2$ the marker is static again and the camera moves using the marker pose $^{m^{\tau_2}}\tilde{\mathbf{T}}_{w}$ as a new reference to estimate its pose $^{c^t}\tilde{\mathbf{T}}_{w}$, closing the cycle.

The *advantages* of MOMA are: An **improved accuracy** with respect to other relative approaches like classical VO, only a **monocular camera** is needed to localize several robots since the markers already provide the scale of the environment, and finally but more importantly, it **does not require features in the environment**. Additionally, this method provides localization to the camera and the marker simultaneously even during movement (both robots in the basic cooperative scheme with only

one measurement device). The *disadvantages* are an increased control and navigation complexity due to the restriction on the movement of the robots since the marker has to stay in the field of view of the camera.

The motion patterns for MOMA odometry have the following movement restrictions:

1. The marker has to be static if the camera moves, and the camera has to be static as long as the marker moves. If more than one marker is used and at least one of the markers should remain static, then the camera and the rest of the markers are able to move (which is not possible for *CPS* [62]).

2. During the transitions, there must be a period of time where at least two devices are static (e.g. both camera and marker in a camera-marker configuration or two markers in a camera-multi-marker configuration).

MOMA implies new considerations in the classical action-perception cycle in robotics. The action-perception cycle is based on the premise of act then perceive or perceive and then act. Now, in MOMA, we have what we call the perception-interaction cycle since the action of the marker affects the perception of the observer and in turn its action as well. Thus, the marker can no longer be considered as a passive entity with no effect on the observer, MOMA is able to provide information regarding its current state to the observer, and the observer can also inform the MOMA which state is needed for the general behavior of the system in a given situation.

4.3 Robotic architectures based on MOMA

In this section, we will describe the possible configurations using mobile robots that we have considered based on a single monocular camera and traditional planar fiducial marker. In the experimental section, the development and testing of a multi-robot system with two of these architectures will be shown.

4.3.1 Caterpillar-like configurations

This is the most basic multi-robot configuration for MOMA. It equals the structure we assumed in Section 4.2 to define the mathematical elaboration.

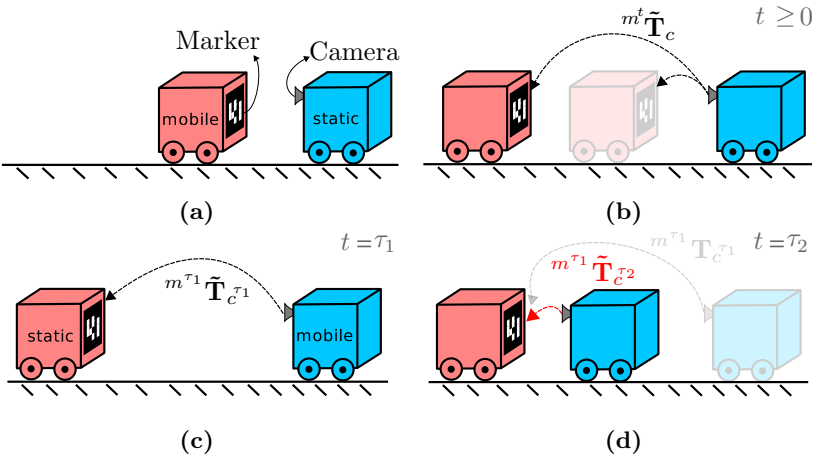


Figure 4.3: Two-robot Caterpillar.

Two-robot Caterpillar

In this configuration, one robot is the mobile marker and the other one is the *observer* (the one with the camera), see Fig. 4.3. The *observer* follows the movement of the mobile marker using the monocular camera. We named this particular kind of movement Caterpillar-like motion since each robot behaves like a segment of the body of a caterpillar.

The MOMA and the *observer* move in turns, following the rules explained in Section 4.2. The error accumulates only during the switching of the reference and is only dependent on the accuracy of the fiducial marker detection, which by using a good camera and proper calibration may be in the range of millimeters [8]. MOMA is able to track the pose of both robots during the movement and not only in the transitions.

Single-robot Caterpillar

In this minimal configuration, a single robot will be pulling a lightweight and rigid sled with a simple pulley mechanism, see Fig. 4.4. The robot can either actuate to pull the sled close to itself (while its wheels are locked to remain static) or let it drag behind. A monocular camera detects a fiducial marker in front of the sled. The robot performs Caterpillar-like motion leaving the sled behind as static reference when it has to move, then stops

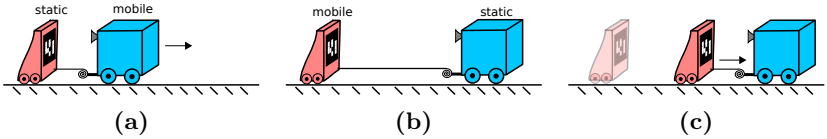


Figure 4.4: Single-robot Caterpillar.

and pulls the sled performing the *MOMA* odometry in the process.

Multi-robot Caterpillar

This is an extension of the basic Caterpillar case for N robots, see Fig. 4.5. Each robot follows the one in front. In this configuration $N-1$ robots with cameras are needed for the relative transformations. If at least one member of the group is static, the rest may move.

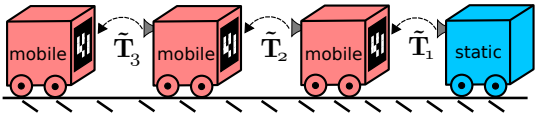


Figure 4.5: Multi-robot Caterpillar.

4.3.2 Top Mobile Observer

This configuration is based on two or more Unmanned Ground Vehicles (UGVs) with fiducial markers on top and an external mobile *observer* (e.g., a UAV), which looks down to all the robots simultaneously using a monocular camera, see Fig. 4.6. The UGVs move in turns as in MOMA odometry but the *observer* is fully mobile.

The *observer* is a very general concept in this configuration. One logical choice is a quadcopter or any other type of UAV with a bottom camera. However, in our tests, we also used a wireless camera in the hand of a person following the robots around the lab. One advantage of this configuration is that the MOMA odometry system will also fully locate the *observer* and the *observer* is allowed to be in continuous movement.

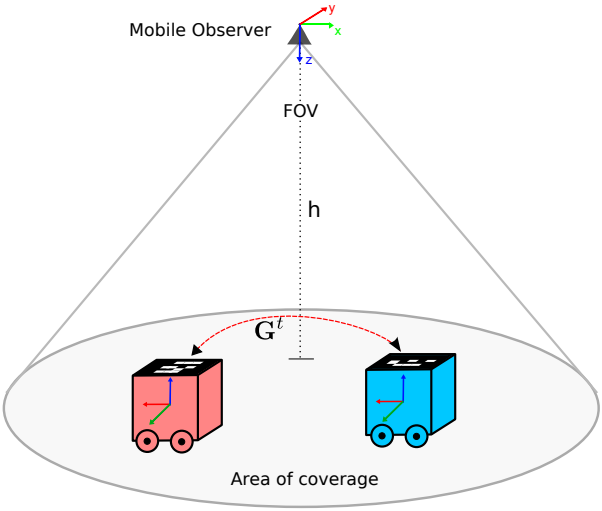


Figure 4.6: Top Mobile Observer.

4.3.3 Summary

The MOMA configurations may appear at first as a big restriction for the movement of a multi-robot system, but in practice, only one of the robots has to remain static to keep the MOMA odometry working, and this robot may be designed as a simple robot, since it only needs to move, not sense. The multi-robot system can use MOMA odometry to get to a particular working space, then the simple robot remains static meanwhile the complex ones move freely using some other odometry systems to perform their tasks, with the advantage that they can return to the static robot to correct drift as necessary. After finishing their tasks, they can move again as a group using MOMA odometry. This allows the robot team to maintain a drift-less odometry estimation meanwhile being able to perform more complex tasks in the environment.

4.4 Experiments on a multi-robot system

4.4.1 Hardware configuration

For a Caterpillar-like configuration our experimental setup consists of two omnidirectional robots (Robotino® from Festo Didactic Inc.). Each

Robotino has Aruco markers on the sides and top (Fig. 4.7). One of the robots was defined as the observer (with a FLIR Blackfly monocular camera) and the other as the mobile marker. For the Top Mobile Observer configuration, we use a quadcopter as an addition.

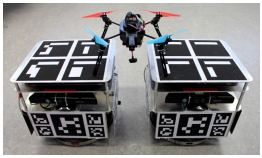


Figure 4.7: Robots used in our experiments.

We calibrated the coordinate frames of the markers and the camera of the robots using the *easy-hand-eye* ROS calibration package and the intrinsic parameters of the cameras using the standard ROS camera calibration package.

4.4.2 Software architecture

Our system's modules are shown in Fig. 4.8. They comprise the tasks of marker detection, MOMA estimation, global navigation planner (MOMA navigation), local navigation planner, and motion control of the ground robots and the quadcopter. The code was implemented in the Robot Operating System (ROS) framework and is openly available at our research's group github account[2], the active mobile marker may be manually controlled by a human operator using a joystick overriding the automatic navigation.

4.4.3 Experimentation and discussion

Two-robot Caterpillar

We performed this test inside a room with a high precision OptiTrack localization system for ground truth. The two Robotinos navigated three loops inside the room using the MOMA scheme in Caterpillar-like motion. The odometry of the robot's monocular camera was recorded both using MOMA odometry and the ground truth from the OptiTrack system. The measured trajectories for the final loop are displayed in Fig. 4.9. The final error (Euclidean distance) was 0.045 m or 0.138 % of the total trajectory,

[2] http://github.com/tud-rmr/tud_momana

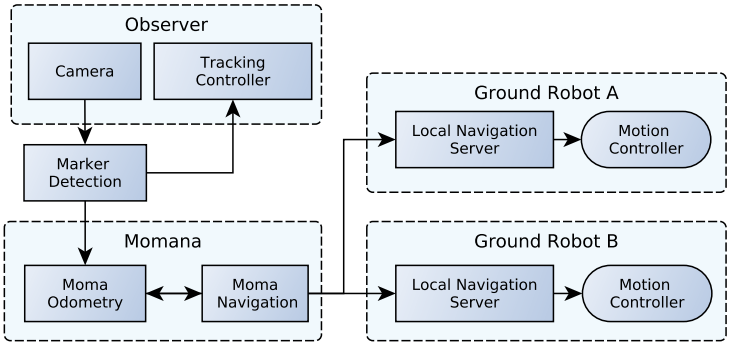


Figure 4.8: MOMA software structure for a multi-robot navigation system.

and the behavior of the error during the navigation is shown in Fig. 4.10. The positional error presents peaks followed by stabilization periods. The peaks happen during the robots' movement since the markers may appear blurred and in non-optimal configurations for the pose estimation algorithm, but as soon as the robots are static while doing the transition, the error decreases. Additionally, in closed-loop motions like this one, the errors are canceled due to symmetric motions. To properly study the performance of the MOMA, a more extended test without repetitive motions was performed.

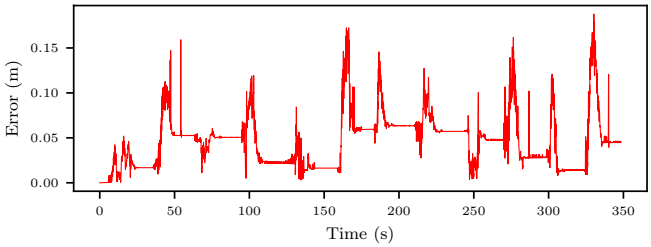


Figure 4.10: Euclidean error of MOMA during the trajectory of the final loop in the Two-robot Caterpillar test.

In Fig. 4.11, the results of a long trajectory are shown. The robots moved from Room A, the one with the Optitrack ground truth system, passed

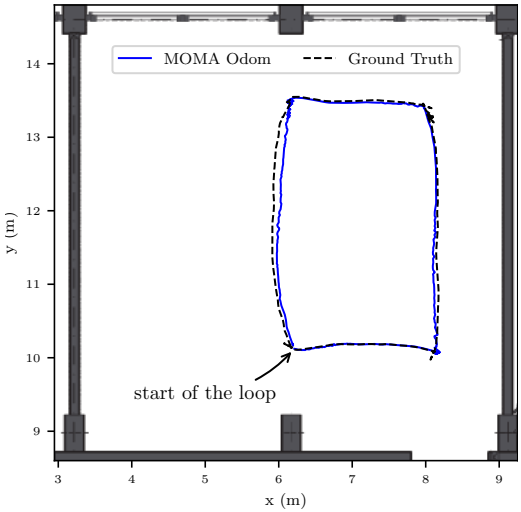


Figure 4.9: Final loop of the navigation for the two-robot Caterpillar configuration. The blue continuous line is the MOMA and the black dashed line the ground truth from the OptiTrack system.

through a low illuminated hallway into Room B, and finally, they returned once again to Room A. With this test, it was possible to measure the robots' start and final positions with the Optitrack system and compare it against MOMA. In Fig. 4.12, a magnification of the starting and final trajectory of the robot and the ground truth are shown. The error at the start and end of the trajectory is shown in Fig. 4.13. The final error was 0.38 m or 0.68 % of the total trajectory.

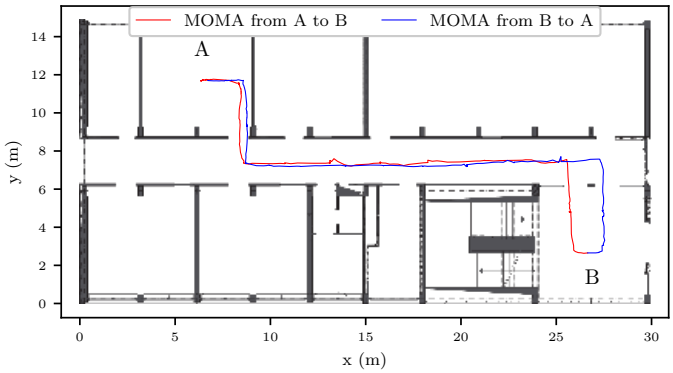


Figure 4.11: Results of the MOMA odometry for a long trajectory from Room A to Room B (red line) and the return trajectory (blue line) over a map of our institute.

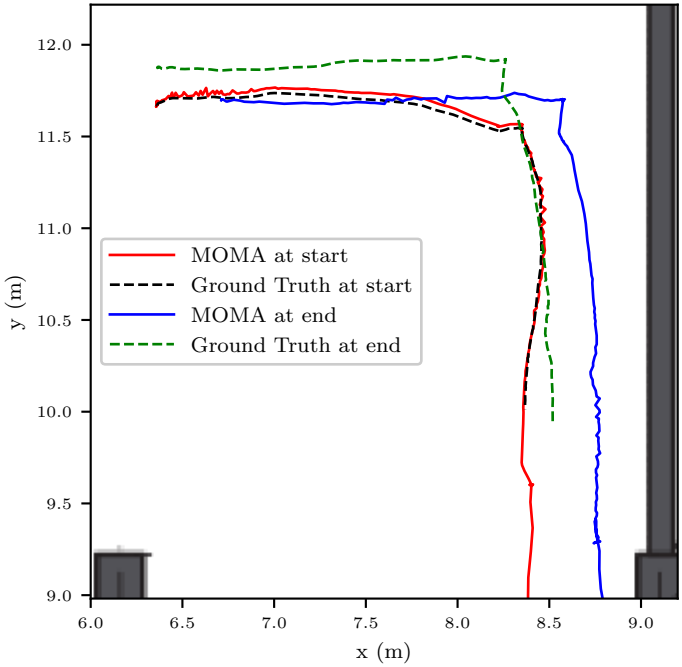


Figure 4.12: Detail of the trajectories shown in Fig. 4.11 for Room A with the ground truth obtained from OptiTrack.

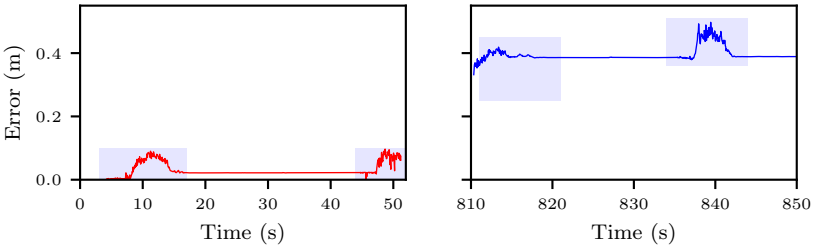


Figure 4.13: Detail of the error for the long trajectory Two-robot Caterpillar test. Left side: error detected in Room A at the start of the trajectory. Right side: error detected in Room A at the end of the trajectory. The highlighted sections correspond to the movement of the robot followed by a static phase, during movement the accuracy decreases but improve when static.

Top Mobile Observer configuration

Finally, we implemented and tested the configuration of Fig. 4.6. The UAV was an Ar.Drone 2.0 quadcopter with a camera attached to the bottom. A simple navigation task similar to the Caterpillar case was defined for our robotic system as a set of goals that form a square shape (side = 1 m). Each goal is a position and orientation in the map coordinate frame (x, y, θ).

This experiment's main goal was to compare MOMA to a regular VO approach in an environment that does not provide enough features. With this intention, we performed a test in a room of our institute with white walls and a floor with a repetitive texture. This set-up is usually a problem for VO systems in our tests.

For comparison, a frontal placed stereo camera was integrated into one of the UGVs, and its video feed was used to perform VO using the VISO2 algorithm [42] during the navigation task. The performance of both MOMA (using the monocular camera on the UAV) and VISO2 were compared against the ground truth provided by fixed ceiling cameras. The final metric of comparison was defined as the main robot's final pose after performing a loop measured by the ground truth system.

This test was executed ten times. In Fig. 4.14, the result of one of the experiments is shown. This test corresponds to the best VISO2 case; in general, VISO2 lost reference entirely in 4 of the 10 cases. For clarity, only the odometry information related to the main UGV is displayed, but MOMA can provide localization for all the robots in the team. VISO2

is only accurate as long as there are enough features in the environment (first quarter of the trajectory) and when the movement does not include pure rotations. When the main robot performs pure rotations at waypoint coordinates $(0, 1), (1, 1)$ and $(1, 0)$, the error in the pose estimation for the VO case increases significantly. A video of this test is also available[3].

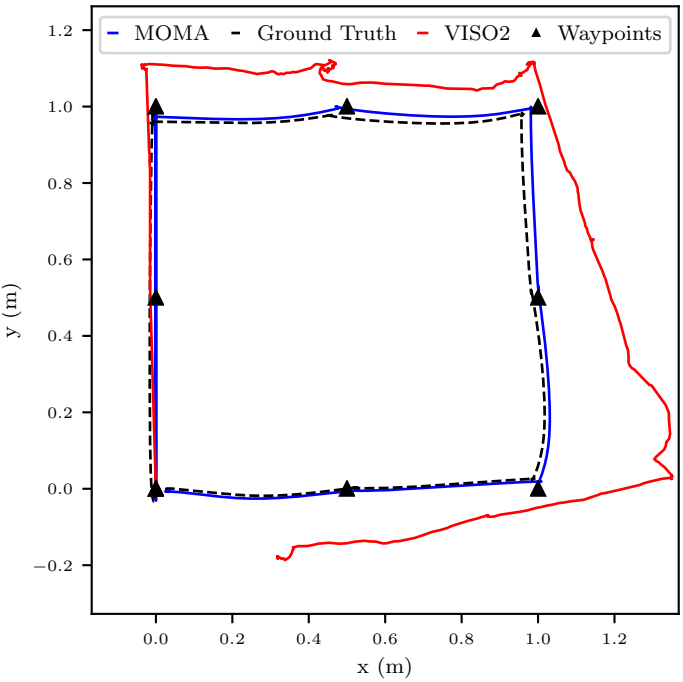


Figure 4.14: Odometry results for the main robot after waypoint navigation. In red is shown the behaviour of VISO2, where the error increases during rotations due to the lack of good features in indoor environments. MOMA (blue) follows the waypoints with low error.

The error on the final position when using MOMA was 0.51 cm or a 0.123 % of the total trajectory. For VISO2 we obtained an error of 33.81 cm or 7.99 %, this was the best case for VISO2. It has to be pointed out that VISO2 and other VO algorithms may achieve close to 1 % accuracy when good features are present in the environment. However, this test

[3] https://youtu.be/0xASGfH8cDM

was designed to be particularly challenging, highlighting the advantages of MOMA in low-feature (or featureless) environments.

4.4.4 Evaluation and extensions to MOMA

As a final evaluation of MOMA, we believe that the proposed method is an exciting tool for existing multi-robot systems in featureless environments. Possible practical applications include: 1) Robot localization in indoor environments low in textures such as empty hangars or buildings. 2) Completely dark environments where the markers may be implemented with LEDs or using computer screens; this saves energy versus illuminating the whole environment. 3) Underwater environments, e.g., pool cleaning robots or seabed exploration. 4) Finally, together with any other odometry system to increase the localization accuracy, for example, stereo odometry, wheel odometry, or inertial, to mention a few. In this last point, we worked on an extension to MOMA by fusing the information coming from a stereo-based V-SLAM system to increase accuracy and robustness. We call this extension Visual Slam Mobile Marker Odometry (S-MOMA).

Starting with the Two-robot Caterpillar configuration of Fig. 4.3, if we now attach a stereo camera to the front robot, we can measure environmental features with the front robot and calculate its pose by performing VO, and at the same time, take monocular-based marker measurements with the back robot. We can use the information from both measurements to improve the pose estimation of the robots. The main idea is that fusing the two sources of pose estimates will be more accurate than each one independently; this will benefit scenarios where the environment is rich in good features while still working in featureless environments, thanks to MOMA. What follows is a short summary of the main aspects of the formulation and results of S-MOMA, which have been published in [69].

The homogeneous 3D coordinates of environmental features in the world frame ${}^{w}\boldsymbol{x}$ can be found by stereo VO triangulation (see Section 4.2) and their projection in the left camera of the stereo camera setup is ${}^{\eta}\boldsymbol{x}$. We can then write the reprojection error of environmental features at any given time in the left camera frame as follows:

$$\epsilon_e = d({}^{\eta}\boldsymbol{x},\, \boldsymbol{\Pi}_0\, {}^{s}\tilde{\mathbf{T}}_{w}\, {}^{w}\boldsymbol{x})^2\,, \tag{4.9}$$

where the transformation between the left camera of the stereo setup and the world can be found by concatenating:

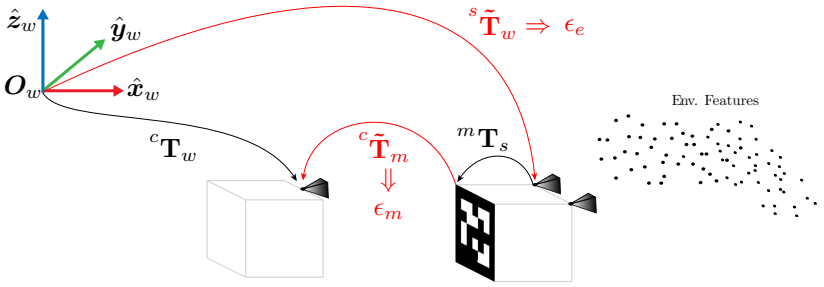


Figure 4.15: S-MOMA configuration.

$$ {}^s\tilde{\mathbf{T}}_w = {}^s\mathbf{T}_m \, {}^m\tilde{\mathbf{T}}_c \, {}^c\mathbf{T}_w \tag{4.10} $$

In a similar way, we can use the homogeneous 3D coordinates of marker features in marker coordinates ${}^m\boldsymbol{y}$ and their projection ${}^\eta\boldsymbol{y}$ (now on the back monocular camera) to define a reprojection error for the marker features:

$$ \epsilon_m = d({}^\eta\boldsymbol{y}, \, \boldsymbol{\Pi}_0 \, {}^c\tilde{\mathbf{T}}_m \, {}^m\boldsymbol{y})^2 \, . \tag{4.11} $$

With these two reprojection errors, which are part of the same loop of transformations (see Fig. 4.15), we can find an optimal ${}^s\hat{\mathbf{T}}_w$ as follows:

$$ {}^s\hat{\mathbf{T}}_w = \underset{{}^s\tilde{\mathbf{T}}_w}{\arg\min} \left\{ \frac{1}{N_e} \sum_{i=1}^{N_e} \epsilon_e^i + \frac{\lambda}{N_m} \sum_{j=1}^{N_m} \epsilon_m^j \right\} . \tag{4.12} $$

We normalize the reprojection errors with their corresponding feature numbers (N_e, N_m) to counter the imbalance in the number of features for each case since it may be that the environment at a given moment has more features than the marker and vice-versa. We also include a factor λ that allows us to control the influence of each kind of reprojection error into the optimization; the value of λ will depend on the relative quality of the cameras used. We evaluated S-MOMA in scenarios without features, scenarios rich in environmental features, and a mixture. We found that when S-MOMA is used in featureless scenarios, we obtained results similar to MOMA, and in feature-rich scenarios, the accuracy of S-MOMA was even better than other V-SLAM approaches.

Some remarks about the planar fiducial marker detection are relevant. In a Caterpillar-like configuration, the estimation of the pose for planar fiducial markers does not provide reasonable depth estimates (Z-axis of the camera); this may be solved by selecting other fiducial marker structures. We will discuss more on this topic in the following chapters. The Top Mobile Observer configuration is more precise than the other configurations since it is based on measurements on the camera's XY plane. Nonetheless, to give more freedom of movement to the UGVs, the UAV has to fly higher (which may decrease marker detection accuracy). Since the moment of switching is the most critical part of the method (when the error accumulates), it is essential to find new ways of improving the marker estimation accuracy.

5 Optimal points configurations

In the previous chapter, we focused on using accurate marker features instead of environmental features for pose estimation. The reason is simple: marker features are well defined and can be extracted reliably with higher accuracy than those present in the environment. However, the improvements obtained were still limited by the control points defined inside the fiducial marker and the algorithm used to calculate the pose. There is a relationship between the pose of the camera in relation to the marker and the accuracy of the pose estimation, which is not easy to model since it depends both on the spatial configuration of the points and the method used to solve the pose; this makes us wonder: is there an optimal set of artificial features? Is there an optimal marker at all?

In this chapter, we go deeper into the study of artificial features by investigating the influence of the spatial configuration of a number of $n \geq 4$ control points on the accuracy and robustness of space resection methods.

5.1 Introduction

The Perspective-n-Point (PnP) problem and the particular case of planar pose estimation via homography estimation are some of the most researched topics in computer vision and photogrammetry. Even though the research in these areas has been comprehensive, there is a surprising lack of information regarding the effect of 3D control point configurations on the estimation methods' accuracy and robustness.

