UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Diego Pavan Soler

2-DOF Laser Structure for Autonomous External Quadrotor Guidance

São Carlos

2019

Diego Pavan Soler

2-DOF Laser Structure for Autonomous External Quadrotor Guidance

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, para obtenção do título de Mestre em Ciências - Programa de Pós-Graduação em Engenharia Mecânica.

Área de concentração: Dinâmica das Máquinas e Sistemas

Supervisor: Prof. Assoc. Marcelo Becker

São Carlos

2019

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Pavan Soler, Diego
P5962 2-DOF Laser Structure for Autonomous External Quadrotor Guidance / Diego Pavan Soler; orientador Marcelo Becker. São Carlos, 2019.

Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia Mecânica e Área de Concentração em Dinâmica e Mecatrônica -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2019.

1. UAV. 2. quadrotor. 3. computer vision. 4. precision agriculture. 5. robot cooperation. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE JULGAMENTO

Candidato: Engenheiro **DIEGO PAVAN SOLER.**

Título da dissertação: "Dispositivo autônomo com 2 graus de liberdade para guiar um quadrotor".

Data da defesa: 06/09/2019

Comissão Julgadora:	**Resultado:**
Prof. Associado **Marcelo Becker (Orientador)** (Escola de Engenharia de São Carlos/EESC)	APROVADO
Prof. Dr. **Roberto Santos Inoue** (Universidade Federal de São Carlos/UFSCar)	APROVADO
Prof. Dr. **Davi Antonio dos Santos – (videoconferencia)** (Instituto Tecnológico de Aeronáutica/ITA)	APROVADO

Coordenador do Programa de Pós-Graduação em Engenharia Mecânica:
Prof. Associado **Carlos De Marqui Junior**

Presidente da Comissão de Pós-Graduação:
Prof. Titular **Murilo Araujo Romero**

This work is dedicated to my love Natália Pereira Nunes and my family Alonso Mazini Soler, Júlia Maria Pavan Soler, Júlio Pavan Soler, Vera Cristina Bastista for their endless support and always remaining me that the sun will rise again.

ACKNOWLEDGEMENTS

I would like to thank:

- To *Fundação de Apoio A Física e A Química* (FAFQ) for the financial aid;
- My advisor Prof. Marcelo Becker, Prof. André Carmona, Prof. Daniel Magalhães for their guidance and knowledge;
- My friends on Labrom, EESC and for their support and companionship;
- Clara Louzada and the Liepo technicians for their help on this project.

"Champion that which you love. He who fights for nothing, dies for nothing." Izaro Phrecius

ABSTRACT

Diego Pavan Soler **2-DOF Laser Structure for Autonomous External Quadrotor Guidance**. 2019. 126p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

This dissertation, 2-DOF Laser Structure for Autonomous External Quadrotor Guidance, has the goal of designing a novel method to attract an aerial robot to a ground vehicle using a system composed of a camera and a laser for autonomous landing. This applications is designed to be used alongside other guiding methods. This is a complex problem that requires work on different fields on both the ground and the aerial robot. On the ground station, know as Mirã, a structure was designed using 3D printed parts, and two stepper motor adjusted for 1:16 microstepping to precisely orient a camera and a laser with 2 DoF. The motors is controlled by a computer vision algorithm that uses four green LED on the aerial robot. With the combination of a light source filter and a color filter along with metric filters, the algorithm can find the LEDs position on the image. By estimating the distance of the center of the LED formation and the center of the image, the algorithm uses the camera calibration parameters to convert it to azimuth and elevation angles, allowing the structure to follow and target the laser on the aerial robot with a error of $elevation = 4.96^o$ and $azimuth = 0.27^o$ with a standard deviation of 1.03 and 0.51 degrees. To call the aerial robot, the structure will perform a linear cut on the elevation motor, because of this, the azimuth precision is made much higher. On the aerial vehicle, using a 6×6 matrix of photodectors, a $0.5 \times 0.5\,m$ sensor was designed to be capable of estimating the position of the laser dot on the sensor at a rate of $200\,Hz$. The aerial robot will decode the sensor reading and acquire multiples instances of the readings, once possible, a linear approximation of the point collection is used to estimate a vector, the direction of the vector is estimated by looking at the older and newer points, resulting in a velocity vector which the aerial robot will follow. Once the vehicle is close enough, the structure will perform a cut in the opposite direction, commanding the aerial robot to stop and allowing for reliable autonomous landing.

Keywords: UAV, quadrotor, computer vision, precision agriculture, robot cooperation.

RESUMO

Diego Pavan Soler **Dispositivo autônomo com 2 graus de liberdade para guiar um quadrotor**. 2019. 84p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

Esta dissertação, *Dispositivo autônomo com 2 graus de liberdade para guiar um quadrotor*, tem como objetivo projetar um novo método para atrair um veículo aéreo para um veículo terrestre usando um sistema composto de uma câmera e um laser para pouso autônomo. Esta aplicação foi projetada para ser usada em conjunto com outros métodos de localização. Este é um problema complexo que requer para sua solução o uso de metodologias em diferentes campos, tanto no robô aéreo quanto no terrestre. No robô terrestre, conhecido como Mirã, uma estrutura foi projetada usando peças impressas em impressora 3D, dois motores de passo ajustados para 1:16 micropasso são usados para orientar com precisão uma câmera e um laser com 2 DoF. O motor é controlado por um algoritmo de visão computacional que usa quatro LEDs verdes no robô aéreo. Com a combinação de um filtro de luz e um filtro de cor auxiliados por um filtro métrico, o algoritmo pode encontrar a posição dos LEDs na imagem. Ao estimar a distância entre o centro do objeto formado pelos LED e o centro da imagem. O algoritmo converte o valor em ângulos de azimute e elevação usando os parâmetros de calibração da câmera, permitindo assim que a estrutura siga e direcione o laser no robô aéreo, com um erro de $elevação = 4.96^o$ e $azimute = 0.27^o$ com desvio padrão de 1.03 e 0.51 graus respectivamente. Para chamar o robô aéreo, a estrutura executará um corte linear, ao mover o motor de elevação e, por este motivo a precisão em azimute é feita maior. No veículo aéreo, usando uma matriz de 6×6 fotodetectores, foi concebido um sensor de $0.5 \times 0.5\,m$ capaz de estimar a posição do laser no sensor a uma taxa de $200\,Hz$. O veículo aéreo decodifica a leitura do sensor e armazena múltiplas instâncias das leituras, então uma aproximação linear dos pontos obtidos é usada para estimar a direcão de um vetor, o sentido é estimado observando os pontos mais antigos e os mais recentes, resultando assim em um vetor velocidade que é enviado para o veículo aéreo seguir. Uma vez que o veículo esteja próximo o suficiente, a estrutura realizará um corte na direção oposta, comandando a parada do veículo aéreo e permitindo uma aterrissagem autônoma confiável.

