TÉCNICO LISBOA

Development of a Lightweight Visual-based Pose Estimation Sensor: Implementation and Validation

Renato Miguel da Silva Severiano

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. Alexandra Bento Moutinho
Prof. José Raul Carreira Azinheira

Examination Committee

Chairperson: Prof. Paulo Jorge Coelho Ramalho Oliveira
Supervisor: Prof. José Raul Carreira Azinheira
Member of the Committee: Prof. Carlos Baptista Cardeira

November 2018

Resumo

Com a procura de soluções para efetuar seguimento em cadeia de comboios de viaturas em auto-estradas convencionais, a necessidade de novos sensores emerge. Os sistemas baseados em visão são geralmente muito flexíveis, no entanto existe falta de métodos testados para a aplicação em causa.

Nesta dissertação é apresentado um sistema baseado em visão que funciona como um sensor para deteção da posição e orientação de um marcador circular. A colocação de um marcador destes na traseira de uma viatura deverá ser suficiente para permitir a deteção da viatura pelo sensor. Fisicamente, o sensor é composto por um *Raspberry Pi 3* e uma câmara dedicada. No *Raspberry Pi* corre o algoritmo de deteção recorrendo ao ambiente *Robot Operating System* para gestão de entrada de imagens e saída de dados de posição e orientação, garantindo a minimização dos tempos de resposta.

O algoritmo de visão utilizado no sensor é baseado num já existente, com a adição de alguns melhoramentos ao nível da deteção de alvos. Estes melhoramentos levaram ao aumento da eficiência do algoritmo, permitindo encontrar o alvo com mais facilidade e utilizando menos recursos computacionais.

Como forma de validar o sensor utilizaram-se dois métodos. Foi feita a comparação dos dados obtidos com o sensor, com os obtidos com um sistema de captura de movimentos externo. Além disso, foi testado um pequeno comboio de viaturas utilizando dois veículos omnidireccionais num sistema líder – seguidor. Em ambos os testes os resultados obtidos demonstram a efetividade da solução proposta.

Palavras-chave: Visão Computacional, Estimação de pose, Marcadores Fiduciais, Deteção de alvos, Identificação de formas

Abstract

Following the demand for platooning solutions on current motorways, the need for new sensors emerges. Vision-based systems are generaly very flexible, however there is a lack of methods tested for platooning.

This dissertation presents a vision-based system working as a sensor for detection of position and orientation of a circular marker. Putting a marker of this sort on the backside of a vehicle should be enough to perform the detection of that vehicle by the sensor. Physically, the sensor is composed by a Raspberry Pi 3 and a dedicated camera. The Raspberry Pi runs the detection algorithm making use of the framework Robot Operating System to manage data input and output aiding to minimise response times. The computer vision algorithm used on the sensor is based on an existing one, with some improvements added to the target detection part. This improvements resulted in an increase of efficiency of the algorithm, enabling an easier detection of the target using fewer computing resources.

As a way of sensor validation, its data was compared with data obtained with a motion capture system. In order to prove the usability of the sensor, it was used as the positioning sensor on a leader - follower system using two omnidirectional vehicles in order to emulate a small platoon. Both tests demonstrated the effectiveness of the proposed solution.

Keywords: Computer Vision, Pose estimation, Fiducial markers, Targets detection, Shapes identification

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

α, β	Magnification parameters.
$\boldsymbol{\epsilon}$	Vector part of a quaternion.
η	Scalar component of a quaternion.
κ_1, κ_2	Terms of radial distortion.

Roman symbols

$\mathcal{F}_A$	Reference frame A.
$\mathcal{K}$	Calibration matrix of a camera.
$\mathcal{M}$	Perspective projection matrix.
O	Origin of a frame.
P	Generic Point in space.
$\mathbf{Q}$	Unit quaternion.
$\mathcal{R}_A^B$	Rotation matrix from reference frame A to frame B.
$\mathcal{T}_A^B$	Transformation matrix from reference frame A to frame B.
$\tilde{u}, \tilde{v}$	Distorted image coordinates.
u, v	Image coordinates.
u_0, v_0	Principal point coordinates.
x, y, z	Cartesian coordinates.

Subscripts

u, v	Digital indexes.
x, y, z	Cartesian coordinates.

Chapter 1

Introduction

1.1 Motivation

The boom of shrinking technology has made possible the creation of powerful and lightweight computers. The most frequent example is today's smartphones which have very good processing power and, at the same time, good cameras usable for augmented reality (AR) applications [1, 2]. Those shrunken bits of technology are fuelling the growth of small single-board computers in the market where the Raspberry Pi (RPi) is a milestone due to its price to performance ratio.

The miniaturisation of digital cameras and computer systems allow it to be integrated on a broader range of mobile systems that can run computationally expensive computer vision algorithms. Small computation boards like the NVIDIA Jetson, the Odroid, the Raspberry Pi and the Thinker Board can be integrated in existing systems to augment its autonomous capabilities. As an example, the Raspberry Pi measures 56 by 85 millimetres and weights 42 grams.

Mobile robots are being created and adapted to work in a number of places of our lives, from the household to the road. The iRobot Roomba is helping people to keep their houses cleaner without effort. Several companies, like Waymo, are working to take the human out of from the driver's seat. In the field of transportation, there is an EU project called "Safe Road Trains for the Environment" (SARTRE) whose goal is to have formations of several cargo trucks as a train. The head truck is driven by a human and the others are following it as big mobile robots. Its main goal is to increase the safety in motorways while having the convenient side-effect of increased fuel efficiency.

A number of applications currently using Global Navigation Satellite System (GNSS) could benefit from the addition of computer vision aided relative positioning systems such as:

- police forces can have an aerial system chasing a suspect car or following the patrol car while reporting the captured data to the central department.
- the military could have an aerial system escorting a ground (or maritime) vehicle relaying ground communications to a distant ground station without the need for satellites.
- the same way that recent cars are able to park themselves, a helicopter pilot could have an assisted landing when landing on a helipad, like the one in Figure 1.1.

Figure 1.1: Helipad [3]

There are environments, such as indoors, where satellite communications are not possible, denying the usage of GNSS positioning. In other situations, where GNSS signal is available, it may not be fast enough for a swiftly moving target or even its accuracy might be questionable.

Nowadays, Unmanned Aerial Systems (UAS) are used to provide service in every sector of the economy: from agriculture to power line inspections, building surveillance, or even for military weapons deployment. The applications are endless and the unexplored potential from such systems is beyond imagination. Any tool that eases the usage of such systems should be welcome.

1.2 State of Art

1.2.1 Historical Background

Photogrammetry is a field of science that studies ways of taking measurements from photographs. It is much older that computing, being almost as old as photography itself (century XIX). What is now trivial by using services like Google Earth, that uses satellite pictures to show aerial pictures, was being done since the forties using the technology available in those times [4]. Military aeroplanes equipped with cameras facing down took pictures of the ground that were later hand-assembled as photo mosaics. The multiple photographs were then produced by normalising the subtle variations in inclination and exposure for each negative in such a way that it could be seen as a single large picture (see Figure 1.2). By knowing the focal length of the camera and the approximate height of the photograph it could then be used for taking ground measurements.

With the advent of computers and the development of digital cameras a new field of study was born. Computer vision is the field of computer science that studies ways of extracting information from digital images computationally. It became a mandatory tool for photogrammetry, delivering better and faster results, while most of the early techniques became obsolete. Nevertheless, the newer field inherited a lot of knowledge from the earlier. Nowadays the two fields share a symbiotic relationship and are often confused.

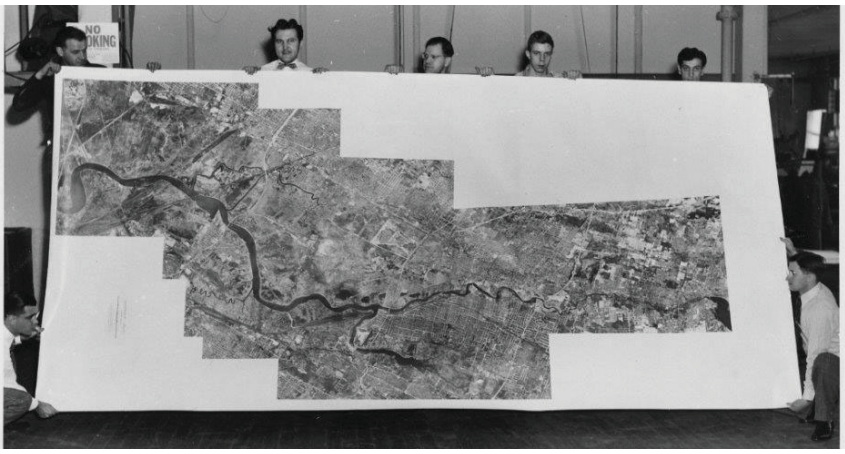


Figure 1.2: Hand-assembled photo mosaics [4].

1.2.2 Mapping and Localization

Photogrammetry has tools commonly used for topographic mapping and nowadays there are other sensors used to aid in that task.

LIDAR sensors are able to scan distances using pulsed laser rays with high accuracy at long distances, by measuring the travel time of the pulses. Three-dimensional LIDARs are able to produce a 3D cloud of points, that can then be fused with the imagery taken with cameras to recreate 3D scenes.

Another way to recreate 3D scenes is using sets of multiple cameras. Sets of two cameras are the most commonly used to extract depth information from pictures because a minimum of two points of view of the same scene is needed.

It is also possible to do mapping and localization with a single camera. However fidutial markers must be added to the scene, as described in [5] where a set of pictures was taken in a room with some markers on the walls. An important step in this procedure is the identification of markers and subsequent positioning of the camera relative to each marker.

1.2.3 Global Positioning

The GNSS (Global Navigation Satellite Systems) is available today for anyone to use, like the well-known Global Positioning System (GPS), which provides a very convenient way of getting a position. Using triangulation of signals received from satellites a position can be estimated in the form of a coordinate in the world geodetic system (WGS 84).

In situations where GNSS usage should be avoided for not being precise or fast enough, or when its usage is not reliable or not available at all (as is the case when inside a building or inside a cave), alternatives must be explored.

1.2.4 Relative Pose Estimation

In Zhang *et al.* [6] the position estimation on an autonomous underwater vehicle is done with the aid of multiple beacons.

Pan *et al.* [7] developed a solution using a single camera with the processing being done on an FPGA for pose estimation. It was made for space applications and yielded high accuracy. The used target was composed of a set of LEDs.

Solutions using letter-shaped targets

A situation commonly explored as an example is landing of a multirotor on a designated target. Bi *et al.* [8] approaches the subject with a custom coloured target and a controller running on an embedded microcontroller on-board the used drone, a Parrot AR.Drone. Their target resembles the letter ”H”, as used on helipads. However, the implementation is very specific to the quadrotor used.

The research done by Xu *et al.* [9] for the autonomous landing of a UAV on a ship uses computer vision to find a custom target shaped as a ”T”. The target is heated and the camera used is a thermal (infrared) camera. This increases the contrast of the target in the image, easing the computational workload for finding the target. The algorithm for recognising the target starts by applying a custom threshold method based on Otsu [10]. Then the edge information is extracted using Sobel's method [11]. Having the edge information and assuming only one contour is present, the Hu moments of the shape are calculated and compared with the target's reference values to verify its presence and extract its orientation.

Solutions using circular targets

Targeting aerospace applications, Wu *et al.* [12] presented a solution for pose estimation using a single camera for detecting a docking ring. They considered the projection of the docking ring onto the image as a set of concentric ellipses. The ellipse detection is done along the strategy stated by Fornaciari *et al.* [13], which presents a ”fast and effective ellipse detector” targeted for usage in smartphones. This ellipse detector extracts the edge information using the Canny edge detector [14] and then extracting and filtering arcs that are joined in clusters of possible ellipses.

Cucci [15] presents a planar target composed by concentric circles (Figure 1.3) with the addition of smaller circles inside the circular section as a coding to differentiate distinct markers following the same system. The target was designed to be accurately found by a small computer on-board of a UAV. The objective was to allow it to follow the target, which was placed on top of a van. The procedure followed by Cucci began by applying the Canny detector and then finding contours according to Suzuki *et al.* [16]. A new circle detection algorithm is presented, assuming ellipses as irregular circles. Then the code is tested for each outer circular section and ellipses are fitted to the inner and outer ellipses. As the projection of the circle centre is not the same as the centre of the projected ellipses, an accurate method for estimating the circle centre is presented. Then the pose is determined similarly to a Perspective-3-Points [17] problem (P3P).

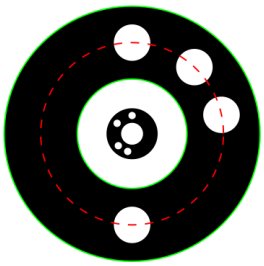


Figure 1.3: Circular target used by Cucci [15].

Other solutions

Karami *et al.* [18] made a comparative study about the performance of popular feature detectors and extractors (SIFT, SURF and ORB) at different poses. They show results that are not on par with specialised algorithms previously presented when the problem is pose estimation, being more proper for complex image matching.

Squared markers such as QR-codes may contain a lot of information, while other markers may contain only enough to distinguish among others of the same type (e.g. ArUco markers in [5]).

A project developed by Inria Lagadic, ViSP (Visual Servoing Platform) contains different solutions for tracking objects, one of them is packaged as a ROS node that, given a live stream from a camera, it returns the pose of a QR-Code.

1.3 Objectives

There are many different algorithms for relative pose estimation, some with specialised platforms (such as FPGAs). However, no solution seems to be ready to deploy with off-the-shelf items.

This dissertation has the goal of building a turnkey solution for the relative pose estimation of a target relative to a camera. With the only sensor used to locate the target being a monocular camera. The algorithm should be tested on a small board computer such as the Raspberry Pi or the Odroid as a ground proof that the algorithm can be easily fitted on a mobile robot for platooning applications.

1.4 Thesis Outline

Chapter 2 describes the kinematics background used as theoretical basis, explaining the reference frames relevant throughout this document and introducing the concepts of pose and rotation matrix. Then, in chapter 3 an overview of the computer vision notions needed to understand the followed methodology is stated, initiating the reader in image projection and contour detection.

By chapter 4 the methodology is explained, based upon the concepts introduced in the preceding chapters 2 and 3. The circular target modelling approach is introduced, and the enhanced detection algorithm proposed for target estimation is presented and compared to the original method. Finally, the

method for extracting the pose of the target is defined after explaining the case of the planar perspective problem at hand.