In Section 2.3, we reviewed the existing PnP methods, and it is clear from the literature that control point configurations are relevant and influence the accuracy and robustness of pose estimates. However, the information available is rather general since it has been based on hands-on experience and thus far only leads to some thumb rules. The most obvious and widely accepted is that increasing the number of control points increases the accuracy of the results in the presence of noise. Further on,

in several studies, when simulations are performed to compare methods, great care is given to avoid some point configurations, such as non-centered data or near-planar cases, which are singularities, or configurations that are degenerate cases for certain estimation methods. There is a need to find out which point configurations are better, so that fair comparisons can be made.

A more thorough evaluation is available for the *normalized* DLT algorithm, whereas the normalization has already been shown to improve the estimation because it is related to the condition number of the set of DLT equations [48]. The only error analysis for homography estimation found so far in the literature shows only statistical results and simulations of the errors in the homography coefficients [17].

However, none of the above answer the question: *Are there optimized perspective-n-point configurations that can increase the accuracy and robustness of space resection methods?*

If there are, this question includes several follow-up questions: Are the optimized configurations dependent on the pose, or is there only one optimal configuration for all poses? What are the specifics of this/these configuration(s) in relation to absolute coordinates and relative distances between coordinates? Are there similarities between configurations that differ in the number of control points? When does an increase in the number of arbitrarily selected points outperform optimized configurations with a smaller number of points?

We search for an answer to these questions first in the planar case by **proposing an optimization objective to find optimized planar control point configurations**. Fig. 5.1 sketches the main idea of optimizing the proposed objective via a gradient descent approach and the stepwise improvement of the accuracy of the pose estimate starting from some initial control point configuration. Each descent step leads to a change in the control point configuration and thus defines stable dynamics for the control points placed on a planar visual fiducial marker (object plane) converging to stable control point configurations. This research is relevant for those interested in developing PnP and space resection algorithms and fiducial marker designers.

This chapter is structured as follows: In Section 5.2, we summarize known findings about control point configurations for pose estimation. In Section 5.3, we summarize the golden standard for PnP. In Section 5.4, we describe our method to find optimal control points. Then, in Section 5.5, we describe the simulation results, and finally, in Section 5.5.3, we discuss the results and give conclusions.

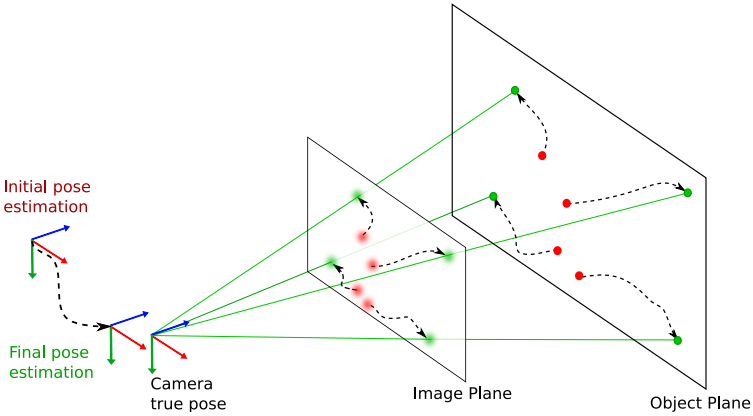


Figure 5.1: Optimizing point configurations: Control points with known 3D coordinates on the object plane (marker) in arbitrary configuration (red) are moved towards optimized configurations (green) for pose estimation from these control points and their corresponding noisy projections on the image plane (blurry red and green). The control points' dynamic (5.16) is given by the gradient descent steps minimizing the optimization objective (5.15) that results in improved pose estimations (from red to green) close to the true camera pose (black).

5.2 Control points configurations

It has been pointed out in the literature [65, 67] that 3D point configurations have an influence on the local minima of the PnP problem. In the RPnP method paper [67], a broad classification of the control points configurations into three groups is presented. The classification is based on the rank of the matrix $\mathbf{M}^\top\mathbf{M} \in \mathbb{R}^{3\times 3}$, where $\mathbf{M} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^\top$, $\boldsymbol{x}_i$ is the 3D coordinate of control point i, and n is the amount of control points.

In EPnP [65], it is shown that if the control points are taken from the *uncentered data* or the region where the image projections of the control points cover only a small part of the image, the stability of the compared methods greatly degrades. In RPnP it is elaborated that based on the previous classification this *uncentered data* is a configuration that lays between the *ordinary 3D case* and the *planar case*.

Some assumptions about the influence of the control points configurations are also present in the IPPE paper [21]. Through statistical evaluations, the authors found out that the accuracy for the 4-point case de-

creases if the points are uniformly sampled from a given region. They circumvent this problem by selecting the corners of the region as the positions for the control points and then refer the reader to the Chen and Suter paper [17], where the analysis of the stability of the homography estimation to 1st order perturbations is presented. In this analysis, it is clear that the homography estimate's error depends on the singular values of the $\mathbf{A}$ matrix in the DLT algorithm (see also next section).

Additionally, in [19, 119] evaluations are presented characterizing pose-dependent offsets and uncertainty on the camera pose estimations. Simulations empirically prove that some camera poses are more stable for the estimation process than others.

5.3 Golden standard algorithm for pose estimation

Before we explain the optimization method for optimizing point configurations, we shortly summarize the *golden standard* optimization methods for pose estimation from planar point configurations, which are the minimization of the reprojection (geometric) error for iterative methods and the minimization of the algebraic error for non-iterative methods via the DLT algorithm.

5.3.1 Minimization of the reprojection error

Given a 3D-2D point correspondence of i-th 3D control point with homogeneous world w coordinates ${}^{w}\boldsymbol{x}_i = [{}^{w}x_i, {}^{w}y_i, {}^{w}z_i, 1]^\top$ and its corresponding projection onto a planar calibrated camera[1] with normalized image homogeneous coordinates ${}^{\eta}\boldsymbol{x}_i = [{}^{\eta}x_i, {}^{\eta}y_i, 1]^\top$, the relation between these points is given by a relative transformation ${}^{c}\mathbf{T}_w$ between world w and camera c frame, ${}^{c}\boldsymbol{x}_i = {}^{c}\mathbf{T}_w \, {}^{w}\boldsymbol{x}_i$, followed by a projection $\lambda \, {}^{\eta}\boldsymbol{x}_i = \boldsymbol{\Pi}_0 \, {}^{c}\boldsymbol{x}_i$ (see Section 2.2); this leads to the relation:

$$\lambda \, {}^{\eta}\boldsymbol{x}_i = \boldsymbol{\Pi}_0 \, {}^{c}\mathbf{T}_w \, {}^{w}\boldsymbol{x}_i \, . \tag{5.1}$$

Including additive noise $\boldsymbol{e}_i = [e_i, \zeta_i, 1]^\top$ on the error-free coordinates ${}^{\eta}\boldsymbol{x}_i$, we get noisy measurements of the image coordinates ${}^{\eta}\tilde{\boldsymbol{x}}_i = {}^{\eta}\boldsymbol{x}_i + \boldsymbol{e}_i$.

[1] Assuming the calibration matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ to be known, the homogeneous pixel coordinates ${}^{p}\boldsymbol{x}_i = [{}^{p}x_i, {}^{p}y_i, 1]^\top$ can be transformed to homogeneous normalized image coordinates in metric units ${}^{\eta}\boldsymbol{x}_i = \mathbf{K}^{-1} \, {}^{p}\boldsymbol{x}_i$.

Thus, we can solve for the reprojection error $\epsilon_i = d\left({}^{\eta}\tilde{\boldsymbol{x}}_i, {}^{\eta}\boldsymbol{x}_i\right)^2$. Minimizing this error for all n points, leads to the following least-squares estimator for the optimal pose:

$$ {}^{c}\hat{\mathbf{T}}_w = \text{argmin}_{{}^{c}\mathbf{T}_w} \sum_{i=1}^{n} \epsilon_i \,, \quad n \geq 3 \,. \tag{5.2} $$

Iterative gradient descent optimization of (5.2) leads to the most accurate pose estimation results in the literature so far, also for planar point configurations. However, this optimization relies in a very good initial estimate to avoid local minima; in the planar case this initial estimate is usually found through homography decomposition [49], so the key element to find is the homography between the object plane and the image plane.

5.3.2 Homography estimation

If the control points $\{({}^{w}\boldsymbol{x}_i) | i = 1, \ldots, n\}$ are all on a plane ρ, we can define a 2D subspace in the 3D world with homogeneous coordinates ${}^{\rho}\boldsymbol{x}_i = [{}^{\rho}x_i, {}^{\rho}y_i, 1]^{\top}$. Plugging the planar constraint in equation (5.1) leads to a homography mapping

$$ \lambda {}^{\eta}\boldsymbol{x}_i = {}^{\eta}\mathbf{H}_{\rho} \, {}^{\rho}\boldsymbol{x}_i \,, \tag{5.3} $$

where the homography matrix ${}^{\eta}\mathbf{H}_{\rho}$ that maps points from the plane ρ to normalized camera plane coordinates η is formed by the first two columns of the rotation matrix ${}^{c}\mathbf{R}_{\rho}$ and the translation vector ${}^{c}\boldsymbol{t}_{\rho}$:

$$ {}^{\eta}\mathbf{H}_{\rho} = \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} . \tag{5.4} $$

Dividing the first row of equation (5.3) by the third row and the second row by the third row, we get two linearly independent equations for each point correspondence:

$$ -h_1 {}^{\rho}x_i - h_2 {}^{\rho}y_i - h_3 + (h_7 {}^{\rho}x_i + h_8 {}^{\rho}y_i + h_9) {}^{\eta}x_i = 0 \tag{5.5} $$

$$ -h_4 {}^{\rho}x_i - h_5 {}^{\rho}y_i - h_6 + (h_7 {}^{\rho}x_i + h_8 {}^{\rho}y_i + h_9) {}^{\eta}y_i = 0 \,, \tag{5.6} $$

which can be converted into matrix form:

$$ \mathbf{A}_i \boldsymbol{h} = \mathbf{0} \,, \tag{5.7} $$

where

$$\mathbf{A}_i = \begin{bmatrix} {}^{\rho}x_i & {}^{\rho}y_i & -1 & 0 & 0 & 0 & {}^{\eta}x_i{}^{\rho}x_i & {}^{\eta}x_i{}^{\rho}y_i & {}^{\eta}x_i \\ 0 & 0 & 0 & {}^{\rho}x_i & {}^{\rho}y_i & -1 & {}^{\eta}y_i{}^{\rho}x_i & {}^{\eta}y_i{}^{\rho}y_i & {}^{\eta}y_i \end{bmatrix} \tag{5.8}$$

and

$$\boldsymbol{h} = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^\top \tag{5.9}$$

Again, assuming noisy measurements of the image coordinates ${}^{\eta}\tilde{\boldsymbol{x}}_i$, we get noisy matrices

$$\tilde{\mathbf{A}}_i = \mathbf{A}_i + \mathbf{E}_i \,. \tag{5.10}$$

From $\tilde{\mathbf{A}}_i\boldsymbol{h} = (\mathbf{A}_i + \mathbf{E}_i)\boldsymbol{h}$ we can solve for the algebraic error $\|\mathbf{E}_i\boldsymbol{h}\|_2^2 = \|(\tilde{\mathbf{A}}_i - \mathbf{A}_i)\boldsymbol{h}\|_2^2 = \|\tilde{\mathbf{A}}_i\boldsymbol{h}\|_2^2$ of each point, because $\mathbf{A}_i\boldsymbol{h} = \mathbf{0}$ holds. Minimizing the squared 2-norm of all points for the optimal homography $\hat{\boldsymbol{h}}$ leads to the following least-squares estimator

$$\hat{\boldsymbol{h}} = \operatorname{argmin}_{\boldsymbol{h}} \sum_{i=1}^{n} \|\mathbf{E}_i\boldsymbol{h}\|_2^2 \,, \quad s.t.\ \|\boldsymbol{h}\| = 1 \,, \quad n \geq 4 \,. \tag{5.11}$$

Since $\boldsymbol{h}$ contains 9 entries, but is defined only up to scale, the total number of degrees of freedom is 8. Thus, the additional constraint $\|\boldsymbol{h}\| = 1$ is included to solve the optimization.

Now, stacking all $\{\tilde{\mathbf{A}}_i\}$ and $\{\mathbf{E}_i\}$ as $\tilde{\mathbf{A}} = [\tilde{\mathbf{A}}_1^\top, \ldots, \tilde{\mathbf{A}}_n^\top]^\top \in \mathbb{R}^{2n\times 9}$ and $\mathbf{E} = [\mathbf{E}_1^\top, \ldots, \mathbf{E}_n^\top]^\top \in \mathbb{R}^{2n\times 9}$, we arrive at solving the noisy homogeneous linear equation system

$$\tilde{\mathbf{A}}\boldsymbol{h} = \mathbf{E}\boldsymbol{h} \,. \tag{5.12}$$

The solution of (5.12) is equivalent to the solution of (5.11) and is given by the DLT algorithm applying a singular value decomposition (SVD) of $\tilde{\mathbf{A}} = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^\top$, whereas $\hat{\boldsymbol{h}} = \tilde{\boldsymbol{v}}_9$, with $\tilde{\boldsymbol{v}}_9$ being the right singular vector of $\tilde{\mathbf{A}}$ associated with the least singular value $\tilde{s}_9$. Usually, an additional normalization step of the coordinates of the control points and its projections is performed leading to the normalized DLT algorithm, which is the golden standard for non-iterative pose estimation because it is very easy to handle and serves as a basis for other non-iterative as well as iterative pose estimation methods.

Once a homography is found, the normal pipeline for pose estimation is to perform homography decomposition to find an initial pose and then refine this pose with a non-linear optimization based on the reprojection error.

5.4 Optimizing points configuration for pose estimation

To find optimal control points configurations, we need a proper optimization criterion. In the following, we propose an optimization criterion that is optimal for planar pose estimation using the (normalized) DLT algorithm, since it is the critical first step in planar pose estimation methods (even the gold standard of the minimization of the reprojection error requires a good initial guess, which is obtained from the DLT). The reader should not confuse the optimization goal, which we will elaborate on in this section (to find better point configurations), with the optimizations used to find an optimal homography from noisy measurements. In this case, we are trying to directly optimize the configuration of the $\mathbf{A}$ matrix by defining optimal 3D spatial positions of the control points $\{{}^{\rho}\boldsymbol{x}_i | i = 1, \ldots, n\}$, so the configuration will be more stable and less affected by the noise present in image measurements $\{{}^{\eta}\tilde{\boldsymbol{x}}_i | i = 1, \ldots, n\}$, this in turn will produce better homography estimates.

We start by availing ourselves of perturbation theory applied to singular value decomposition of noisy matrices [108] and have a look at the first-order perturbation expansion for the perturbed solution of the DLT algorithm, presented in [17], which is

$$\hat{\boldsymbol{h}} = \tilde{\boldsymbol{v}}_9 = \boldsymbol{v}_9 - \sum_{k=1}^{8} \frac{\boldsymbol{u}_k^{\top} \mathbf{E} \boldsymbol{v}_9}{s_k} \boldsymbol{v}_k \,, \tag{5.13}$$

where $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$ are the left and right singular vectors of the unperturbed matrix $\mathbf{A}$, s_k the corresponding singular values and $\mathbf{E}$ the measurement errors. Equation (5.13) clearly shows that the optimal solution for the homography that equals the right singular vector of the unperturbed matrix $\mathbf{A}$, associated with the least singular value[2] $s_9 = 0$, is perturbed by the second term in (5.13). The second term is a weighted sum of the first eight optimal right singular vectors $\boldsymbol{v}_k$, whereas the weights $\boldsymbol{u}_k^{\top} \mathbf{E} \boldsymbol{v}_9 / s_k$ are the influence of the measurement errors $\mathbf{E}$ on the unperturbed solution $\boldsymbol{v}_9$ along the different k dimensions of the model space, and $\boldsymbol{u}_k$ are the left singular vectors. The presence of very small s_k in the denominator can give us very large weights for the corresponding model space basis vector $\boldsymbol{v}_k$ and dominate the error. Hence, small singular values s_k cause the estimation $\hat{\boldsymbol{h}}$ to be extremely sensitive to small amounts of noise in the data

[2] The singular values are arranged in descending order: $s_1 \geq s_2 \geq \cdots \geq s_8 \geq s_9 = 0$.

and correlates with the singular value spectrum[3] $(s_1 - s_8)$ as follows: The smaller the singular value spectrum, the less perturbed the estimation is. It is also well known, that the condition number of a matrix with respect to the 2-norm is given by the ratio between the largest and, in our case, the second-smallest[4] singular value [43]

$$c(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2 = \frac{s_{max}}{s_{min}} = \frac{s_1}{s_8}\,, \tag{5.14}$$

which is minimal if the singular value spectrum is minimal. The normalization of the control points and its projections leads to the normalized DLT algorithm, which has shown to improve the condition of matrix $\mathbf{A}$ [48]. Thus, we simply try to minimize the condition number c of matrix $\mathbf{A}$ with respect to all n control points $\{{}^{\rho}\boldsymbol{x}_i | i = 1, \ldots, n\}$ as follows:

$$\{{}^{\rho}\hat{\boldsymbol{x}}_i\} = \text{argmin}_{\{{}^{\rho}\boldsymbol{x}_i\}} c\left(\mathbf{A}(\{{}^{\rho}\boldsymbol{x}_i\})\right)\,. \tag{5.15}$$

Notice that this optimization can be performed offline, as in simulation, thus without noise in the projected points $\{{}^{\eta}\boldsymbol{x}_i\}$. Optimization of (5.15) is realized with a gradient descent minimization, whereas to calculate the gradient vector, we use automatic differentiation[5] [92]. This leads to the final discrete control points dynamics

$${}^{\rho}\boldsymbol{x}_i(t+1) = {}^{\rho}\boldsymbol{x}_i(t) - \alpha(t)\nabla c\left(\mathbf{A}\left(\{{}^{\rho}\boldsymbol{x}_i\}(t)\right)\right)\,, \tag{5.16}$$

for each iteration t and stepsize $\alpha(t)$, which is adapted using SuperSAB [111]. The control points dynamics can now be used to find optimal control point configurations for pose estimation from planar markers.

Error bounds on homography estimation when noise is present

Now, if we assume that there are perturbations on the matrix $\tilde{\mathbf{A}}$, as a product of noisy measurements ${}^{\eta}\tilde{\boldsymbol{x}}_i$, we can define the relative error on the estimation of the homography parameters as $\boldsymbol{\xi} = (\hat{\boldsymbol{h}} - \boldsymbol{h})/\boldsymbol{h}$, and from perturbation theory, the following inequality defines an upper bound for the relative error:

$$\|\boldsymbol{\xi}\| \leq c(\mathbf{A}) \frac{\|\tilde{\mathbf{A}} - \mathbf{A}\|}{\|\mathbf{A}\|}\,. \tag{5.17}$$

[3] Here, the singular value spectrum between the first and second-last singular value is relevant, because $s_9 = 0$ holds.

[4] In our case, the smallest singular value is zero because (5.7) is a linear homogeneous equation system.

[5] In our implementation we used *autograd* [28].

To find a lower bound, we can use the error of using a perturbed matrix $\tilde{\mathbf{A}}$ with the true homography $\boldsymbol{h}$, defined as $\tilde{\mathbf{A}}\boldsymbol{h}$, and the error of using the optimal homography estimation $\hat{\boldsymbol{h}}$ with the same perturbed matrix, defined as $\tilde{\mathbf{A}}\hat{\boldsymbol{h}}$, to build the following inequality:

$$\|\tilde{\mathbf{A}}\hat{\boldsymbol{h}} - \tilde{\mathbf{A}}\boldsymbol{h}\| = \|\tilde{\mathbf{A}}(\hat{\boldsymbol{h}} - \boldsymbol{h})\| \leq \|\tilde{\mathbf{A}}\|\|\hat{\boldsymbol{h}} - \boldsymbol{h}\|, \tag{5.18}$$

which then divided by $\|\boldsymbol{h}\|$ leads to a lower bound of the relative error:

$$\|\boldsymbol{\xi}\| \geq \frac{\|\tilde{\mathbf{A}}(\boldsymbol{h} - \hat{\boldsymbol{h}})\|}{\|\tilde{\mathbf{A}}\|\|\boldsymbol{h}\|}\,. \tag{5.19}$$

The upper bound implies that there are only two ways to improve the maximum values of the relative error, either by reducing the perturbation of the $\mathbf{A}$ matrix, which in practice usually cannot be controlled or by improving the condition number of the $\mathbf{A}$ matrix, which can be done by a normalization step in the DLT transform and by **selecting optimized control point configurations** $\{{}^{\rho}\hat{\boldsymbol{x}}_i\}$, as in equation (5.15).

5.5 Simulation and real experiment results

In this section, we will demonstrate how the optimization method of equation (5.15) improves the accuracy of different pose estimation algorithms when noise in image coordinates is present. We are interested mainly in the control points' final position after the optimization, but we will also show how the accuracy changes during the optimization. It is important to remark that this optimization can be executed offline to find the optimal control points for a particular setup, which later can be used to define a static fiducial marker for a pose estimation task.

Our simulation setup is based on a perspective camera model and a planar visual marker on the $\hat{\boldsymbol{x}}\hat{\boldsymbol{y}}$ plane of the world coordinates, centered in the origin $\boldsymbol{O}_w = [0, 0, 0]^\top$. We impose an arbitrary circular limit with a radius of 0.15 m; this allows a smooth movement of the control points during the optimization while keeping them inside the camera image. Rectangular limits were also tested, but the discontinuities on the corners restrict the points' movement, so they tend to stay on the corners.

A set of control points are randomly defined inside the limits of this circular plane, which are then projected onto the camera image[6]. A uniform distribution of 400 camera poses is defined around the marker as

[6] Image size: 640×480 px.
Camera intrinsic parameters $\mathbf{K} = [800, 0, 320; 0, 800, 240; 0, 0, 1]$, no distortion.

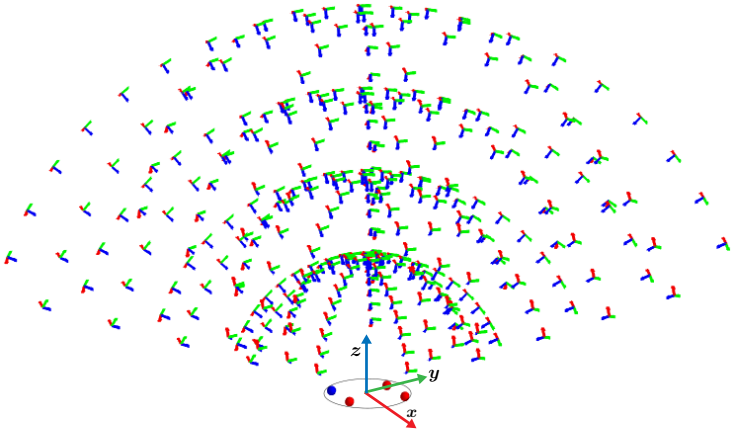


Figure 5.2: Distribution of 400 camera poses used in simulations. A plane ρ, delimited by a circle with n inner control points $\{{}^{\rho}\boldsymbol{x}_i | i = 1, \ldots, n\}$, is shown at the bottom of the figure. The cameras are distributed on spheres of evenly sampled radii. Each camera is looking at the center of the circular plane keeping all the points inside the camera's field of view.

displayed in Fig. 5.2; this distribution provides a wide combination of rotation and translations (without lack of generalization) in the whole range of detection and allows us to compare the final point configurations in image coordinates properly.

5.5.1 Optimization influence in homography estimation

To evaluate the effect of the optimization (5.15) on the underlying homography matrix estimate $\hat{\mathbf{H}}(t)$ when using a given set configuration of n homogeneous control points $\{{}^{\rho}\boldsymbol{x}_i(t) | i = 1, \ldots, n\}$ at time t, we define

$$HE(t) = \frac{1}{M} \sum_{j=1}^{M} d({}^{\eta}\tilde{\boldsymbol{x}}_j(t),\, \hat{\mathbf{H}}(t)\, {}^{\rho}\boldsymbol{x}_j(t))^2 \,, \tag{5.20}$$

as the homography reprojection error induced by the optimal homography $\hat{\mathbf{H}}(t)$ for a fixed set of M validation control points $\{{}^{\rho}\boldsymbol{x}_j\} \notin \{{}^{\rho}\boldsymbol{x}_i(t)\}$ that are evenly distributed on the object plane covering an area larger than the limits of the circle. Thus, it is possible to measure how good $\hat{\mathbf{H}}(t)$ is

able to represent the true homography $\mathbf{H}$ beyond the space of the control points.

Each simulation for a given camera pose is then performed in the following way:

- 1) An initial random n-point set $\{{}^{\rho}\boldsymbol{x}_i(t_0)\}$ is defined inside the circular plane.
- 2) For each iteration step t, an improved set of control points $\{{}^{\rho}\boldsymbol{x}_i(t)\}$ is obtained by (5.15) and projected to camera pixel coordinates $\{{}^{p}\boldsymbol{x}_i(t)\}$ using the true camera pose ${}^{c}\mathbf{T}_{\rho}$ and the calibration matrix $\mathbf{K}$. We add Gaussian noise to the projected points $\{{}^{p}\tilde{\boldsymbol{x}}_i(t)\}$ and use the correspondences $\{{}^{p}\tilde{\boldsymbol{x}}_i(t), {}^{\rho}\boldsymbol{x}_i(t)\}$ to calculate $\mathbf{A}(t)$ and $c(\mathbf{A}(t))$.
- 3) For each iteration t, we performed 1000 runs of the homography estimation using the normalized DLT algorithm[7] since we want a statistically meaningful measure of the homography estimation robustness against noise.
- 4) Finally, we calculate the error metric $HE(t)$ for each run and the average $\mu\left(HE(t)\right)$ and standard deviation $\sigma\left(HE(t)\right)$ of this error for all runs.

As illustration of the gradient minimization process, we present an example case of a simulation in a fronto-parallel camera pose for a 4-point configuration. A Gaussian noise of $\sigma_G = 4$ pixel is added to image coordinates for the homography estimation runs. In Fig. 5.3, the initial object and image point configurations are shown.

The evolution of $c(\mathbf{A}(t))$, as well as $\mu\left(HE(t)\right)$ and $\sigma\left(HE(t)\right)$, is presented in Fig. 5.4. The condition number decreases drastically in the first iterations of the gradient descent, and by doing so, the mean and standard deviation of $HE(t)$ is also reduced. With more iterations, both metrics slowly and smoothly converge to a stable minimum value.

[7] The homography estimation method presented in [46] and the method based on gradient descend of OpenCV were also tested. The results almost do not differ for low point configurations to the DLT, so it was the chosen one for the experiments.