Palavras-chave: VANT, quadrotor, visão computacional, agricultura de precisão, cooperação robótica.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

FAO	Food and Agriculture Organization of the United Nations
IMech	Institution of Mechaical Engineer
LabRoM	Mobile Robotics Laboratory
Liepo	Laboratório de Instrumentação Eletrônica
USP	Universidade de São Paulo
USPSC	Campus USP de São Carlos
EMBRAPA	Brasilian Agicultural Research Corporation
LIBS	Laser-Induced Breakdown Spectroscopy
UAV	Unmanned Aerial Vehicles
GPS	Global Positioning System
DoF	Degree of Freedom
CG	Center of Gravity
RGB	Red, Green and Blue
HSV	Hue, Saturation and Value
LED	Light-Emitting Diode
CCD	Charge-Coupled Device
ROS	Robot Operation System
GUI	Graphical user interface
PTU	Pan-Tilt Unity
IMU	Inertial Measurement Unit
PID	Proportional, Integrative, Derivative
EKF	Extended Kalman Filter

CONTENTS

1 INTRODUCTION

This chapter introduces the topics of this dissertation: 2-DOF Laser Structure for Autonomous External Quadrotor Guidance. The goal of section 1.1 is to discuss the inspiration behind this work. Then section 1.2 will explain the problems that will be addressed in this work and section 1.3 will present the solution for those problems. Lastly, section 1.4 will bring a summary of the following chapters on this dissertation.

1.1 Motivation

By 2050 forecast estimates that the world population will be approximately 9.7 billion (Population Reference Bureau, 2017). This growth in population could result in a major world wide food shortage. According to Food and Agriculture Organization of the United Nations (FAO), there is a need for a 70% increase in global food production to avoid this problem (FAO, 2011).

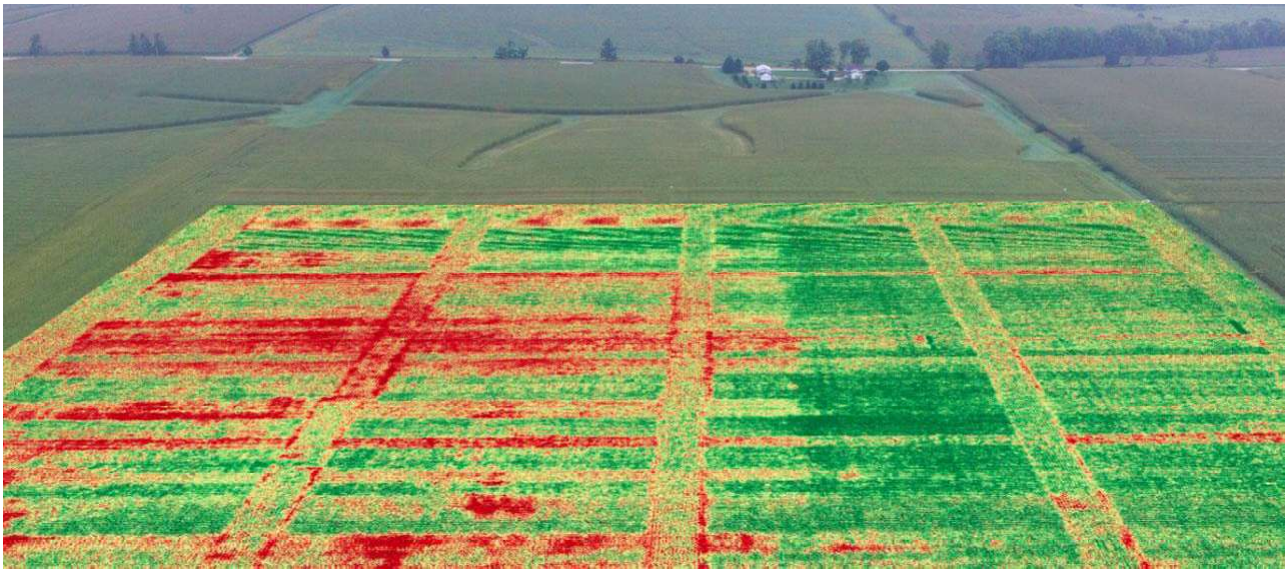
Just increasing the farming area is not a viable solution, as agriculture already consumes 70% of all water (WORLDFOODSUMMIT,) which could result in clean water shortage. Also, with the growth in population, there is also a growth in the size of cities since it is predicted that 75% of the world population will leave in cities by 2050, relying on the 25% who remain in the country to meet the quota (KELLY, 2012).

What could be done is to reduce waste towards the beginning, improving productivity. A 2013 report by The Institution of Mechanical Engineer (IMechE) shows the existence of a great amount of food waste on the beginning of production due to inefficient harvesting, inadequate local transportation and poor infrastructure (IMECHE, 2013). Besides this problem, plant diseases may be catastrophic to the whole crop if not diagnosed soon. To solve those problems, studies have been conducted to use robots in the agriculture context.

The main limitation for robotics on the field is the need for sophisticated algorithm for sensing, planning and controlling with the complicated agricultural environment. On top of that, the technology, equipment and means required for a specific crop will not be applicable to another crop or different environment (BECHAR; VIGNEAULT, 2016). In the end, the applicability for an autonomous robots for a given task on the field must depend of its economic viability.