The set of tools used for the deployment of the marker detector is referred in chapter 5, briefly detailing the Robot Operating System operation and the cameras used.

The results obtained throughout this are stated and discussed in chapter 6, where a comparison of the algorithm to the robotic arena is presented using the latter as ground truth. Some insights on the usage of this sensor for train following are also stated here.

In chapter 7 a retrospective of the work is presented with the resulting achievements and contributions besides some follow-up possibilities as future work.

Chapter 2

Kinematics Background

2.1 Reference Frames

For this study several reference frames are considered. Reference frames are important not only for positioning, but also to understand the orientation of a frame relative to another. In this section the reference frames used throughout this work are presented in detail. All reference frames are defined in the Cartesian space while using the metre as unitary reference.

The main reference frames used in this document are the camera frame $\mathcal{F}_C$ and the target frame $\mathcal{F}_T$. The other frames used, which are very important in computer vision applications, such as the projected image frame $\mathcal{F}_I$ and its distorted and normalised counterparts. These last frames are explored ahead in this document, in section 3.2.

The camera reference frame (schematically depicted in Figure 2.1) is a frame rigidly attached to the camera body, with the z axis having a direction normal to the sensor and pointing to the camera lens. The xy plane is parallel to the sensor plane, with the y axis typically pointing downwards of the camera. Its origin O_C corresponds to the optical centre of the camera, which depends on both camera and lens specifications. For the purpose of this explanation, the optical centre can be thought of as the geometric centre of the camera lens. This topic will be recalled in chapter 3, when introducing the camera model followed as reference.

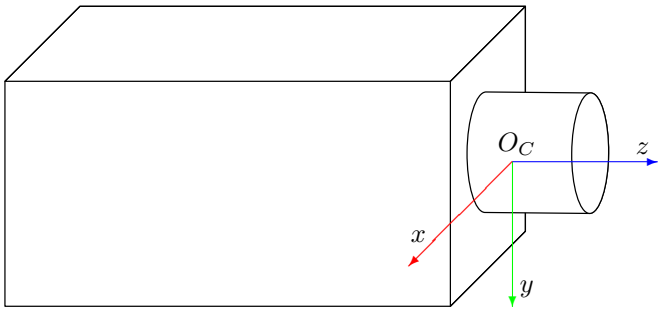


Figure 2.1: Camera frame schematic

Besides the camera frame, the other main reference frame used is the target frame $\mathcal{F}_T$. Depending on the target used, it is placed conveniently following some rules assuming a planar target:

- The origin O_T is placed at its geometrical centre;
- The plane xy is coincident with the target plane;
- The y axis should be positive to the side of the target plane considered as being the bottom;
- The x axis should be positive to the side of the target plane considered as being the right side;
- The z axis must be negative on the side the target is visible.

This set of rules guarantees that when the camera and the target frames are aligned and the target is within the field of view of the camera it can be detected without rotation, thus allowing a possible control application to nullify rotations without further transformations. Figure 2.2 represents a sketch of a generic target with its reference frame schematised above it. Section 4.2 has more details about the target used.

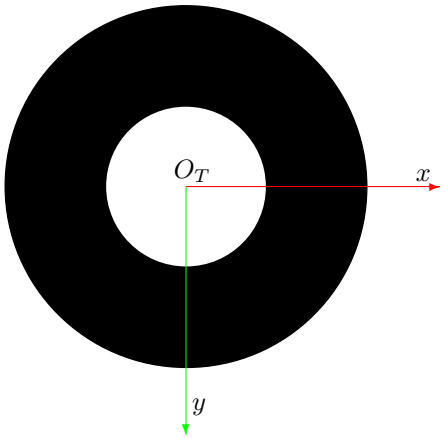


Figure 2.2: Target frame schematic

2.2 Rotation Matrix

A rotation matrix $\mathcal{R}$ is a 3×3 matrix that can be used to represent the orientation of a frame with respect to another reference frame. It is an orthogonal matrix denoted as $\mathcal{R}_B^A$ for a rotation from $\mathcal{F}_A$ to $\mathcal{F}_B$. The special case when $\mathcal{R}_B^A = \mathcal{I}_3$[1] happens when $\mathcal{F}_A$ and $\mathcal{F}_B$ share the same orientation, meaning there is no rotation from $\mathcal{F}_A$ to $\mathcal{F}_B$.

The consecutive multiplication of rotation matrices results in another rotation matrix.

$$\mathcal{R}_B^A \mathcal{R}_C^B = \mathcal{R}_C^A \tag{2.1}$$

The transpose of a rotation matrix is the same as its inverse. This transformation yields the opposite rotation.

$$\left(\mathcal{R}_B^A\right)^T = \left(\mathcal{R}_B^A\right)^{-1} = \mathcal{R}_A^B \tag{2.2}$$

[1] $\mathcal{I}_n$ denotes the identity matrix of size n.

Further along in this section, the notation introduced by (2.3) is used to introduce some inverse solutions of the rotation matrix. The elements of the rotation matrix are generally referred to as r_{ij}, where i and j represent the matrix indices of the rotation matrix.

$$\mathcal{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.3}$$

2.2.1 Roll-Pitch-Yaw

The rotation matrices describing rotations about a single axis are named elementary rotation matrices. These can be used to build rotation matrices. When a rotation matrix is built to transform the orientation of a vector from a reference frame to another, it is made by multiplying elementary rotations one by one until the frames are aligned. In a three dimensional space at most 3 axial rotations may be needed to reach any orientation. These subsequent rotations may be defined relative to the frame being moved or relative to the static (original) frame.

The three possible axial rotations in three dimensions are stated as follows:

$$\mathcal{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{2.4}$$

$$\mathcal{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{2.5}$$

$$\mathcal{R}_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

In the aeronautical field it is common to use the roll-pitch-yaw set of angles for orientation and reference. The yaw is measured by a compass and the roll and pitch have its dedicated indicator: the Primary Flight Display (PFD), as shown in Figure 2.3. To aid the pilots job, the PFD shows figuratively the roll and pitch angles. The yaw is displayed by a compass.

The rotations roll-pitch-yaw correspond to the rotations in the axes (x, y, z) relative to a fixed frame, which can applied through a rotation matrix.

RPY Rotation Matrix

This set of Euler angles may be defined as a XYZ rotation around its respective (fixed frame) axes by the angles $(\psi, \vartheta, \varphi)$. This is equivalent to a sequence of rotations Z, Y, X about the axes of the rotating frame [20], which is what matters when joining the elementary axial rotations.

Figure 2.3: Primary Flight Display of a Boeing 737-800 [19].

$$\mathcal{R}_z(\varphi)\mathcal{R}_y(\vartheta)\mathcal{R}_x(\psi) = \begin{bmatrix} c_\varphi c_\vartheta & c_\varphi s_\vartheta s_\psi - s_\varphi c_\psi & c_\varphi s_\vartheta c_\psi + s_\varphi s_\psi \\ s_\varphi c_\vartheta & s_\varphi s_\vartheta s_\psi + c_\varphi c_\psi & s_\varphi s_\vartheta c_\psi - c_\varphi s_\psi \\ -s_\vartheta & c_\vartheta s_\psi & c_\vartheta c_\psi \end{bmatrix} \tag{2.7}$$

The expressions c_ϑ and s_ϑ in equation (2.7) are, respectively, short for cosine and sine of the angles indicated in subscript. In this matrix, the angle φ represents the roll as a rotation about z, the angle ϑ is the pitch and the angle ψ is the yaw.

Inverse solution of a rotation matrix to RPY angles

The inverse solution of a rotation matrix into the XYZ Euler angles is computed using the function of the inverse tangent that takes into account the signs of both the numerator and the denominator to place the returned angle in the correct quadrant. This function is known as $\mathrm{atan2}(y, x)$ as the arc-tangent of y/x with two inputs: numerator and denominator.

For a set of angles in the range $[-\pi/2, \pi/2]$ the inverse solution is given by (2.8). This solution can be inferred from (2.7).

$$\begin{aligned} \varphi &= \mathrm{atan2}(r_{21}, r_{11}) \\ \vartheta &= \mathrm{atan2}(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \\ \psi &= \mathrm{atan2}(r_{32}, r_{33}) \end{aligned} \tag{2.8}$$

2.2.2 Unit Quaternion

The unit quaternion is a way of representing an orientation that brings a more proper tool for some usage scenarios in robotics relatively to Euler angles.

The unit quaternion $\mathcal{Q} = \{\eta, \varepsilon\}$ is a convenient way of expressing an orientation using four parameters [20]. It is composed by the scalar part defined by η and the vector part defined by $\boldsymbol{\varepsilon} = (\varepsilon_x, \varepsilon_y, \varepsilon_z)^T$.

The fact that it is a unit quaternion implies the restriction in (2.9) applies.

$$\eta^2 + \varepsilon_x^2 + \varepsilon_y^2 + \varepsilon_z^2 = 1 \tag{2.9}$$

Given a unit quaternion $\mathcal{Q}$, it takes the form of a rotation matrix $\mathcal{R}$ according to (2.10)

$$\mathcal{R}(\eta, \boldsymbol{\varepsilon}) = \begin{bmatrix} 2(\eta^2 + \varepsilon_x^2) - 1 & 2(\varepsilon_x\varepsilon_y - \eta\varepsilon_z) & 2(\varepsilon_x\varepsilon_z + \eta\varepsilon_y) \\ 2(\varepsilon_x\varepsilon_y + \eta\varepsilon_z) & 2(\eta^2 + \varepsilon_y^2) - 1 & 2(\varepsilon_y\varepsilon_z - \eta\varepsilon_x) \\ 2(\varepsilon_x\varepsilon_z - \eta\varepsilon_y) & 2(\varepsilon_y\varepsilon_z + \eta\varepsilon_x) & 2(\eta^2 + \varepsilon_z^2) - 1 \end{bmatrix} \tag{2.10}$$

Inverse solution of a rotation matrix to unit quaternion

Using a similar reasoning to get the unit quaternion from a rotation matrix as followed for the case of RPY angles, the inverse solution of a rotation matrix to a unit quaternion is inferred from its direct solution (2.10).

In this case another function is used, the function $\mathrm{sign}(x)$ returns the value 1 whenever $x \geq 0$ and returns -1 otherwise.

$$\begin{aligned} \eta &= \frac{\sqrt{r_{11} + r_{22} + r_{33} + 1}}{2} \\ \varepsilon &= \frac{1}{2} \begin{bmatrix} \mathrm{sign}(r_{32} - r_{23})\sqrt{r_{11} - r_{22} - r_{33} + 1} \\ \mathrm{sign}(r_{13} - r_{31})\sqrt{r_{22} - r_{33} - r_{11} + 1} \\ \mathrm{sign}(r_{21} - r_{12})\sqrt{r_{33} - r_{11} - r_{22} + 1} \end{bmatrix} \end{aligned} \tag{2.11}$$

2.3 Pose

The term pose comprises the position and orientation of a rigid body. To fully define the placement of a rigid body relative to another rigid body, a position and an orientation are needed. The bodies in study for relative placement are the camera and the target. As rigid bodies, their poses may be defined by their respective frames.

2.3.1 Position

The position of a point P in a reference frame $\mathcal{F}_A$ can be defined by a vector starting at its origin O_A and ending at point P, this vector can be represented as ${}^{A}P$. For three dimensional spaces three values are needed to define a vector (e.g. (x, y, z) in a Cartesian coordinate system).

To define the position of $\mathcal{F}_B$ in $\mathcal{F}_A$ the point used for reference is its origin O_B. When referring to the origin of $\mathcal{F}_B$ in $\mathcal{F}_A$ the notation used is O_B^A, which corresponds to the translation vector of from $\mathcal{F}_A$ to $\mathcal{F}_B$ in the coordinate system of $\mathcal{F}_A$. The notation O_B by default has the same meaning as O_B^B.

2.3.2 Orientation

Following a translation from $\mathcal{F}_A$ to $\mathcal{F}_B$ the two frames may not coincide, this is the case when those two frames have different orientations.

Often the orientation is referred to as attitude. It may defined by Euler angles, by a 3×3 rotation matrix or as a unit quaternion.

2.4 Transformations

Given any point P with its position known in $\mathcal{F}_A$ as P^A, it can be defined in $\mathcal{F}_B$ provided that the pose of $\mathcal{F}_B$ in $\mathcal{F}_A$ is fully known. This means that $\mathcal{R}_A^B$ and O_A^B must be known in order to perform the transformation. If instead of $\mathcal{R}_A^B$, the matrix $\mathcal{R}_B^A$ is known, review equation (2.2).

In case both frames have the same orientation, it is a simple translation performed by a vector addition.

$$P^B = P^A + O_A^B \tag{2.12}$$

On the other side, if the origin is common ${}^BO_A = 0$, only a rotation is applied by multiplying the rotation matrix by the point.

$$P^B = \mathcal{R}_A^B P^A \tag{2.13}$$

And if O_B^A is known, it can be rotated like any other point to calculate ${}^BO_A$ using the knowledge that the origin of a frame is a null vector when defined in its own reference frame (e.g. $O_B = \mathbf{0}$).

$$O_B = \mathcal{R}_A^B O_B^A + O_A^B \iff O_A^B = -\mathcal{R}_A^B O_B^A \tag{2.14}$$

In short this means that the transformation of point O_A^B to the frame $\mathcal{F}_B$ can be calculated as the rotation and translation of its defining vector:

$$P^B = \mathcal{R}_A^B P^A + O_A^B \tag{2.15}$$

To clear the notation, a new matrix, called transformation matrix, is introduced as $\mathcal{T}$. It is a 4×4 matrix containing the rotation matrix on the upper left corner, a null vector under it, the translation vector on the upper right corner and a 1 under it.

$$\mathcal{T}_A^B = \begin{bmatrix} \mathcal{R}_A^B & O_A^B \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.16}$$

$$\begin{bmatrix} P^B \\ 1 \end{bmatrix} = \begin{bmatrix} \mathcal{R}_A^B & O_A^B \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} P^A \\ 1 \end{bmatrix} \tag{2.17}$$

To make use of the transformation matrix, the notation $\tilde{P} = (p_x, p_y, p_z, 1)$ can be created be adding fourth unitary component to a vector $P = (p_x, p_y, p_z)$. It allows a compact notation of transformations to be employed, without loss of correctness or meaning.