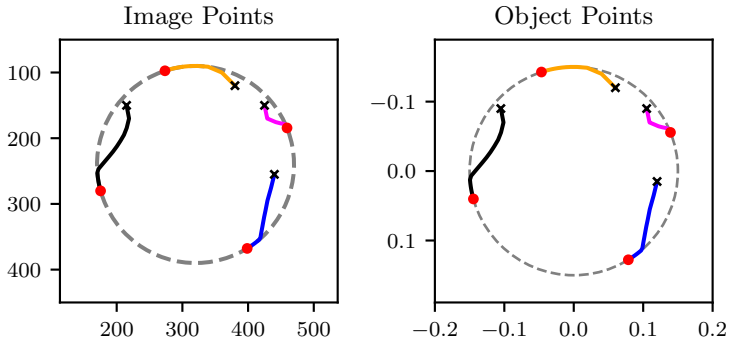


Figure 5.3: (Fronto-Parallel). Movement of control points in image and object coordinates during optimization for a fronto-parallel camera configuration simulation until an optimized configuration (red dots) limited by the circle (dashed grey line). Notice how the final object shape is close to a square.

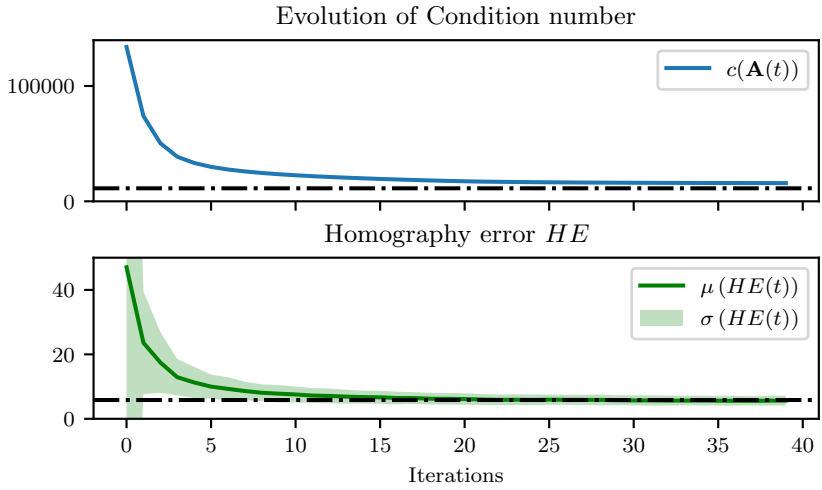


Figure 5.4: (Fronto-Parallel). Condition number $c(\mathbf{A}(t))$ and homography reprojection error $HE\,(t)$ during gradient descent. For comparison, the dashed-dotted black line represents the mean value for an ideal 4-point square.

This first result in itself is highly representative as it proves that some point configurations increase the accuracy of the homography estimation because they are more robust to noise. The results also prove that it is

indeed possible to obtain optimized control point configurations with a better performance than randomly selected points.

5.5.2 Optimization influence in pose estimation

Motivated by the homography results, we now want to test if the optimization of control point configurations could improve the accuracy of planar pose estimation algorithms. Thus, three different pose estimation algorithms[8] were run at each step t of the optimization process, namely: 1) a non-iterative PnP method **EPnP** [65], 2) a planar pose estimation method **IPPE** [21], and 3) an iterative one based on the Levenberg-Marquardt optimization denoted as **LM**.

As in similar works [21, 65], we denote $\left(\hat{\mathbf{R}}(t), \hat{\mathbf{t}}(t)\right)$ as the optimal estimated rotation and translation for a given camera pose at iteration t and by $(\mathbf{R}, \mathbf{t})$ the true rotation and translation. The error metrics for pose estimation are defined as follows:

- $\mathrm{RE}\left(\hat{\mathbf{R}}(t)\right)$ is the rotational error (in degrees) defined as the minimal rotation needed to align $\hat{\mathbf{R}}(t)$ to $\mathbf{R}$, which can be obtained from the axis-angle representation of $\hat{\mathbf{R}}(t)^\top \mathbf{R}$. We define the operator $\angle()$ to represent the extraction of the angle of the axis-angle representation of a rotation matrix, with the help of this operator we define the rotational error as:

$$\mathrm{RE}\left(\hat{\mathbf{R}}(t)\right) = \angle(\hat{\mathbf{R}}(t)^\top \mathbf{R})\,. \tag{5.21}$$

- $\mathrm{TE}\left(\hat{\mathbf{t}}(t)\right)$ is the relative error in translation defined as:

$$\mathrm{TE}\left(\hat{\mathbf{t}}(t)\right) = \frac{\|\hat{\mathbf{t}}(t) - \mathbf{t}\|_2}{\|\mathbf{t}\|_2} \times 100\%\,. \tag{5.22}$$

Similar to the homography simulation, for each iteration t, 1000 runs of the pose estimation with noisy image points (added 4-pixel Gaussian noise in image coordinates) for each of the PnP methods were performed. Then, the mean and standard deviation of RE and TE for the 1000 runs were calculated for each iteration. For the fronto-parallel case, we show in Fig. 5.5 how the mean of RE and TE change during the evolution of the condition number, and in Fig. 5.6 we present a more detailed view of the errors for each method, including the standard deviation.

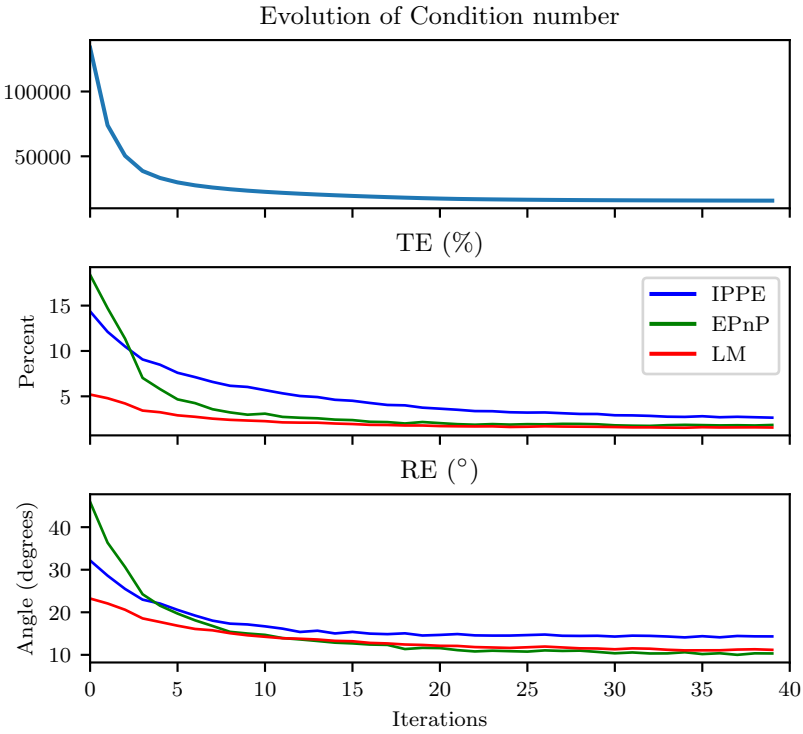


Figure 5.5: (Fronto-Parallel). Comparison of the evolution of the mean errors for different planar pose estimation methods during the iterative optimization process. The initial points were the same for all runs.

The pose error decreases during the optimization of the condition number for all methods, even for the LM algorithm, which is already using an optimization based on the reprojection error. The improvement on LM is because this algorithm needs a good initial estimate to converge to the correct pose and optimized control points produce a better initial estimate. We chose the fronto-parallel camera configuration because it is the most challenging; this is evidenced by the high values of the RE (more than 10 degrees); if the camera is inclined, we see also the same improvement

[8]For the EPnP and LM methods, the OpenCV implementations were used, and for IPPE the Python implementation provided in the author's GitHub repository.

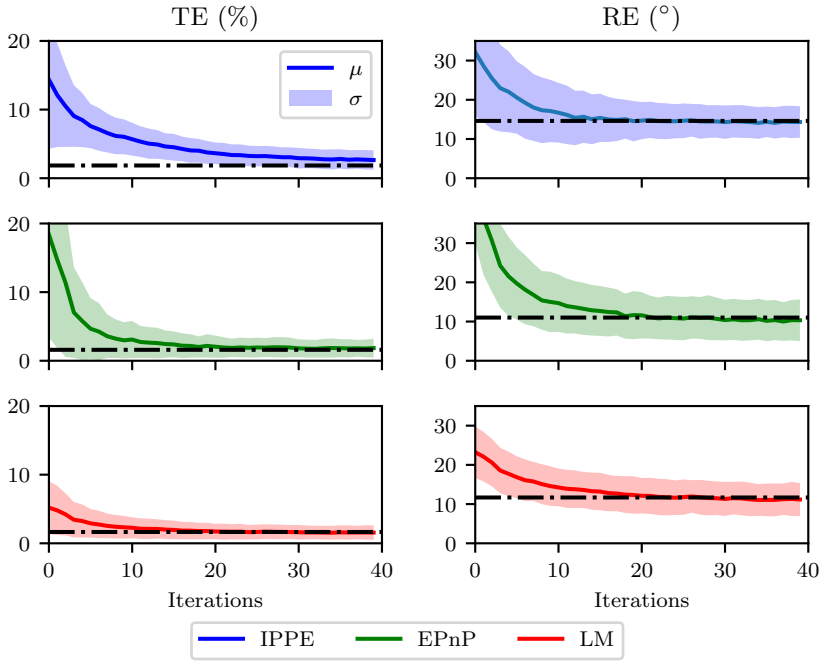


Figure 5.6: (Fronto-Parallel). Mean values (lines) and standard deviations (filled colored areas) of translational and rotational error for each method. The dashed-dotted black line represents the mean value for a 4-point square.

during the optimization, but the final errors in rotation are less than 5 degrees.

A real experiment[9] was also implemented in order to test if the simulation assumptions (Gaussian image noise and perfect intrinsics) may affect the results in practical applications. A computer screen was used as the planar fiducial marker to display the points during gradient descent dynamically. A set of 4 circles was displayed for each iteration of the optimization. These circles were then captured by a PointGrey Blackfly camera[10] and detected using a circle detector based on the Hough transform. We performed 100 detections for each gradient descent iteration.

[9] A video of this experiment: https://youtu.be/a6lDrwgqNmY.

[10] Camera image size: 1288×964 px. Camera intrinsic parameters: $\mathbf{K} = [1070.82, 0, 647.98; 0, 1071.20, 488.27; 0, 0, 1]$.

An Optitrack system was used to measure the camera's ground truth pose relative to the marker screen. The results of running the optimization process for a set of 4 random initial points are shown in figures 5.7, 5.8 and 5.9.

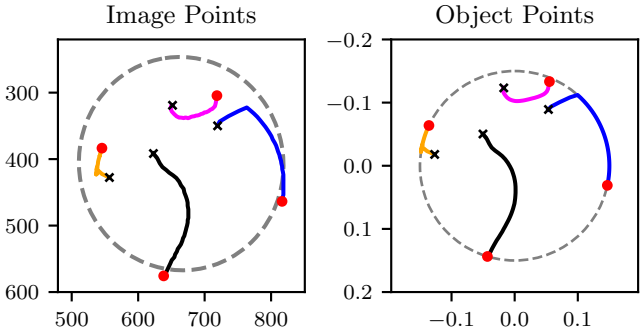


Figure 5.7: (Real). Movement of control points in image and object coordinates during gradient descent for the experiment with a real camera.

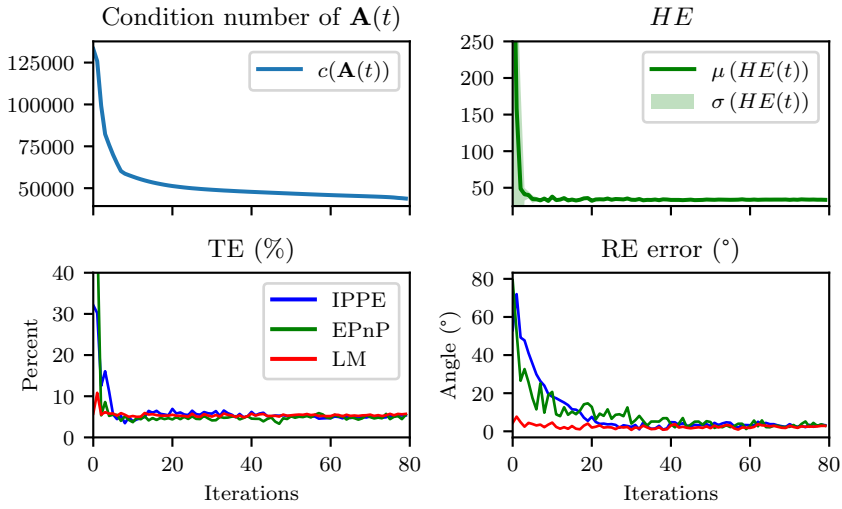


Figure 5.8: (Real). Evolution of the condition number and the homography reprojection error during gradient descent using a real camera.

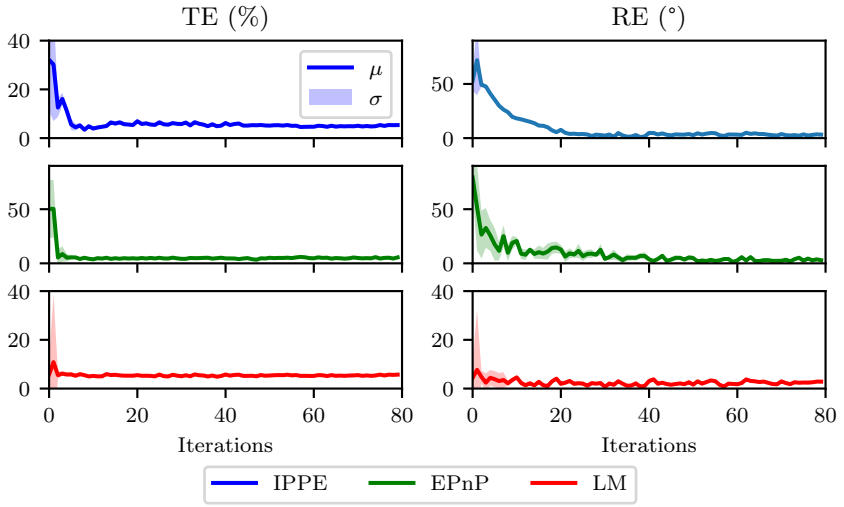


Figure 5.9: (Real). Detailed view of the standard deviation of each method represented by the filled, lightly colored areas.

We also studied the relationship between badly configured points and optimized points. For each camera pose in the distribution of Fig. 5.2, 100 different initial random n-point configurations with $n \in \{4, 5, 6, 7, 8\}$ were simulated and the optimization process was performed. In this case, only the initial and final values of the point configuration metrics are stored. Thus, it is possible to compare the methods based on *ill-conditioned* (random initial points) and *well-conditioned* (after optimization) point configurations. We present the results for the homography estimation in Fig. 5.10 and the results of the pose estimation in Fig. 5.11. Finally, in Fig. 5.12, the final point configurations for all the camera poses are shown as a 2D histogram, and some example configurations are shown for the 4-point, 5-point, and 6-point case.

5.5.3 Discussion

The results show that control point configurations have a substantial effect on the accuracy of homography and planar PnP methods. There are indeed optimized configurations that are better than random, and it is possible to find them using our method.

For the 4-point case, our empirical results show that a square-like shape

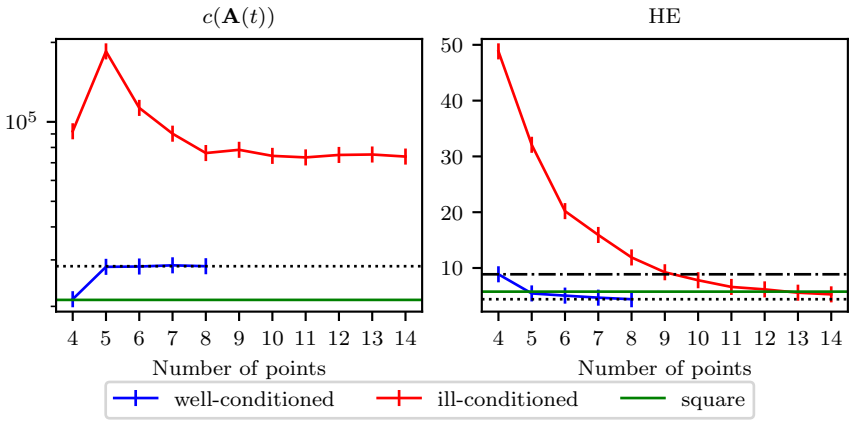


Figure 5.10: Robustness (cond. num.) and accuracy (homography error) dependent on the number of points for well- and ill-conditioned point configurations as well as an ideal 4-point square (green line).

is the most common minima and a very stable and robust configuration for all camera poses (see Fig. 5.12a); for 5-point, a common minimum is a pentagon (Fig. 5.12b) and for the 6-point a hexagon (Fig. 5.12c). The positions of the optimized point configurations do not show any strong dependency on the pose of the camera (besides scale and image limits). The real dependency is mainly related to the distribution of the points in camera image coordinates. As we can see from the 2D histograms, the points are driven to distribute themselves as much as possible in the available space (hence high repetition in circle boundaries), and they tend to increase the distance to each other, which favors the creation of some common regular polygons (squares, pentagons, and hexagons). For higher point numbers, the trend is still to place as many points as possible in the boundaries.

The first iterations of the optimization are when the increase in accuracy is more substantial, which means that the condition number is a good optimization objective. For example, the improvement in accuracy from a square-like configuration to a perfect square is tiny, but the increase of accuracy from random points to the square-like shapes obtained on the first iterations of the optimization is radical; this means that with less than five iterations, it is possible to obtain very stable and accurate configurations.

The smaller the number of control points, the higher is the relative

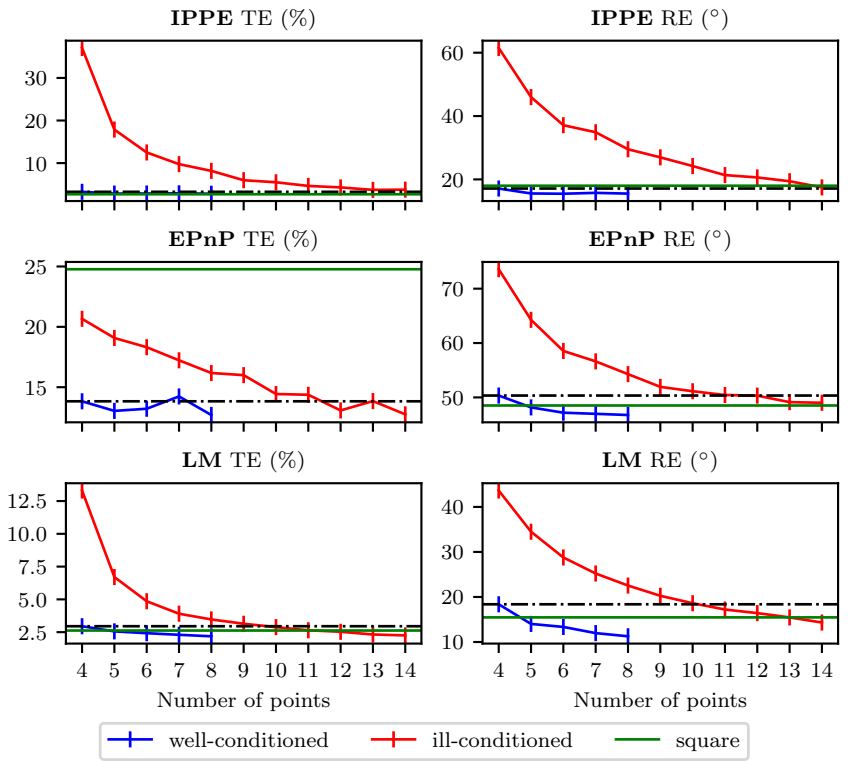


Figure 5.11: Comparison of different pose estimation methods for different numbers of control points for well- and ill-conditioned point configurations as well as an ideal 4-point square (green line).

improvement on the estimates for all of the evaluated methods. For example, the accuracy using 4 points is always better than random point configurations with more points $4 < n \leq 9$, as can be seen in Fig. 5.10 for homography, and in Fig. 5.11 for PnP. Thus, the control points' configuration has more effect on the accuracy than the number of control points.

The improvement in the EPnP and IPPE methods is more pronounced than for LM, which is an interesting result since those methods take considerably less computation time. For well-configured points, the methods converge to similar error values, and both mean and variance are reduced; this means that well-conditioned points can be used for a fair comparison of

pose estimation algorithms. LM also has increased accuracy, even though our optimization objective is not directly related to the minimization of the reprojection error; this shows the importance of having a good initial guess. The results of the real experiment closely match the simulations.

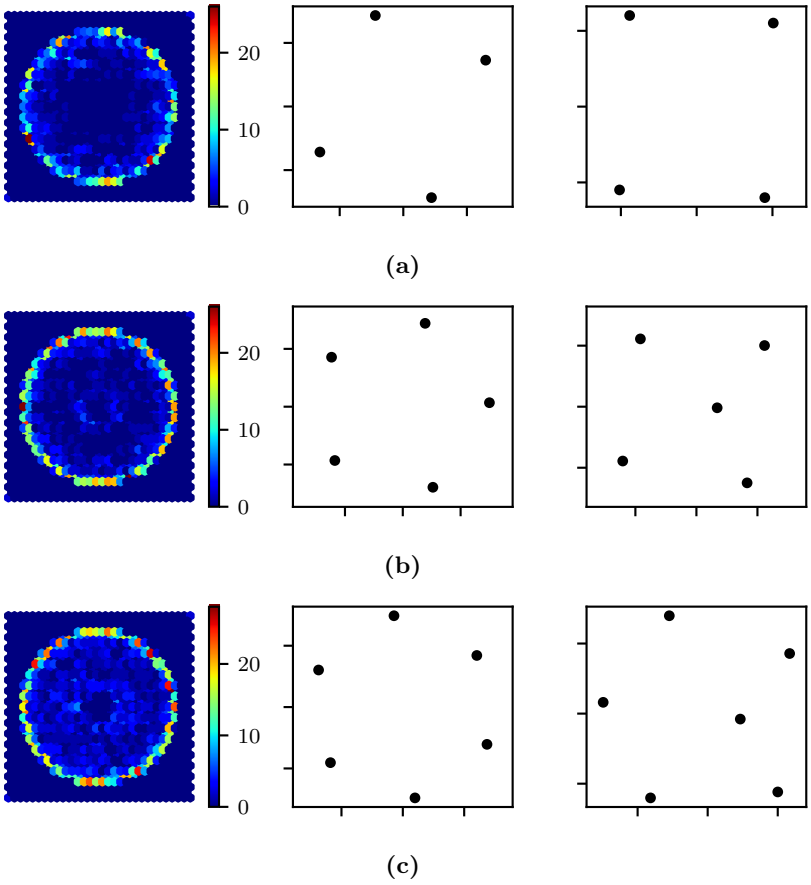


Figure 5.12: Final point configurations for 4 points (a), 5 points (b) and 6 points (c). On the left side: 2D histogram of final configurations for all 400 camera poses. Middle and right side: Examples of optimized final point configurations in object coordinates.

These results are significant for fiducial marker design; they validate that a square is a very stable configuration, and, additionally, some other regular polygonal shapes are proven to be stable and robust, e.g., the pentagon and hexagon. The hexagon is particularly interesting because it is also the most compact way of packing circles inside some planar limits; this would make a hexagonal grid an excellent and stable way of optimally distributing control points on a given surface.

We also performed some extensions of this work to non-planar configurations in [112], where we applied the DLT method to directly estimate a pose from 6 or more control points (non-planar). In this case, the limits of the optimization were a sphere instead of a circle. The final results showed again that the optimal points are distributed on the available space (the sphere borders) and arranged in regular polygonal shapes. For the 6-point configuration, the most common shape was a triangular prism inscribed in the sphere. In the 3D case, however, there may exist better metrics for optimization than the DLT since the algebraic error of the DLT does not match the geometric error (in the planar case, it does for the homography when the weights of the homogeneous coordinates are equal to 1 [49]), one possible goal for the optimization would be the condition number of the error covariance matrix [112].

6 Dynamic fiducial markers

In this chapter, we introduce a novel fiducial marker design, the dynamic marker. This new fiducial tries to optimize the pose estimation process by using the dynamism of a display device. The dynamic fiducial marker can change its appearance according to the spatiotemporal requirements of the visual perception task, especially in the case of a mobile robot using a camera as the sensor, see Fig. 6.1. We present a control scheme that changes the marker's appearance to increase the range of detection and assure better accuracy.

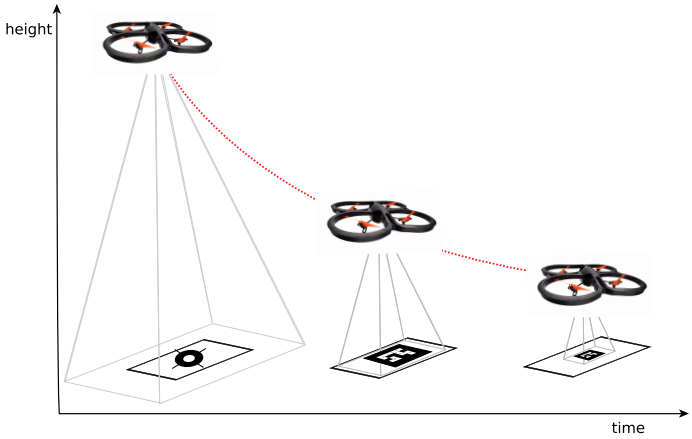


Figure 6.1: We propose a dynamic fiducial marker that can adapt over time to the requirements of the perception process. This method can be integrated into common visual servoing approaches, e.g., a tracking system for autonomous quadcopter landing. The fiducial marker's size and shape can be changed dynamically to better suit the detection process depending on the relative quadcopter-to-marker pose.

We first introduce the concept of a dynamic marker using as a base the theory of visual servoing, then we describe the modules necessary to implement a dynamic marker controller, and finally, we present two

different dynamic marker designs. The first marker design, the DDYMA, is based on discrete changes of the fiducial by selecting the appropriate shape from a family of known fiducial markers. The second design, the FDYMA, is a fiducial designed from scratch to better suit the dynamism of the screen; this fiducial changes its shape smoothly with pose changes, and the shape of the fiducial is designed to be optimal for pose estimation problems based on most of the results obtained in previous chapters. The DDYMA was tested on a quadcopter landing scenario and the FDYMA using a fixed optical tracking system as a comparison.

Our proposal is novel and simple: Instead of using a static fiducial marker, we propose using a screen to change the marker shape dynamically. A marker that changes requires a controller, which we must couple with the perception algorithm and the camera's movement. This section presents the minimal hardware/software set-up for what we call a discrete dynamic marker and introduces a control scheme that integrates conveniently into visual servoing. We will demonstrate that including a dynamic marker in the action-perception-cycle of robots improves the detection range of the marker, the accuracy of the pose estimation and improves the robot's performance compared to a static marker, which can be advantageous despite the increase in system complexity. It is worth noting that in previous publications, screens were used to display structured light sequences of images for camera calibration [107] [125] [44]. However, to the best of our knowledge, none of those applications exploited the possibilities of performing dynamic changes to the image based on the perception task's feedback. It is precisely this feedback that makes a dynamic marker an exciting concept for control applications.