Even with those problems, research institutes are developing autonomous robots for different tasks. (MONTGOMERY et al., 2006) and (LEHNERT et al., 2017) have designed autonomous harvester for rice and sweet pepper with good efficacy, which could, one day, reduce the harvesting labor cost. The Mobile Robotics Laboratory (LabRoM)

Figure 1: Plant Health Estimation by DroneDeploy Software



Source: <https://www.caseih.com/northamerica/en-us/products/advanced-farming-systems/field-solutions/case-ih-uav> - accessed on May/12/2018.

from São Carlos School of Engineering (EESC) of USP in partnership with Brazilian Agricultural Research Corporation (EMBRAPA) are developing a robotic platform capable of monitoring field condition and detecting diseases in soybean (HIGUTI et al., 2017).

Mirã is a 4WDS (Four Wheels Drive and Steeering) Vehicle designed to navigate between the soybean rows and is equipped with a Laser-Induced Breakdown Spectroscopy (LIBS) system described in (Santos Junior et al., 2006), (Larenas, M.C, Magalhães, D.V. and Milori, 2016) and (Larenas, M.C, Milori, D.M.B.P. and Magalhães, 2016) for monitoring the ground characteristic of the crop. The platform also carries a UAV for aerial crop investigation, which is the scope of this work.

The use of Unmanned Aerial Vehicles (UAVs) for aerial images acquisition for precision agriculture is greatly increasing. The reason to UAVs be conquering market for this application is because of the high cost and good weather requirement of the previously top technology: satellites and airplane images. High resolution aerial images are being used to measure plant health, as seen in Figure 1, identifying crop stress and solution in plant counting. All this method are already implemented and being used in many platforms, such as the work done in (SANKARAN et al., 2015) and (MENENDEZ-APONTE et al., 2016), and the commercial platform DroneDeploy (REGGIANI; POLITECNICO, 2015).

1.2 Problem Statement

The goal of the cooperation between the ground platform and the UAV is the ability for more detailed data acquirement. The UAV is capable of agile flight, quickly flying through the field while high enough to avoid crop damage. During this process, the UAV would acquire images that could estimate disease, weed and plagues, feeding this information to the ground station, which is able to carry a larger number of sensor than the UAV, and then evaluating the problem with more detail and complexity.

Nowadays, commercial UAVs are already capable of flying autonomously through waypoints, such as the DJI Mavic and multiples open-source projects, (Fraundorfer et al., 2012). Another problem on the planned robot cooperation was autonomous landing on a mobile ground platform. This was solved on (RODRIGUES, 2017), where the author divided the problem in two categories: the motion control, where he commands the UAV to first position himself on the top of the landing area and then descend, and the state estimation task, where the UAV uses computer vision and a AprilTag, described on (OLSON, 2011) and (WANG; OLSON, 2016), on the ground robot to solve for the six degrees of freedom (DoF) problem.

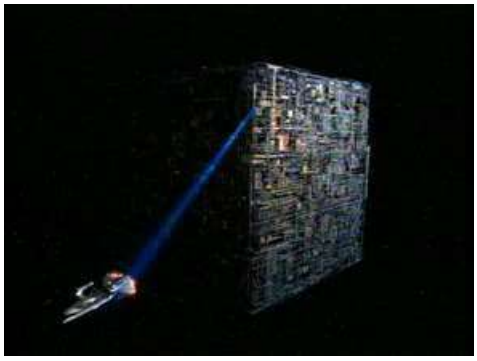
The problem of this solution is the range in which it works, which depends on the size of the AprilTag. For an 0.11 x 0.11 m AprilTag, the state estimation works for as far as a horizontal distance of $2 : \mu$ and a height of 1 m from the tag, while using a camera with a fisheye lens. Then it is imperative that both robots have very good position estimation and path planning algorithm.

There is extensive research on the field of quadrotor position estimation. The most basic position estimation is by using an onboard GPS, but the average GPS error for civilian applications are 4.7 m (vertical) and 3.4 m (horizontal) (FAA GPS Product Team, 2014), the solution being to apply filter on the GPS reading, such as EKF, Extended Kalman Filter, introduced on (KALMAN, 1960) and explained on (WELCH; BISHOP, 2006), finally used for quadrotor navigation on (OH; AHN, 2015). Another method to improve position estimation is using visual odometry, such how it is used on (Fraundorfer et al., 2012) and (FORSTER et al., 2017), some being also capable of live 3D-mapping and navigation (FAESSLER et al.,). Visual method, such as Optical Flow, when used alongside EKF can better improve quadrotor stability (CONROY et al., 2009). Then, there is also the possibility to fuse all of the methods above using sensor fusion algorithm, such as the ROS package (LYNEN et al., 2013), achieving even higher precision.

In order for both robots to meet, it is viable to plan a meeting region, where both robot should head on a certain time frame. This could also be improved by establishing a communication link between both robots, that could be bluetooth, radio or Wi-Fi.

But all those methods are not immune to fail. Without a reliable GPS reading,

Figure 2: Borg cube holding USS Melbourne in a "Tractor Beam" on Star Treck



Source: <https://memory-alpha.fandom.com/wiki/Battle_of_Wolf_359> - accessed on Jun/26/2019.