The variables in equation (2.15) mean the same as in (2.17), allowing the shorter notation introduced in equation (2.18).

$$\tilde{P}^B = \mathcal{T}_A^B \tilde{P}^A \tag{2.18}$$

Chapter 3

Computer Vision Background

Computer vision is the field of science that studies ways to extract information from digital images computationally. There are many different ways to approach a problem, depending on the specific task at hand.

Countless applications for computer vision are possible. In industrial environments computer vision is commonly used for quality assurance of fabricated parts on an assembly line. Mainstream computer mouses use optical flow to track its movement and do it using a dedicated integrated circuit. The smartphone application Snapchat processes the camera feed in real time tracking the users' faces to place cartoons on top of them. The app Seene[1] allowed 3D reconstruction from multiple shots instantly, but it is no longer available and the company is exploring other ways to monetise the technology.

3.1 General Overview

This chapter introduces the base knowledge of computer vision needed to explain subsequent choices. Here you can find a baseline explanation of the reference frames used, the pinhole camera model, camera calibration details, the importance of the homography matrix for pose estimation, pre-processing algorithms used, a few feature detection and extraction algorithms and how elliptic and circular shapes can be found in an image.

To successfully achieve the objective of target following through computer vision one needs to find the position and orientation of the target relative to the camera. Typically this is done by extracting and classifying features from a reference image and a live camera feed. They are then compared by matching the extracted features on the two images to use them as reference points for the estimation of a perspective projection matrix between both target planes.

[1] `https://seene.co/`

3.2 Pinhole Camera Model

The pinhole camera is a way to create a projection of a real world scene to a plane[21]. A pinhole is a hole so small that only light rays coming directly at it can pass through it.

In a box with no other source of light but a pinhole on one of its faces, the opposite face will get a projection of the scene as seen through the pinhole – that is the image projected onto the image plane. Taking Figure 3.1 as an example of the projection of a candle through a pinhole. The image projected onto the image plane is inverted, but in most situations a virtual image can be taken into account. The virtual image is not inverted and is placed on an imaginary plane parallel to the image plane and at the same distance from the pinhole on the opposite side of the pinhole plane.

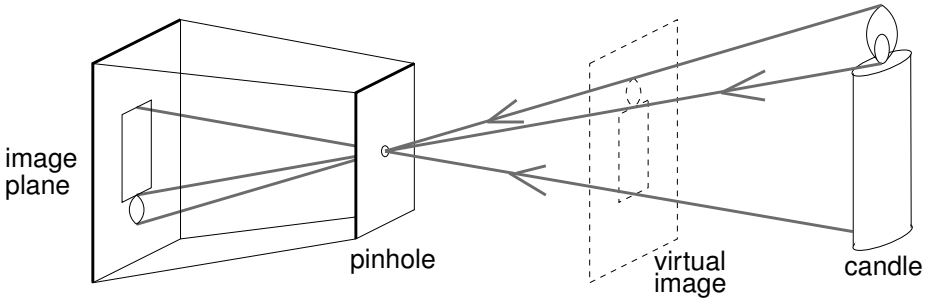


Figure 3.1: Candle projected through a pinhole. Adapted from [21].

In the case of classic film cameras, the film is on the plane opposite to the pinhole (image plane) and the shutter covers the pinhole letting or shielding the light from burning the film. In the place of the shutter, modern cameras use a moving mirror to quickly switch from the light going to the viewfinder to burning the film. With digital cameras there is a camera sensor in the image plane which is an array of coloured light sensors. Typical reflex cameras use the same mirror technique to protect the delicate sensor, however cheaper cameras like webcams and the ones equipping smartphones have cheaper sensors that are much smaller and do not need the same kind of protection as professional grade cameras.

3.2.1 Perspective Projection

To extract positioning information of objects from a camera it is important to know mathematically how an image is formed, therefore the pinhole camera model is presented here for convenience, according to Forsyth *et al.* [21].

Note that Figure 3.2 is a 2D representation of the perspective projection in a pinhole camera. The point O_C represents the pinhole position and is also the origin of the camera frame ($\mathcal{F}_C$, which is defined by the axes x, y and z). The axis z is the optical axis.

The 3D scene is projected onto the projection plane ($\mathcal{F}_P$), defined by the axes x' and y'. It is a translation of $\mathcal{F}_C$ by $-f$ along the z axis with its origin – the centre of projection – being O_P. The focal length is always positive ($f > 0$).

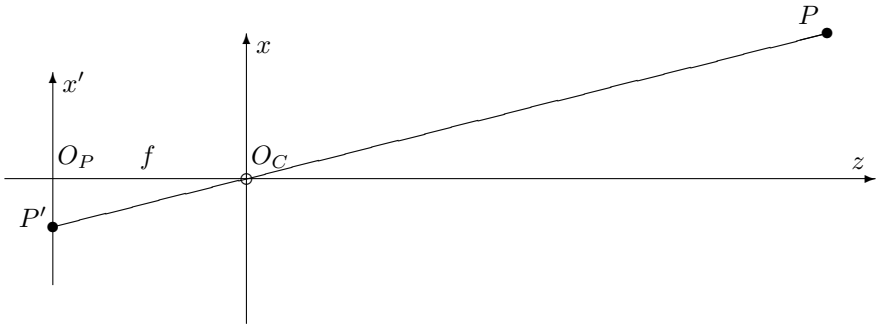


Figure 3.2: Pinhole camera reference frames

Point $P^C = (x, y, z)$ is a point in space with $z > 0$ projected as $P'^C = (-x', -y', -f)$. As P, P' and O_C are collinear: equation (3.1) can be written for some real number λ.

$$\overrightarrow{O_C P'} = \lambda \overrightarrow{O_C P} \tag{3.1}$$

Which leads to the equivalence shown in (3.2).

$$\begin{cases} x' = \lambda x \\ y' = \lambda y \\ f = \lambda z \end{cases} \iff \lambda = \frac{x'}{x} = \frac{y'}{y} = \frac{f}{z} \tag{3.2}$$

Resulting in the perspective projection formula in (3.3).

$$\begin{cases} x' = f\dfrac{x}{z} \\ \\ y' = f\dfrac{y}{z} \end{cases} \tag{3.3}$$

3.2.2 Radial Distortion

The perspective projection of the pinhole camera is a straightforward model that allows a simple mathematical formulation of how a camera works. However, for accurate measurements it is not enough. Real cameras have finite sized apertures, not infinitesimal holes as the depicted pinhole. This allows more light going into it, thus producing clearer pictures.

In order to keep a camera focused while having a larger aperture, lenses must be used. The main downside of using lenses is that it produces a distortion which must be accounted for in scenarios where world measurements need to be taken.

As shown in Figure 3.3, the radial distortion is highly dependent on the distance to its centre – the centre of radial distortion – which is the the principal point, where the optical axis crosses the picture. To mitigate the effects of distortion, it can be modelled to improve the results on the full field of view of each picture.

As was stated by Zhang in [23], the distortion model used is described in (3.4). The coordinates (u, v) correspond to the ideal image as per the pinhole camera model, $(\tilde{u}, \tilde{v})$ are the real – distorted – image coordinates and r is the distance of the point (u, v) to the centre of radial distortion – the principal

Figure 3.3: Common types of radial distortion[22].

point (u_0, v_0). The parameters k_1 and k_2 are the two first terms of radial distortion and no others were modelled as Zhang notes it is more numerically stable and sufficient for a camera.

$$\begin{cases} \tilde{u} = u + (u - u_0)(k_1 r^2 + k_2 r^4) \\ \tilde{v} = v + (v - v_0)(k_1 r^2 + k_2 r^4) \end{cases} \tag{3.4}$$

3.2.3 Intrinsic Camera Parameters

Due to the physical characteristics of the cameras, its sensors and its lens: an image captured by a camera does not produce exactly the same projection as described in equation (3.3) neither is it read in world distance units (like metres). Moreover digital sensors have a finite resolution, thus some loss of information is expected while discretising the projection into pixels.

The image sensor is the canvas where the projection takes place on a digital camera. It is safe to assume that it is a rectangular array of pixels on a plane. As an image sensor has a certain resolution (in pixels), its pixels also have a fixed size. Commonly the pixels are not square, given its dimensions are a by b it is acceptable to write down its units in $m/pixel$. The size of the pixels are the conversion factor from the world units in the projection plane (x', y') to pixels as shown in equation (3.5), considering the units of the image plane (u, v) are pixels.

$$\begin{cases} u = x'/a = \dfrac{f}{a}\dfrac{x}{z} \\ \\ u = y'/b = \dfrac{f}{b}\dfrac{y}{z} \end{cases} \tag{3.5}$$

Following the fact that digital images are processed by computers, its origin is not in the centre of the image because computer indexing of vectors and matrices usually starts at index 0 (e.g. C++, Python) or index 1 (e.g. MATLAB), having the first index (e.g. $(0, 0)$) on a corner of the image. Besides, due to manufacturing errors, the optical centre of the camera is frequently a bit shifted from the centre of the sensor. Assuming the image is shifted (u_0, v_0), in pixels, from the projection model origin: the conversion from the projected plane to the captured image can be calculated as (3.6).

$$\begin{cases} u = \dfrac{f}{a}\dfrac{x}{z} + u_0 \\ \\ v = \dfrac{f}{b}\dfrac{y}{z} + v_0 \end{cases} \tag{3.6}$$

To simplify the notation the parameters $\alpha = \dfrac{f}{a}$ and $\beta = \dfrac{f}{b}$ are used as magnification parameters resulting in (3.7).

$$\begin{cases} u = \alpha\dfrac{x}{z} + u_0 \\ \\ v = \beta\dfrac{y}{z} + v_0 \end{cases} \tag{3.7}$$

The intrinsic camera parameters add physical meaning to the image captured by the retina of the camera. Another parameter – the skew θ – exists but is ignored since it shows no impact on preliminary data.

The intrinsic camera parameters, often called just camera parameters: $(\alpha, \beta, u_0, v_0)$ can now be assembled in matrix form, called the camera matrix $\mathcal{K}$, containing the four intrinsic camera parameters presented.

$$\mathcal{K} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

The projection of a point $\mathbf{P} = (x, y, z, 1)^T$ in the camera frame, to a point $\mathbf{p} = (u, v, 1)^T$ in the image can now be expressed as stated by eq. (3.9), taking $\mathcal{M}$ as the perspective projection matrix.

$$\mathbf{p} = \frac{1}{z}\mathcal{M}\mathbf{P}, \qquad \text{where} \qquad \mathcal{M} = \begin{pmatrix} \mathcal{K} & \mathbf{0} \end{pmatrix} \tag{3.9}$$

3.2.4 Extrinsic Camera Parameters

The general case in which the camera frame ($\mathcal{F}_C$) is not the same as the world frame ($\mathcal{F}_W$) a coordinate system change must happen according to the general rule:

$$\begin{bmatrix} P^C \\ 1 \end{bmatrix} = \begin{bmatrix} \mathcal{R}_W^C & O_W^C \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} P^W \\ 1 \end{bmatrix} \tag{3.10}$$

When mixing the intrinsic and the extrinsic parameters in the perspective projection matrix $\mathcal{M}$ to map points in the world frame to points in the image, the result is closely related to a generic form of (3.9). However, the point $\mathbf{P}$ is now defined in $\mathcal{F}_W$ and $\mathcal{M}$ adopts the form denoted by equation (3.11), where $\mathbf{r}_i$ is the *i-th* row of the rotation matrix $\mathcal{R}$ and $\mathbf{t} = (t_x, t_y, t_z)$ is the translation vector.

$$\mathcal{M} = \mathcal{K}\begin{pmatrix} \mathcal{R} & \mathbf{t} \end{pmatrix} = \begin{pmatrix} \alpha\mathbf{r}_1^T - \alpha\mathbf{r}_2^T + u_0\mathbf{r}_3^T & \alpha t_x - \alpha t_y + u_0 t_z \\ \beta\mathbf{r}_2^T + v_0\mathbf{r}_3^T & \beta t_y + v_0 t_z \\ \mathbf{r}_3^T & t_z \end{pmatrix} \tag{3.11}$$

3.3 Image Feature Detection

Features are pertinent distinguishable areas or points in an image, much like the reader might visually identify someone by the colour and shape of the eyes and the face. In computer vision the good features depend on the classifier used and the best classifier depends heavily on the task in hand. Frequently the features are corners [24], corner-based features [25], or oriented gradients as those features are found by well described mathematical algorithms and have distinct properties (e.g. the inside of a block of solid colour itself does not contain much information, but its corners and edges might). Some commonly used feature detectors are: Harris Corner Detector, Histogram of Oriented Gradients (HoG) and Features from Accelerated Segment Test.

Popular feature classifiers like Scale-Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF) are robust enough for general applications. However, for tasks that require real-time speed they are too slow to use for every frame and for commercial usage a licence is needed because both are patented.

3.4 Camera Calibration

Often the intrinsic parameters and distortion terms of a camera are not available from the manufacturer sheet or the available information might not be reliable. In those cases it is convenient to perform the camera calibration to recover its parameters. An easy technique of calibration is described by Zhang in [23].

The process of camera calibration does not mean any physical change to the camera itself. All that is needed is a series of pictures of a calibration target (like the chessboard without distortion in Figure 3.3) in varying poses. The calibration targets are chosen by having a set of points easy to be reliably detected and at known world distances. The most commonly used calibration target is the chessboard-like target, because its features are on the corners of the inner squares. Other targets may be used for calibration, like the square grid of circles in Figure 3.4, whose detected points are the centres of the circles. The detection of the target points in each picture for calibration is all that is needed from each image and results in an array of points for each image.

3.5 Ellipse Notations

Ellipses can be seen as the shapes that arise from the projection of a circle on an image plane, where a circle is a special case of ellipse.

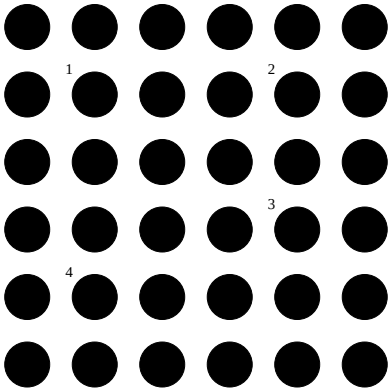


Figure 3.4: A symmetrical circle pattern proposed as calibration target [26].