6.1 Motivation

As explained in Chapter 3, a planar visual fiducial marker is a known shape, usually printed on a paper, located in the environment as a point of reference and scale for a visual task. Fiducial markers are commonly used in augmented reality, virtual reality, object tracking, and robot localization. In robotics, they are used to obtain the absolute 3D pose of a robot in world coordinates; this usually involves distributing several markers around the environment in known positions or fixing a camera and detecting markers attached to the robots. A fiducial may not be convenient for some applications due to the required environment intervention. For unknown environments, the preference is for other types of localiza-

tion systems that do not rely on artificial features or previous knowledge of the environment, i.e., Simultaneous Localization and Mapping (SLAM) or VO. Nonetheless, fiducial marker-based SLAM systems are still a topic of interest [71] [83] [80], mainly in controlled environments where ground truth is required and especially when the size of the environment is significant, and it is not practical to use an external localization system such as VICON.

Fiducial markers in cooperative robotics serve as a convenient and straightforward inter-robot relative localization system (e.g., MOMA). Since the markers are attached to the robots, no environment intervention is required. For example, in the work of Howard et. al [52], fiducial markers fixed on a team of robots are detected by the leader robot to combine the relative location of the other team members and fuse it with its global position. Dhiman et al. [25] proposed a multi-robot cooperative localization system that uses reciprocal observations of camera-fiducials to increase the accuracy of the relative pose estimation.

In teams of UAVs and UGVs, it is common to use fiducial markers mounted on top of the UGVs; this configuration was used for coordinated navigation of a heterogeneous team of UAVs-UGVs in the work of Saska et. al [102] and also by Mueggler et. al [79] for guiding a ground robot among movable obstacles using a quadcopter, and to perform odometry for a team of robots without external environmental features, as explained in Chapter 4. The vision-based autonomous landing of UAVs, e.g., autonomous quadcopter landing on static or moving platforms, is a representative example of fiducial markers for robotic systems. The landing point for the vehicle is usually defined using a marker that can be detected by a downward-looking camera in the UAV, and this marker can then be tracked for landing by a visual servoing controller [101] [68] [63] [10]. Complex fiducial markers allow the extraction of more information, such as full 3D pose and identification of the marker between an extensive library of possible markers. Additionally, the number of features used for pose calculation improve the accuracy of the calculated pose. However, there is a limit on the number of features that can be present in a given market area, and this directly affects the detection distance; this means that a complex marker is more challenging to detect at longer distances than a simpler one, and a simple marker shape may not be able to provide full 3D pose and identification.

The maximum range of fiducial marker detection is especially relevant for autonomous landing. For example, a large marker is wanted to increase the detection distance; however, if the marker is too big, and the camera is close, it will not be detected. In other robotic cooperative systems like

MOMA, not only the range is important but also the accuracy, and ideally, the magnitude of the pose estimation error should be constant and not depend on the pose. Another process in which the problems of range and accuracy in fiducial detection are relevant is camera calibration. To calibrate a camera, the user must sample several images of a fiducial in different relative poses. To get a good calibration, a high point density is ideal. However, the point density is limited since the static marker is usually designed for long-range detection, but this requires more display area, which means fewer control points. The range and accuracy requirements of pose estimation and camera calibration problems cannot be satisfied simultaneously by a static fiducial marker, and this is why a marker that can change and adapt would be useful. The transition from printed markers to screens is also natural since screens are now ubiquitous on robotic and Internet of Things (IoT) applications; it makes only sense to use their capabilities.

6.2 Dynamic fiducial marker definition

We define a dynamic marker as any known feature with a structural configuration that can be changed in time as needed. Since the marker is usually physically separated from the system that performs the perception, a dynamic marker must be an intelligent system capable of communicating with the detection system.

The following modules are needed for a minimum system configuration:

1. A screen of any kind of technology (LCD, OLED, E-INK, etc.).
2. A processing unit capable of changing the image on the screen on demand.
3. A communication channel between the perception system and the display system.

From now on, we will refer to these three modules as the Dynamic Marker. All these elements are commonplace nowadays. During our research, we used a foldable laptop, Ipads, and smartphones as dynamic markers.

6.2.1 Pose based visual servoing

Traditional monocular visual servo control uses the image information captured by the camera to control the movement of a robotic platform. Visual servo control can be separated into two categories: Image-Based Visual Servoing (IBVS), which is based directly on the geometric control of the image features, and Position-Based Visual Servoing (PBVS), which projects known structures to estimate a pose that is in turn used for robot control. We are going to focus on PBVS for our dynamic marker analysis, but the same concepts apply to IBVS [35].

The goal in a visual servoing approach is to minimize an error $\boldsymbol{e}(t)$ defined by

$$\boldsymbol{e}(t) = \boldsymbol{s}^* - \boldsymbol{s}(\boldsymbol{m}(t), \boldsymbol{a}), \tag{6.1}$$

where $\boldsymbol{s}$ is a vector of visual features in the camera image, these features are calculated using a set of image measurements $\boldsymbol{m}(t)$ (e.g., image pixel positions) plus the information contained in the vector $\boldsymbol{a}$, see Fig. 6.3. The vector $\boldsymbol{a}$ contains our knowledge about the system (e.g., camera intrinsics and target object structure). Finally, the vector $\boldsymbol{s}^*$ represents the desired value of the features in the image.

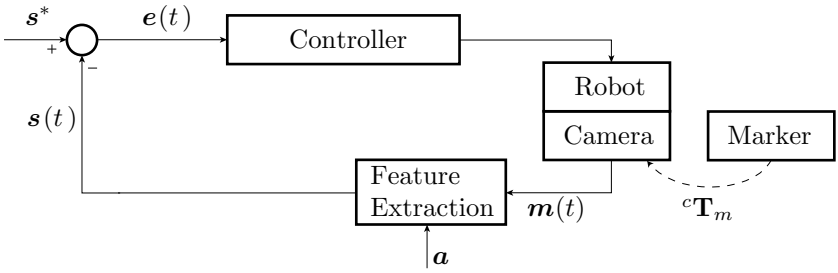


Figure 6.2: Image based visual servoing control architecture.

For IBVS, $\boldsymbol{s}$ is defined as a set of features that can be immediately obtained from the image measurement (without knowing the pose). For PBVS, the vector $\boldsymbol{s}$ include additionally the 3D parameters which are calculated using the immediately available image features together with the information of vector $\boldsymbol{a}$ (i.e., a pose). In PBVS, the vector $\boldsymbol{a}$ must include the camera intrinsic parameters and the 3D model of the object (in our case, the fiducial marker) so a pose can be estimated.

As an illustrative example, to control the camera pose in relation to a fiducial marker using the PBVS frame conventions, we denote the current camera frame as $\boldsymbol{F}_c$, the desired camera frame as $\boldsymbol{F}_c^*$ and the marker reference frame as $\boldsymbol{F}_m$. A translation vector ${}^c\boldsymbol{t}_m$ gives the coordinate of the marker frame relative to the camera frame, and the coordinate vector ${}^{c^*}\boldsymbol{t}_c$ gives the coordinate of the current camera frame relative to the desired camera frame. The matrix $\mathbf{R} = {}^{c^*}\mathbf{R}_c$ is the rotation matrix that defines the orientation from current camera frame to the desired frame, and $\theta\boldsymbol{u}$ is the angle-axis representation of this rotation. Both $\mathbf{R}$ and $\boldsymbol{t}$ can be obtained from a pose estimation algorithm based on visual features such as PnP.

With these definitions, the vector of visual features can be rewritten as $\boldsymbol{s} = ({}^{c^*}\boldsymbol{t}_c, \theta\boldsymbol{u})$, our desired $\boldsymbol{s}^* = 0$ and the error is then $\boldsymbol{e} = s$. A common control approach to minimize the error by moving the camera is to design a controller for the camera velocity. The rotational and translational motion can be decoupled in this case, and the control scheme will be:

$$\begin{cases} \boldsymbol{v}_c = -\lambda\, {}^{c^*}\mathbf{R}_c^\top\, {}^{c^*}\boldsymbol{t}_c & (6.2) \\ \boldsymbol{w}_c = -\lambda\theta\boldsymbol{u}\,, & (6.3) \end{cases}$$

where $\boldsymbol{v}_c$ and $\boldsymbol{w}_c$ are the camera translational and rotational velocities and λ is a proportional parameter. Using the previously defined strategy, we can control the translational and rotational velocities of the camera separately, to converge to the desired pose. A PBVS approach is very similar to traditional pose robot control where the camera is attached to the robotic platform (e.g., a robotic arm), see Fig. 6.3.

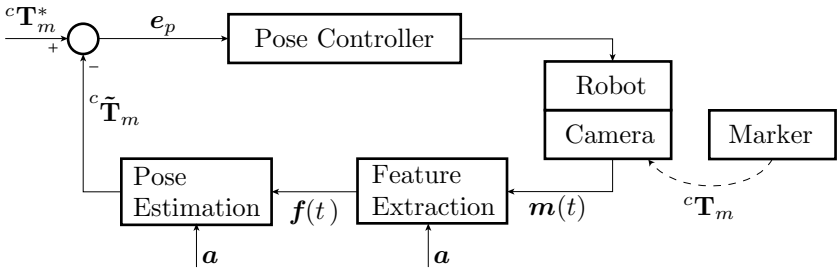


Figure 6.3: Position based visual servoing control architecture.

To summarize, a PBVS controls the camera pose to minimize feature space errors (or what the camera sees). However, the vector of visual

features $\boldsymbol{s}$ depends on image measurements, which change according to pose, and the parametrization $\boldsymbol{a}$, which is assumed static. $\boldsymbol{a}$ is static because the camera is always the same, and more importantly, the 3D model of the object does not change; what changes is the camera pose.

6.2.2 Dynamic marker controller

A dynamic marker is a fiducial marker that we control to optimize the extraction of image features and posterior pose estimation. The main control objective is that the marker appearance should be selected so that it increases the accuracy of the computer vision task for the current state. For example, in pose estimation, to increase the detection range, we may use a simple marker for the long-range and a more complex one (with full 6DOF pose capabilities) for the close range with varying scales.

We define the following basic modules of a dynamic marker as follows: 1) **Marker controller** is the module that calculates the control rules and defines the next marker shape; 2) **Display** is the module that draws the fiducials on the screen based on the parameters provided by the controller; 3) **Detect** is the processing unit in charge of detecting the fiducial in camera images and calculating the pose and relevant detection parameters to be fed back to the controller.

We propose a control architecture for the dynamic marker based on the theory of PBVS with the difference that now we control the displayed features instead of the camera pose. The key difference is that in this case the vector $\boldsymbol{a}$ changes over time since the marker shape can be modified on the screen. We define the vector of visual features as $\boldsymbol{s}(\boldsymbol{m}(t), \boldsymbol{a}(t))$, which shows the change of the displayed 3D object explicitly over time.

The proposed control loop for the marker shape on the screen is presented in Fig. 6.4. For the initial analysis, we assume that both the camera and the dynamic marker are static, with a relative pose from a marker frame to the camera frame defined by ${}^{c}\mathbf{T}_{m}$. We assume that there should exist an desired marker shape and scale that improves the performance of the pose calculation; we use this premise as the basis for our control architecture design.

We start with the vector $\boldsymbol{s}^{*}$ which contains a feature configuration in image coordinates optimal for pose estimation. For example, lets assume that a single circle is the ideal shape that we want to detect on camera images, to properly detect the circle it must occupy a minimum amount of pixels on the image (e.g., a diameter of 50px); in this minimal example we can define the vector of optimal features as $\boldsymbol{s}^{*} = [{}^{p}\phi^{*}]$, where ${}^{p}\phi^{*}$ is

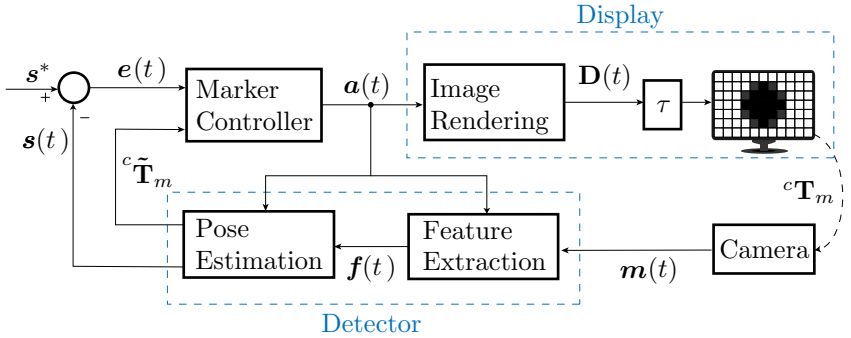


Figure 6.4: Dynamic marker control diagram.

desired circle diameter in image pixels units.

Analogous to the visual servoing approach, the control's goal is to minimize the error $\boldsymbol{e}$ defined by

$$\boldsymbol{e}(t) = \boldsymbol{s}^* - \boldsymbol{s}(\boldsymbol{m}(t), \boldsymbol{a}(t))\,. \tag{6.4}$$

The task of the controller is the generation of the vector $\boldsymbol{a}(t)$ which defines the 3D shape of the marker to be displayed. Using the same example, if the shape that we want to display is a circle, the vector $\boldsymbol{a}$ can be defined as $\boldsymbol{a}(t) = [\boldsymbol{k}^\top, \phi, x, y]^\top$, where $\boldsymbol{k}$ is the camera matrix in vectorized form, ϕ is the circle diameter in meters and (x, y) define the position in meters of the circle's center in marker coordinates; these basic parameters are then converted to a matrix of pixels by a rendering step and sent to the screen for display. We include the camera matrix since it can change during time due to variations on camera focus.

The plant in this control scheme is the display device or screen which is in essence a matrix of light producing dots (pixels) distributed in a flat surface. Most screens use LEDs to display light, each pixel is a combination of three different LEDs named RGB, R for red, G for green and B for blue; the combined intensity of the RGB LEDs allows the display of most of the colors perceived by the human eye. The separation between each pixel is the pixel pitch and its value is provided by the screen manufacturer. What we control on the screen is the luminosity of the RGB LEDs of each pixel. The luminosity can be represented with a 8 bit range of values where 0 represents totally black and 255 maximum intensity.

The control signal for the screen is then a matrix $\mathbf{D}(t)$ of size $(w \times h \times 3)$

where (w, h) are the number of pixels in the horizontal and vertical direction respectively. Each element $\mathbf{D}_{ij}(t) = [r, g, b]^\top$ contains the required intensity of the RGB LEDs at the (i, j) row and column of the screen.

The dynamics of the plant depend on the individual LED switching characteristics. There is a finite amount of time required for changing the intensity level of a single LED and this time depends on the LED technology, for example, regular grade consumer LEDs have a 80% stabilization time of around 4.8 ms, see Fig. 6.5, while Organic LEDs have a much faster response time of less than 0.1 ms, see Fig. 6.6.

For the dynamic marker the transitory state of the LED is not relevant, what matters is the final stable state of the LED, with this premise, we model our plant as an ideal screen with LEDs that have 0 ms switching time preceded by a time-delay with the value of the real LED switching time as shown in the Display section of Fig. 6.4. In the time-delay block we can also include other potential delays such as the communication between the controller and the display device or rendering delays which are device dependent and will be analyzed in detail in further sections.

We define the center of the screen as the origin of the marker coordinate system and since we know the pixel pitch and the (w, h) values we also know with sub-millimeter accuracy the position of each pixel $\mathbf{D}_{ij}$ in relation to the marker coordinate system.

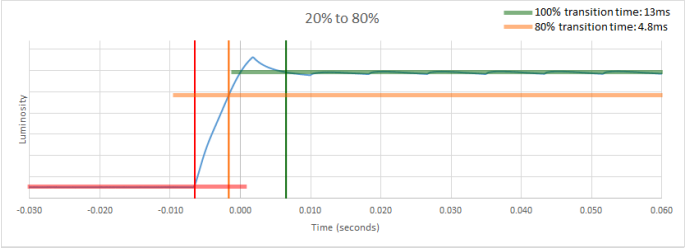


Figure 6.5: Response time of consumer grade photodiodes.

Once the information of $\mathbf{D}(t)$ is displayed on the screen it is captured by the camera which has a relative pose to the screen ${}^{c}\mathbf{T}_{m}$ that is unknown. The position of each pixel $\mathbf{D}_{ij}$ represented in marker homogeneous coordinates ${}^{m}\boldsymbol{r}_{ij}$ is then projected by the camera using the camera projection matrix as explained in Chapter 2:

$$\lambda\, {}^{p}\boldsymbol{r}_{ij} = \mathbf{K}\, {}^{c}\mathbf{T}_{m}\, {}^{m}\boldsymbol{r}_{ij}\,. \tag{6.5}$$

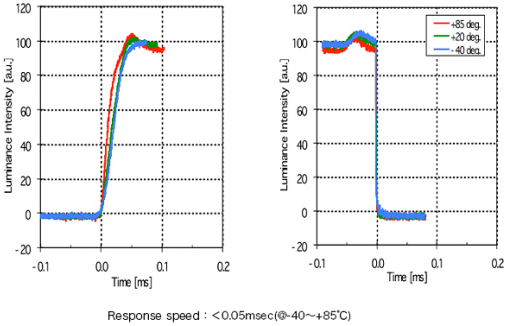


Figure 6.6: Organic LED response times.

Each position ${}^{p}\boldsymbol{r}_{ij}$ in camera pixel coordinates plus the color measurement of that pixel for all the screen pixels comprise the vector of measurements $m(t)$. In the next step a Feature Extraction algorithm will obtain the main desired parameters from the $m(t)$ vector. In our circle example the task of this step is to detect the circle and calculate its diameter ${}^{p}\phi$ in image pixel units, however we can extract additional features useful for pose estimation, like the circle border, the circle center and the circle inner color. All the detected features are collected into a vector of general features $\boldsymbol{f}(t)$ and passed to the pose estimation step, where, depending on the available features, an estimate of the marker to camera pose ${}^{c}\tilde{\mathbf{T}}_{m}$ can be calculated and fed back to the controller. The pose estimation module also builds the final vector $\boldsymbol{s}(t)$ which is then compared to $\boldsymbol{s}^{*}(t)$ to produce the error and close the loop.

At this point there are two scenarios, the first one is when the available data is enough to estimate a pose (i.e., we have a calibrated camera and enough features), in this case the pose estimation will output the estimated pose ${}^{c}\tilde{\mathbf{T}}_{m}$ and will set the $\boldsymbol{s}(t)$ vector to zero, so the input error to the controller will be directly $\boldsymbol{e}(t) = \boldsymbol{s}^{*}(t)$; this is because in this scenario the controller can directly compute the output of $\boldsymbol{a}(t)$ from ${}^{c}\tilde{\mathbf{T}}_{m}$, the camera matrix and the vector $\boldsymbol{s}^{*}(t)$ by simply doing an inverse projection of the desired features.

For example, to calculate the circle diameter in meters ϕ which is one of the $\boldsymbol{a}(t)$ parameters, we can build an homogeneous vector with the desired diameter in camera pixels $[{}^{p}\phi^{*}, 0, 1]^{\top}$ and calculate directly the diameter in meters ϕ to be displayed by the screen as follows:

$$\begin{bmatrix} \phi \\ 0 \\ 0 \\ 1 \end{bmatrix} = {}^{m}\tilde{\mathbf{T}}_{c}\mathbf{K}^{-1} \begin{bmatrix} {}^{p}\phi^{*} \\ 0 \\ 1 \end{bmatrix} . \tag{6.6}$$

The second scenario is when it is not possible to obtain a pose estimate, in this case the pose estimation block will set the output of the estimated pose to zero and output the vector $\boldsymbol{s}(t)$ of detected features which is then substracted to $\boldsymbol{s}^{*}(t)$ to obtain the error. In our circle example, the measured vector of features will be simply the measured circle diameter in pixel units $\boldsymbol{s} = [{}^{p}\phi]$ and the error input to the controller: $\boldsymbol{e}(t) = [{}^{p}\phi^{*} - {}^{p}\,\phi]$.

The error can then be used by a simple controller (e.g., a PID controller) to generate the $\boldsymbol{a}(t)$ vector closing the loop.

6.2.3 Dynamic marker PBVS

The dynamic marker can also be integrated into a PBVS approach. We assume, in this case, that the camera is part of a robotic platform and that its movement can be controlled. Fig. 6.7 shows the proposed control diagram. It is possible to define two separate control loops, one related to camera movement and another to marker changes, both of them try to minimize the overall pose error simultaneously. The dynamic marker tries to maximize the marker detection and pose accuracy, which results in better measurements for the PBVS.

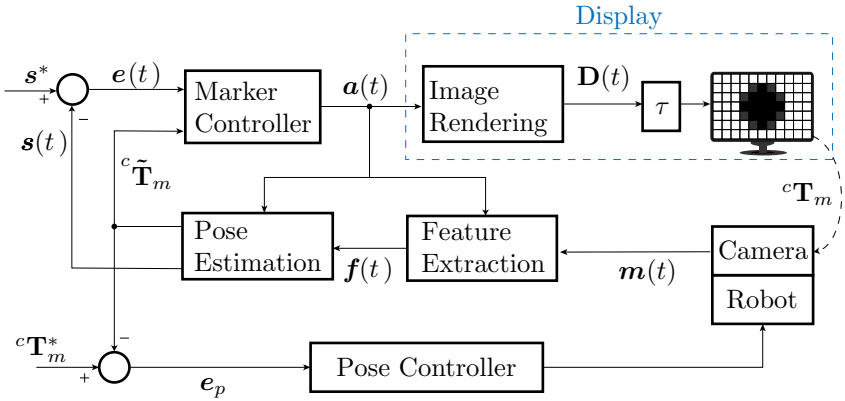


Figure 6.7: A dynamic marker used with a virtual servoing control approach.

There is an interesting consequence of having a dynamic marker in a PBVS control loop. If the marker scale and orientation $\boldsymbol{a}(t)$ changes on display without updating the detector, it is possible to directly control the pose of the robotic platform by only changing the marker.

6.2.4 System delay analysis

There is a race condition on the control loop of the dynamic marker that we must consider. From the moment the controller calculates a new $\boldsymbol{a}$, it may take some time for the information to arrive at both the display and the detector due to communication delays, and additionally, the device in charge of the display takes additional time to update the pixels with the new fiducial. Since the update path of the display can take more or less time than the update path of the detector, it is necessary to consider this difference to calculate the pose with the current parameters and not with the previous invalid ones.

We start the delay analysis assuming the most straightforward case when we execute all the steps one after the other in a synchronized manner. The details of the timing are shown in Fig. 6.8. The control loop starts with the Control module, an initial $\boldsymbol{a}(t_0)$ is generated and sent both to the Display and Detect modules. The Display module draws the new fiducial on the screen, followed by the Detect module, which sends a trigger signal to the camera to capture a new image. The Detect module receives the image, detects the fiducial using the information of $\boldsymbol{a}(t_0)$, calculates the pose and image features, and finally sends the information back to Control to repeat the cycle.

The are several relevant events in the synchronized loop. The initial time t_0 defines the instant when control sends the initial command with the new marker to both the display and detector. Signal received time t_s is the moment when the display receives the new marker command; we assume that the detector receives the signal before or close to this time; t_s can change in a non-deterministic way depending on the type of communication we can calculate it or calibrate it beforehand. The time of refresh t_r is the point in time when the screen is refreshed with the updated marker information, and the camera is signaled to capture the new marker, t_r depends on the display device and the screen technology. Image time t_i is when the new image has been captured by the camera and transmitted to the detector. For the sake of simplicity, we assume that the camera is working in trigger mode; if not, the camera's frame rate has to be considered. Finally, the Pose time t_p marks when the

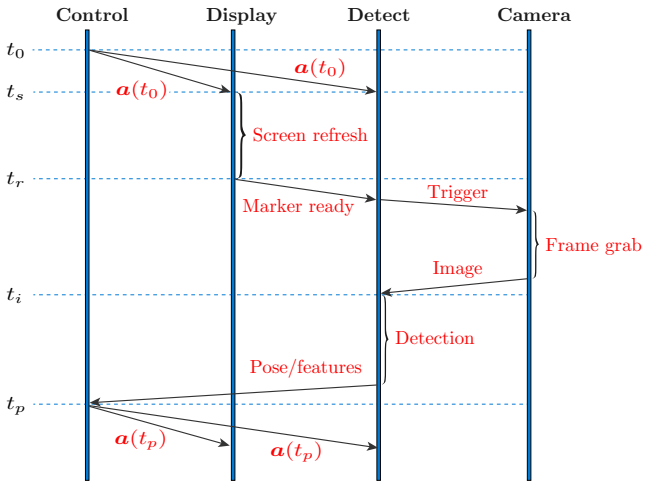


Figure 6.8: Relevant delays in the control loop of a dynamic marker.

detector estimated the marker features, calculated the camera pose, and transmitted the information back to the controller. At this point, the cycle can be repeated.

The delays that define the loop duration are then the communication delays, the screen refresh delay $(t_r - t_s)$, the image capture delay $(t_i - t_r)$, and the detection delay $(t_p - t_i)$. We are going to analyze each one of them independently.

Communication delay: If all the modules (control, display, and detect) are running on the same computer, the delays are relatively short and are mostly related to the software architecture required to pass messages, mainly due to the use of threads with sleep() functions. We will assume process switching latencies of around 1 ms in well configured OS. If the modules are running on different computers, the Ethernet or Wifi delay has to be added, e.g., Ethernet local network latencies range from 1 to 5 ms.

Screen refresh delay: We are interested in finding how much time it will take to update the marker on the screen, which depends on many screen properties. The first significant value is the refresh rate of the display device. The refresh rate indicates at what rate the monitor can

change images on the screen, and it defines an absolute minimum for the screen refresh delay. The achievable refresh rate depends on the screen technology. For example, LCDs have a higher refresh rate than e-ink displays. In the year 2020, commercial gaming monitors reach refresh rates of up to 360 Hz and E-Ink displays up to 7 Hz; these numbers may seem fast enough compared to regular camera refresh rates. However, the refresh rate number of a monitor can be misleading. The actual time required to change a single pixel's brightness on a monitor is called reaction time, and for gaming monitors, this delay is around 1 ms, which is faster than the refresh rate. However, to change a pixel, the information has to be rendered and transmitted to the monitor through a cable using some protocol (e.g., High-Definition Multimedia Interface (HDMI)); once transmitted, most monitors do some preprocessing to the input image before displaying it line by line at the refresh rate of the monitor. The amount of time that takes from the moment a new frame is sent at the monitor's input and the moment it is displayed is called the input lag, and it is generally much higher than the monitor's refresh rate. For competitive gaming monitors, the input lag can be around 5 ms and for general purpose monitors, greater than 15 ms. Moreover, the input lag does not include the amount of time needed by the processing unit to render the image in the graphic card and send it through the cable, and since there are many variables involved (graphics library, operating system, drivers, input lag, refresh rate), it is necessary to measure the complete path of latency.