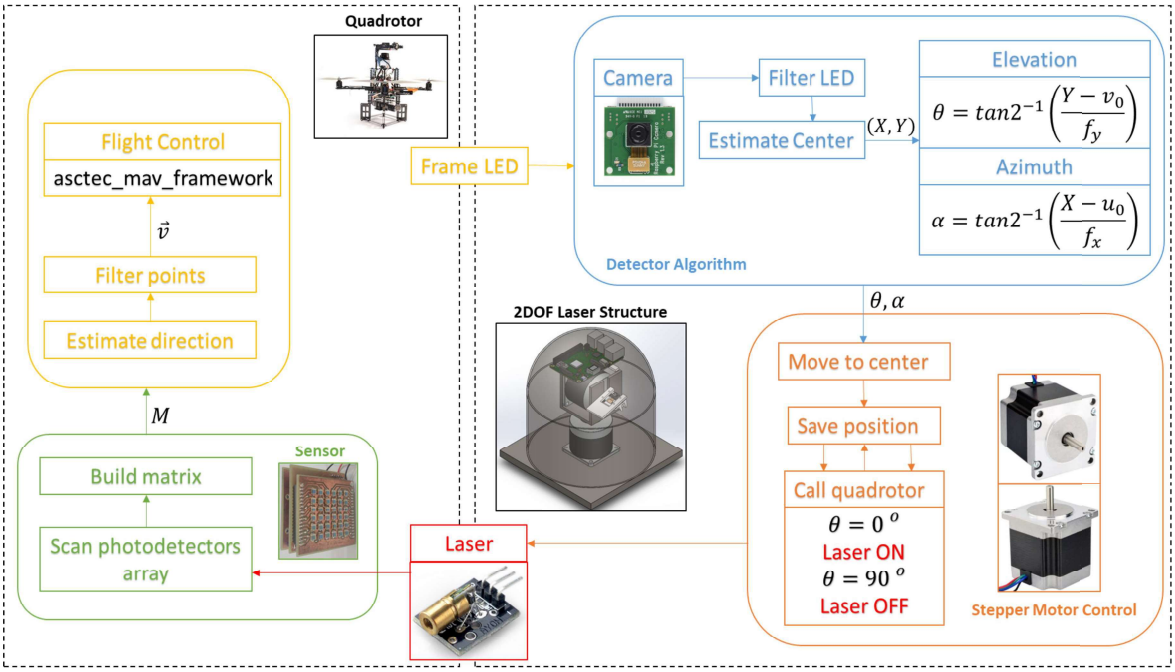
which depends on the number of available satellites and a favorable positioning, even the best filter will diverge, which happens a lot on rural regions or bad weather conditions. Optical flow is used alongside GPS, as it is a measurement of velocity, not position, and is highly dependent on the amount of features on the image, which reduces as the plant grows. Lastly, communication between robot is not reliable, as the messages could be easily corrupted or delayed due the same weather conditions as before. To better improve this cooperation, the author proposes a new method to bring the quadrotor back to the landing pad to be used along side other methods.

1.3 Proposed Solution

To achieve the goal of developing a system to be used alongside other methods with the goal of bringing the UAV back to the ground robot, a novel device is created, the 2-DOF Laser Structure for Autonomous External Quadrotor Guidance, also referred in this work as "Tractor Beam". The solution consists of using a laser to guide the quadrotor on the direction it should move. The inspiration for this work range from science fiction, such as the tractor beam used by the *Borg cube*, Figure 2, on *Star Trek* and also used by the *Death Star* on *Starwars*, and from patents on laser guidance, (TEACH, 1978) and (Andrew A. FrankMasahiko Nakamura, 1991), laser targeting (Andrew A. FrankMasahiko Nakamura, 1991) and for example using laser for autonomous comet landing (Johnson; Miguel San Martin, 2000). This project is more similiar to the science fiction examples, because the laser is used by a third-person device rather then the quadrotor itself.

The Tractor Beam design problem consists in multiple specific configurations. First, a laser must be mounted on a structure, with at least 2 DoF: azimuth, ranging from 0 to 180^o, and elevation, ranging from 0 to 90^o, such as sun-tracking system for solar panel (ROTH; GEORGIEV; BOUDINOV, 2004). Then, a RGB camera is mounted alongside the

Figure 3: Laser Structure and Quadrotor interaction diagram



Source: Authors own

laser on the structure. Using computer vision, the camera algorithm will locate the UAV's position in the image and estimate the rotation needed to hit it with the laser. The UAV will be equipped with LEDs with the goal to communicate with the Tractor Beam, for instance, when it is ready to land, and to make it more visible, facilitating the computer vision algorithm.

A sensor is designed to solve the problem of how the UAV will sense the laser. It will be positioned underneath the UAV and it consists of a matrix of photodetectors sensor, which will detect the position of the laser dot, in a way similar to how a CCD camera works. The sensor will estimate the direction which the laser is moving and feed this information to the UAV control structure as a velocity vector. A diagram illustrating the interaction between each component of this project is presented on Figure 3.

The system was designed to be capable of communicating with a quadrotor at a maximum distance of 20 m from the Tractor Beam. This distance was obtained assuming the quadrotor would fly at a reasonable height of 10 m and the landing platform would be at the edge of the image, assuming the fisheye lens opening of 60^o.

1.4 Dissertation Outline

The topics of the following chapters are:

- **Chapter 2 - Preliminaries:** Introduces the reader with all the necessary background and resources that will be used in the rest of the work;
- **Chapter 3 - Quadrotor:** Exposes all the necessary information about the UAV used in this work;
- **Chapter 4 - Laser:** Explains all design consideration for the Tractor Beam structure and drone detecting method;
- **Chapter 5 - Sensing:** Explains method used in sensing the laser and controlling the UAV towards the Tractor Beam;
- **Chapter 6 - Conclusions:** Concludes this work and propose future works.

2 PRELIMINARIES

This chapter will introduce the basic concepts used in this work. The section 2.1 introduces the coordinate frame and the notations used, being based on (CRAIG, 2004). Inspired on the explanation in (CORKE, 2011), (Richard Hartley, 2003) and (Kostas Daniilidis, 2017). section 2.2 introduces the basic of computer vision and camera calibration. section 2.3 will explains the middleware used on the aerial and ground vehicle, this section is based on (JOSEPH, 2015). Lastly, based on (CRAIG, 2004) and (Vijay Kumar,), section 2.4 introduces the quadrotor coordinate frame formulation and the equation of motion used to fly the vehicle in a 3D space.

2.1 Basic Definitions

2.1.1 Notations

For this dissertation we will use the following notation convention. This convention was inspired by (CRAIG, 2004):

1. Mathematical variables are written in italic, uppercase letters represent points and matrix, lowercase are scalars and lower case bold vectors.