The size and shape of an ellipse are defined by two values, the major axis and the minor axis. The major axis is the distance between the two furthest apart opposite points on the circumference of the shape. Opposite points on the circumference of an ellipse is the pair of points that are intersected by a straight line crossing the ellipse through its centre. The minor axis is the closest distance between two opposite points on the circumference of an ellipse.

$$\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} = 1 \tag{3.12}$$

Equation (3.12) is the Cartesian equation of an ellipse centred on (x_0, y_0) without rotation. The parameters a and b are the semi-major and semi-minor axes, which are half the major and minor axes. In the particular case of a circle, the values of a and b are equal to its radius.

When an ellipse is rotated by an angle θ the formulation becomes:

$$\frac{((x-x_0)\cos\theta + (y-y_0)\sin\theta)^2}{a^2} + \frac{((x-x_0)\sin\theta - (y-y_0)\cos\theta)^2}{b^2} = 1 \tag{3.13}$$

In a different approach, ellipses can also be denoted as conic sections. Conic sections can be parabolas, ellipses, circles or hyperbolas and are defined by a set of coefficients to get the following Cartesian form:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{3.14}$$

If the ellipse parameters a, b, x_0, y_0 and θ are known, then its parameters for the conic section form can be analytically calculated.

$$
\begin{aligned}
A &= (\cos\theta/a)^2 + (\sin\theta/b)^2 \\
B &= 2\cos\theta\sin\theta(1/a^2 - 1/b^2) \\
C &= (\cos\theta/b)^2 + (\sin\theta/a)^2 \\
D &= 2\sin\theta(y_0\cos\theta - x_0\sin\theta)/b^2 - 2\cos\theta(x_0\cos\theta + y_0\sin\theta)/a^2 \\
E &= -2\cos\theta(y_0\cos\theta - x_0\sin\theta)/b^2 - 2\sin\theta(x_0\cos\theta + y_0\sin\theta)/a^2 \\
F &= \left(\frac{x_0\cos\theta + y_0\sin\theta}{a}\right)^2 + \left(\frac{y_0\cos\theta - x_0\sin\theta}{b}\right)^2
\end{aligned}
\tag{3.15}
$$

It can also be represented in matrix form [27] expressed in (3.16) and used as (3.17) with the same meaning as (3.14), giving that $\tilde{\mathbf{x}}^T = [x \quad y \quad 1]$.

$$
\mathcal{Q} = \begin{bmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{bmatrix} \tag{3.16}
$$

$$
\tilde{\mathbf{x}}^T \mathcal{Q} \tilde{\mathbf{x}} = \mathbf{0} \tag{3.17}
$$

3.6 Image Processing Tools

Pre-processing an image correctly is a core step for having success on obtaining the best results in computer vision. In situations where the shapes are present in an image, the object of study and textures can be discarded. It might be interesting to explore algorithms that try to emphasise shapes. One way to do it is through thresholding algorithms that return monochromatic images.

3.6.1 Colour Map Conversion

The standard colour map used for defining digital pictures uses a primary colour addition model. The intensity of each of the three primary colours – Red, Green and Blue – is used to refer the colour of each pixel. This is known as the RGB colour map. Typically a byte is used for each colour channel, when that is the case it is often referred to as 24-bit colour.

Usually, computer vision algorithms do not take into account multiple colour channels at once. To overcome this it might be possible to run it independently for each channel and then to join the multiple results, but more commonly the multiple colour channels are joined in one single channel. The most common way of converting a RGB colour map into a single channel colour map is simply by averaging the three channels into one. This single channel colour map can be referred to as greyscale, intensity or value.

3.6.2 Finding Contours

Once an image is binarized, its contours can be searched more easily. Suzuki *et al.* described an efficient algorithm for finding contours in binary images that focuses on following borders[16]. Borders are external contours, i.e. contours not encircled by other contours. To find internal contours the algorithm may be rerun inside the detected contours.

At the beginning, it scans the image horizontally for the first pixel that has a value different from the previous, marking it as the start of a border. Then it searches clockwise around the last detected pixel in the border for the next pixel until the contour is closed by finding the starting pixel, if it goes out of the frame that contour is discarded. Then it goes on looking for the next border on the horizontal opposite side of the last detected border. The original algorithm also contains a way to efficiently skip previously detected contours, improving speed of scanning and guaranteeing the uniqueness of each contour.

Canny Edge Detector

The method described by Canny [14] is an efficient algorithm for edge detection, oriented for computational applications. The Canny edge detector is a step detector for images that use adaptive thresholding to avoid noise on the contours.

This edge detector goes through five steps to return a binary image containing only the detected edges. The first step applies a Gaussian filter which removes most of the noise while maintaining the well contrasted shapes. Squares are softly smoothed, but round shapes like ellipses and circles keep their shapes. Then computes the gradients of 2 by 2 squares in the image. Those gradients are then compared with the neighbours for validation with a global threshold. Once a neighbour belongs to an edge its threshold is lowered to follow along that edge. This allows good robustness of the algorithm in varying lightning conditions.

After its publication, the Canny method has been studied for improvements with many publications following it, always maintaining the core idea.

Chapter 4

Methodology for Target Pose Estimation

This chapter proposes a solution for the stated problem of estimating the pose of a planar target. This solution is better for real-time pose estimation than methods based on feature descriptors like SIFT or SURF by using a custom algorithm optimised for the main goal.

The proposed method is based on the solution presented by Cucci [15] which showed very promising results that could be applied for the objectives in hand. Both methods make use of the same family of targets, these are composed primarily by a pair of concentric circles.

This method may be decomposed into two steps. Firstly, an image is processed in order to extract clusters of pairs of concentric circles, which represent possible targets. Secondly, for each cluster its code is verified until it matches the reference code. If it passes the verification, the pose is calculated. A brief overview of the process is summarised in Figure 4.1.

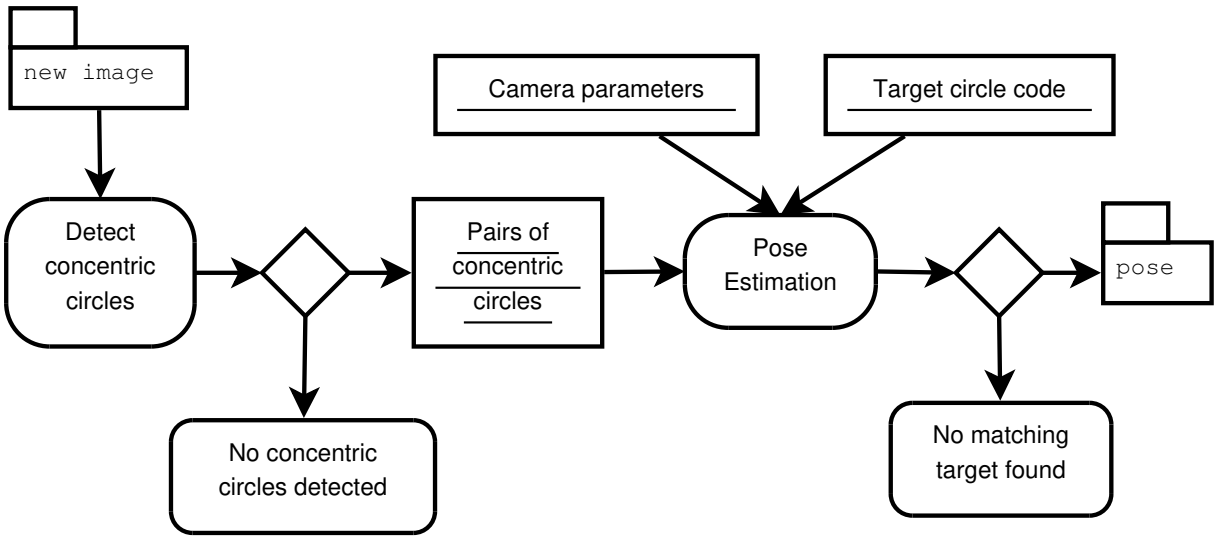


Figure 4.1: Brief summary of the algorithm

4.1 Algorithm Overview

The algorithm for detecting the target can be partitioned in 5 stages. In short, those stages are:

- Image preprocessing
- Finding contours in image
- Detecting concentric circles
- Looking for the right target
- Extracting the pose of the target

The schema in Figure 4.2 places the mentioned stages in its respective locations of the algorithm, based on the summary of Figure 4.1.

The image preprocessing prepares the image for the contour scan. A pair of concentric circles composes the primary shape of any possible target of the same family. The circles are detected from the contours as ellipses. This is because when circles are projected onto planes (such as the camera retina) they take elliptic shapes.

The method used to confirm that a certain pair of concentric circles matches the right target is developed in section 4.2. Once the signal matches the one being looked for its pose is extracted.

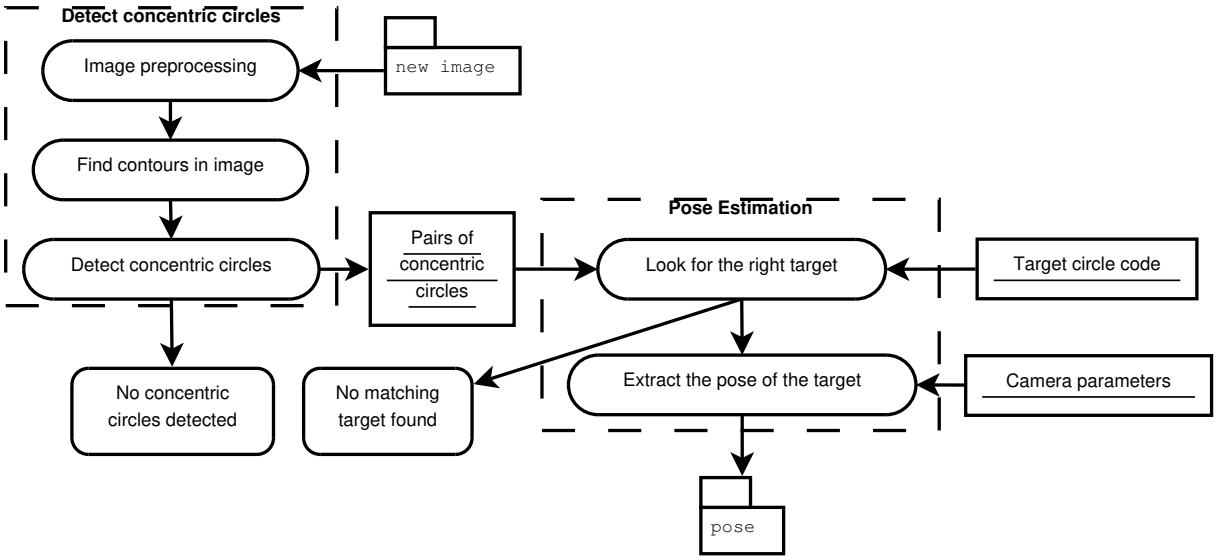
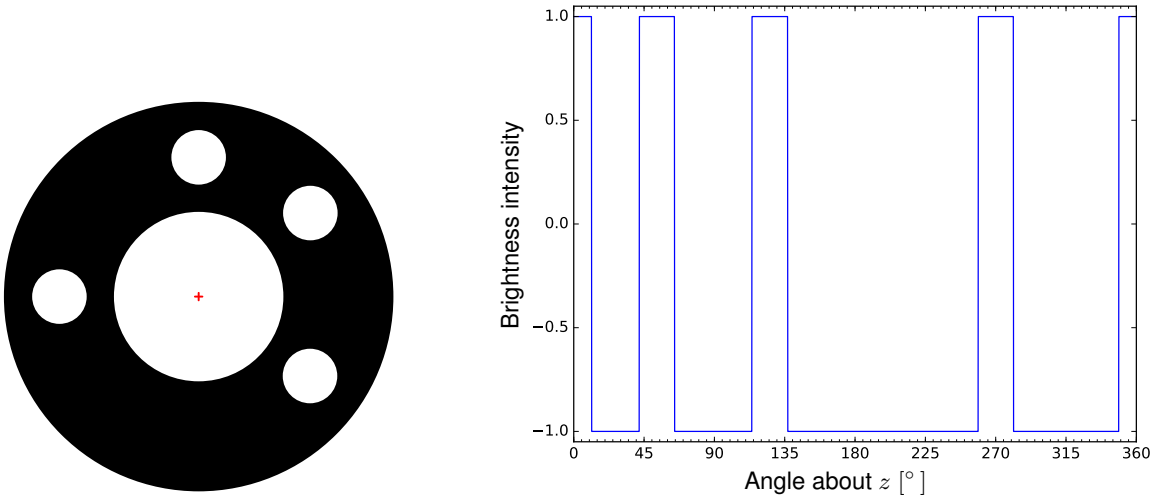


Figure 4.2: Overview of the algorithm

4.2 Planar Target

The target used is based on Cucci's target. It can be printed with black ink on white paper. Its black region is composed by a black circle with a smaller concentric white circle. As can be seen in Figure 4.3(a), in the black region there are some smaller white circles. Those smaller circles are used to create distinguishable coded targets that can be recognised by the method stated next. The smaller circles forming the code are all centred on top of an imaginary circumference concentric with the target. The intensity of the pixels along that circumference can be read as a signal, taking the value -1 for the

darkest pixels and the value 1 for the brightest, as shown in Figure 4.3(b). This reduction to a bar-code like uni-dimensional circle-code makes it easier to match the code.



(a) Target (b) Signal

Figure 4.3: A target used on the proposed approach.

To match the cluster of concentric circles with the reference target, the signal is extracted from a mean circumference between the inner and the outer ellipses, that crosses the centre of the circles composing the code. The reference signal is repeated to emulate a periodic signal. Then the correlation between the extracted signal and the reference signal is calculated. The signal is rotated so that it starts at the peak correlation value. Afterwards the absolute error is computed and if is under a reasonable threshold the correct target should have been identified. The lag at the peak correlation value is directly related to the rotation from the camera z axis to the target z axis.

4.3 Image Preprocessing

In order to find the desired target efficiently, the algorithm must find the shapes that look like the target with the least effort.

Instead of using the Canny edge detector, an adaptive thresholding could be used. However, there is no thresholding technique delivering results as good as Canny with changing light conditions. The Canny edge detector is a fast and robust algorithm that provides a binary image with edges highlighted in white and everything else in black by taking a greyscale image as input (as an example). In the case that the image provided is coloured, it must be converted into a single channel image or else a single colour channel can be used.

(a) Colour image (b) Grey image (c) Edge image

Figure 4.4: Preprocessing steps

This dynamic is pictured in Figure 4.4, where the coloured image – in 4.4(a), as taken from the camera – is converted into a greyscale image by taking the mean of the three colour channels – red, green and blue. The greyscale image in 4.4(b) is then fed to the edge detector algorithm that outputs 4.4(c) (displayed with inverted colours).