The best way to measure the lag from generation to display, from now on, "display lag", is to create a program that changes the screen on user input, and then use a high framerate camera to record the screen and the user input (e.g., mouse movement) simultaneously. On the recorded video, one can count the number of frames between the mouse movement and the associated change on the screen, and then use the camera framerate to calculate the amount of time; this is assuming that we use a mouse with low and known input lag. Another way to measure the display lag is to do it automatically by using a high frame rate camera connected to the same computer that generates the image. Since the computer will know precisely when it generated the frame and the instant of time the camera took the images, it can automatically perform the calculation. As a reference, in our tests, we used a 144 Hz gaming monitor model LG 27GL850 with an input lag of 4.7 ms and a display lag of 12.5 ms at native resolution.

Image capture delay: This delay is measured from the moment that a signal is sent to the camera to capture a new frame (in trigger mode)

until the moment that the image data is in the detector device's memory. The image capture delay will be comprised of 3 parts, the amount of time for the camera firmware to receive and process the trigger signal, the duration the shutter remains open (exposure), and finally, the amount of time required to transmit the image to the computer that does the detection. The shutter mainly drives the image delay since it is the one that takes the longest. A typical value for the image capture delay in indoor conditions is around 5 to 15 ms.

Detection delay: There are mainly two tasks that have to be performed by the detector. The first task is the processing required to detect the marker's main features in the image, which includes the position of the control points in image coordinates and a general evaluation of image quality. The second task consists of calculating the pose ${}^{c}\mathbf{T}_{m}$ using $\boldsymbol{a}$ and the detected control points. Both tasks combined can take from 5 to 20 ms depending on how well optimized the image processing algorithms are and how complicated is the non-linear optimization for the pose estimation.

Including all the above delays, the synchronized control loop can last from 25 to 55 ms, this translates into a loop frequency range from 18 to 40 Hz. The frequency range of a synchronized control loop may be too slow for some cameras since it would not take advantage of the full-frame rate of a fast camera (e.g., 60 fps). The problem is that the control loop works sequentially, but we can parallelize some of the steps. Lets assume for simplicity that the delays are: communication 1 ms, screen refresh 20 ms, image capture 10 ms, and detection 10 ms; in this scenario, the sequential control loop will be as shown in Fig. 6.9. However, this can be optimized if we execute some of the stages in parallel. In Fig. 6.10, we present a concurrent execution for the same delays; this exploits the fact that the actual change of the pixel happens at the end of the 20 ms screen refresh delay, and this change only last 1 ms (reaction time). Notice that we can take two consecutive images per displayed marker, and only the first loop has a latency; afterward, we obtain the right pose at the camera's maximum speed.

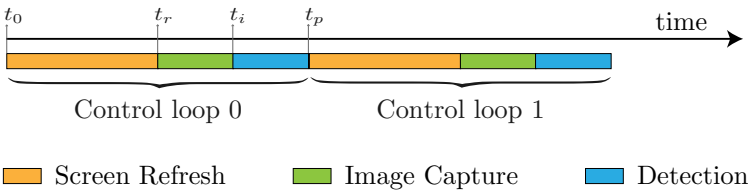


Figure 6.9: Example of a sequential control loop for the dynamic marker where each one of the stages is executed in sequence. The control loop will take more time than the camera refresh rate, which is not ideal although easier to implement.

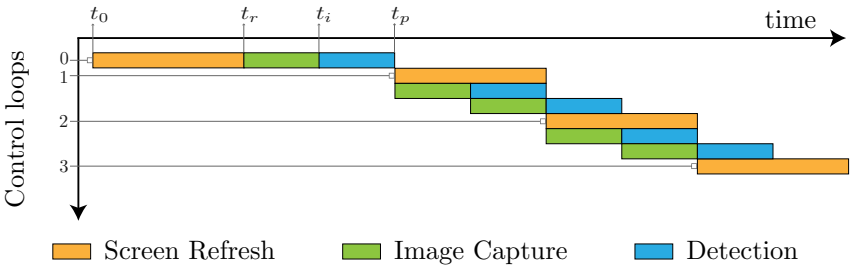


Figure 6.10: Example of a parallelized control loop for the dynamic marker. The stages are overlapped; hence, the output of the estimated pose is at the same rate as the camera.

6.3 The Discrete Dynamic Marker

The first marker design is a simple approach. We replace printed paper by a screen, but we still use the same fiducials available in the State of the Art. The focus of this design is simplicity but also increasing the range of the detection and the accuracy in short distances compared to static fiducials. We propose discrete changes of scale and marker family selection based only in camera-to-marker distance, we select the best markers for close range and the best ones for far range, and we scale them accordingly based on distance. We name this design as the Discrete Dynamic Marker (DDYMA).

To validate our proposal, we present in this section, the design of a DDYMA controller for PBVS applications. We have chosen the landing of an autonomous quadcopter as a test-bed since it presents the typical problems related to PBVS on a dynamic platform using fiducial markers. For the design of the dynamic marker controller, it is necessary to characterize the perception problem first with static fiducial markers to understand which dynamic changes are required.

6.3.1 DDYMA design: fiducial markers for landing

The major problem for fiducial marker quadcopter landing is the detection range of the marker. It is preferred to have long detection distance while having full pose information. Additionally, marker identification is required. The final centimeters of the landing are also critical because the controller has to perform slight adjustments for proper alignment, which requires reasonable pose estimates; this means that the marker should be observable at a short-range. This set of requirements presents a problem when choosing a marker, and usually, a trade-off is done; thus, the design of the controller will be based on two requirements: first, the ability to display markers from different marker families, and second, the possibility to scale the marker based on the camera-to-marker distance.

Fiducial markers are a research area, in general, more connected to the augmented reality field, and there is surprisingly a low amount of marker comparisons focusing on pose detection accuracy at different distances and angles. Most of the fiducial marker comparisons focus on the detection's performance during occlusion or the marker family identification capabilities. However, it is particularly interesting for the robotics community to know precisely the range and orientations at which a given marker provides a reliable pose. Due to this, the first step for the DDYMA design is

to study the ranges and characteristics of the maker families.

From the results of our state of the art study on fiducial markers, we propose to use two different marker families that can complement each other. The first one, a high complexity fiducial that can provide full Six Degrees Of Freedom (6DOF) pose estimates with good close-range accuracy and provides robust identification capabilities. The second one, a marker that has a more simple shape with a focus on long range detection. From the planar fiducial marker analysis of Section 3.2, we know that square fiducials such as Aruco [38] fit the requirements for the first desired marker family; meanwhile, simple circular fiducials such as Whycon [60] fulfill the requirements for the second marker family.

From the high complexity marker families, Aruco is a convenient choice since it is now part of the well known open-source computer vision library OpenCV, and there are several implementations available in Robot Operating System (ROS). Similarly, the low complexity marker Whycon has been successfully used in several robotic applications due to its accuracy, simplicity, and low processing time, and also has a convenient ROS implementation. However, Whycon is not capable of providing yaw angle measurements. From the Aruco and Whycon papers plus our tests, we arrived at the following conclusions: The accuracy of Aruco dramatically decreases with the distance to the camera, while Whycon is more stable at all distances. Whycon is a good alternative for large distances (when only position estimation is needed), while Aruco is only good at small marker-to-camera distances. In our tests, we found that the maximum detection distance of Whycon is at least three times larger than Aruco's when using the same display area and the same camera. Nonetheless, yaw angle estimation is still a vital requirement for quadcopter landing because it is necessary to align the heading of the quadcopter to the landing platform. Regarding the rotation estimation results, we observed that Whycon, in some cases, calculates completely wrong rotation estimates and, in other cases, correct values but with the wrong sign; in contrast, Aruco performs better for rotation estimates. We also made comparisons between the markers printed on paper and the markers displayed on the screen, and we did not find any significant difference. For both marker families, the accuracy is better when the marker is the center of the camera image; this is because the effect of lens distortion is more pronounced on the borders (even in a calibrated camera). Base on all of the above, we defined that the DDYMA will use Whycon as a long-range position only marker and Aruco as a close-range full pose marker. The limits of the transition between Aruco and Whycon have to be defined depending on the camera

that is going to be used and the size of the display screen.

6.3.2 DDYMA controller definition

The marker controller should define the optimal marker for the current pose of the camera. The controlled marker features $\boldsymbol{a}(t)$ are the scale of the marker and the marker family (Aruco or Whycon). The primary measured feature $\boldsymbol{s}$ is the marker-to-camera distance mcd obtained from camera image measurements. The mcd is the euclidean distance from the origin of the marker coordinate frame $\boldsymbol{O}_m$ to the origin of the camera coordinate frame $\boldsymbol{O}_c$

We need to define a rule for automatic marker scaling based on mcd. We propose a scaling rule based on the camera field of view ϕ, see Fig. 6.11. The desired marker size ms at a given marker-to-camera distance mcd has to fit inside a reduced field of view of the camera ϕ_r; the limits of this reduced field of view define how much the camera can change its orientation before the marker is out of the image for a given mcd and ms. Having a marker smaller than what fits inside ϕ_r allows for some camera movement freedom without losing the detection. We define the marker size by a function with the following form:

$$ms = f\left(\phi, mcd, \alpha\right), \tag{6.7}$$

were ϕ can be calculated from the camera intrinsic parameters, mcd is the marker-to-camera distance and α is a scaling factor.

The scale factor α defines the value of ϕ_r as a fraction of the maximum field of view: $\phi_r = \alpha * \phi$, which means that α is defined for the range $0 < \alpha \leq 1$, when $\alpha = 1$ then $\phi_r = \phi$. By using a simple geometrical calculation, we define the following equation for the optimal marker size ms for a given mcd:

$$ms = 2 * mcd * tan\left(\alpha * \phi\right). \tag{6.8}$$

The optimal value for α depends on both the minimum amount of pixels required for the marker identification algorithm and how much can the camera move before losing the marker from the field of view. Choosing a value for α can also be seen as choosing an angle of camera freedom θ:

$$\theta = \phi/2 - atan(ms/(2 * mcd)), \tag{6.9}$$

this angle indicates how much the relative orientation of the camera can change before losing the marker.

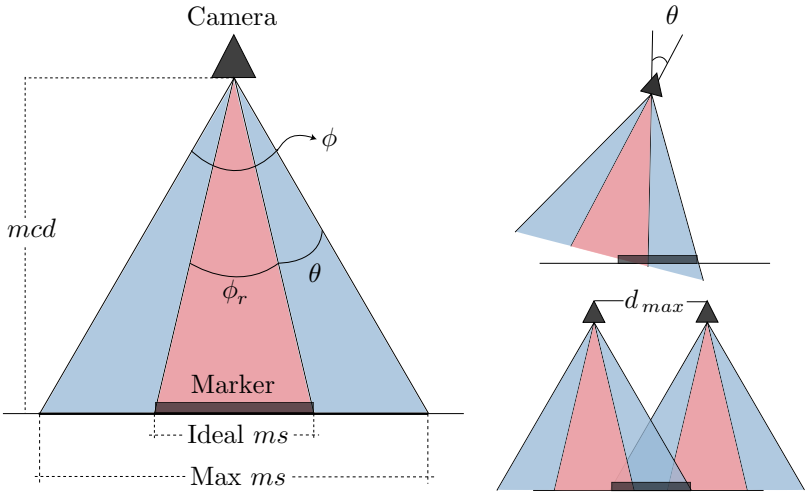


Figure 6.11: An ideal marker for landing should have a size that allows a successful detection while leaving some extra room in the field of view for both lateral and rotational movements of the camera.

The angle α is an input defined by the user, and in each loop, the controlled variable ms will be calculated by the controller by applying eq. 6.8 using the measured mcd and α.

Once we defined the marker's automatic scaling, the only part missing is the control rule for the discrete change between different marker families. We define a switch distance sd, where we decide the marker family to use. When $mcd > sd$, the marker Whycon is selected, and if $mcd <= sd$, Aruco is selected. The value of sd is obtained experimentally by measuring the maximum mcd at which Aruco is still reliably detected. We also exploit another feature of Aruco, the so-called board of markers, which is a grid of Aruco markers on the same surface, each one with a different ID but sharing a common coordinate origin. When the size of the Aruco marker is smaller than the available screen space, the rest of the screen will be filled with additional Aruco markers from a predefined Aruco board, all of them will form part of the same coordinate system, increasing the accuracy on close ranges. At the start of the system, the initial marker will be Whycon to assure detection.

6.3.3 Autonomous quadcopter landing using a DDYMA

In this section, the designed DDYMA will be applied to autonomous quadcopter landing to test the key concepts of a dynamic marker. Our experimental setup for quadcopter landing consists of an AR.Drone Parrot 2.0 quadcopter with a custom wireless camera and landing legs. All the image processing and control is done in a ground station that sends the commands back to the quadcopter through Wifi using the ROS ardrone autonomy package.

We implemented an observer and a predictor module to cope with the Wifi delay problems of the AR.Drone and a velocity controller based on these predictions. A foldable laptop with a 13.1 inch OLED screen was selected as the Display module of the DDYMA. The code for the dynamic marker was implemented in ROS.

For marker recognition, the *ar_sys* and *whycon_ros* packages were used for Aruco and Whycon detection, respectively, with some modifications to the *ar_sys* package for dynamic marker reconfiguration. On top of this setup, we have a simple PBVS controller based on the velocity controller that we developed for the AR.Drone. For this camera/display configuration the maximum size of the marker is 15 cm and the switch distance was defined as $sd = 1.2$ m, for distances greater than 1.2 m we use Whycon, and for the rest, Aruco.

The quadcopter was flown manually to a height greater than 2 m to a position where the dynamic marker was in the field of view (with Whycon displayed on the screen), see Fig. 6.12. The PBVS was activated to track the marker at the height of 2.5 m, and finally, the landing signal was sent. The quadcopter then descended at a constant speed until the final landing was performed.

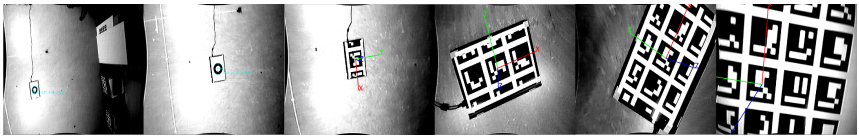


Figure 6.12: Camera frames during the landing procedure. Notice the change from Whycon to Aruco in the third frame and the start of the yaw rotation correction. In the frames 4, 5 and 6 it is possible to see the dynamic change of the scale.

The position of the quadcopter during landing can be seen in Fig. 6.13. Notice how the yaw angle of the quadcopter is corrected as soon as the dynamic marker changes into Aruco at $t = 14\,\text{s}$. The landing was performed smoothly with a final position error of 3.5 cm from the center of the marker. The subsequent frames of Fig. 6.12 show how the display changes according to dynamic marker design, first from Whycon to Aruco at *sd* and then gradually reducing Aruco scale and filling empty spaces with the Aruco board of markers. Extensive testing was performed with this setup with more than 50 successful landings, with an average error of $4,8\ cm$. In comparison, if a static marker is used, either the detection range is limited, so landing from the same height is impossible when using Aruco, or it is impossible to align the quadcopter with the landing platform when using Whycon; this proves the advantages of a DDYMA for visual servoing based landing.

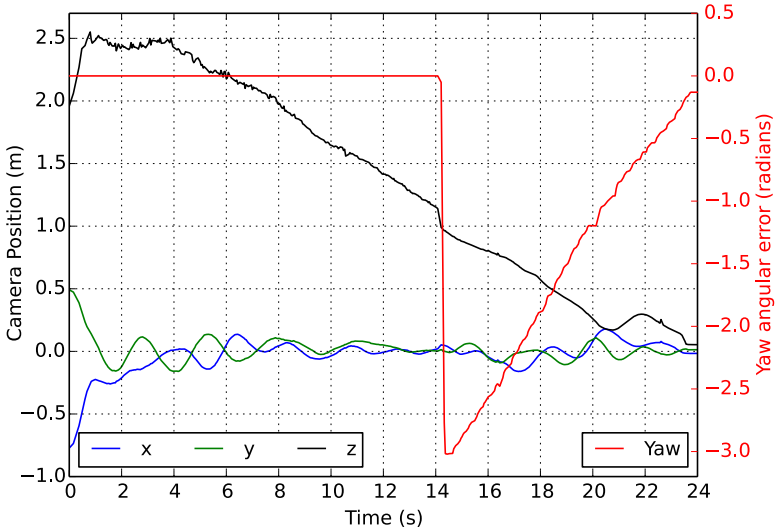


Figure 6.13: A successful landing using a dynamic marker. From $t = 0$ to $t = 14$ the displayed marker was Wycon, then Aruco board with a dynamic change of scale. Notice the yaw angle correction as soon as Aruco is detected.

The PBVS and the marker are tightly coupled in a dynamic marker. Both systems are intertwined. If one changes the marker without updating the $\mathbf{a}(t)$ parameter, then it will have a similar effect on the PBVS as changing the reference. For example, if the size of the marker is reduced

by half without updating $\mathbf{a}(t)$, then the pose estimator will calculate a "virtual" height that is twice as high as the real one, and the platform will move down to compensate; this can be used to control the platform unilaterally by only changing the marker, e.g., the heading of the quadcopter may be controlled by rotating the dynamic marker.

6.3.4 Discussion of the DDYMA design

The results of this section show that the DDYMA represents a novel concept on the field of fiducial markers, which can be successfully integrated into PBVS applications such as UAV landing. From the first results obtained, we were able to identify many more interesting problems to study. The coupling between the dynamic marker and the PBVS is an excellent consequence of the design. The dynamics of the marker can be changed instantly in time and thus much faster than the visual servoing control of the mobile robot.

The DDYMA, even though it is simple and easy to implement, does not take full advantage of the dynamism of the screen. The discrete selection from different marker families based on the range is a very basic objective function, and the use of predefined maker families limits the optimization of the shape to only scale changes. It should be possible to obtain additional objective functions that define an optimal marker shape for a given camera-to-marker pose. If we do not restrict the shapes displayed on the screen to some predefined markers, it should be possible to define more creative ways of presenting the control points. The design of the marker could be extended to take advantage of the temporal domain fully, e.g., showing marker codes for identification in a temporal sequence and the number of features and their configuration could be dynamically optimized to improve pose estimation accuracy. Also, a dynamic marker could be used as a visible light communication system to guide the control process, e.g., to trigger or switch between different control tasks. Additionally, a more formal comparison of traditional static fiducials vs. the dynamic marker in terms of accuracy is also necessary. For example, we can use an external camera system (e.g., Optitrack) for different camera poses. We also need to analyze the potential disadvantages of using a screen, such as view angle restrictions on LCD screens due to reflections.

In the next section, we will focus on these open topics with the introduction of a new dynamic marker design that is not based on discrete changes of already available marker families.

6.4 The Fully Dynamic Marker FDYMA

A dynamic marker based on discrete changes of different fiducial families proved useful to increase the range of the detection. Still, it does not take full advantage of the flexibility of a screen compared to static (printed) fiducials. There are several aspects of the fiducial design that can adequately benefit from the dynamism of a screen.

The main characteristics to be optimized are the scale, the marker identification, and how the marker shape changes from frame to frame. With a surface that changes over time, the identification can be based on time instead of on display area, and we should present the identification information only when strictly required, for example, only when the detector finds other similar shapes in the same image. Moreover, the marker should change over time in a more organic way. The shapes, amount of control points, scale, colors, and other marker features, should transform from capture to capture fluidly and not in discrete changes as in the initial approach. Finally, the chosen shapes should focus on improving the performance of pose estimation methods as much as possible. All the above means that in order to have a genuinely dynamic marker, we have to design it from scratch.

In this section, we will describe the design and implementation of a Fully Dynamic Marker (FDYMA), and we will evaluate its performance versus traditional static fiducial markers.

6.4.1 FDYMA Design

The design is based on the fulfillment of the criteria for good fiducials described in section 3.1.3, while considering the particularities of a screen. To describe our design, we will follow step by step the items of the criteria in the same order.

Shape

We use the term shape, in this context, as the total structural configuration of the fiducial and the term features as the essential structural elements used to build the shape. The most common fundamental structural element used in planar fiducial markers for feature representation are circles and corners (as discussed in section 3.2). Other basic features exist, but they are either harder to detect or less invariant to perspective transformations. When designing a new marker, the main question is then: Which

features are better for detection? Circles or corners? The answer depends on both the achievable detection accuracy of the feature and the desired flexibility of the detection process. To define the features that best suit our design, we need to understand the details of their detection process.

For corners, the algorithms used for detection work either by detecting edge intersections or by finding the saddle point using surface fitting of intensity around a corner point [11]. Of the two options, edge intersections are more accurate. The currently preferred pipeline is an initial pass with a fast edge-based corner detection algorithm such as Harris Corner Detector [47], followed by a sub-pixel corner refinement step based on local gradients.

For ellipse detection, there are mainly two options: centroid extraction, and ellipse fitting [75], where the latter is more accurate than the first. The best algorithm for ellipse detection in computation speed and accuracy is arguably the "Direct least square fitting of ellipse" [33].

In terms of complexity and detection time (which becomes critical with many features), the corner detection methods are slightly faster than ellipse fitting methods. However, both are highly optimized nowadays and easily accessible in computer vision libraries (e.g., OpenCV).

The accuracy of corners and circle features is hard to evaluate since it depends on the final problem that wants to be solved. For example, if one is working with these features in synthetic images without perspective projection, circle features are the most accurate [75]. However, in camera calibration or pose estimation problems, images are influenced by both perspective transforms and camera lens distortion. The real position represented by a feature will be affected first by the intrinsic errors of the method used for the detection (detection bias), and then by the perspective and distortion bias. Some of these biases can be corrected or minimized, others not.

The saddle point methods for corner detection are unaffected by perspective and distortion bias (but they have a higher detection bias). Edge-based methods do not suffer from perspective bias since lines project as lines in captured images. However, a distorted line is a curve, and this affects the detection of the corner, producing a distortion bias. Luckily, the subsequent sub-pixel corner refinement step minimizes the effect of the distortion bias since it works in a local region where the effect of distortion is minimal.

Circles, on the other hand, are subject to both perspective and distortion bias. The perspective transform converts circles into ellipses, and the center of the ellipse is not the same as the projected circle's center. The

perspective bias can be corrected by using the center of the projected conic instead [75]. Additionally, the perspective bias is negligible if the diameter of the circle is between 10 and 20 pixels in the camera image.

There is an additional bias, called the bloom effect, that can occur on cameras when capturing images of scenes with varying illumination, such as outdoor scenes [76]. Blooming happens when a high amount of light causes the white areas of the image to bloom out (or bleed) into the surrounding areas; this effect cannot be countered by exposure time alone. Blooming is particularly relevant for screens since full white pixels can leak into full black pixels. Corner features are significantly affected by this problem since the bloom moves the edges inwards, which influences the corner position estimation. The center of circles, on the other hand, are more invariant to this effect because the circle edges are affected relatively equally by the blooming, leaving the center unperturbed.

For camera calibration, the corner edge detection methods with subpixel refinement are the best option due to a combination of lower detection bias plus a minimized distortion bias. When calibrating a camera, the user who performs the calibration also controls the environment's lighting conditions, so the bloom effect is not determinant in this case. As an alternative, it is possible to use small radius circles to minimize the introduced distortion bias while keeping the detection bias low [75].

In the case of pose estimation, which is the main aim of a Dynamic Marker design, circles are the ideal feature due to several reasons. First, the camera intrinsic parameters and the lens distortion coefficients are known beforehand and used to undistort the captured image, so the distortion bias in circles is no longer a problem. Even though both corners and circles with appropriate measures are free of perspective bias, circles are less affected by the blooming effect, and they have lower detection bias than corners.

Now that we discussed the characteristics of the features and found the optimal, we can dwell more in-depth into the fiducial's actual shape. A fiducial marker for pose estimation is not made of a single individual feature. Several features have to be arranged in a geometric configuration that provides enough information to extract the pose. In essence, a shape is built from a group of individual features, and the designed shape should provide a structure that allows a detector to recover distance, rotation, and identification. We can analyze how markers in state of the art are built using the aforementioned basic features and use it as a base for our design. For example, in the case of corner features, the minimal structural shape is a square or rectangle (4-corner features) with a coding inside;

examples of this design are AprilTags, Aruco, and other similar markers presented in section 3.2. For circles, full pose recovering requires at least two circles; a minimal structural shape example of this configuration is the Whycode [70], which uses two concentric circles with a ring of circular coding inside. Fiducials build only from circles, such as Pi-tag and Rune-tag, suffer from increased detection time because many contours in an image can be similar to circle blobs, and they may be present in a wide variety of scales. That is why circular fiducial designs need to introduce other geometric relationships between the circles to separate them from other circular blobs that occur naturally. Squared fiducials, on the other hand, are faster to detect since 4-corner polygon shapes are not that common in nature and easier to detect.

Based on the previous information, the proposed shape for the FDYMA is a mixture of circles and corner features in the shape of a quadrilateral polygon with circles inside, see Fig. 6.14. This marker can be detected fast and efficiently by exploiting the already available pipeline for square markers; besides, the circles inside the quadrilateral will provide extra accuracy, which is desired in pose estimation. We define the distribution of the circle centers inside the quadrilateral as a honeycomb (hexagonal packing), this is based on the results obtained in Chapter 5: the homography estimation, which is critical for planar PnP, is more accurate when the control points are distributed appropriately in space, and one of the more uniform distributions of six control points in space are hexagonal grids. Hexagons are also a convenient configuration for packing circles inside quadrilaterals since the distance of each circle's center to its neighbors is always the same.

Color

White and black color combinations are the most common designs in planar fiducials because of the complexity of working with color. In our case, we will maintain this principle to keep simplicity, with a minor exception for orientation and identification, as we will explain below. On printed fiducials, the most common design is a black filled shape with a white foreground (e.g., a black square with internal black/white squares for code); this is because the designers usually assume that the fiducials will be printed on white paper, so black filled shapes will provide the highest contrast on the white paper surface. In our case, we have the flexibility of a screen with its full range of brightness; however, for the color selection, we have to consider the border of the screen (aka monitor bezel), which

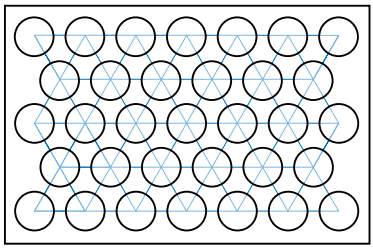


Figure 6.14: The Proposed FDYMA structural shape consist on a set of circles placed in a honeycomb configuration inside a quadrilateral. The blue equilateral triangles are shown only as a reference of the internal hexagonal distribution of the centers. The aspect ratio of the quadrilateral can be changed to suit the needs of the detection.

in most monitors is black, but it can also be white or gray. The marker should use the maximum amount of pixels on the screen; this means that we can use the monitor bezel as the marker border on the biggest marker size. Based on the bezel constraint, we configure the Dyma marker as a white background with black circles when the monitor bezel is black and the inverse when the bezel is white, see Fig. 6.15. When we scale the marker down, we expand the same color of the bezel to the inside of the screen for maximum contrast.