2. The absolute value of a scalar a is $|a|$. The 2-norm of a vector a is represented as $||a||$;

3. Leading subscripts and superscripts identify which coordinate system a quantity is written in. For example, ${}^{A}P$ represents a position P written in coordinate system {A}, and ${}^{A}_{B}R$ is a rotation matrix from coordinate system {B} to {A};

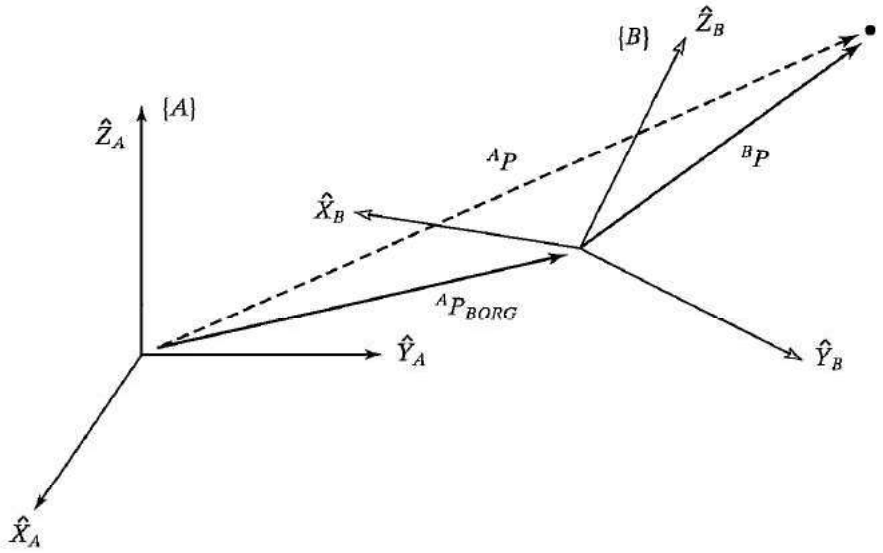
4. Trailing superscripts are used for indicating the inverse or transpose of a matrix, respectively as follows, R^{-1} and R^{t};

5. Trailing subscripts may indicate a vector component, such as v_x, or may represent a description P_{camera}, the position of the camera;

6. Vectors are taken to be column vectors; hence, row vectors will have the transpose indicated;

7. A matrix $A_m \in \mathbb{R}^m$, which contains m-rows and n-columns. Then, we have that a_{ij} is the element in the i-th row and j-th column.

Figure 4: Coordinate Frame Transformation



Source: (CRAIG, 2004)

2.1.2 Coordinate Frame

In robotics it is common to have many sensors returning the position and orientation of a body in respect to the sensor, such as a camera. For convenience, we define a frame that is a set of four vectors giving position and orientation information (CRAIG, 2004). A coordinate frame will be represented as a uppercase letter between brackets, e.g. the camera frame is represented by {C}.

With different sensors, it is possible that the definition of a vector is known, with respect to some frame {B}, but if we would need with respect to frame {A}, we could have the same case as in Figure 5.

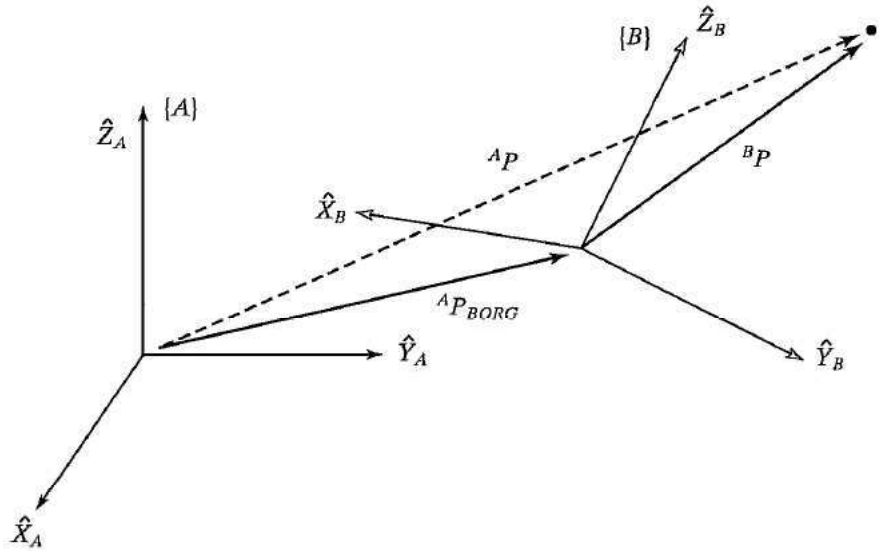
This calculation is possible when we know the combination of rotation and translation between frame {B} and frame {A}. For example, we want the position vector ${}^BP$ in frame {A} and we know that frame {B} is defined in frame {A} as the rotation ${}^A_BR$ and the translation ${}^A\mathbf{p}_B$, then we can map ${}^BP$ in frame {A} as:

$$ {}^AP = {}^A_B R\, {}^BP + {}^A\mathbf{p}_B \tag{2.1} $$

We can then define a homogeneous transformation T, as:

$$ \begin{bmatrix} {}^AP \\ 1 \end{bmatrix} = \begin{bmatrix} {}^A_BR & {}^A\mathbf{p}_B \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^BP \\ 1 \end{bmatrix} \tag{2.2} $$

Figure 5: Coordinate Frame Transformation



Source: (CRAIG, 2004)

then,

$$^AP = {}^A_B T\, {}^BP \tag{2.3}$$

2.2 Camera Model

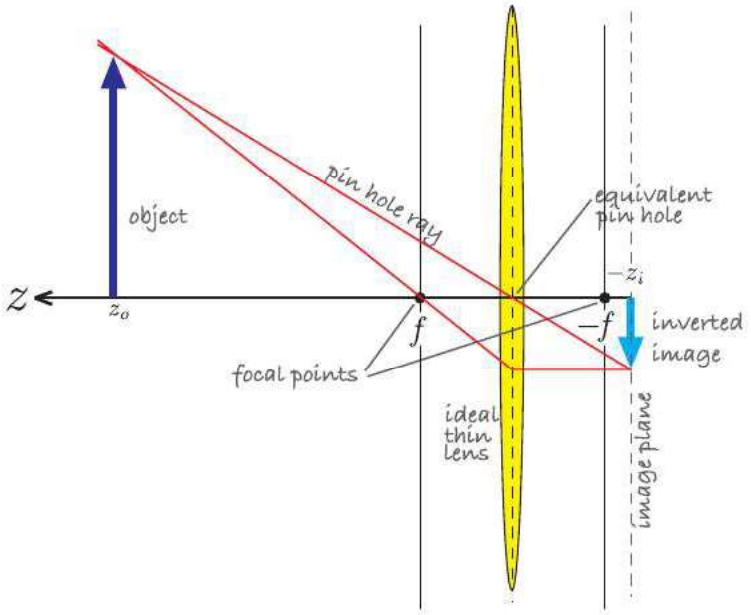
2.2.1 Perspective Projection

The pinhole camera is based on the fact that a small hole, a pin-hole, is able to create an inverted image on the wall of a darkened room. Following this same principle, a digital camera uses a glass or plastic lens to project an image on the surface of a light sensitive sensor.