4.4 Binary Ellipse Classifier

To detect the target in an image, its contour must be successfully classified as an ellipse.

If this classification is not restrictive enough by allowing too many false positives, then too many shapes that are not the target will have its circle code checked without needing. Leading to a waste of resources.

In turn, if the classification is too strict resulting in too many false negatives it results in opportunities of detection lost, limiting the usability of the algorithm.

4.4.1 Description

When analysing a contour to detect if is elliptic, or measure its “ellipticness”, there are different methods. The method proposed by Cucci classifies too few true positives.

The classifier proposed is based on the comparison of areas. The first area is the contour area A_c, which is the number of pixels inside the contour. The second area is the area of the ellipse A_e inscribed on the rotated rectangle of minimum area that contains the full contour. If it is the contour of a perfect ellipse the difference of the two areas is zero. In order to tolerate a finite resolution and a reasonably low lens distortion, a relative error should be tolerated.

$$\frac{A_c - A_e}{A_e} < e_{max} \tag{4.1}$$

The tolerance e_{max} was set to 0.11 to allow the detection of moving targets.

4.4.2 Comparison

To compare the enhanced method proposed with the one originally employed by Cucci, performance evaluation tests were conducted, to assess the ability of classification of elliptic shapes of both algorithms.

As no suitable data-set was found for validation, one was made by creating two sample images containing 1635 shapes each. The first sample image contains ellipses with sizes starting from 15 pixel as major axis, minor/major ratios ranging from 0.15 to 0.95 and rotations up to $162°$ in steps of $18°$. The shapes in the second sample image have the same dimensions as the first, but instead of ellipses, rectangles were drawn.

Table 4.1: Test results of the ellipse classifiers

Cucci		Actual	
		True	False
Result	Positive	479	0
	Negative	1635	1156

Proposed		Actual	
		True	False
Result	Positive	1391	0
	Negative	1635	244

In Table 4.1, the values in the "Result" rows represent shapes that were classified positively as ellipses or negatively as not ellipses, as a result of each classifier; the values in the "Actual" columns display the amount of shapes whose results were truly or falsely classified.

After fine tuning the threshold parameter to get the best results for both classifiers, the results shown in Table 4.1 were obtained, giving some evidence that the proposed classifier should be more reliable by failing to classify ellipses less often.

To quantify the quality of each classifier a performance measure is introduced. Accuracy is a quantity that can be directly compared in this situation, because the results were obtained using the same input data and also because there are as many positive input data as negative input data.

$$\text{Accuracy} = \frac{\text{total amount of truly classified data}}{\text{total amount of classified data}} \tag{4.2}$$

The method proposed by Cucci obtained an accuracy of $64,6\%$ and the method proposed here demonstrated an accuracy of $92,5\%$. Note that a blind classifier would have an accuracy of around 50%, so it is a large increase in classification performance.

In addition, the proposed approach has a considerable advantage in regard to computational efficiency. Comparing both solutions in respect to computing time, the total time for classifying with the proposed classifier was $122\,\text{ms}$ and the Cucci method took $570\,\text{ms}$, nearly five fold as much.

4.5 Concentric circles detector

To detect a shape based on its contour, the first thing to do is to extract the contours from an image. Then, we must look up for ellipses that may be the projection of the outer circle of the target and verify is it contains an inner central contour.

4.5.1 Hierarchy tree

For convenience, while identifying contours it is possible to arrange a tree of information stating inside what contour is each contour segmenting the contours in levels. The outer contours in an image are all part of the trunk of the tree, it can be called level zero.

When a contour A is encircled by a contour B, it is said that A is a child of B and, reciprocally, that B is the parent of A. In Figure 4.5 a target with the contours highlighted in red and green is drawn. The only top level contour is the outer contour (in red), which is parent of the green contours. Typical pictures have more contours (check Figure 4.4).

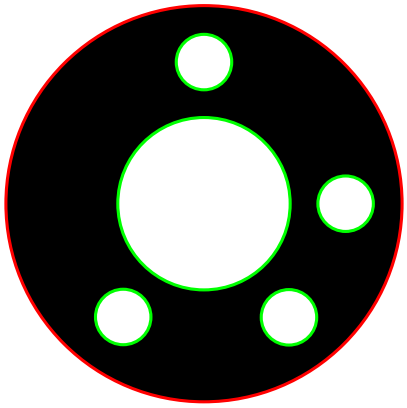


Figure 4.5: Target with contours coloured.

The most explicit advantage of making use of a hierarchy tree is that when searching for an inner concentric circle, one may look for it only amongst the inner levels of the outer circle. This brings a significant performance boost when the scene is cluttered with contours.

4.5.2 Detection algorithm

The algorithm for clustering concentric circles in pairs of inner and outer circle is summarised in Figure 4.6. It is a recursive algorithm, as it calls itself into inner hierarchic levels. This algorithm can reach as many recursive levels as there are levels of hierarchy of contours minus one, as if a contour does not contain any inner contour it cannot be the outer contour limiting the ellipse.

It starts with the index of the first contour, which is zero and is in the base level. The size of contour is tested, and if it has a perimeter under 30 pixels it is considered too small to be readable. Otherwise its proportions are tested, if the smallest length is smaller than 15 pixels it is too thin, being considered oddly shaped. If it has an acceptable thickness and size it gets tested whether it is an elliptic shape. In case it is not considered an elliptical shape the algorithm restarts in the first contour inside this contour.

4.6 Planar Perspective-3-Point Problem

Assuming the target is successfully detected in the image by knowing its outer contour and its projected centre, it becomes possible to estimate its pose. Due to a recursion in the algorithm the centre is

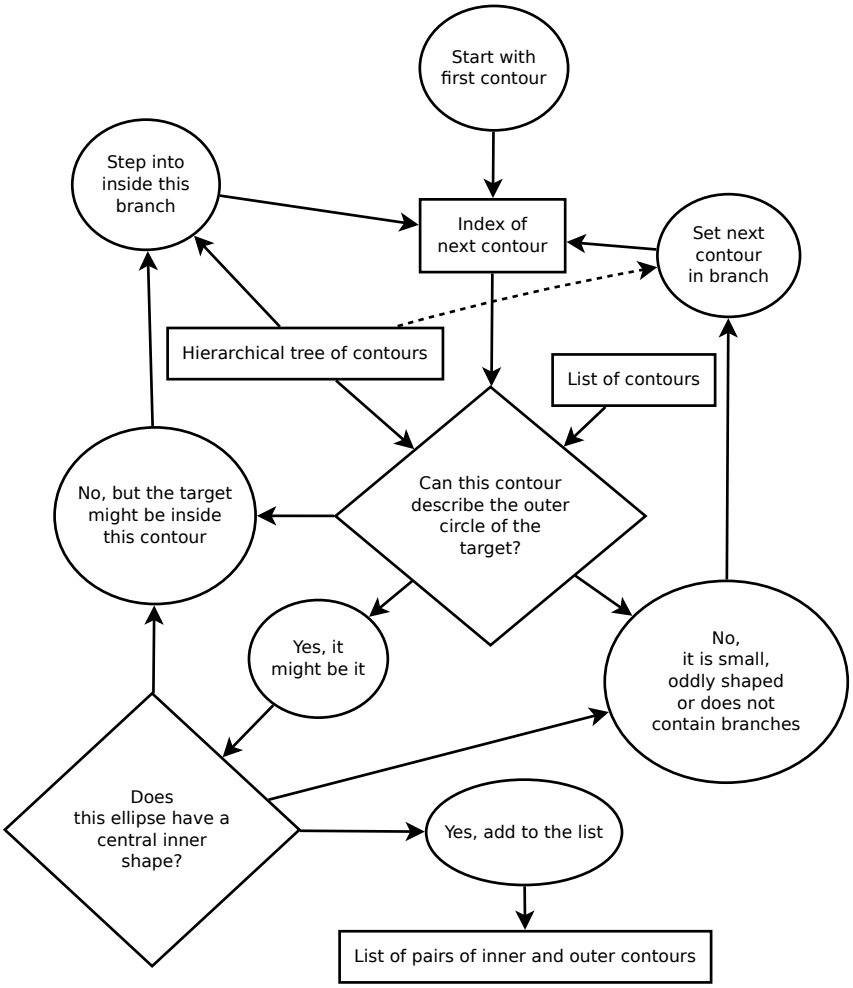


Figure 4.6: Algorithm for the concentric circles detector

computed in section 4.7, by now assuming it is correctly known.

Figure 4.7 represents a simplification of the Perspective-3-Point (P3P) problem as projected onto the yz plane of $\mathcal{F}_C$. An objective is to estimate the angle ψ as the rotation about axis x, when computing on the yz plane, and to estimate the angle θ as the rotation about axis y when computing on the xz plane.

The capital Latin letters represent points:

- O is the camera optical centre;
- C is the target centre;
- A and B are the extremes of the target along the y direction;
- A', B' and C' are the respective projection of points A, B and C onto $\mathcal{F}_P$.

The position of C defines the position of the target, as it is the origin of its frame.

The unit vector from A' to O is the same unit vector as from O to A or from A' to A, it is denoted as $\vec{v}_A$ and calculated in (4.3). The equivalent reasoning is used to obtain $\vec{v}_B$ and $\vec{v}_C$. Save for the point O, none of the referred points and vectors must lie on the yz plane, as can be wrongly perceived.

$$\vec{v}_A = \frac{K^{-1}A'}{\|K^{-1}A'\|} \tag{4.3}$$

The Greek letters represented in Figure 4.7 represent angles. The angles θ_1 and θ_2 can be computed

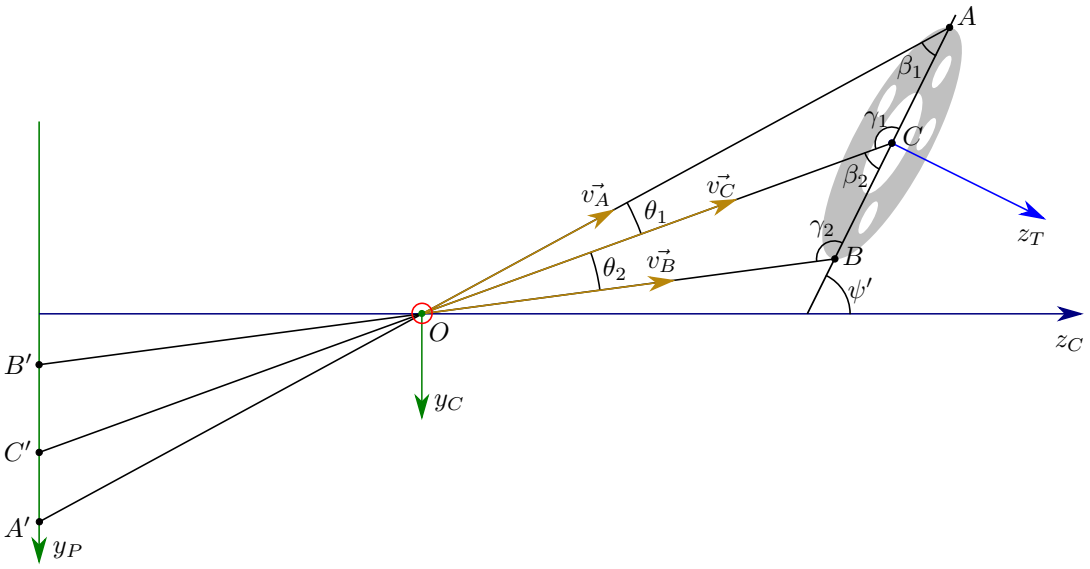


Figure 4.7: A P3P problem case projected on the yz plane

as the inverse cosine of the dot product of their adjacent vectors.

$$\theta_1 = \arccos\left(\vec{v}_A.\vec{v}_C\right) \tag{4.4}$$

$$\theta_2 = \arccos\left(\vec{v}_B.\vec{v}_C\right) \tag{4.5}$$

The segments $\overline{AC}$ and $\overline{BC}$ are external radii of the target, therefore their lengths are equal to each other $\overline{AC} = \overline{BC} = r$. This assumption creates the conditions for a special case of the P3P problem, where a closed form solution can be computed through the cosine theorem. The general form of this approach is known as Random Sample Consensus (RANSAC) enunciated by Fisher *et al.* [28].

$$\begin{cases} r^2 = \overline{OA}^2 + \overline{OC}^2 - 2\overline{OA}\,\overline{OC}\cos\theta_1 \\ r^2 = \overline{OB}^2 + \overline{OC}^2 - 2\overline{OB}\,\overline{OC}\cos\theta_2 \\ (2r)^2 = \overline{OA}^2 + \overline{OB}^2 - 2\overline{OA}\,\overline{OB}\cos\left(\theta_1 + \theta_2\right) \end{cases} \tag{4.6}$$

Solving the system in $\overline{OC}$ gives an analytic expression for the distance of the target to the camera centre. This value allows the computation of the position of the target $C_P = \overline{OC}\vec{v}_C$

$$\overline{OC} = \frac{\sqrt{2}r\sin\left(\theta_1 + \theta_2\right)}{\sqrt{3 - 2\cos\left(2\theta_1\right) - 2\cos\left(2\theta_2\right) + \cos\left(2(\theta_1 + \theta_2)\right)}} \tag{4.7}$$

To find the angle $\psi = \psi' - \pi/2$, the angles γ_1 and β_2 should be computed with the pair of solutions that lets (4.10) to be true giving that (4.8) and (4.9) both have two solutions each ($\sin x = \sin \pi - x$).

$$\gamma_1 = \arcsin\left(\frac{\overline{OC}}{r}\sin\theta_1\right) \tag{4.8}$$

$$\beta_2 = \arcsin\left(\frac{\overline{OC}}{r}\sin\theta_2\right) \tag{4.9}$$

$$\theta_1 + \gamma_1 + (\pi - \gamma_2) = \theta_2 + \gamma_2 + \beta_2 = \pi \tag{4.10}$$

Then, calculating $\overline{OB}$ it becomes possible to determine ψ'.

$$\overline{OB} = r\frac{\sin(\pi - \beta_2 - \theta_2)}{\sin\theta_2} \tag{4.11}$$

$$\vec{v}_{CB} = \overline{OB}\vec{v}_B - \overline{OC}\vec{v}_C \tag{4.12}$$

$$\psi' = \arctan(\vec{v}_{OB,y}, \vec{v}_{OB,z}) \tag{4.13}$$

The angle ψ' is the angle between the z_C axis and the straight line crossing the points A, B and C on a projection onto the yz plane. The angle defining the rotation of the target about x_C is ψ, the angle formed between the axes z_C and z_T on the projection onto the yz plane.