The lack of colors on the basic marker design may seem restrictive when one thinks about the color gamut of modern computer screens, but the design is targeted for detection simplicity, and colors are usually complex to detect accurately. However, the presence of other colors can be helpful if used sparingly, for example, in marker orientation definition and identification.

Orientation

The process of full pose estimation needs non-symmetrical elements to estimate the orientation. Planar marker designs usually introduce these non-symmetrical elements within the internal code that is used for identification. The Dyma design, in contrast, solves the identification and orientation problem with a layered approach based on the screen dynamism; this means that we can define asymmetry only by the modification of a single feature. We propose a simple change in the circle feature closer to the bottom left corner of the quadrilateral, as shown in Fig. 6.16. This

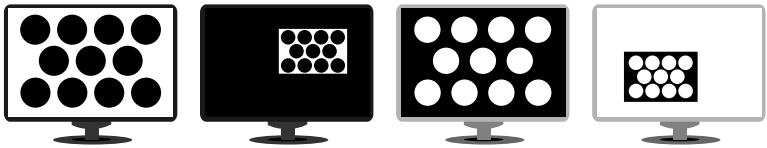


Figure 6.15: Basic black and white color schemes for the FDYMA. a) White quadrilateral with black circles when the screen bezel has a dark color, b) the dark bezel is extended with black pixels when the marker is scaled down, c) black quadrilateral with white circles when the screen bezel has a lighter color, d) the light bezel is extended with white pixels when the marker is scaled down.

circle is going to be denominated as the key circle. There are several options to make this feature different than the others. The first option is to remove the circle completely; this has the advantage of not requiring extra processing; however, it is one less control point for pose estimation. Another alternative is to replace the circle with another basic structural shape, e.g., a triangle, but this requires extra processing. The better option is to introduce color or a shade of gray in the inside of the circle while maintaining proper border contrast to assure contour detection. A colored circle with a border would work, but, depending on the inner color, the contour detector may detect two concentric circles, which increases detection complexity. A solution is to use a gradient from the border to the desired color. We selected this last option for our design; we defined a filled circle with a color gradient as the default option.

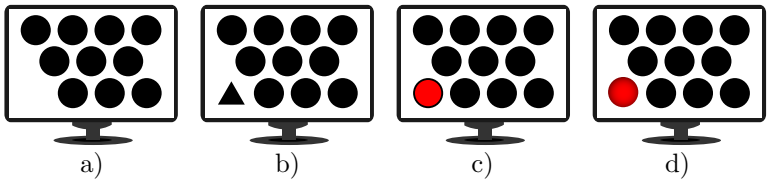


Figure 6.16: Possible alternatives for orientation definition on the FDYMA. The lower left corner is selected as the key feature for asymmetry. a) complete removal of the circle. b) Substitution by another basic form (triangle). c) Colored circle with high contrast border. d) Circle filled with a gradient (default configuration).

Size

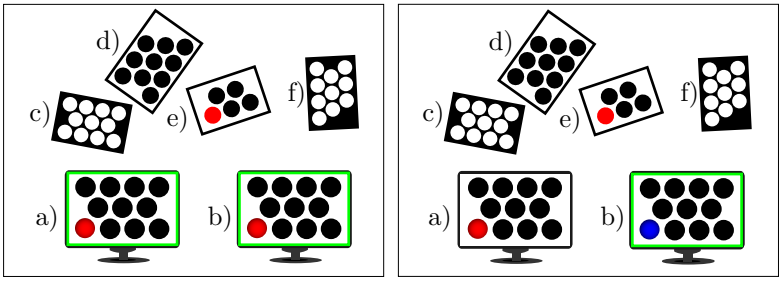
Similarly to the DDYMA, in this design, we can change the scale of the FDYMA fluidly, but now fine-tuning the shape to the particular camera pose. We can select, for example, the size of the quadrilateral and the size of the circles independently. Moreover, we can move the center of the quadrilateral to any place inside the screen coordinates. Scaling the circles also changes the number of circles on the marker (the bigger the circles, the less quantity of them can be fitted inside the quadrilateral). The FDYMA controller is then in charge of changing several positional/scale parameters of the marker simultaneously depending on the camera pose: diameter of the circles, circle separation, amount of circles, quadrilateral size, and quadrilateral position.

Identification

Fiducial identification is, at its essence, a block of information that has to be transmitted to the detector. For static markers, this block of data has to be statically coded inside the fiducial, and this requires a significant portion of the available display area. The amount of data transmitted in the block is directly related to the size of the marker family (the number of markers that have to be identified uniquely). The amount of data transmitted is also increased if error-correcting codes are included.

For marker identification, we are going to use a layered approach. Each layer will provide extra complexity, allowing us to separate the marker from other similar shapes and discern one marker from another. The first and second layers are the quadrilateral border and the inner circles, respectively. Every contour that is not a quadrilateral with circles inside is discarded. The third layer is the number of circles, only the quadrilaterals with the correct number of circles inside at any given moment are selected. A fourth layer is the color of the key circle. These initial four layers already segment most of the environment's shapes with no extra overhead on the detection algorithm, except perhaps another dynamic marker with the same scale. The design allows the definition of an internal code in the circles, if necessary, by alternating black and white circles; however, this is not necessary due to a vital characteristic of the dynamic marker: a feedback loop. In the rare case that another shape has the exact characteristics of a displayed marker at a given time, then the controller can change the displayed marker in the next iteration by either inverting the colors of the circles and background or changing the color of the

key ellipse, see Fig. 6.17. This means that the identification information for complex scenarios (several dynamic markers working together) can be transmitted to the detectors by any method of optical communication via visible light; this is an advantage of the FDYMA since traditional fiducial shapes require additional space in the form of extra features to transmit the identification. With a FDYMA, we exploit the additional display area to present more accurate control points. In essence, we are moving the data transmission complexity from a feature domain to a time domain, thanks to the screen.



Capture n Capture $n+1$

Figure 6.17: FDYMA identification process using feedback-based disambiguation. In frame n there are several shapes that have ellipses inside quadrilaterals, two of them are impossible to separate since they have the exact number of ellipses with the same scale and color. The detector sends this information to the controller which changes the color of the key circle, in the frame $n+1$ the marker is uniquely identified.

Robustness

The FDYMA design optimizes the displayed shape for detection and pose estimation, but the robustness of the detection will depend on additional factors related to the screen. Depending on the screen technology used, there may be problems related to glare and view angles. The typical commercial LCD screens are intended in general for indoor use, and they are designed to emit light optimally when looking entirely straight into the screen; if the view angle is increased, the viewer will perceive certain areas of the screen dimmed; cameras can also perceive this effect. Nonetheless, if angle view is a limiting factor, other screen technologies such as OLED displays can be selected with a wide field of view because each pixel emits

their light. An advantage of screens vs. paper besides dynamism is that they are also suitable for marker detection in low light environments. Another problem is that almost all commercial monitors and TVs will present problems outdoors due to the glare effect on the screen glass, which will impact the fiducials' detection. However, LED grids for outdoor events are already commercially available, and electronic ink displays do not have this problem since they use reflected light. Finally, screens that have a glass in front can introduce an additional source of bias due to light refraction; this effect may be minimal for pose estimation, but it can be meaningful for camera calibration (tenths of a millimeter of deviation)[26], we can correct this bias if we know the screen properties beforehand. The characterization of the bias introduced by refraction and its correction is out of the scope of this dissertation; nonetheless, we assume that the bias introduced by refraction is less meaningful than the improvements provided by a dynamic marker.

6.4.2 Detection

The detection of the designed shape on captured images follows the same initial steps of the Aruco marker detection implemented in OpenCV v4.2 and comprises the following steps:

1) Adaptive threshold. The first step is to binarize the input to get a black and white image for contour extraction. For every pixel, we apply a threshold value. If the pixel value is smaller than the threshold, we set it to 0, otherwise, to a maximum pixel value. If the threshold is predefined, the binarization will have different results depending on the scene lighting. There are methods to find the threshold automatically, such as OTSU's algorithm, but they increase computation time if the input image is large. One solution that keeps computation time low is to use adaptive thresholding; this technique obtains the best threshold for a region surrounding a pixel, and it works best in images that vary in illumination. When using adaptive thresholding, the size of the window to evaluate around a pixel is the main parameter to configure. In order to be flexible to different illumination changes, we binarize the input image three times using a different window size each time, which will produce three different binarized images. The user can configure the size of the three windows in runtime.

2) Contour extraction and classification. For each of the three binary images, we extract the contours (the edges of the shapes). We

filter contours by perimeter using minimum and maximum thresholds that depend on image size; in a Dynamic Marker, these thresholds can be very narrow since we control the scale. Now we approximate the detected contours into polygons and classify them depending on the number of corners, and we then proceed to filter out all the polygons with less than four corners (unless we are searching for triangles as well). We assign 4-corner polygons to a list of quadrilaterals candidates, and polygons with more than four corners to a list of potential ellipse candidates.

3) Remove contour nesting. Since we are processing three different binary images simultaneously, for each real edge we are going to obtain a set of three different nested contours. For example, a squared shape will produce three quadrilateral candidates with the same center of mass, but with different sizes. However, only one of those candidates will be closer to the actual edge. The color of the shape related to the background matters; if in the binarized image the square is black and the background white, then we select the smallest quadrilateral candidate; on the other hand, if the colors are inverted, we select the biggest. The same applies to the ellipse contours.

4) Quadrilaterals with the correct number of candidate ellipses. For each quadrilateral, we check how many ellipse candidates are inside of it. A quadrilateral candidate needs to have at least the amount of ellipses that we are displaying, but it can have more due to contour artifacts; we will remove these artifacts in the next step.

5) Ellipse refinement. The image's adaptive thresholding allowed us to find contour edges without having any information about the illumination of the image, but the window sizes are predefined, which means that they are not optimal. Now that we want to accurately detect the edges of the ellipse and remove potential contour artifacts, we can apply the OTSU'S method in the image regions where we know the potential ellipses are, which reduces computation time significantly without sacrificing accuracy. Now we check the ellipse candidates in each quadrilateral, we perform OTSU's thresholding in the region of interest of each ellipse candidate and calculate its contour again. Since ellipse fitting is a computing-intensive task, we first filter the candidates by performing a fast circularity check, then we fit the ellipse using Fitzgibon's algorithm [33]. We remove all the bad ellipse candidates and check if the quadrilateral has the correct number of remaining ellipses; if not, we

remove the quadrilateral candidate. Since we know the area ratio of the quadrilateral to the circles inside of it, we can further remove candidates that have the wrong ratio because it is perspective invariant.

6) Identification. We need to find the key circle in the quadrilateral, knowing that it has a different color than the rest. The key circle is always closest to the lower-left corner of the displayed quadrilateral. We find those quadrilateral corners which have the closest ellipses, and we select these ellipses as key circle candidates. We test the color of these ellipses in two ways. First, we calculate the mean color of each ellipse and find the ellipse that has the most different mean color, and second, we compare this ellipse color to the displayed color. If it passes both tests, then the key ellipse is marked. If there are two or more marker candidates with the same amount of ellipses, the same quadrilateral to ellipse area ratio, and the same desired key circle color, we send the position of all of them to the controller so the marker can be adapted in the next iteration. When the marker is changed through the controller and uniquely identified, the correct pose for this iteration can be also be calculated.

6.4.3 Controller

The controller of the Discrete Dynamic Marker (DDYMA) used only the relative distance between the marker and the camera to control the scale and select the marker family. Now, the FDYMA controller will control several parameters of the marker simultaneously depending on the requirements of the detection: background/foreground colors (bgc, fgc), key circle color (kc), circle diameter (d), circle separation (cs), screen separation (ss), quadrilateral scale (sx, sy) and quadrilateral position in screen coordinates (x, y). All these parameters are included in the control signal $\boldsymbol{a}(t)$, which is sent to the Display and Detection modules. The positional and size parameters are defined in meters. A summary of the controlled parameters is presented in Fig. 6.18.

Each of the parameters can be manually configured by the user or controlled automatically. To make the automatic marker changes from pose to pose more organic, we can define some relative parameters. We define the parameters ss and cs as a percentage of the circle diameter cd, and the scale of the quadrilateral in X and Y direction as a fraction of the screen size: $0 \leq sx, sy \leq 1$. For example, when $sx = 1$ and $sy = 1$, the marker is using the whole screen. Based on these parameters, we then calculate the maximum amount of circles that fit on the marker

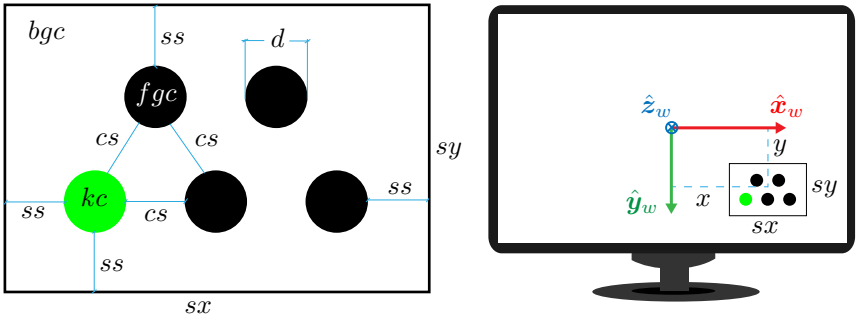


Figure 6.18: Main controlled parameters on a FDYMA. Left side: colors, position and size of the circles inside the quadrilateral. Right side: scale and position of the quadrilateral in the display device. All the parameters are defined in meters. The center of the right-handed marker coordinate system is in the center of the display device, the positive direction of the X-axis to the right, Y-axis to the bottom and Z-axis going inside the display plane.

boundaries and their hexagonal grid placement.

Now we need to control $\boldsymbol{a}(t)$ by using the features detected from previous captured camera frames $\boldsymbol{m}(t)$. We control the scale of the quadrilateral in a similar way to the scale of the DDYMA, as explained in section 6.3.2, but now we also control the position of the quadrilateral's center (x, y) so it stays as close as possible to the middle of the camera image. The colors are defined by default depending on the monitor color, and they only change in case of an identification ambiguity. What is left to control is the circle distribution inside the quadrilateral. Given that we defined the positional parameters of the circles as relative to the circle diameter, then the only parameter that we need to control is the circle diameter.

To control the circle diameter, we need to define first what is an optimal diameter. The main question is: Which size of a circle allows more precise detection of its center? When we talk about the size, we refer to the circle's size in camera image pixels. From [75] we know that small circles (less than 10 pixels in diameter) are the best in order to avoid the effect of perspective and distortion bias. Another argument in favor of small circles is that they occupy less space allowing us to display more control points. However, in practice, the distortion bias is not relevant to us because we are working with undistorted images, the perspective bias can be corrected in the pose estimation step, and small circles have more detection bias in

the presence of noise.

In terms of detection bias, which is the most relevant for our case, we studied the effect of the size of the circle on the bias in [113] and [53] and found that the bigger the circle, the less the detection bias when using and edged detection method like Fitzgibon's algorithm. In additional tests, we found that a good trade-off for all these different constraints is a circle diameter that ranges from 20 to 50 pixels, and the optimal value in our tests was a diameter of 25 pixels (approximately a circle area of 500 pixels).

Our control rule for the diameter is then to maintain an average diameter of 25 pixels in camera image coordinates for all poses. We need to measure this diameter in camera images; however, the circles will be transformed into ellipses due to perspective distortion. In order to get a unique value of the diameter, we calculate first the average area of all the ellipses in the detected marker, and then we apply the equation of the area of a circle to obtain an average diameter; this diameter is an approximation that represents a circle with the same area as the average ellipses. Nonetheless, it is a sufficiently good approximation for our purposes.

The control of the marker diameter in marker coordinates ${}^{m}d$ (in meters), based on the diameter measured in camera image pixels ${}^{p}d$, can be implemented in two ways depending on the main target application of the dynamic marker.

The first way is a direct computation of ${}^{m}d$ based on the estimation of the pose ${}^{c}\mathbf{T}_{m}$ from a previous measurement and the desired diameter that we want in camera image pixels (e.g. ${}^{p}d = 25$). Using the transform between the marker and the camera ${}^{c}\mathbf{T}_{m}$, and the camera intrinsics $\mathbf{K}$, we calculate the homography ${}^{m}\mathbf{H}_{p}$ between the camera image plane and the marker plane and finally apply the following equation to obtain ${}^{m}d$:

$$\begin{bmatrix} {}^{m}d \\ 0 \\ 1 \end{bmatrix} = {}^{m}\mathbf{H}_{p} \begin{bmatrix} {}^{p}d \\ 0 \\ 1 \end{bmatrix}. \tag{6.10}$$

The second way does not require the pose; this is useful for camera calibration applications or when the camera intrinsics are not available. Remember that the process of camera calibration requires the collection of control points displayed on a plane for different poses of the camera. We found that to control the circle diameter without a pose, it is enough to use a simple proportional controller with the following form:

$$u(t) = Kp\ {}^{p}e(t). \tag{6.11}$$

The error is defined as ${}^{p}e(t) = {}^{p}\hat{d} - {}^{p}d$, where ${}^{p}\hat{d}$ is the reference diameter in camera pixel coordinates and ${}^{p}d$ is the measured diameter. The controlled variable $u(t)$ is the new diameter in marker coordinates ${}^{m}d$. The gain Kp has to be manually tuned since it depends on the camera. With a proper Kp, we found convergence to the desired diameter in 3 to 4 frames, and afterward, it will compensate appropriately for the camera movement in 1 or 2 frames; this is because camera pose has smooth changes over time. However, if the camera intrinsics are available, the first control method is preferable since it gives a direct solution and does not require tuning.

6.4.4 Pose estimation

From the detected control points of the marker, which comprise the four corner points and the inner circle centers, we can calculate a pose using any of the methods for Planar Pose Estimation (PPE) introduced in section 2.3.3. However, since we want the highest possible accuracy, we are going to base our pipeline on the golden standard explained in section 2.3.5. To recap, the golden standard for pose estimation consists in using a direct one-shot solution (that may be inaccurate but fast) to get an initial pose estimate, and then we proceed to optimize this pose with a non-linear optimization based on the minimization of the reprojection error.

Of all the modern methods for PPE, perhaps the most stable and with better results is the Infinite Plane Pose Estimation (IPPE) method [21]. We use IPPE for a first fast initial estimate using all the detected control points (both from circles and corners features). For the non-linear optimization, we could use the Levenberg–Marquardt (LM) algorithm to minimize the reprojection error of all the control points. However, this will not give the best pose estimate because we would be merging features with different properties (circle centers and corners) and treating them equally. We have to remember that the detection of circle centers is subject to perspective bias, and we need to consider this in the optimization.

We will follow the same premises of [99], where the authors applied perspective bias correction to calculate homographies for camera calibration, but we apply the process to pose estimation instead.

We start by highlighting the relationship between a circle and the projected ellipse, knowing that both are conics. A conic is a curve described

by a second-degree equation in the plane [49]. Given the 2D cartesian coordinates (x, y) we can represent conics by

$$s_1x^2 + s_2xy + s_3y^2 + s_4x + s_5y + s_6 = 0\,, \tag{6.12}$$

where $s_1, s_2 ... s_6$ are the conic coefficients. We can convert (6.12) to homogeneous coordinates $\boldsymbol{x} = [x_1, x_2, x_3]$ by doing the following replacements $x \mapsto x_1/x_3, y \mapsto x_2/x_3$:

$$s_1x_1^2 + s_2x_1x_2 + s_3x_2^2 + s_4x_1x_3 + s_5x_2x_3 + s_6x_3^2 = 0\,, \tag{6.13}$$

and convert it to quadratic matrix form

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} s_1 & s_2/2 & s_4/2 \\ s_2/2 & s_3 & s_5/2 \\ s_4/2 & s_5/2 & s_6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0\,, \tag{6.14}$$

where $\mathbf{S}$ is the conic matrix

$$\mathbf{S} = \begin{bmatrix} s_1 & s_2/2 & s_4/2 \\ s_2/2 & s_3 & s_5/2 \\ s_4/2 & s_5/2 & s_6 \end{bmatrix}. \tag{6.15}$$

We can calculate the center of symmetry of a conic matrix from its 2×2 top block and the first two elements of its last column:

$$C(\mathbf{S}) = -\begin{bmatrix} s_1 & s_2/2 \\ s_2/2 & s_3 \end{bmatrix}^{-1} \begin{bmatrix} s_4/2 \\ s_5/2 \end{bmatrix}. \tag{6.16}$$

Now, if a point in the marker plane m maps to another in the camera normalized image plane η through a homography ${}^{\eta}\boldsymbol{x} = \mathbf{H}\,{}^{m}\boldsymbol{x}$, then a conic in the marker plane will map to its correspondent conic in the camera normalized image plane as follows:

$${}^{\eta}\mathbf{S} = \mathbf{H}^{-\top}({}^{m}\mathbf{S})\mathbf{H}^{-1}. \tag{6.17}$$

The center of the transformed ellipse $C({}^{\eta}\mathbf{S})$ is not the same as the center of the marker circle transformed by the homography $\mathbf{H}C({}^{m}\mathbf{S})$, see

Fig. 6.19. When we detect the center of the captured ellipse using the ellipse detector, we are obtaining an estimate of the center $\tilde{C}({}^{\eta}\mathbf{S})$, hence, we cannot use $\mathbf{H}C({}^{m}\mathbf{S})$ to calculate the reprojection error .

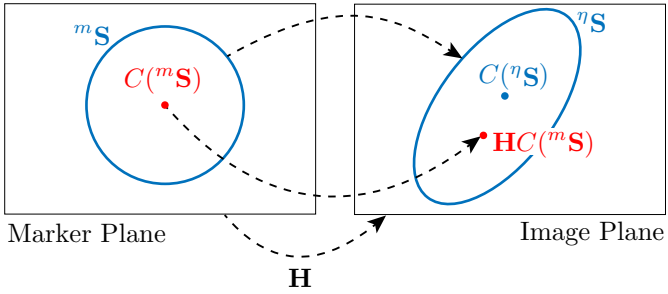


Figure 6.19: The homography transformation of the circle's center $\mathbf{H}C({}^{m}\mathbf{S})$ is not the same as the center of the transformed conic $C({}^{\eta}\mathbf{S})$. This is important to avoid perspective bias in circle based pose estimation.

In order to correctly calculate the reprojection error for each circle i in the marker, we reproject the circle conic with a homography estimation $\tilde{\mathbf{H}}$ that we obtained using the initial estimate of the marker-to-camera transformation ${}^{c}\tilde{\mathbf{T}}_{m}$ and the known camera matrix $\mathbf{K}$. Finally, we calculate the center of the projected conic using the estimated $\tilde{\mathbf{H}}$ and compare it to the ellipse center extracted from the image to get the circle center reprojection error:

$$\boldsymbol{e}_{\circ}^{i} = d\left(C(\tilde{\mathbf{H}}^{-\top}({}^{m}\mathbf{S}^{i})\tilde{\mathbf{H}}^{-1}),\, \tilde{C}({}^{\eta}\mathbf{S}^{i})\right)^{2} \tag{6.18}$$

$$\boldsymbol{e}_{\circ}^{i} = d\left(C({}^{\eta}\tilde{\mathbf{S}}^{i}),\, \tilde{C}({}^{\eta}\mathbf{S}^{i})\right)^{2}. \tag{6.19}$$

Now, it is only missing the formal definition of the reprojection of the corners of the marker's rectangular border. For each corner j, in marker coordinates ${}^{m}\boldsymbol{x}_{j}$, we apply the estimated transform from marker-to-camera coordinates ${}^{c}\tilde{\mathbf{T}}_{m}$, followed by a projection to normalized image coordinates. Finally, we calculate the geometric distance of the resulting point to its corresponding detected corner ${}^{\eta}\tilde{\boldsymbol{x}}_{j}$, obtaining the square corners reprojection error:

$$\boldsymbol{e}_{\square}^{j} = d\left({}^{\eta}\tilde{\boldsymbol{x}}_{j},\, \boldsymbol{\Pi}_{0}\, {}^{c}\tilde{\mathbf{T}}_{m}\, {}^{m}\boldsymbol{x}_{j}\right)^{2}, \tag{6.20}$$

With these two errors we can now solve the following least squares optimization to obtain an optimal estimation of the pose ${}^{c}\hat{\mathbf{T}}_m$

$$ {}^{c}\hat{\mathbf{T}}_m = \text{argmin}_{{}^{c}\tilde{\mathbf{T}}_m} \sum_j \boldsymbol{e}_{\square}^{j} + \sum_i \boldsymbol{e}_{\circ}^{i} \,. \tag{6.21} $$

6.4.5 FDYMA experiments

In this section, we will demonstrate the performance of the FDYMA in comparison to traditional planar fiducials. We are interested in testing the FDYMA in the following main metrics: the range, detection rate, and accuracy.

For all the following experiments we used a Flir Blackfly camera model BFLY-U3-13S2C-CS with a resolution of 1288×964 px.

We start by demonstrating how changes on some reference parameters reflect on the displayed marker through the controller, then we will test the detection range of the marker compared to Aruco markers, and finally, we are going to study the accuracy of the FDYMA for different positions and the accuracy in the presence of noise.

Area and circle separation

The first parameter to test is the reference of area for the circles, which defines the diameter reference ${}^{p}d$. We placed the monitor at a fixed distance of 1.5 m from the camera. We started with an reference of area that generated the biggest circle that could fit in the marker (including separation), then we proceeded to change the reference on steps until it was so small that it could not be detected anymore. The results can be seen in Fig 6.20. At an area reference of $100\,\text{px}^2$, the marker was no longer detected. Notice how the amount and position of the circles change for different area references; these changes are automatic and controlled by the rules defined in previous sections.