The problem with the pinhole camera is due to the size of the pinhole; only a small amount of light is capable of traversing the hole, resulting in a very dim image. With the goal of accomplishing brighter images one may use a lens or curved mirror to collect light over a larger size. For most of the digital cameras, a convex lens is used.

Figure 6 presents the elemental process of image formation with an ideal thin lens. The point in which the light rays converge is called *center of projection*. The origin of the z axis is the *center of projection*, this axis being called the *optical axis* of the camera. The

Figure 6: 2D image formation for a thin convex lens



Source: (CORKE, 2011)

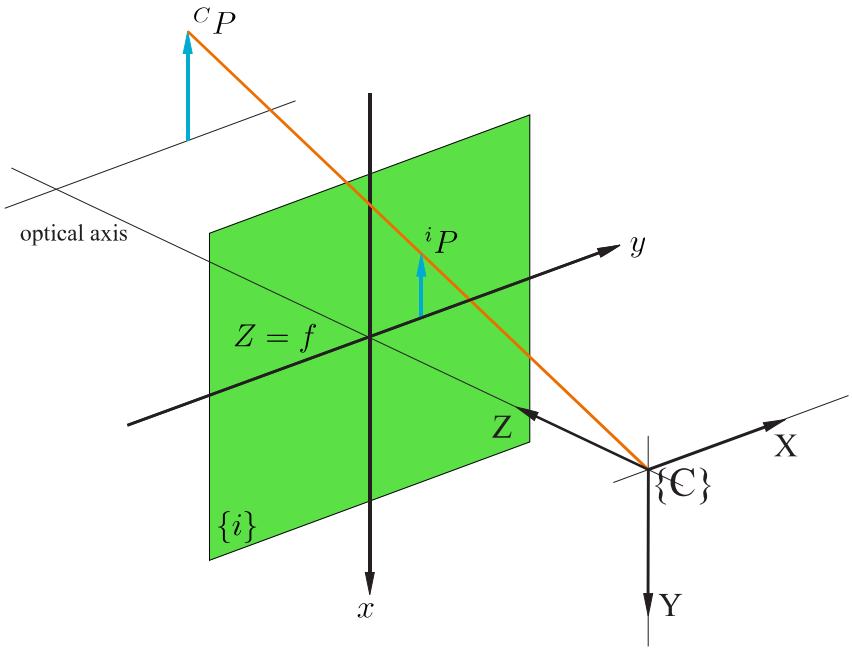
position of the object and its image are related by the lens law, given by:

$$\frac{1}{z_o} + \frac{1}{z_i} = \frac{1}{f} \tag{2.4}$$

where z_o is the distance from the lens to the object, z_i the distance from the lens to the image plane, and f is the focal length of the lens. The lens law gives the distance of the inverted image plane z_i, that a given object must be for a sharp image. In a digital camera, the image plane is fixed at the surface of the sensor, so the distance z_i is adjusted by moving the lens through the focus ring. By inspecting the lens law, for a given object at infinite, or for a point where $z_o >> f$, we have that $z_i = f$, which would be the same for a pinhole at a distance f from the image. The need to focus is a tradeoff for the sharper image.

To find the perspective transformation from a point at the world coordinate ${}^{C}P = (X, Y, Z)$ to its projection on the image plane $\{i\}$ at ${}^{i}P = (x, y)$, the central-projection model is commonly used, as shown in Figure 7, where the image plane is fixed at $Z = f$, also known as principal point, and $\{\mathrm{C}\}$ is the camera frame origin. Using similar triangles, we can show that the projective transformation is represented by Equation 2.5

Figure 7: The Central Image Projection Model



Source: Authors own, inspired on (CORKE, 2011)

$$x = f\frac{X}{Z} \tag{2.5a}$$

$$y = f\frac{Y}{Z} \tag{2.5b}$$

This transformation represents a mapping from 3-dimensional space to the 2-dimensional image plane: $\mathbb{R}^3 \mapsto \mathbb{R}^2$. Each point in the image plane represents a ray CP in the world-coordinate, resulting in no unique (x, y) mapping. On this projection lines and conics are projected as lines and conics, respectively, but angles and shapes are not preserved. Parallel lines in the world are projected as lines that intersect at a point in the image, those points being called vanishing points. The line formed by connecting two vanishing point on the ground plane results in the horizon image, which shows the orientation of the ground plane in respect to the camera.