To get θ, the angle of the target about y_C, an analogous procedure is taken but using a projection onto the xz plane.

4.7 Find the Projected Target Centre

The projection of the circle is an ellipse, however the projection of the centre of the circle is not the geometric centre of the projected ellipse. The accuracy of estimation of the coordinates of this point influences the accuracy of estimation of the pose of the target.

4.7.1 First Guess

Kim *et al.* [29] present a very accurate method to extract the centre of projection of concentric circles. It is based on the assumption that both circles are sections of the same cone. Those circles are projected as ellipses and the projected centre of the concentric circles can be extracted using the notation of a conic section.

By calling $\mathcal{Q}_o$ to the conic section matrix of the outer circle and $\mathcal{Q}_i$ to the conic section of the inner circle, a matrix $\mathcal{H}_1$ can be computed as in (4.14).

$$\mathcal{H}_1 = \mathcal{Q}_o^{-1}\mathcal{Q}_i \tag{4.14}$$

The eigenvalue of $\mathcal{H}$ closest to zero provides the key to get to the searched point, that eigenvalue can be denoted as λ to get $\mathcal{H}_2$.

$$\mathcal{H}_{\in} = \mathcal{Q}_o^{-1} - \lambda \mathcal{Q}_i^{-1} \tag{4.15}$$

The mean values of the first and second columns of $\mathcal{H}_2$ provide, respectively, an approximation for the values of x_p and y_p. These can be taken as a first approximation for the centre of projection of the concentric circles $\hat{C}' = (\hat{x}_p, \hat{y}_p)$.

4.7.2 Refinement and Validation

In section section 4.6 a method for getting the distance to the target was described. However it relied on an accurate estimation of the position of point C'. If that point is not accurate, different pairs of opposite points will produce different values for the distance $\overline{OC}$. Based of this fact, a heuristic for centre refinement and error checking can be produced.

Computing the standard deviation σ_d^2 of the estimated distance for different, equally spaced, pairs of opposed points on the circumference of the outer ellipse, it is possible to get some insight about the accuracy of estimation. An estimated point with a lower value for σ_d^2 means that it is most probably a better estimation that another point having a higher value for σ_d^2.

With this assumption, a centre refinement heuristic was devised. It starts by computing σ_d^2 for the initially guessed centre, if that value falls under some certain tolerance t it is deemed acceptable and the heuristic stops returning the initial guess. Otherwise the values for σ_d^2 are computed at different testing points around the initial guess about a radius of search r_s 60° apart from each other, if there is a better centre candidate the heuristic restarts with that point as initial guess, otherwise it it restarted with a new radius of search $r_s/2$.

4.8 Proposed Algorithm

The proposed algorithm is inspired by Cucci [15]. It introduces a new method for classifying elliptic contours and adds instrumental enhancements on some parts of the algorithm, as described in the previous sections.

The proposed algorithm is briefly described in section 4.1 and the steps followed are detailed in subsequent sections. The target design and the solution to the P3P problem were followed from Cucci without modifications, all the other steps were inspired by Cucci's findings with more or less modifications.

The proposed algorithm achieves best results by following this steps:

Image steam stabilisation Open a stream of images from a camera with a constant rate, that rate should be adjusted to the resolution and computer used;

Image preprocessing Process each image, if its coloured convert it into greyscale;

Canny contour detector Detect all contours in the image being processed;

Concentric circles detection Find the contours that might define the outer circumference of a target;

Cross check code Check if the code if found on any of the possibly found targets by cross correlating the signal in the target with a reference signal, if it matches: the resulting lag indicates the z rotation of the target;

Find the centre Estimate the point of projection of the centre of the concentric circles of the target and refine it if needed;

Estimate the pose Compute the distance to the target and its rotations in x and y, the position of the centre of the target if given by $C^C = \overline{OC}\vec{v}_C$.

This concludes the description of the proposed methodology for pose estimation. The following chapter covers a set of tools useful for the implementation aspects.

Chapter 5

Experimental Implementation

5.1 Robot Operating System (ROS)

Robot Operating System (ROS) is an open-source framework designed as a modular set of tools for robotics applications. The distribution used in this project is the *ROS Kinetic Kame* on top of *Ubuntu 16.04*, which will both be supported until April 2021 (as LTS versions).

ROS can easily inter-operate with many other frameworks as it officially supports the writing of C/C++ and Python nodes. This means that virtually any existing library for C/C++ and Python can be used.

This tool has other concepts that may be useful for different usages but are not detailed in this document[30].

5.1.1 ROS Fundamental Concepts

The structure of a working ROS implies at first that its core module (`roscore`) is running and then the needed **nodes** can connect to it through IPv4, any nodes running on the same machine can connect to the *localhost* IP (127.0.0.1). A machine does not require a network interface (i.e. wifi or ethernet) to be able to run a fully fledged ROS.

Nodes

Nodes are programs that use ROS as a means of communication. They may subscribe or publish messages in the ROS network. A node can be used as a driver or a service like in any other operating system.

Any executable software using the ROS framework can be considered a node. There are hundreds of ROS packages available on the ROS website [31]. Each package includes at least one ROS node.

The nodes are what allows ROS to be called modular. Each one may have more or less tasks in hand depending on what the developer wants it to do. A node can be developed in C++ or in Python.

As an example, the implementation of the algorithm can be run as a ROS node called `target_finder`, developed in Python. The `target_finder` is fed with a stream of images from another node called

usb_cam_node, developed in C++. The image streamer node can be found as a ROS package called usb_cam and works as a USB camera driver.

roscore

In the ROS Wiki page for roscore [32] is described as being:

> (...) a collection of nodes and programs that are pre-requisites of a ROS-based system.

It is has three main modules:

1. **Master**
2. **Parameter Server**
3. rosout

The **Master** is the main piece of roscore. It gives the nodes the needed information for establishing the connections to relevant nodes depending on the published or subscribed topics. Each new node connects to **Master** to announce to it its intention to **publish** and/or **subscribe** to the **topics** needed.

Figure 5.1 displays a ROS connection between a Camera node and an Image viewer node. The Camera node represented here is a camera driver for ROS that is responsible for getting frames from the camera and publishing them as *Image* messages containing that data, so it advertises to **Master** it is publishing *images*, but no *images* are sent until there is any subscriber. The Image viewer node needs *images* to show the user, so it subscribes to that topic from the **Master**. Now the **Master** has a publisher and a subscriber for the same topic so it tells both nodes about each other to allow them to transfer *images* from one to the other.

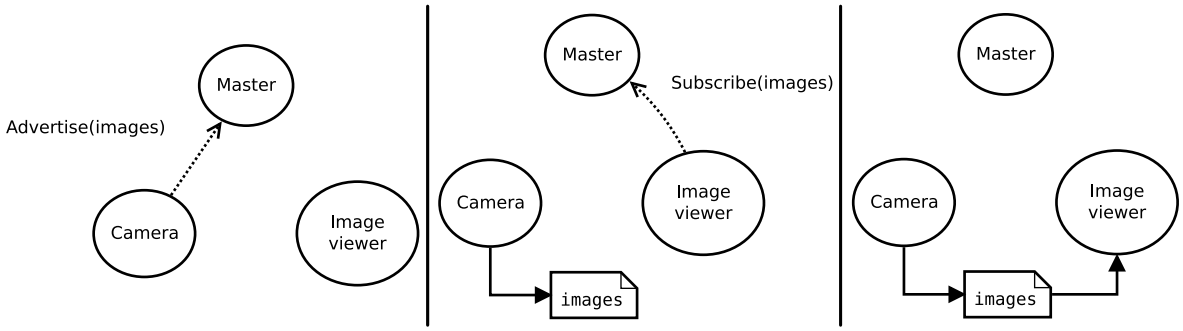


Figure 5.1: **Left**: Camera node announces to **Master** it is publishing *images*. **Centre**: Image viewer node subscribes to *images*; **Right**: Camera sends *images* to Image viewer. Adapted from [33].

The **Parameter Server** is an auxiliary unit that runs inside the **Master** and is intended for static parameters that can be changed during runtime.

The rosout is a logging mechanism. It is a node that subscribes to the topic \rosout and appends the income messages to a log file, a new file is created every time roscore is launched.

Topics

Each **topic** is characterised by its name (i.e. `\camera\image`) and its message type (i.e. *Image*). The name of the topic works like a web address by being a ROS URI (Uniform Resource Identifier), while the message type is the hint for the structure of the messages transmitted on that topic.

5.1.2 Message Types

The common messages are a comprehensive set of message types and should be used for better compatibility and code readability whenever possible[34]. There are also the standard messages which are more basic message types and may be used as a starting point for new message types[35].

The most relevant message types used in this dissertation are succinctly described next and are part of the `common_msgs` set.

Image

The *Image* message is included in the subset `sensor_msgs` and its main content is an uncompressed image.

An *Image* message contains:

- The message header has:
  - timestamp (should be the aquisition time)
  - frame_id
  - frame origin
  - (x,y,z) directions according to an NED frame
- height
- width
- encoding (of data)
- data (the uncompressed image itself)

CameraInfo

The *CameraInfo* message is used by camera driver nodes to publish camera calibration information, as described in the subset `sensor_msgs`.

Some information on this message type is identical to *Image*: the message header, the height and the width are common to both types. Adding to that, an amount of relevant data gotten from the camera calibration is included in the message:

- distortion_model typically ”plumb_bob”
- D the distortion parameters vector (κ_1, κ_2)
- K intrinsic camera matrix for the distorted image
- R rectification matrix (relevant for multiple cameras, equals the identity matrix in the monocular case)

- P intrinsic camera matrix of the rectified (undistorted) image

Pose

Pose is a message type belonging to the subset `geometry_msgs` and describes a position and orientation:

- position Point x,y,z
- orientation Quaternion $\eta, \epsilon_x, \epsilon_y, \epsilon_z$

5.1.3 Auxiliary Tools

To aid the developer and the user to work with ROS there are some essential tools included in a ROS desktop installation and of those, the featured ones are mentioned next.

rostopic

It is a command-line interface (CLI) tool for getting information about a topic on the command-line. The highlighted information one can get from a topic with this tool are its frequency, its bandwidth usage, its data and its message type.

rqt

`rqt` is a graphical user interface (GUI) tool that can plot data from topics, show topics, view published images, amongst many other possibilities.

rosrun

It is a CLI tool used to get a node from a package running.

roslaunch

`roslaunch` is a CLI tool to launch ROS nodes with its configurations on a XML (eXtensible Markup Language) file with `.launch` extension. Multiple nodes can be started from a single launch file and if `roscore` is not running it also launches it, so it is a powerful tool to quickly get a ROS environment running.

5.1.4 Using ROS

ROS brings significant advantages for the application under study in this work, hence was the development framework selected for this project. The modular design of ROS allowed the different implementations to be tested independently as different ROS nodes and the change of camera settings or hardware did not require any changes to the developed nodes.

As a starting point one may imagine a simplified ROS network as schematised in Figure 5.1. Notwithstanding that is a simplification of what happens in the simple case of a camera viewer because the only part of `roscore` displayed is the **Master** and that is acceptable in this use case.

The developed node was given the name `target_finder`. At its minimum operating state the developed node subscribes to a pair *Image* and *CameraInfo* topics and publishes the results to a *Pose* topic.

The node working as the ROS camera driver is called `usb_cam`, is available from the ROS repositories and interfaces with cameras through the *Video4Linux* drivers, so it has a broad compatibility with USB cameras. It was chosen due to its rich set of features and low processor usage. The camera used for testing features auto-focus, but that is an undesirable feature because the intrinsic camera parameters would change dynamically. This node is able to set a fixed focus on the camera, so is was fixed to infinity on all tests. Other tested camera drivers (e.g. `cv_camera`) were heavier on CPU usage.

5.1.5 Base Network

To improve diagram readability the `roscore` related nodes are suppressed, but are fundamental for a ROS network to function and need to be running anytime a node is running.

Figure 5.2 displays a diagram representing the flow of data in the ROS network when the nodes `usb_cam` and `target_finder` are running. At this point the developed node is receiving the subscribed messages and publishing the pose found in each image.

If the camera info is not expected to change dynamically (i.e. constant: camera, used sensor region, resolution, lens and focus), once it is received it may be safely unsubscribed because all the following *CameraInfo* messages are expected to have the same data.

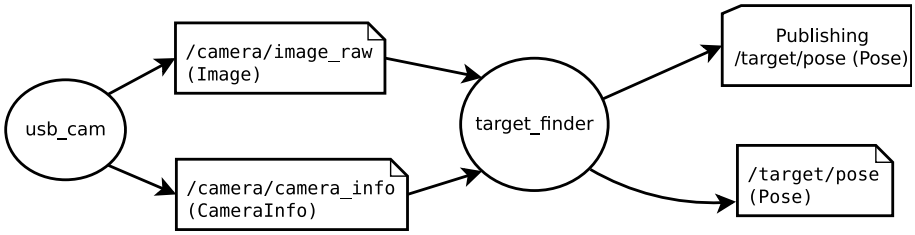


Figure 5.2: Flow Diagram of `usb_cam` and `target_finder` nodes launched.

To watch the data flowing to and from the developed node the tool `rqt` was used.

Possible Practical Network

To show a practical use case scenario let us consider a hypothetical robot that can move and rotate in a 3D space. Assuming it had a ROS node able to control its pose called `robot_poser`, one can sketch a possible network for controlling that robot pose.

The diagram in Figure 5.3 shows a simplified network showing only a developed node and its output `Pose` being fed to a node controlling a robot. The reference pose to be followed by the robot can be managed by the **ROS Parameter Server** included in `roscore`.

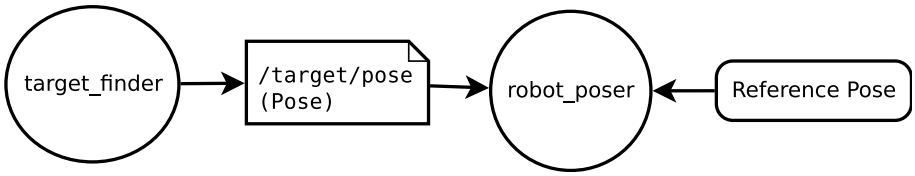


Figure 5.3: Simplified flow diagram of a practical ROS network with a hypothetical robot

5.2 OpenCV

To circumvent the re-implementation of common computer vision algorithms, the library OpenCV was used. It stands for Open Source Computer Vision Library and contains many computer vision algorithms implemented in C++ with bindings for usage with other programming languages like Python.