Another essential parameter is the separation between circles cs, which we express in terms of percentage of the circle diameter. For the same camera-to-marker distance of the previous experiment, we left the area reference fixed at $2000\,\text{px}^2$ and varied the cs reference parameter instead. The results are shown in Fig 6.21. We can observe how the separation parameter changes the position and number of circles in the marker, but the circle diameter remains constant. Both parameters can be controlled

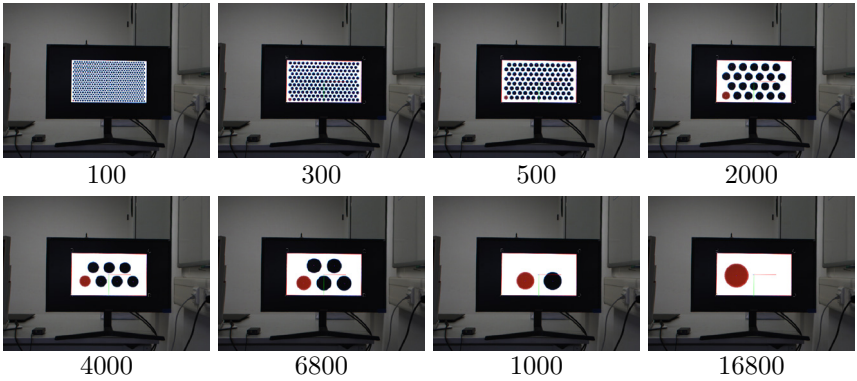


Figure 6.20: Shape of the controlled FDYMA for different circle area reference values in camera image px^2.

simultaneously either manually or automatically.

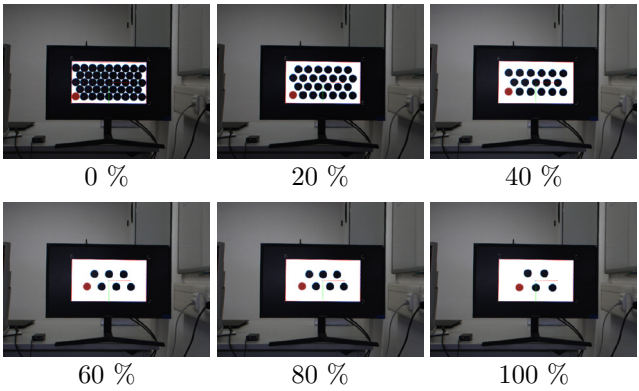


Figure 6.21: Shape of the controlled FDYMA for different percentage of separation between circles (cs) and between circle and screen border (ss).

Detection range

One of the dynamic marker design's main goals is to increase the range of detection; in this regard, we will compare the FDYMA to state-of-

the-art fiducial markers planar. We selected the Aruco marker as the planar fiducial marker family since it is arguably the standard squared marker with a robust codebase already integrated into OpenCV, which is, in turn, the de facto computer vision library. Of course, there are some differences in accuracy and detection rate between Aruco and some similar competitors like AprilTag, but they are relatively small and do not justify comparing the FDYMA to all the other marker families.

To test the range, we placed the monitor at different distances, from very close to the maximum distance at which the marker was still detectable, while measuring the distance with a laser. Since the range of distances to measure is very large (from 0 m to more than 30 m), the camera focus had to be adjusted manually during the testing.

For Aruco, we selected two different sizes: small with 1 cm side length and big with 25 cm side length. The size of big Aruco was the maximum that could fit in the screen while leaving enough space for a border of white pixels large enough for detection. The small Aruco size works well in close distances, and the big one is better for far away; there is an overlap of distances between the two of them. The FDYMA was configured for automatic control with a circle area reference of 3000 px^2 for distances less than 10 cm and an area reference of 1000 px^2 for the rest, the circle and border separation was fixed at 30 %. The FDYMA, in contrast to Aruco, can use the full display since its quadrilateral shape can be adapted to the aspect ratio of the monitor, providing higher flexibility.

For each distance, we placed the camera pointing to the center of the screen, and then we displayed the three selected markers in the monitor without moving the camera. For each displayed marker, we captured 100 images and calculated the rate of detection (a value of 1 means 100 %). For distances less than 3 m, the marker and monitor were in an office space with almost ideal lighting conditions, and for distances greater than 3 m, they were placed on a hallway with more challenging light conditions. The results of the range test are displayed in Fig. 6.22 in the form of a plot of detection rate vs. distance for each marker, and in Fig 6.23 we show some of the images captured for each marker at particular distances.

The valid range (100 % detection rate) for the small Aruco was 0.05 m to 0.4 m, for the big Aruco 0.60 m to 14 m, and for FDYMA 0.05 m to 25 m. Not even two sizes of Aruco can cover the whole range of the FDYMA, the new fractal Aruco or AprilTag could cover the range from 0.05 m to 14 m due to the use of nested markers, but they will not increase the maximum distance; this is because the inside of the marker has to be used for the identification code and the nesting, which increases marker complexity. At

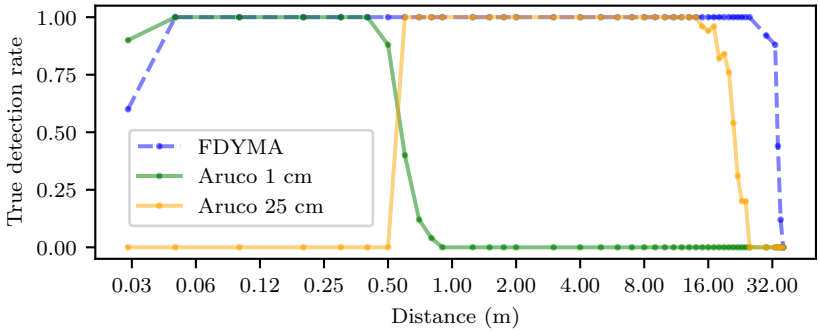


Figure 6.22: True detection rate vs distance of the FDYMA and two sizes of Aruco markers.

large distances, the marker in the captured image does not cover enough image pixels necessary to discern the individual features of a complex marker. In contrast, the FDYMA, in its minimum shape, is a rectangle with a single circle inside (see Fig. 6.20); this minimum amount of features (4 corners and one circle center) is detectable at longer distances than any more complex marker.

When the camera is too close to the screen, around 3 cm, the pixel pitch of the screen starts to matter. At close range, it is possible to discern on the camera image the individual pixels of the screen (see Fig. 6.23); this complicates contour detection if the displayed shape is not aligned with the screen pixels. The effect is more pronounced in circles because there will be a low amount of pixels available to represent them and the circles start to look like squares for the contour detection. In this scenario, squared fiducial markers have an advantage; the solution for the FDYMA is to use retina displays instead, which have a high pixel density.

Positional accuracy

Now we are interested in studying how accurate is the newly designed fiducial when the pose of the camera changes. It is known that fiducials present a lot of variation on the accuracy of the pose estimation depending on the true pose. In the case of squared planar fiducials, fronto-parallel camera-to-marker positions give the worst results since small errors in

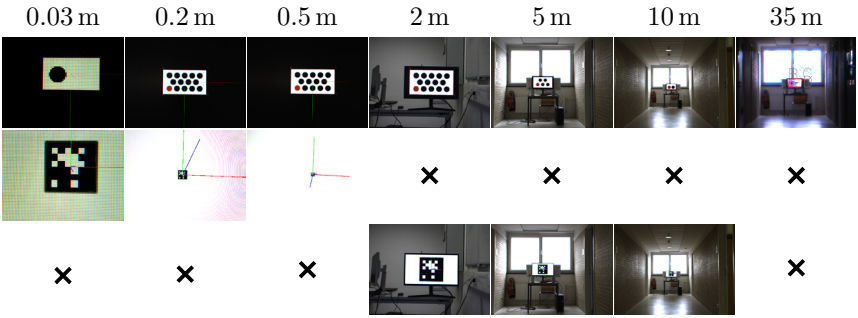


Figure 6.23: Image captures of the markers at different distances. First row the FDYMA, second row the small Aruco with 1 cm side length, and third row the big Aruco with 25 cm side length. For the FDYMA at 35 m, the image was zoomed in to make the marker visible for the reader.

the detection of the corners create significant variations on the rotation matrix estimate. Additionally, it is known that increasing the camera-to-marker distance also increases the error due to the scale reduction in image coordinates, which makes the detection more susceptible to noise. On top of all the above, we have to add all the possible corner cases that happen in ill-posed configurations and are not easy to predict. Many studies rely on simulations to compare the accuracy of the markers. However, it is almost impossible to simulate the exact influence of the camera capture pipeline on the final measurements, especially the behavior of the noise.

In order to properly test the accuracy of the fiducial under all the previously described scenarios, we are going to perform our measurements in a room with an Optitrack multi-camera tracking system with millimeter accuracy, as shown in Fig. 6.25. We placed the camera at different distances and different angles, the maximum distance case was limited by the dimensions of the room (max 4 m), and we always oriented the camera aiming to the center of the screen. The exact points at which the camera was placed are described in Fig. 6.25. The selected angles (0°, 22.5° and 45°) cover from the fronto-parallel case (0°) to the highest inclination (45°) that would also fit in the room for the selected distances 0.5 m to 3.5 m.

The Optitrack system requires the placement of small reflecting spheres around the object that needs to be tracked. We placed a set of spheres in the camera and in the screen to measure their pose in the world. However, we are interested in finding the real value of the marker-to-camera pose

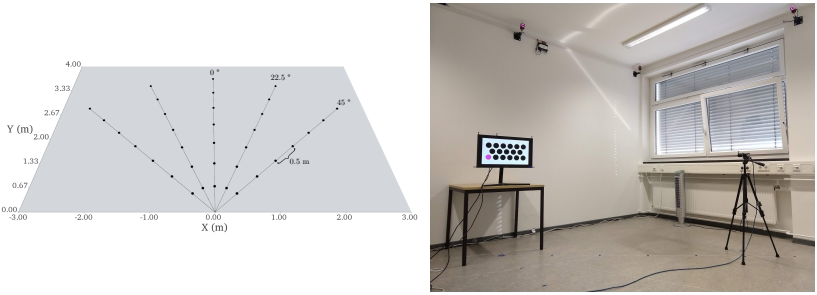


Figure 6.24: Camera placement in 2D coordinates for the marker accuracy evaluation. The screen that displays the marker was fixed at coordinate (0, 0) and the camera was placed along the lines of 0°, 22.5° and 45° at a distance from the marker of 0.5 m to 3.5 m in intervals of 0.5 m. For each position, the camera was oriented so that the marker was in the center of the image and the vertical height of the camera was the same as the center of the marker. The ground truth pose of the camera was captured with millimeter accuracy by the multi-camera localization system Optitrack.

${}^{c}T_{m}$, and to measure ${}^{c}T_{m}$ from the coordinates of the Optitrack spheres, we have to perform a process called hand-eye calibration. A hand-eye calibration finds the transformations from the coordinate frame of the Optitrack system to the center of the screen and to the camera coordinate frame. For our experiments, we performed the hand-eye calibration using the ROS package `easy_handeye`, which conveniently encapsulates the state of the art hand-eye calibration methods of the Visual Servoing Platform (ViSP). The selection of angles and distances and the Optitrack ground truth measurements will allow us to make a 2D map of accuracy for both position and rotation estimation.

For each of the camera positions defined in Fig. 6.25, we displayed the Aruco marker with a side length of 18 cm, and the FDYMA in automatic direct control mode with a circle area reference value of $500\,\mathrm{px}^2$, which gives a high point density for better accuracy. For each marker and each position, we took 100 image captures, estimated the pose from each capture, and calculated the mean μ and standard deviation σ of the translation error metric TE using (5.22) and the rotation error metric RE using (5.21). The main results of this experiment are presented in Fig. 6.25.

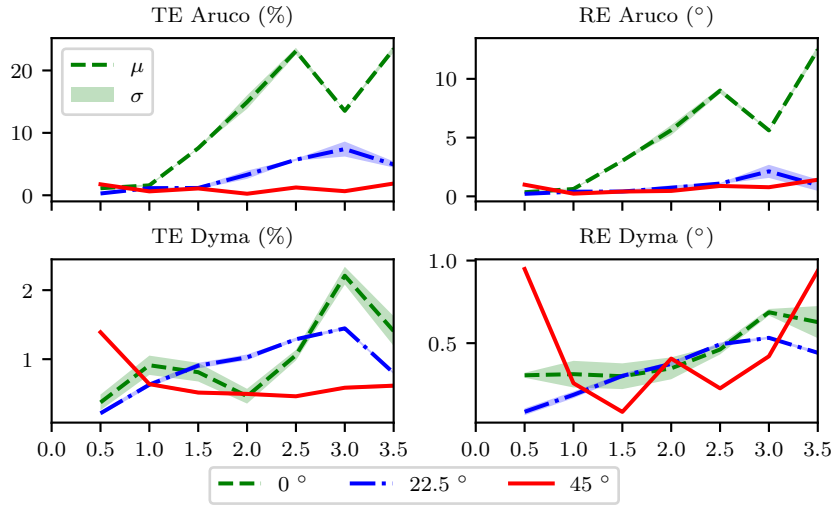


Figure 6.25: Translational and rotational accuracy of the estimated pose from Aruco and FDYMA for different distances and angles. Notice that there is an order of magnitude of difference between the errors of Aruco and those of FDYMA (there is a different scaling of the Y-axes).

The experiments validate what is a known problem of traditional planar fiducials. As we can see from the Aruco results, there is significant variability in the pose error; this variation depends on the relative orientation and the distance. The higher the distance, the greater the error. The worst possible angle is the fronto-parallel configuration (angle of 0°). Increasing the relative angle also increases the accuracy of Aruco, and we found the best performance at an angle of 22.5° at close distances.

The maximum errors of FDYMA for both the translation and rotation are an order of magnitude less than the maximum errors of Aruco. Additionally, the FDYMA has higher stability with less variation for different angles and distances. The FDYMA has errors under 2.2 % for the translation and under 1° for the rotation. Greater distances also decrease the accuracy of the FDYMA, but nowhere near to the drastic change seen on Aruco.

For the view angle of 45 °, and a distance of 50 cm, the FDYMA and Aruco present a higher error compared to other angles, which we believe is caused by the usage of a screen. Some screen technologies have a limited view angle, and in the case of the screen used in this experiment, the

effect of this limitation can be already perceived at 45°. In Fig. 6.26, we can observe that the right side of the screen looks brighter than the left side; this will affect the edge detection algorithm and displace the features from their correct positions. However, even with this disadvantage, the error of the FDYMA is less than the error of Aruco at the same position, and we do not see this effect at an angle of 22.5°.

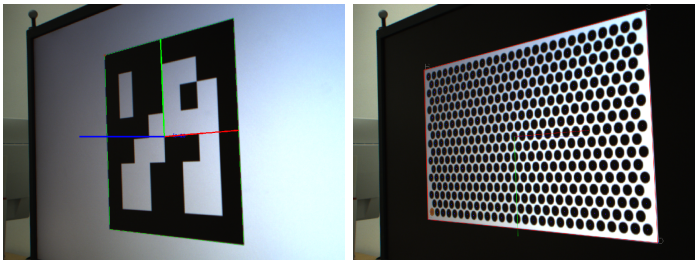


Figure 6.26: Effect of using an inclined view angle when capturing a screen with a camera. These images were captured with an angle of 45° at a distance of 50 cm which is an angle that is outside the optimal viewing region of the screen, hence, the colors on the screen do not look uniform; the left side of the screen appears darker than the right side, which affects slightly the pose estimation accuracy.

Due to the configuration of our experiment, we can make a 2D interpolation of the sampled positions, which provides a better insight into the stability of the pose estimation. In Fig. 6.27, we present a 2D map of the positional and rotational accuracy for both markers; this allow us to appreciate the behavior of the markers in space.

From the 2D maps, we can appreciate that the FDYMA, compared to Aruco, has a flat error behavior with a low variation for all the sampled distances and angles, which is ideal for a fiducial marker. It can be noticed that the fronto-parallel configuration is the most challenging configuration, even for the FDYMA. The effect is more pronounced on the rotation than in translation, but it is still a minimal variation.

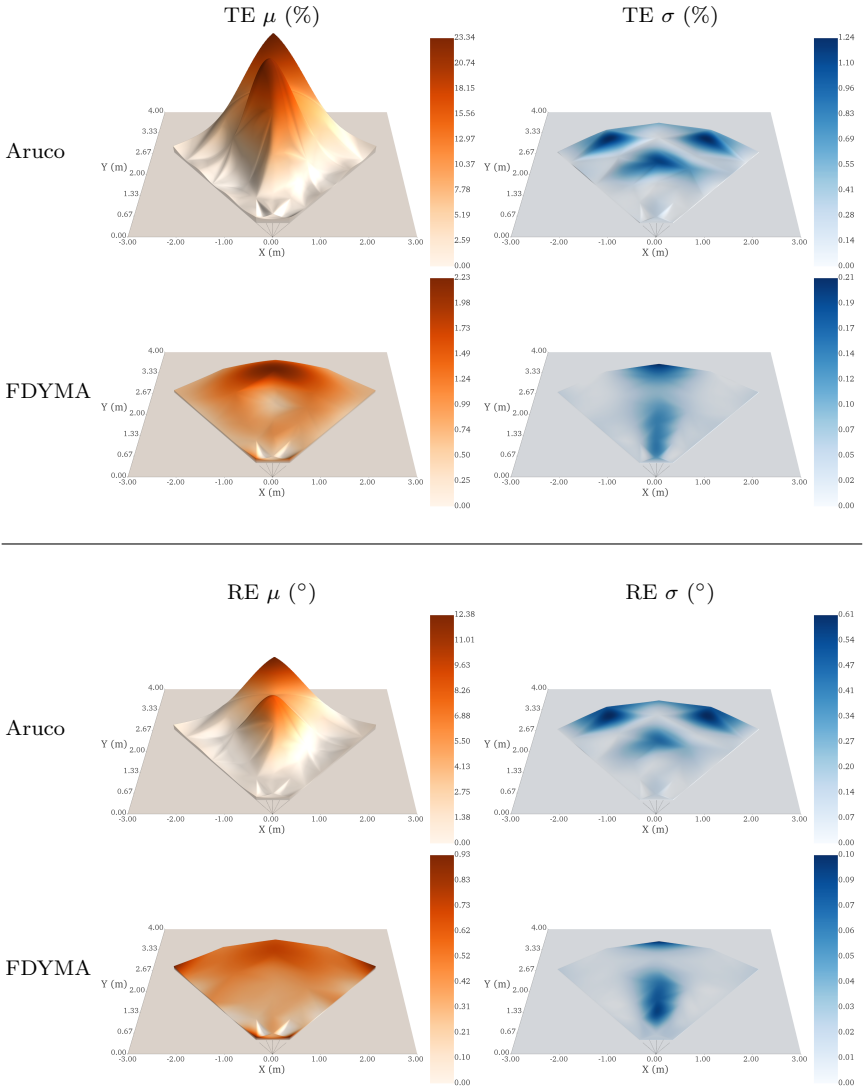


Figure 6.27: Spatial distribution of the translation and rotation errors for the Aruco marker and FDYMA. The errors are represented by a color-coded height map, the mean on the left side in orange, and the standard deviation on the right side in blue. The elevation scale is the same for all the figures, but the color code changes for each one to represent the plotted values better.

Accuracy in the presence of noise

From the results of the previous section, one can argue that the only reason for the accuracy difference between Aruco and FDYMA is because the latter changes in size; meanwhile, we left Aruco at a fixed size. However, FDYMA allows the display of more control points since it does not require to show a complex identification, and additionally mixes different kinds of shapes to display the control points (circles and corners), which makes it more accurate. In order to test this claim, we place the camera at a fixed distance from the screen, and we display both the Aruco and FDYMA but this time, we configure the FDYMA, so the bounding quadrilateral occupies the same display area as the Aruco marker; in this way, we try to remove the scale constraint. In this configuration, we also test the influence of the image noise in the stability of the pose estimation, see Fig. 6.28.

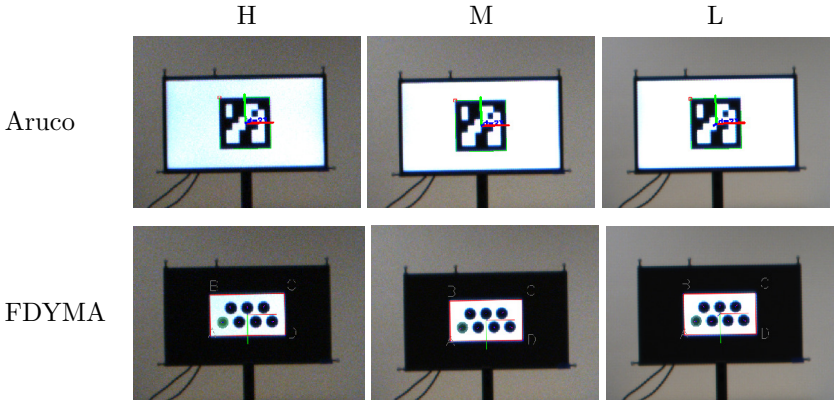


Figure 6.28: Images of Aruco and FDYMA for high (H), medium (M) and low (L) image noise. The letter H stands for high noise, M medium noise and L low noise.

To manually increase the amount of image noise, we change the exposure time while also changing the camera sensor gain to get pictures with similar brightness. Short exposure times require more amplification of the camera sensor measurement, which in turn amplifies the noise. We selected three different pairs of exposure-gain. The letter H stands for high noise, M medium noise, and L low noise. For each of these settings, we took 100 images of each marker and calculated the error metrics for translation and

rotation. The results of this test are presented in Fig. 6.29. As expected for the Aruco marker, the noise greatly affects the accuracy of the pose estimation both for translation and rotation. For Aruco, the change from noise setting L to M is small; however, there is a big jump from setting M to setting H, which evidences that a prediction of this marker response to noise can be problematic. In contrast, the FDYMA has a more stable response to noise, due to more and diverse control points. The mean of the error and the deviation is less than those of Aruco for all the noise settings.

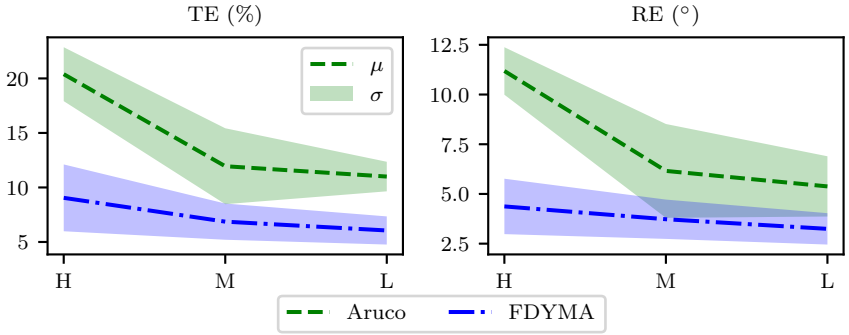


Figure 6.29: Translational and rotational accuracy of FDYMA and Aruco markers in the presence of image noise. The letter H stands for high noise, M medium noise and L low noise.

The results from this section confirm that the FDYMA design provides superior stability than traditional planar markers even without dynamically changing the scale. Since a FDYMA does not require much space for the identification, the extra control points improve the pose estimation accuracy and stability. This better basic design, coupled with the dynamic scale changes, make the FDYMA an outstanding fiducial marker for pose estimation. The added benefits come with some disadvantages; the FDYMA is intrinsically more complex to implement than traditional fiducials since it requires communication and collaboration between several modules. Moreover, the usage of screens brings its challenges due to the influence of the screen technologies on detection speed, the limited usage in outdoor environments, and the limited view angles. Nonetheless, we believe that the FDYMA is an excellent tool for all the applications where the accuracy is the most relevant factor, especially for cooperative robotics, since those systems usually have the screens and communication

modules integrated. We believe that the benefits of a FDYMA greatly surpass the complexities of its implementation; as screen technologies advance and the IoT becomes more of a reality, there is going to be a broader range of applications where a FDYMA can be used.

7 Conclusions and outlook

In this chaper we present a review of the direct achievements of this dissertation and give an outlook of future research possibilities.

7.1 Overview

In the first chapter, we introduced the reader to the topic of vision-based localization, emphasizing the relationship between action and perception and we gave a first peek at the potential sources of errors in pose estimation and how we were going to address them. In Chapter 2, we presented the basics of camera-based pose estimation while introducing the reader to our notation, which is a mixture of the one used in robotics and computer vision. In the next chapter, we went deeper into the concepts of control points and features and performed a comparative study of artificial features for pose estimation; this study allowed us to find points in common among the different fiducial designs and also identify their problems. With this grounds, we started with our main contributions in Chapter 4, where we presented the MOMA, a new cooperative odometry system that uses only artificial features, and by doing so, we avoid some of the errors introduced by automatically detected natural features and increasing accuracy and robustness of pose estimation in single and multi-robot scenarios. The work on this odometry system highlighted some of the problems on the spatial configuration of the control points, so we elaborated a methodology to obtain optimal configurations in Chaper 5 that allows a better selection of control points for future fiducial designs. The extensive study on fiducial designs of Chapter 3, the experience on interaction that we gathered from MOMA, and the optimal configuration for six control points obtained in Chaper 5 were used as the basis for the design of a new kind of interactive fiducial marker in Chapter 6, closing in this way the dissertation.

The results of this dissertation also paved the way for some other work, which is out of the scope of this dissertation, but still worth mentioning, such as contributions in the field of UAV control and landing [1, 78] and in the field of camera calibration [2]. On the last one, many of the results

obtained in the chapter about optimal control points and in the one about the FDYMA were applied to the development of a new single pose camera calibration system that employs curved screens; this research has led to the creation of a startup with the name "Caliberation".

Next, we will present more detailed conclusions for the main highlights of the dissertation and future work.

7.2 MOMA

The new cooperative mobile marker odometry demonstrated a high accuracy cooperative visual odometry system without the need of environment features and with better accuracy than state of the art markerless-based methods such as VO in featureless environments and better accuracy than VO methods in feature-rich environments. Our system exploits interaction between the observer and the marker. In MOMA the marker moves and cooperates with the observer. This cooperation was mentioned in the introduction chapter as the perception-interaction cycle.

We demonstrated in different real cooperative robot configurations the feasibility of the implementation and the advantages of the system. Our proposed method proved easy to integrate into existing multi-robot systems since it only requires a cheap monocular camera and cheap printed fiducial markers. We believe that MOMA is particularly interesting for challenging environments, e.g., underwater environments or with the absence of light, with potential applications in search and rescue scenarios. We found that the poses one selects for the transitions mater. Some relative camera-to-marker poses are better than others in terms of accuracy. For example, a fronto-parallel arrangement is less accurate than inclined. We think that a possible avenue of research for the future is to study which relative poses are optimal for the switch and integrate this knowledge into a planner that defines the best relative motions of the robot to maintain accuracy. Another potential research path would be to use three-dimensional fiducial markers that could be observable from any angle or use the 3D model of the shape of each robot as the marker.

7.3 Optimal points

A method for obtaining optimized control points for homography estimation is presented. The lower the number of control points, the more the

point configuration influences the accuracy of homography and PnP estimation methods. Our empirical results show that a square is a very stable and robust configuration for all camera poses. Optimized points configurations follow simple rules, they are driven to distribute in space, and they tend to increase the distance to each other; this includes, in many cases, the optimized 4-point configuration as a subset. We found that there is almost no difference in accuracy between the best pose estimation methods when optimized point configurations are used, which could serve for better comparisons between different kinds of methods. It was a pleasant surprise to see how such beautiful symmetrical polygonal shapes emerged from the optimization in the cases of 4, 5 and 6 points; these geometrical shapes can serve as the basis of other future planar fiducial designs.