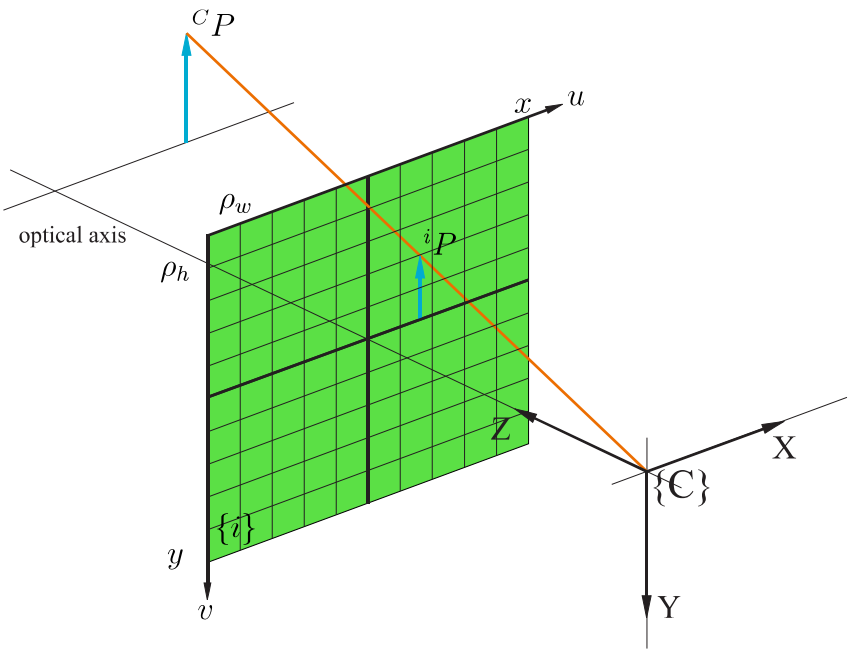
The homogeneous coordinate space is able to represent points and lines to infinity in a convenient way. By converting the world coordinates in homogeneous form $^{C}P = (X, Y, Z, 1)$, and also writing the image plane point in homogeneous coordinates, $^{i}P =$

(x, y, z), the perspective projection can be written in linear form as:

$$ {}^{i}P = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.6} $$

The image output of a camera consists of a grid pixels, as shown in Figure 8, which are picture elements, corresponding to the light sensitive elements of the camera. Each pixel in the image has its coordinate (u, v) in the 2-dimensional space, where the origin, by convention, is located at the top left hand corner of the image plane. The pixels are not necessarily squares, they have a width ρ_w and a height ρ_h. Lastly, the pixel coordinate is related to the image plane coordinate by

Figure 8: The Pixel Plane



Source: Authors own, inspired on (CORKE, 2011)

$$ u = \frac{x}{\rho_w} + u_0, \tag{2.7a} $$

$$ v = \frac{y}{\rho_h} + v_0, \tag{2.7b} $$

where (u_0, v_0) is the principal point, the point where the optical axis hits the image plane.

Now, Equation 2.6 can be written for pixel coordinates in homogeneous coordinates as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.8}$$

where $\alpha_x = \frac{x}{\rho_w}$, $\alpha_y = \frac{x}{\rho_h}$. As long as the intrinsic camera parameters K is known, defined as ${}^iP = K{}^CP$, it is possible to map camera coordinate into pixel coordinates. The matrix K can be found by a process called camera calibration.

The last step in perspective transformation is calculating the rotation and translation from the world frame to the camera frame. This transformation can be easily represented by applying a rotation followed by a translation, as in Equation 2.3. This will be used to estimate the position and rotation of the drone in world coordinate frame based in visual feedback.

Adding this transformation to the previous model yields,

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \alpha_x f & 0 & u_0 & 0 \\ 0 & \alpha_y f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^A_B R & {}^A\mathbf{p}_B \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.9}$$

which can be written as,

$${}^iP = K^C_W T^W P \tag{2.10}$$

where ${}^C_W T$ is the world frame to camera frame transformation matrix.

2.2.2 Camera Calibration

The goal in camera calibration is to calculate the intrinsic and extrinsic parameters of the camera. In the real world camera, the principal point (u_0, v_0) is not at the center of the sensor. Also, according to (CORKE, 2011), the focal length f of a lens is only accurate to 4%, and also, the focal length can be adjusted. The only parameter that is constant is ρ_w and ρ_h (pixel size), and they can be found in the sensor manufacturer's data sheet.

The are many great works in the topic of camera calibration. In this dissertation, every camera was calibrated using the ROS package *image_pipeline: camera_calibration*, (James Bowman,). This package uses OpenCV's camera calibration algorithm which is based on (ZHANG, 2002) and (J.Y.BOUGUET, 2014). Similar to the Matlab package, this ROS package works with the user sending the algorithm images of a known planar

Figure 9: Camera Calibration GUI

Source: (James Bowman,)

chessboard pattern, which is called calibration rig, and is used to estimate the extrinsic camera matrix and the distortion parameters.

The main advantage of the ROS package over the Matlab package is its automation, making it simpler to acquire many more images in unique rotations and translations, improving the accuracy. By running the calibration node, a GUI will open, as the one on Figure 9. The goal is to move the calibration rig in different position, and holding it until the image is highlighted. The X, Y and Size show the amount of unique position on the horizontal, vertical and depth of the image, respectively. Once all bars are full, it is possible to start the calibration process, which is the same process as the Matlab package.

2.2.3 Image Distortion

A problem in robotic vision is the radial image distortion due to a wide angle camera lens, as seen in Figure 10, in which the distortion is much bigger on the side of the image. If the camera is calibrated properly it is possible to use a rectification algorithm to correct the image distortion, obtaining a image.

A image coordinate point (u, v) and its distorted coordinate (u_d, v_d) are related by

Figure 10: Example of image distortion due to a wide angle camera lens. Left: Image after correction, Right: Image before correction.

Source: (James Bowman,)

Equation 2.11.

$$(u_d, v_d) = (u + \delta_u, v + \delta_v) \tag{2.11}$$

where the displacement of each pixel (δ_u, δ_v) is a function of the radial and tangential distortion components and the distance r of the pixel element from the principal point given from Equation 2.12.

$$\begin{bmatrix} \delta_u \\ \delta_v \end{bmatrix} = \begin{bmatrix} u * (k_1 r^2 + k_2 r^4 + k_3 r^6 + ...) \\ v * (k_1 r^2 + k_2 r^4 + k_3 r^6 + ...) \end{bmatrix} + \begin{bmatrix} 2p_1 uv + p_2(r^2 + 2u^2) \\ p_1(r^2 + 2v^2) + 2p_1 uv \end{bmatrix} \tag{2.12}$$

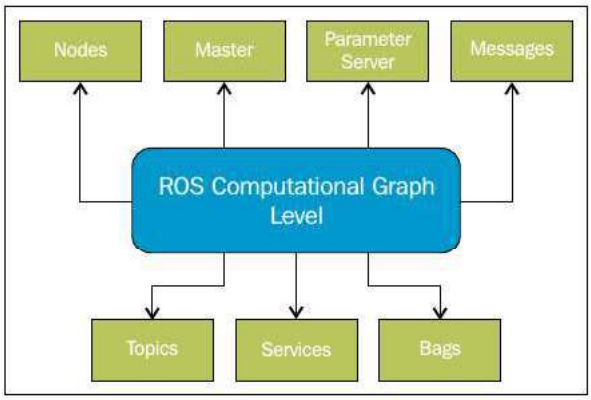
Typically tree coefficients k_i are enough to describe the distortion. The distortion is modeled using $(k_1; k_2; k_3; p_1; p_2)$, which are considered as addition intrinsic parameters to be estimated during the calibration. Those values can also be estimated by calibrating the camera. This process is done by the ROS package *image_proc*, which access the camera intrinsic parameters and publish the rectified image of the camera.