Currently a version (3.3.1) of OpenCV is included in ROS, easing the porting of any node made for a version of ROS to other platforms where is can be installed.

5.3 Testing Setup

The computer used as test bench is a laptop computer with the specifications stated next:

- CPU: Intel i7-720QM (1.60 GHz to 2.80 GHz)
- GPU: ATI Mobility Radeon HD 5730 1GB
- RAM: 8GB DDR3 1333 MHz Dual-Channel
- OS: Ubuntu 16.04 (amd64)

5.3.1 Test Camera

The camera used for testing the implementation was the Logitech HD Pro C920 (shown in Figure 5.4). This camera was chosen because of its apparent good build quality, interesting specifications, good price to quality ratio, and prompt availability at physical stores.

Figure 5.4: View of the camera Logitech HD Pro C920. (From [36])

Specifications

The specifications listed next are transcripts from the manufacturer website[37]:

- USB Protocol: USB 2.0
- Lens and Sensor Type: Glass
- Focus Type: Auto
- Optical Resolution: True 3MP, Software Enhanced 15MP
- Diagonal Field of View (FOV): 78°
- Focal Length: 3.67mm
- Image Capture: 2.0 MP, 3 MP*, 6 MP*, 15 MP*
- Video Capture: 360p, 480p, 720p, 1080p
- Frame Rate (max): 1080p@30fps

The USB 2.0 specification limits the bandwidth to 192Mb/s (24MB/s), that may be a limitation on the rate of data the camera can handle.

Further analysis of information revealed by the camera driver showed a few interesting features:

- Pixel formats supported are YUYV, MJPEG and H264.
- MJPEG and H264 support the same resolutions and frame rates (up to 1920x1080 at 30fps).
- The pixel format YUYV is not so simple, supporting a larger resolution while being more limited on the maximum frame rate per pixel (in Table 5.1).

The pixel format YUYV (a form of YUV, which is a luminance-chrominance colour space) is an uncompressed format that for each pair of pixels has two luminance (brightness) bytes (Y) and two chrominance (colour) bytes (U and V). The U and V bytes are the same for each pair of pixels and the Y bytes are for each pixel, which means that for a grey scale image the data is uncompressed and lossless.

Table 5.1: Maximum frame rate for different resolutions for the Logitech C920 with the pixel format YUYV

Resolution	2304x1536	1920x1080	1280x720	1024x576	864x480	800x448
Max. Frame rate	2	5	10	15	24	30

The format MJPEG is a video format based on the well known JPEG picture format standing for Motion JPEG. MJPEG and H264 are compressed video formats that can be decoded to raw formats (e.g. RGB, HSV) losing some data.

The ROS node `usb_cam` supports by default the pixel formats YUYV and MJPEG, so all the resolution/framerate combinations are available to use.

Camera Parameters

For camera calibration, the ROS package `camera_calibration` was used with the chessboard suggested by OpenCV printed on a A4 paper as the calibration target. The results obtained for common resolutions with the Logitech camera are shown on Table 5.2.

5.4 Final Setup

The final implementation of the proposed method can be run on any platform supporting ROS.

Table 5.2: Camera parameters estimated for the Logitech C920 used

Resolution	α	β	u_0	v_0	κ_1	κ_2
640x480	621.34	624.75	331.22	242.33	0.09373	-0.17168
800x600	783.65	784.14	424.66	310.07	0.11841	-0.20147
1280x720	923.66	922.05	664.03	371.90	0.10440	-0.18081
1920x1080	1395.7	1385.0	1013.0	571.14	0.10567	-0.17602

5.4.1 Raspberry Pi

The Raspberry Pi 3 was used to test the achievable performance of the implemented algorithm on a small board computer.

This lightweight computer has the following technical specifications:

- Raspberry Pi 3
  - CPU: Broadcom BCM2837 (Quad-core ARM Cortex A53 at 1.2GHz)
  - RAM: 1GB DDR2
- RPi Camera Module v1
  - 5 Megapixels
  - Video output up to 1080p at 30fps, 720p at 60fps and 480p at 90fps
  - Sensor resolution: 2592x1944 pixels
  - Sensor size: $3.76\,\text{mm} \times 2.74\,\text{mm}$
  - Focal length: $3.60\,\text{mm}$

Resulting from the calibration of the Camera Module v1, the parameters estimated are displayed in Table 5.3. As resolutions higher than VGA showed poor performance with the Raspberry Pi 3, calibration values for those resolutions were not estimated.

Table 5.3: Camera parameters estimated for the Raspberry Pi Camera Module v1

Resolution	α	β	u_0	v_0	κ_1	κ_2
640x480	604.92	613.29	318.72	244.96	0.12422	-0.32018

The Camera Module v2 NoIR (a version without the stock infrared filter) was also tested. The v2 has an 8MP sensor with the same size and has a higher field of view when using the full resolution, however for poor lighting environments the v1 showed slightly clearer results due to having larger pixels. The narrower field of view of the v1 allows the detection of objects at a larger distance.

5.4.2 Omni-directional Vehicle

For demonstrations purposes it became convenient to use the promptly available platform OMNI-ANT. As can be seen in Figure 5.5, it is an omni-directional vehicle with a set of three wheels equally spaced mounted on a flat round aluminium plate.

This platform contains a low-level controller that accepts commands for velocity in x and y (in the directions indicated by the letters X and Y in Figure 5.5) and also a command for rotation speed control.

Figure 5.5: Top view of the OMNI-ANT platform

The commands order changes in the reference velocity for the low-level controller, the command values are proportional but are not in any known units. The OMNI-ANT platform receives its input references wirelessly.

Chapter 6

Results

This chapter presents the implementation results obtained using the proposed methodology for a set of experimental tests. To validate the solution, a comparison of the measurements taken using the method stated by this document with the measurements taken with the Qualisys robotic arena on a simultaneous run. To extract the limits of the solution the target was moved until a limit was reached. Finally, in order to demonstrate the solution in action in a dynamic system controlled in closed-loop system, a pair of ground vehicles were used in a configuration Leader-Follower.

6.1 Validation

In order to validate the proposed methodology, a concise and effective plan was designed:

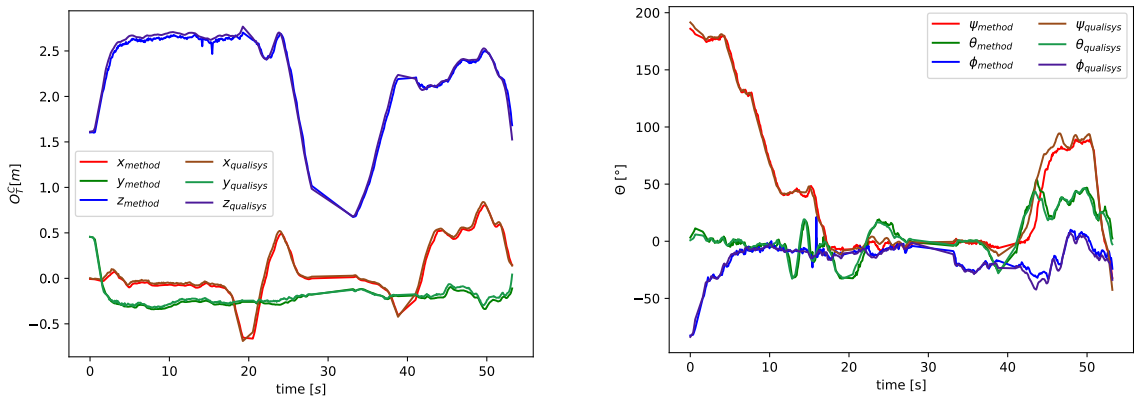
- Reflective markers were placed around the camera and around the target, to allow detection of both with the Qualisys robotic arena;
- The Qualisys software was set to start recording and the detector node was launched with the `rosbag` recording the coordinate transforms calculated;
- The target or the camera were moved manually while trying to keep the target in the field of view of the camera.

Some trials were made with the camera still, the target moving and others with camera moving, target still. The latter situation showed better results, due the lighting conditions inside the laboratory.

The values shown in Figure 6.1 were measured using a target printed in a A3 sheet and the Logitech camera with a resolution of 1280×720 (720p) with the implementation running on the test bench. The data points for the times where either the arena or the algorithm produced no output were removed. As the Qualisys system has a much higher rate of data, only the data points with the closest timings were considered in order to have the same amount of data for both data series.

Table 6.1: Standard deviation values of the pose validation trial

$\sigma_x[m]$	$\sigma_y[m]$	$\sigma_z[m]$	$\sigma_\psi[^\circ]$	$\sigma_\theta[^\circ]$	$\sigma_\phi[^\circ]$
0.0296	0.0190	0.0435	3.34	3.82	6.31



(a) Position

(b) Orientation

Figure 6.1: Pose of target in the camera frame as detected by both methods for validation

The standard deviation values in Table 6.1 were computed from the absolute error of each parameter in the same units as Figure 6.1, taking the robotic arena values as ground truth. The standard deviation of z is higher than the standard deviations of x and y because for each computation the error of z is always larger that the x or y errors. The estimated translation errors are most frequently under the 5% margin, as the algorithm aims for. The error for ϕ is often double than the error for ψ and θ due to the nature of its computation, a lower error can be achieved by taking more data points to read the circle code. The ψ and θ values share close standard deviations as both are computed with the same steps, differing only in the values used.

The results obtained show that the solution proposed is on par with a much more sophisticated tracking system for the desired application, thus is well-suited for integration with mobile robotics applications, since the error is acceptable for most instances.

6.2 Limitations and Specifications

This solution has some limitations inherent to the camera, the algorithm and the processing hardware.

The limitations of the solution were extracted empirically by moving either the camera or the target starting at a surely detectable starting position. That starting position of the target relative to the camera had approximately zero rotation in all three axes and its centre was in the optical axis at a distance of 2 metres. From the starting position, different steps were taken to detect different limitations:

- Moving the **target** towards the camera until detection fails, to fetch the **closest** z distance possible;
- Moving the **target** outwards from the camera until detection fails, to fetch the **farthest** z distance possible;
- Rotating the **target** around its x axis, to extract the ψ detection limits;
- Rotating the **target** around its y axis, to extract the θ detection limits;
- Rotating the **camera** around its x axis, to find the yz field of view;
- Rotating the **camera** around its y axis, to find the xz field of view.

This procedure was followed to detect the limits of a Raspberry Pi 3 with the 5 megapixel camera, which are summarised in Table 6.2. The values for the range are acceptable values for using the solution without largely affecting the frequency of estimation. The largest z value detected was close to 5 metres, but at a very low frequency due to the small size of the target in the image hindering the detection process. The field of view (FoV) was measured at a distance of approximately 2 metres, for longer distances it is wider and at closer positions is is narrower.

Table 6.2: Limitations found when using the Raspberry Pi Camera v1

Resolution	Frequency	ψ Range	θ Range	z Range	xz FoV	yz FoV
640×480	10 Hz	$\pm 60°$	$\pm 55°$	0.6 m to 3.5 m	$49.7°$	$36.8°$

The frequency of 10Hz was manually defined at the camera capturing settings to synchronise it to the typical processing rate on the Raspberry Pi 3, as higher frequencies would increase its power output making it reduce its processing power due to high CPU temperature (thermal throttling). It was set with stock settings, a small aluminium heat sink covering most of the area of the CPU, an ambient temperature of $25\,°C$ and an open case allowing the ambient air to flow freely. After some hours of having the system running with the described settings, the CPU temperature settled at $64\,°C$. The processing time varies depending on the amount of clutter in the scene and whether the right target is visible enough for pose estimation. Typical processing times go from $30\,ms$ to $70\,ms$ when the target is not visible, and from $50\,ms$ to $100\,ms$ when the target is detected, save for some random peaks taking over $200\,ms$ due to background workloads inherent to the operating system. These processing times are enough for low-speed control, as if the target exits the field of view of the camera between processed frames it can be missed.

6.3 Application

For the last part, the solution was used as a positioning sensor on a closed-loop control system.

6.3.1 Hat Approach

A hat for the OMNI-ANT platform was designed and 3D printed in order to place a marker on top of it, as shown in Figure 6.2. The objective was to control the OMNI-ANT while being localised with a camera placed on the ceiling pointing to the floor. However, the proved to be unfeasible due to the height of the ceiling in the lab being too high, whether the Logitech or the Raspberry Pi cameras were used. An option would be to disable binning in the Raspberry Pi camera, halving the field of view in both axis while increasing the range of operation. The binning in a camera is a feature that allows it to produce a low resolution picture using more pixels of the sensor for each pixel in the image. The Raspberry Pi cameras do 2×2 binning by default when capturing images under the resolution of 1296×976, in order to reduce the signal to noise ratio (SNR) in low light conditions. Disabling binning resulted in very poor images with the laboratory lighting.

Figure 6.2: OMNI-ANT with hat

The hat approach was successful for controlling an OMNI-ANT into going to a specific point in sight of a wall-mounted camera.

6.3.2 Follow the Leader

The second, and final, approach into using the solution as a positioning sensor was to equip an OMNI-ANT as an autonomous follower, as depicted in Figure 6.3. Besides the minimum equipment needed for the operation of an OMNI-ANT, a Raspberry Pi 3 with a Camera Module v1 and a power-bank to power the Raspberry Pi. The power-bank was used because it was readily available for usage as a LiPo battery with a regulated output of $5\,\mathrm{V}$ and up to $2.1\,\mathrm{A}$.

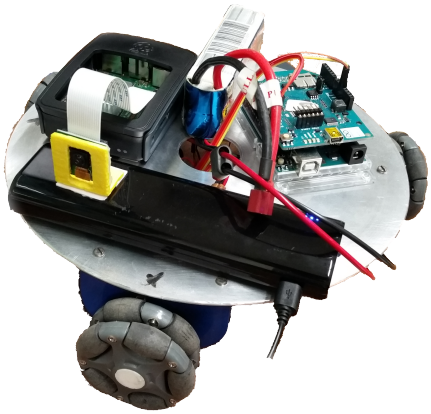


Figure 6.3: OMNI-ANT equipped as Follower

The follower was set to locate the target placed on top the leader (headed backwards) and to follow the leader using a proportional controller. From the beginning, the follower has the information about the fixed transformations needed:

- Follower to Camera, transforms the follower OMNI-ANT reference frame into the camera reference frame;
- Target to Leader, transforms the target reference frame into the leader OMNI-ANT reference frame.