We think that the condition number can serve as a metric also to evaluate point configurations "live" during pose estimation tasks when using non-artificial features (e.g., SIFT, SURF, or ORB features). Usually, too many features are extracted for each image, even after filtering with RANSAC or other analogous methods. We think there must be a minimum set of these features that are the best at representing the pose. The condition number can be used as a way of evaluating sets of features and use only the optimal; this will increase pose estimation accuracy, reduce computation time and use less storage space for bundle adjustment applications.

In future work, more work is needed in generalizing the results to non-planar point configurations and include other optimization metrics in the optimization; an excellent candidate is the trace of the posterior covariance matrix in the reprojection error, which is commonly used in the research field of optimal sensor placement.

7.4 Dynamic fiducial markers

The dynamic fiducial marker has been a consolidation of the research done during this project. In a world that is getting more and more connected through technology and where the IoT seems to be the future, it makes only sense that traditional paper fiducials transform into a connected and intelligent device.

Our design is based on more than 30 years of fiducial knowledge, and it outperforms the state of the art designs in terms of accuracy and range. Similar to the MOMA system, the core of a dynamic marker is the interaction, the cooperation between the observer and the marker, all together

working towards a better pose estimation.

The results, both simulated and experimental, proved that this concept has higher accuracy and robustness than traditional fiducials at the expense of complexity. However, this additional complexity in several scenarios (especially in robotics) is a fair price to pay to have a cheap, reliable, and accurate localization device. Even though the focus of this dissertation has been on pose estimation, the dynamic marker can be an invaluable tool for camera calibration applications, where usually the process of acquiring the points for calibration is complicated, cumbersome, and where accuracy is of the utmost importance.

The practical limits of a dynamic fiducial are yet to be seen; the technology behind displays has a powerful momentum and is rapidly improving. It will not be long before we see cheap low latency electronic ink displays at the frame rates of regular LCD monitors, making the use of dynamic fiducials in outdoor environments a reality. Independently of the technology, the core concepts presented in this thesis in terms of block design and control are going to be maintained.

An immediate path for future research is to integrate a FDYMA into the MOMA system. Instead of using planar markers, we can place a screen on the robots and perform the odometry. The marker will interact in two different ways with the observer, through shape, and relative positioning, improving the overall accuracy and adding more flexibility due to the increased range of detection.

Publications and patent

[A1] Raul Acuna, Zaijuan Li, and Volker Willert. MOMA: Visual Mobile Marker Odometry. In *2018 Int. Conf. Indoor Position. Indoor Navig.*, pages 206–212. IEEE, 2018.

[A2] Raul Acuna and Volker Willert. Dynamic Markers: UAV landing proof of concept. *Proc. IEEE Lat. Am. Robot. Symp.*, pages 496–502, 2018.

[A3] Raul Acuna and Volker Willert. Insights into the robustness of control point configurations for homography and planar pose estimation. *CoRR*, abs/1803.0, 2018.

[A4] Raul Acuna and Volker Willert. Method for determining calibration parameters of a camera, Patent WO 2020/109033, 2020.

[A5] Raul Acuna, Ding Zhang, and Volker Willert. Vision-based UAV landing on a moving platform in GPS denied environments using motion prediction. In *Proc. IEEE Lat. Am. Robot. Symp.*, pages 515–521, João Pessoa, Brazil, 2018. IEEE.

[A6] Raul Acuna, Robin Ziegler, and Volker Willert. Single pose camera calibration using a curved display screen. In *Forum Bildverarbeitung*, pages 25–36, Karlsruhe, 2018. KIT Scientific Publishing.

[A7] Zaijuan Li, Raul Acuna, and Volker Willert. Cooperative Localization by Fusing Pose Estimates from Static Environmental and Mobile Fiducial Features. In *Proc. IEEE Lat. Am. Robot. Symp.*, pages 65–70. IEEE, nov 2018.

[A8] Dinu Mihailescu-Stoica, Raul Acuna, and Jurgen Adamy. High performance adaptive attitude control of a quadrotor. *2019 18th Eur. Control Conf. ECC 2019*, pages 3462–3469, 2019.

Bibliography

[1] Raul Acuna, Ding Zhang, and Volker Willert. Vision-based UAV landing on a moving platform in GPS denied environments using motion prediction. In *Proc. IEEE Lat. Am. Robot. Symp.*, pages 515–521, João Pessoa, Brazil, 2018. IEEE.

[2] Raul Acuna, Robin Ziegler, and Volker Willert. Single pose camera calibration using a curved display screen. In *Forum Bildverarbeitung*, pages 25–36, Karlsruhe, 2018. KIT Scientific Publishing.

[3] B Atcheson, F Heide, and W Heidrich. CALTag: High Precision Fiducial Markers for Camera Calibration. In *Vision, Model. Vis.* The Eurographics Association, 2010.

[4] Ruzena Bajcsy, Yiannis Aloimonos, and John K. Tsotsos. Revisiting Active Perception. pages 1–22, 2016.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *Comput. Vis. – ECCV 2006*, volume 3951 LNCS, pages 404–417, 2006.

[6] Burak Benligiray, Cihan Topal, and Cuneyt Akinlar. STag: A stable fiducial marker system. *CoRR*, abs/1707.0, 2017.

[7] Filippo Bergamasco, Andrea Albarelli, Emanuele Rodola, and Andrea Torsello. RUNE-Tag: A high accuracy fiducial marker with strong occlusion resilience. In *CVPR 2011*, pages 113–120, 2011.

[8] Filippo Bergamasco, Andrea Albarelli, and Andrea Torsello. Pi-Tag: A fast image-space marker design based on projective invariants. *Mach. Vis. Appl.*, 24(6):1295–1310, 2013.

[9] Tolga Birdal, Ievgeniia Dobryden, and Slobodan Ilic. X-Tag: A fiducial tag for flexible and accurate bundle adjustment. In *Proc. 4th Int. Conf. 3D Vis.*, pages 556–564, 2016.

[10] Matevž Bošnak, Drago Matko, and Sašo Blažič. Quadrocopter hovering using position-estimation information from inertial sensors and a high-delay video system. *J. Intell. Robot. Syst. Theory Appl.*, 67(1):43–60, 2012.

[11] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. First edition, 2008.

[12] Martin Buczko and Volker Willert. How to distinguish inliers from outliers in visual odometry for high-speed automotive applications. In *IEEE Intell. Veh. Symp. Proc.*, number IV, pages 478–483, 2016.

[13] L. Calvet, P. Gurdjos, and V. Charvillat. Camera tracking using concentric circle markers: Paradigms and algorithms. In *Int. Conf. Image Process.*, pages 1361–1364, 2012.

[14] L Calvet, P Gurdjos, C Griwodz, and S Gasparini. Detection and Accurate Localization of Circular Fiducials under Highly Challenging Conditions. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 562–570, 2016.

[15] Luca Carlone and Sertac Karaman. Attention and Anticipation in Fast Visual-Inertial Navigation. *IEEE Trans. Robot.*, 35(1):1–20, 2019.

[16] Patrick Cavanagh. Visual cognition. *Vision Res.*, 51(13):1538–1551, 2011.

[17] Pei Chen and David Suter. Error analysis in homography estimation by first order approximation tools: A general technique. *J. Math. Imaging Vis.*, 2009.

[18] Youngkwan Cho, Jongweon Lee, and Ulrich Neumann. A Multi-ring Color Fiducial System and an Intensity-invariant Detection Method for Scalable Fiducial-Tracking Augmented Reality. In *IWAR*, pages 147–165, 1998.

[19] David H S Chung, Matthew L. Parry, Philip A. Legg, Iwan W. Griffiths, Robert S. Laramee, and Min Chen. Visualizing multiple error-sensitivity fields for single camera positioning. *Comput. Vis. Sci.*, 2014.

[20] Louis G. Clift and Adrian F. Clark. Determining positions and distances using collaborative robots. In *Comput. Sci. Electron. Eng. Conf.*, pages 189–194, 2015.

[21] Toby Collins and Adrien Bartoli. Infinitesimal plane-based pose estimation. *Int. J. Comput. Vis.*, 2014.

[22] Gabriele Costante, Christian Forster, Jeffrey Delmerico, Paolo Valigi, and Davide Scaramuzza. Perception-aware Path Planning. *arXiv*, pages 1–16, 2016.

[23] G David. Object recognition from local scale-invariant features. *Proc. IEEE Int. Conf. Comput. Vis.*, 2:1150–1157, 1999.

[24] Joseph Degol, Derek Hoiem, and Example Chromatag Detections. ChromaTag : A Colored Marker and Fast Detection Algorithm. In *ICCV*, volume 2, pages 2–3, 2017.

[25] Vikas Dhiman, Julian Ryde, and Jason J. Corso. Mutual localization: Two camera relative 6-DOF pose estimation from reciprocal fiducial observation. *IEEE Int. Conf. Intell. Robot. Syst.*, pages 1347–1354, 2013.

[26] H Dierke, M. Petz, and R. Tutsch. Photogrammetrische Bestimmung der Brechungseigenschaften von Flüssigkristallbildschirmen. In *Forum Bild. 2018, Karlsruhe*, pages 13–24, 2018.

[27] Diego Brito Dos Santos Cesar, Christopher Gaudig, Martin Fritsche, Marco A. Dos Reis, and Frank Kirchner. An evaluation of artificial fiducial markers in underwater environments. *MTS/IEEE Ocean. 2015 - Genova Discov. Sustain. Ocean Energy a New World*, 2015.

[28] Dougal Maclaurin. *Modeling, Inference and Optimization with Composable Differentiable Procedures.* PhD thesis, Harvard University, 2016.

[29] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. *Proc. IEEE Int. Conf. Comput. Vis.*, pages 1449–1456, 2013.

[30] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. PAMPC: Perception-Aware Model Predictive Control for Quadrotors. *arXiv cs.RO*, 2018.

[31] Mark Fiala. ARTag, a fiducial marker system using digital techniques. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2:590–596, 2005.

[32] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, 1981.

[33] Andrew Fitzgibbon, Maurizio Pilu, and Robert B. Fisher. Direct least square fitting of ellipses. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):476–480, 1999.

[34] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. Probabilistic approach to collaborative multi-robot localization. *Auton. Robots*, 8(3):325–344, 2000.

[35] Chaumette Francois and Seth Hutchinson. Visual Servo Control Part I : Basic Approaches. *IEEE Robot. Autom. Mag.*, 13(December):82–90, 2006.

[36] Friedrich Fraundorfer and Davide Scaramuzza. Visual odometry: Part II: Matching, robustness, optimization, and applications. *IEEE Robot. Autom. Mag.*, 19(2):78–90, 2012.

[37] Borko Furht. *Handbook of Augmented Reality*. Springer, 2011.

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

[39] S Garrido-Jurado, R Muñoz-Salinas, F J Madrid-Cuevas, and R Medina-Carnicer. Generation of fiducial marker dictionaries using Mixed Integer Linear Programming. *Pattern Recognit.*, 51:481–491, 2016.

[40] Valeria Garro, Fabio Crosilla, and Andrea Fusiello. Solving the PnP problem with anisotropic orthogonal procrustes analysis. *3DIMPVT*, 2012.

[41] Lance B. Gatrell, William A. Hoff, and Cheryl W. Sklair. Robust image features: Concentric contrasting circles and their image extraction. In *Coop. Intell. Robot. Sp.*, volume 2, pages 235–244, Boston, 1992. SPIE.

[42] Andreas Geiger, Julius Ziegler, and Christoph Stiller. StereoScan: Dense 3d Reconstruction in Real-time. In *IEEE Intell. Veh. Symp.*, Baden-Baden, Germany, 2011.

[43] Gene H. Golub. *Matrix Computations*, volume 6. The Johns Hopkins University Press, 1393.

[44] Hyowon Ha, Yunsu Bok, Kyungdon Joo, Jiyoung Jung, and In So Kweon. Accurate camera calibration robust to defocus using a smartphone. *Proc. IEEE Int. Conf. Comput. Vis.*, 11-18-Dece(2011):828–836, 2016.

[45] Verena Händler and Volker Willert. Accuracy Evaluation for Automated Optical Indoor Positioning Using a Camera Phone. *Zeitschrift für Geodäsie, Geoinf. und Landmanagement*, 137(2):114–122, apr 2012.

[46] M. J. Harker and P. L. O'Leary. Computation of Homographies. In *BMVC*, 2005.

[47] Chris Harris and Mike Stephens. A Combined Corner and Edge Detector. In *Proc. Fourth Alvey Vis. Conf.*, pages 147—-151, 1988.

[48] R.I. Hartley. In defence of the 8-point algorithm. In *Proc. IEEE Int. Conf. Comput. Vis.*, 1997.

[49] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[50] Adam Herout, Michal Zachariáš, Markéta Dubská, and Jiří Havel. Fractal marker fields: No more scale limitations for fiduciary markers. *ISMAR 2012 - 11th IEEE Int. Symp. Mix. Augment. Real. 2012, Sci. Technol. Pap.*, November:285–286, 2012.

[51] Joel A Hesch and Stergios I Roumeliotis. A direct least-squares (DLS) method for PnP. In *IEEE Int. Conf. Comput. Vis.*, 2011.

[52] Andrew Howard, Lynne E. Parker, and Gaurav S. Sukhatme. Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection. *Int. J. Rob. Res.*, 25(5-6):431–447, 2006.

[53] Victor Jimenez. Design of a dynamic fiducial marker for optimal pose estimation. Bachelor's thesis, TU Darmstadt, 2019.

[54] Pengju Jin, Pyry Matikainen, and Siddhartha S. Srinivasa. Sensor fusion for fiducial tags: Highly robust pose estimation from single frame RGBD. *IEEE Int. Conf. Intell. Robot. Syst.*, September:5770–5776, 2017.

[55] Martin Kaltenbrunner and Ross Bencina. reacTIVision: A computer-vision framework for table-based tangible interaction. In *First Int. Conf. Tangible Embed. Interact.*, pages 69–74, 2007.

[56] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images. *arXiv cs.CV*, 2017.

[57] Dawar Khan, Sehat Ullah, and Ihsan Rabbi. Factors affecting the design and tracking of ARToolKit markers. *Comput. Stand. Interfaces*, 41:56–66, 2015.

[58] Laurent Kneip, Hongdong Li, and Yongduek Seo. UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability. In *Eur. Conf. Comput. Vis.*, 2014.

[59] T Krajník, M Nitsche, J Faigl, T Duckett, M Mejail, and L Preučil. External localization system for mobile robotics. In *16th Int. Conf. Adv. Robot.*, nov 2013.

[60] Tomás Krajník, Matías Nitsche, Jan Faigl, Petr Vanek, Martin Saska, Libor Preucil, Tom Duckett, and Marta Mejail. A Practical Multirobot Localization System. *J. Intell. Robot. Syst. Theory Appl.*, 76(3-4):539–562, 2014.

[61] Maximilian Krogius, Acshi Haggenmiller, and Edwin Olson. Flexible Layouts for Fiducial Tags. In *IEEE Int. Conf. Intell. Robot. Syst.*, pages 1898–1903, 2019.

[62] Ryo Kurazume, Shigemi Nagata, and S. Hirose. Cooperative positioning with multiple robots. *Proc. 1994 IEEE Int. Conf. Robot. Autom.*, pages 1250–1257, 1994.

[63] Daewon Lee, Tyler Ryan, and H. Jin Kim. Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing. *Proc. IEEE Int. Conf. Robot. Autom.*, pages 971–976, 2012.

[64] Thomas Lemaire, Cyrille Berger, Il Kyun Jung, and Simon Lacroix. Vision-based SLAM: Stereo and monocular approaches. *Int. J. Comput. Vis.*, 74(3):343–364, 2007.

[65] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An accurate O(n) solution to the PnP problem. *Int. J. Comput. Vis.*, 2008.

[66] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *Int. J. Rob. Res.*, 34(3):314–334, 2015.

[67] Shiqi Li, Chi Xu, and Ming Xie. A robust O(n) solution to the perspective-n-point problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2012.

[68] Wei Li, Tianguang Zhang, and Kolja Kühnlenz. A vision-guided autonomous quadrotor in an air-ground multi-robot system. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 2980–2985, Shanghai, 2011. IEEE.

[69] Zaijuan Li, Raul Acuna, and Volker Willert. Cooperative Localization by Fusing Pose Estimates from Static Environmental and Mobile Fiducial Features. In *Proc. IEEE Lat. Am. Robot. Symp.*, pages 65–70. IEEE, nov 2018.

[70] Peter Lightbody, Tomáš Krajník, and Marc Hanheide. A Versatile High-performance Visual Fiducial Marker Detection System with Scalable Identity Encoding. *Proc. Symp. Appl. Comput.*, pages 276–282, 2017.

[71] Hyon Lim and Young Sam Lee. Real-Time Single Camera SLAM Using Fiducial Markers. In *ICCAS-SICE*, pages 177–182, 2009.

[72] Diego López De Ipiña, Paulo R.S. Mendonça, and Andy Hopper. TRIP: A Low-Cost Vision-Based Location System for Ubiquitous Computing. *Pers. Ubiquitous Comput.*, 6(3):206–219, 2002.

[73] Chien Ping Lu, Gregory D. Hager, and Eric Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2000.

[74] Pieter Jan Maes, Marc Leman, Caroline Palmer, and Marcelo M. Wanderley. Action-based effects on music perception. *Front. Psychol.*, 4(JAN):1–14, 2014.

[75] John Mallon and Paul F Whelan. Which pattern? Biasing aspects of planar calibration patterns and detection methods. *Pattern Recognit. Lett.*, 28(8):921–930, 2007.

[76] Joshua G. Mangelson, Ryan W. Wolcott, Paul Ozog, and Ryan M. Eustice. Robust visual fiducials for skin-to-skin relative ship pose estimation. *Ocean. 2016 MTS/IEEE Monterey, OCE 2016*, 2016.

[77] Eric Marchand, Hideaki Uchiyama, Fabien Spindler, Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose Estimation for Augmented Reality: A Hands-On Survey. *IEEE Trans. Vis. Comput. Graph.*, 1(1), 2016.

[78] Dinu Mihailescu-Stoica, Raul Acuna, and Jurgen Adamy. High performance adaptive attitude control of a quadrotor. *2019 18th Eur. Control Conf. ECC 2019*, pages 3462–3469, 2019.

[79] Elias Mueggler, Matthias Faessler, Flavio Fontana, and Davide Scaramuzza. Aerial-guided Navigation of a Ground Robot among Movable Obstacles. In *Proc. IEEE Int. Symp. Safety, Secur. Rescue Robot.*, 2014.

[80] Rafael Muñoz-Salinas, Manuel J. Marín-Jimenez, Enrique Yeguas-Bolivar, and Rafael Medina-Carnicer. Mapping and Localization from Planar Markers. *CoRR*, abs/1606.0:1–14, 2016.

[81] Leonid Naimark and Eric Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Int. Symp. Mix. Augment. Real.*, pages 27–36, 2002.

[82] Gaku Nakano. Globally Optimal DLS Method for PnP Problem with Cayley parameterization. In *BMVC*, 2015.

[83] Michael Neunert, Michael Bloesch, and Jonas Buchli. An Open Source, Fiducial Based, Visual-Inertial Motion Capture System. In *Int. Conf. Inf. Fusion*, 2016.

[84] Matias Nitsche, Tomáš Krajník, Petr Čížek, Marta Mejail, and Tom Duckett. WhyCon: An Efficent, Marker-based Localization System. In *IROS Work. Open Source Aer. Robot.*, 2015.

[85] Denis Oberkampf, Daniel F. DeMenthon, and Larry S. Davis. Iterative Pose Estimation Using Coplanar Feature Points. *Comput. Vis. Image Underst.*, 1996.

[86] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. *Proc. IEEE Int. Conf. Robot. Autom.*, pages 3400–3407, 2011.

[87] C. B. Owen, Fan Xiao, and P. Middlin. What is the best fiducial? In *Proc. 1st IEEE Int. Work. Augment. Real. Toolkit*, Darmstadt, 2002.

[88] Oxford. *Oxford English dictionary*. Oxford University Press, 1989.

[89] Andrew Parker. *In The Blink Of An Eye: How Vision Sparked The Big Bang Of Evolution*. Perseus Pub, 2004.

[90] I Poupyrev, H Kato, and M Billinghurst. ARToolkit User Manual. Technical report, University of Washingtoon, 2000.

[91] Meghshyam G. Prasad, Sharat Chandran, and Michael S. Brown. A motion blur resilient fiducial for quadcopter imaging. *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, pages 254–261, 2015.

[92] Louis B. Rall. *Automatic Differentiation: Techniques and Applications*, volume 120. Springer, 1981.

[93] Prasanna Rangarajan and Panos Papamichalis. Estimating homographies without normalization. In *Proc. Int. Conf. Image Process.*, 2009.

[94] Jun Rekimoto. Matrix : A Realtime Object Identification and Registration Method for Augmented Reality. In *Proc. Asia Pacific Comput. Hum. Interact.*, pages 63–69, Kangawa, Japan, 1998. IEEE Computer Society.

[95] Jun Rekimoto and Yuji Ayatsuka. CyberCode: Designing augmented reality environments with visual tags. In *Proc. DARE 2000 Des. Augment. Real. Environ.*, pages 1–10, NY, USA, 2000. ACM.

[96] Michael Rohs and Beat Gfeller. Using Camera-Equipped Mobile Phones For Interacting with Real-World Objects. *Adv. Pervasive Comput.*, 176:265–271, 2004.

[97] Francisco Romero Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Fractal Markers: a new approach for long-range marker pose estimation under occlusion. *preprint*, 2019.

[98] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proc. IEEE Int. Conf. Comput. Vis.*, number November, pages 2564–2571, Barcelona, 2011.

[99] Victoria Rudakova and Pascal Monasse. Camera matrix calibration using circular control points and separate correction of the geometric distortion field. *Proc. - Conf. Comput. Robot Vision, CRV 2014*, pages 195–202, 2014.

[100] Jon Russell. The Humble QR Code Is Being Disrupted... And Going Dotless, 2015.

[101] S. Saripalli, J.F. Montgomery, and G.S. Sukhatme. Visually guided landing of an unmanned aerial vehicle. *IEEE Trans. Robot. Autom.*, 19(3):371–380, 2003.

[102] M Saska, V Vonásek, T Krajník, and L Přeučil. Coordination and navigation of heterogeneous UAVs-UGVs teams localized by a hawk-eye approach. In *2012 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, volume 33, pages 2166–2171, 2012.

[103] Junaed Sattar, Eric Bourque, Philippe Giguère, and Gregory Dudek. Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction. In *Proc. - Fourth Can. Conf. Comput. Robot Vision, CRV 2007*, pages 165–171, 2007.

[104] Florian Schweiger, Bernhard Zeisl, Pierre Georgel, Georg Schroth, Eckehard Steinbach, and Nasir Navab. Maximum detector Response markers for SIFT and SURF. In *Proc. Vision, Model. Vis. Work.*, pages 145–154, 2009.

[105] Gerald Schweighofer and Axel Pinz. Robust pose estimation from a planar target. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2006.

[106] Gerald Schweighofer and Axel Pinz. Globally Optimal O(n) Solution to the PnP Problem for General Camera Models. *BMVC*, 2008.

[107] Zhan Song and Ronald Chung. Use of LCD panel for calibrating structured-light-based range sensing system. *IEEE Trans. Instrum. Meas.*, 57(11):2623–2630, 2008.

[108] G W Stewart. Perturbation Theory for the Singular Value Decomposition. Technical Report October, University of Maryland at College Park, College Park, MD, USA, 1990.

[109] Peter Sturm. Algorithms for Plane-Based Pose Estimation. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 706–711, Hilton Head Island, 2000.

[110] H Tanaka, Y Sumi, and Y Matsumoto. Avisual marker for precise pose estimation based on lenticular lenses. In *2012 IEEE Int. Conf. Robot. Autom.*, pages 5222–5227, 2012.

[111] Tom Tollenaere. SuperSAB: Fast adaptive back propagation with good scaling properties. *Neural Networks*, 3(5):561–573, 1990.

[112] Dialekti Athina Voutyrakou. 3D Fiducial Marker Configurations for Optimal Position Estimation. Master's thesis, TU Darmstadt, Darmstadt, 2019.

[113] Magdalena Wache. Ellipse detection analysis and usage in a dynamic fiducial marker. Bachelor's thesis, TU Darmstadt, 2017.

[114] Daniel Wagner and Dieter Schmalstieg. ARToolKitPlus for Pose Tracking on Mobile Devices. In *Comput. Vis. Winter Work.*, St. Lambrecht, Austria, 2007.

[115] Ken Waldron. Handbook of Robotics Chapter 1 : Kinematics. *Reading*, 2005.

[116] Hao Wang, Zongying Shi, Geng Lu, and Yisheng Zhong. Hierarchical fiducial marker design for pose estimation in large-scale scenarios. *J. F. Robot.*, 35(6):835–849, 2018.

[117] Hao Wang, Xiongfeng Wang, Geng Lu, and Yisheng Zhong. HArCo: Hierarchical Fiducial Markers for Pose Estimation in Helicopter Landing Tasks. *Proc. IEEE Int. Conf. Syst. Man, Cybern.*, pages 1968–1973, 2015.

[118] Dennis Wildermuth and Frank E. Schneider. Maintaining a common coordinate system for a group of robots based on vision. *Rob. Auton. Syst.*, 44(3-4):209–217, 2003.

[119] Volker Willert. Optical indoor positioning using a camera phone. In *Proc. 2010 int. conf. indoor Position. indoor Navig.*, 2010.

[120] Volker Willert and Julian Eggert. A Stochastic Dynamical System for Optical Flow Estimation. *IEEE Int. Conf. Comput. Vis. (ICCV), 4th Int. Work. Dyn. Vis.*, 2009.

[121] Yihong Wu. PnP problem revisited. *J. Math. Imaging Vis.*, 24(1):131–141, 2006.

[122] Guoxing Yu, Yongtao Hu, and Jingwen Dai. TopoTag: A Robust and Scalable Topological Fiducial Marker System. In *arXiv cs.CV*, 2019.

[123] Zhengyou Zhang. A flexible new technique for camera calibration. Technical report, Microsoft, 2002.

[124] Yinqiang Zheng, Yubin Kuang, Shigeki Sugimoto, Kalle Astrom, and Masatoshi Okutomi. Revisiting the PnP problem: A fast, general and optimal solution. In *Proc. IEEE Int. Conf. Comput. Vis.*, 2013.

[125] Zongqian Zhan. Camera calibration based on liquid crystal display (LCD). *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, XXXVII, 2008.