The chain is closed by the solution by providing the transformation from the Camera to the Target,

allowing the follower to be aware of the positioning of the leader. The goal position of the follower is the point located $1\,\mathrm{m}$ behind the leader, on a pure translation in the direction of the negative x semi-axis of the leader.

The leader was remotely controlled using an Android app created with MIT APP INVENTOR. This theme is deepened in Appendix A.

Both, the leader and the follower used the same OMNI-ANT platform without changing its controlling method of sending the commands through UDP (User Datagram Protocol).

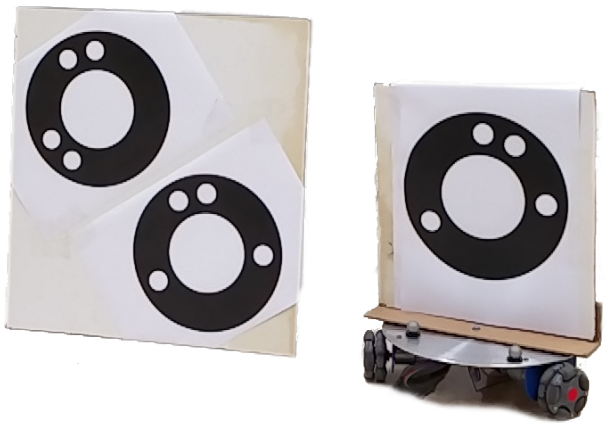


Figure 6.4: **Left**: plate with two dummy targets, **right**: OMNI-ANT equipped as Leader

At the left side of Figure 6.4 there is a pair of dummy targets made using the same template as the real target of the leader, at the right. As expected and desired, the follower follows only the target in the leader, after testing with all three targets inside the field of view of the camera.

Figure 6.5: QRCode containing a hyperlink to the demo video (`https://youtu.be/5D9tKhgqPAQ`)

Some trials were documented in video and compiled into a single clip that can be viewed from the link stated in Figure 6.5 as a direct link in the description or as a QRCode that can be scanned to quickly access the video from a smart-phone.

Chapter 7

Conclusions

7.1 Achievements

The objectives stated initially were globally accomplished, namely the turnkey solution for relative pose estimation with the possibility to be applied on platooning.

The preliminary validation of the algorithm by confrontation of its results with the robotic arena data proved it is a good pose estimation approach with good results.

This solution was easily fitted on an existing mobile robot without adding any custom parts. It can be considered an inexpensive solution, as the sum of the parts added to the robot cost about the same as the Logitech camera used for validation alone (around 90€).

As for the platooning, a preview demonstration was accomplished with only a leader carrying a target and a follower carrying the final solution with a Raspberry Pi 3 and a camera module attached.

7.2 Contributions

Using Cucci [15] as baseline for a working marker detection algorithm, an improved method for binary ellipse classification was developed.

A solution for autonomous OMNI-ANT vehicles in formation was demonstrated with working results.

This document can be used as an introduction to ROS for the development of future work.

7.3 Future Work

There are several paths that can be taken from the point left by this work. Some works might focus on the algorithm while others might give more detail to the control scheme.

Using the same hardware solution: different algorithms, targets and control techniques can be employed and compared using this work as a baseline to improve upon.

Following the same algorithm several improvements could be achieved. One in particular could be the method for testing the target code, as cross correlation methods are very heavy for real-time appli-

cations. The approach employed by the uni-dimensional bar-codes seen everyday in product packaging could be studied to employ here as it would allow the identification of the target by a number instead of a signal, easing human readability while providing faster code checking.

The platooning preview can be extended to more vehicles, as there are more OMNI-ANTs available each could be mounted with a target facing backwards and the camera to the front. It could be coupled with a computer more powerful than the Raspberry Pi 3 to test in faster situations, as in traffic. It might also provide new control challenges by keeping each node independent, as each one should keep following the next while easing the movements to allow the previous to keep following.

Bibliography

[1] Snapchat. Lens studio. https://lensstudio.snapchat.com/, 2017. Online, accessed: 22-December-2017.

[2] Google. ARCore — google developer. https://developers.google.com/ar/, Jan. 2018. Online, accessed: 3-April-2018.

[3] H. Bain. Helipad at Blessington Lakeside Resort. http://www.geograph.org.uk/photo/262175, 2006. Online, accessed: 22-September-2017.

[4] K. Lutz. 61-years of working aerial photogrammetry history. *Photogrammetric Engineering & Remote Sensing*, 81(2):89 – 93, 2015. ISSN 0099-1112. doi: https://doi.org/10.1016/S0099-1112(15)30301-3. URL http://www.sciencedirect.com/science/article/pii/S0099111215303013.

[5] R. Muñoz-Salinas, M. J. Marín-Jimenez, E. Yeguas-Bolivar, and R. Medina-Carnicer. Mapping and localization from planar markers. *Pattern Recognition*, 73:158 – 171, 2018. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2017.08.010. URL http://www.sciencedirect.com/science/article/pii/S0031320317303151.

[6] J. Zhang, C. Shi, D. Sun, and Y. Han. High-precision, limited-beacon-aided auv localization algorithm. *Ocean Engineering*, 149:106 – 112, 2018. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2017.12.003. URL http://www.sciencedirect.com/science/article/pii/S0029801817307254.

[7] H. Pan, J. Huang, and S. Qin. High accurate estimation of relative pose of cooperative space targets based on measurement of monocular vision imaging. *Optik - International Journal for Light and Electron Optics*, 125(13):3127 – 3133, 2014. ISSN 0030-4026. doi: https://doi.org/10.1016/j.ijleo.2013.12.020. URL http://www.sciencedirect.com/science/article/pii/S0030402614000849.

[8] Y. Bi and H. Duan. Implementation of autonomous visual tracking and landing for a low-cost quadrotor. *Optik - International Journal for Light and Electron Optics*, 124(18):3296 – 3300, 2013. ISSN 0030-4026. doi: https://doi.org/10.1016/j.ijleo.2012.10.060. URL http://www.sciencedirect.com/science/article/pii/S0030402612008698.

[9] G. Xu, Y. Zhang, S. Ji, Y. Cheng, and Y. Tian. Research on computer vision-based for UAV autonomous landing on a ship. *Pattern Recognition Letters*, 30(6):600–605, apr 2009. doi: 10.1016/j.patrec.2008.12.011. URL https://doi.org/10.1016/j.patrec.2008.12.011.

[10] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, Jan 1979. ISSN 0018-9472. doi: 10.1109/TSMC.1979.4310076.

[11] I. Sobel. An isotropic 3× 3 image gradient operator. *Machine vision for three-dimensional scenes*, pages 376–379, 1990.

[12] B. Wu, D. Ye, Y. Guo, and G. Chen. Multiple circle recognition and pose estimation for aerospace application. *Optik - International Journal for Light and Electron Optics*, 145:148–157, sep 2017. doi: 10.1016/j.ijleo.2017.07.024. URL https://doi.org/10.1016/j.ijleo.2017.07.024.

[13] M. Fornaciari, A. Prati, and R. Cucchiara. A fast and effective ellipse detector for embedded vision applications. *Pattern Recognition*, 47(11):3693 – 3708, 2014. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2014.05.012. URL http://www.sciencedirect.com/science/article/pii/S0031320314001976.

[14] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, nov 1986. doi: 10.1109/tpami.1986.4767851. URL https://doi.org/10.1109/tpami.1986.4767851.

[15] D. A. Cucci. Accurate optical target pose determination for applications in aerial photogrammetry. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-3:257–262, jun 2016. doi: 10.5194/isprsannals-iii-3-257-2016. URL https://doi.org/10.5194%2Fisprsannals-iii-3-257-2016.

[16] S. Suzuki and K. be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, apr 1985. doi: 10.1016/0734-189x(85)90016-7. URL https://doi.org/10.1016/0734-189x(85)90016-7.

[17] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003.

[18] E. Karami, S. Prasad, and M. S. Shehata. Image matching using sift, surf, brief and orb: Performance comparison for distorted images. *CoRR*, abs/1710.02726, 2017.

[19] Westnest. A boeing 737's primary flight display. https://en.wikipedia.org/wiki/Primary_flight_display#/media/File:Primary_Flight_Display_of_a_Boeing_737-800.png, 2015. Online, accessed: 12-July-2018.

[20] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer London, 2009. doi: 10.1007/978-1-84628-642-1. URL https://doi.org/10.1007/978-1-84628-642-1.

[21] D. A. Forsyth and J. Ponce. *Computer Vision A Modern Approach*. Prentice Hall, 2003.

[22] OpenCV. OpenCV: Camera calibration and 3d reconstruction, Oct. 2017. URL https://docs.opencv.org/3.3.1/d9/d0c/group__calib3d.html.

[23] Z. Zhang. A flexible new technique for camera calibration. In *article*, volume 22, page 1330–1334, December 2000. URL https://www.microsoft.com/en-us/research/publication/a-flexible-new-technique-for-camera-calibration/.

[24] L. Sroba, R. Ravas, and J. Grman. The influence of subpixel corner detection to determine the camera displacement. *Procedia Engineering*, 100:834 – 840, 2015. ISSN 1877-7058. doi: https://doi.org/10.1016/j.proeng.2015.01.438. URL http://www.sciencedirect.com/science/article/pii/S1877705815004658. 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014.

[25] E. Karami, S. Prasad, and M. Shehata. Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images. *ArXiv e-prints*, Oct. 2017.

[26] Inria. VISUAL SERVOING PLATFORM – tutorial: Camera calibration. http://visp-doc.inria.fr/doxygen/visp-daily/tutorial-calibration.html#calibration_circle/, Apr. 2018. Online, accessed: 5-April-2018.

[27] D. A. Brannan, M. F. Esplen, and J. J. Gray. *Geometry*. Cambridge University Press, 1999. ISBN 978-0-521-59787-6.

[28] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL http://doi.acm.org/10.1145/358669.358692.

[29] J.-S. Kim, P. Gurdjos, and I.-S. Kweon. Geometric and algebraic constraints of projected concentric circles and their applications to camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):637–642, April 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.80.

[30] Open Source Robotics Foundation. ROS/Concepts - ROS Wiki, Jan. 2018. URL http://wiki.ros.org/ROS/Concepts.

[31] Open Source Robotics Foundation. ROS packages. http://www.ros.org/browse/list.php?package_type=package&distro=kinetic/, Jan. 2018. Online, accessed: 29-March-2018.

[32] Open Source Robotics Foundation. roscore - ROS Wiki, Jan. 2018. URL http://wiki.ros.org/roscore/.

[33] Open Source Robotics Foundation. Master - ROS Wiki, Jan. 2018. URL http://wiki.ros.org/Master/.

[34] Open Source Robotics Foundation. common_msgs - ROS Wiki, Jan. 2018. URL http://wiki.ros.org/common_msgs/.

[35] Open Source Robotics Foundation. common_msgs - ROS Wiki, Jan. 2018. URL http://wiki.ros.org/std_msgs/.

[36] Logitech. Logitech c920 hd pro webcam, Jan. 2018. URL https://www.logitech.com/en-roeu/product/hd-pro-webcam-c920.

[37] Logitech Support. Hd pro webcam c920 - logitech support, Oct. 2017. URL http://support.logitech.com/en_us/product/hd-pro-webcam-c920/specs.

Appendix A

OMNI-ANT Wifi App

For controlling the Bluetooth OMNI-ANT an android application already existed, but to remote control all the other existing OMNI-ANTs only custom code existed. For the sake of flexibility and not having the need to allocate the Bluetooth OMNI-ANT, which is equipped with a LIDAR for obstacle avoidance, a new application was made.

This new app, conveniently called *OmniWifi*, is visually inspired by its Bluetooth counterpart and was also made using MIT APP INVENTOR.

The look of the app can be seen in Figure A.1.

The app allows the controlling of any of the 8 existing OMNI-ANTs by selecting its ID from the drop-down menu at the top, in **Omni ID:**. At the bottom, two sliders allow the selection of the maximum linear and rotational velocities separately.

The virtual joystick layout at the centre allows the simultaneous control of the x (linear) and z (rotation) axes velocities. The slider underneath it allows the analog control of the y axis, and the ”Move” buttons allow a digital control on that axis by ordering half the maximum velocity to the OMNI-ANT.

When a control is untouched, a stopping order is sent to the vehicle.

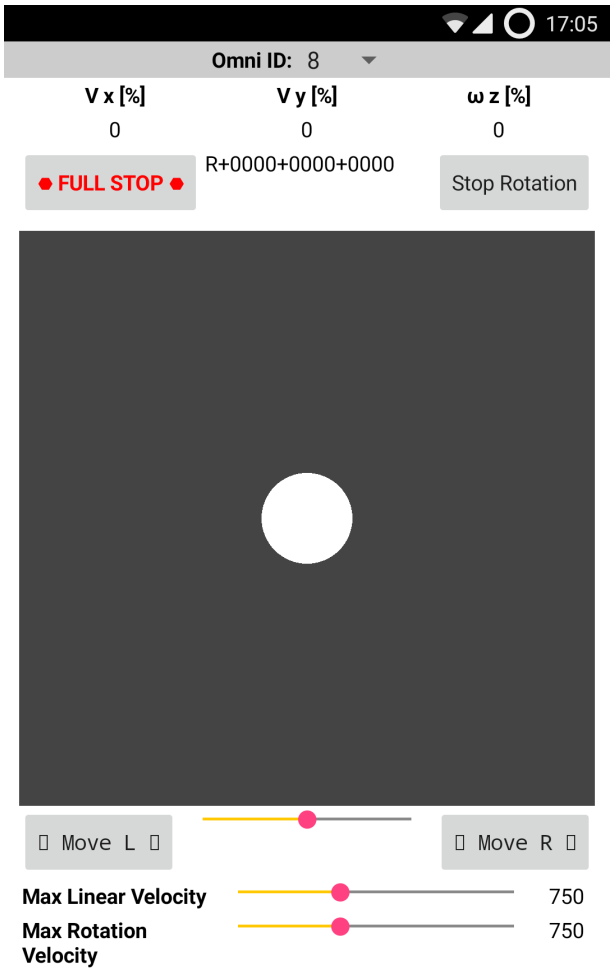


Figure A.1: Screenshot of the OmniWifi App ready