2.3 Middleware: ROS

This section was inspired in the book (JOSEPH, 2015). The Robot Operating System (ROS) is a flexible framework for writing robot software, that provides many features that simplify the task of creating complex robot behavior. It is supported by a growing community with users worldwide that develop great pieces of work that are available for free and can be adapted easily for other applications.

The advantages of ROS on other middleware applications are: High-end capability, such as SLAM (Simultaneous Localization and Mapping), helpful debugging tools, support high-end sensors and actuator, such as camera drivers, modularity, active community and more.

Figure 11: ROS Computation Graph



Source: (JOSEPH, 2015)

The basic unit of the ROS software is a package that contains the ROS runtime process (nodes), libraries and configuration files, all in a single unit. ROS supporters share their package to be used by others in different applications, for example, *asctec_mav_framework* and *camera_calibration*.

The computation in ROS is done using a network of the process called ROS nodes. Figure 11 presents the main concepts in the computation graph.

The basic elements of the ROS computation are as follow:

- **Nodes:** Nodes are the process that perform computation. A node can be a publisher, subscriber or, more commonly, both. A publisher publishes messages in a specific topics. The subscriber subscribes and receives new messages from topics.

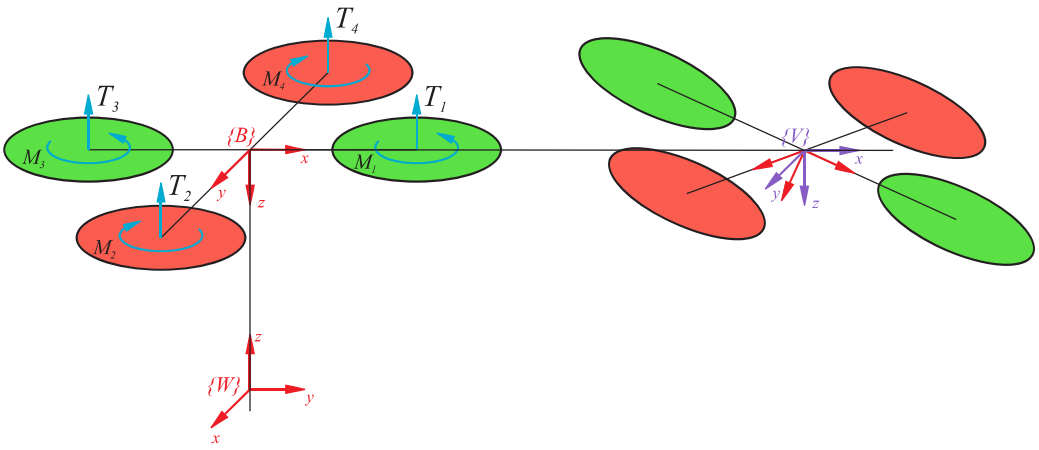
- **Master:** A ROS master provides name registration and lookup to the other nodes. By creating a single master, nodes are able to find each others and exchange messages.

- **Parameter Server:** Is a single location to store date. All nodes can access and modify these values.

- **Messages:** A node will send and receive data through a message. Messages are a simple data structure field, which can hold a set of data. Nodes must know which type of message they work with.

- **Topics:** A ROS topic is like a pipeline. In one side, a single publisher publishes a message through the pipeline, and in the other end, a subscriber receives this same message. Each topic has a well defined message topic and name.

Figure 12: Quadrotor Coordinate Frame



Source: Authors own, inspired on (CORKE, 2011).

- **Services:** Services work similar to messages, but they are a one shot communication between two nodes.

- **Bags:** Bags are a format for saving and playing back ROS message. They can record multiple nodes and then playing back with the same structure as before, making it possible to record a bag and test it with a different node.

2.4 Quadrotor Formulation

Flying robots differ from ground robot in two important ways. First, they have 6 degrees of freedom, three translational, forward, lateral and vertical translations and three rotations along the x, y and z axis, called roll, pitch and yaw, respectively. Secondly, their motion are expressed by a combination of forces and torques, rather than velocities.

Quadrotors are flying vehicles that consist of a rigid frame connecting four propeller, and by adjusting the torque on each propeller, it can move along the three translational axes. Compared to fixed wing aircraft, they are highly maneuverable, easier to takeoff and land and can safely fly indoors. Compared to conventional helicopters, their mechanisms are much simpler and are easier to model and control.

2.4.1 Quadrotor Coordinate Frame

Before modeling the quadrotor, we define the quadrotor notation in Figure 12. First, we have a world frame $\{W\}$, where the axes orientation will follow the North, east, down (NED) system, where we have the axes x_w, y_w and z_w, respectively, pointing North, East and Down.

The body fixed frame $\{B\}$ is defined with origin located in the quadrotor center of gravity (CG), where the z_B-axis is downward by convention and the x_B-axis is aligned with the front and rear rotors.

To move the quadrotor parallel to the ground, for example in the x direction, the vehicle must pitch the nose down to generate a force. To accommodate those changes in directions we define a vehicle frame $\{V\}$ with the same origin as $\{B\}$ but with its x and y axes parallel to the ground.

2.4.2 Quadrotor Equation of Motion

For a quadrotor, each rotor rotates at an angular velocity ω_i, for $i = 1, 2, 3, 4$. Then, each rotor generates a thrust, T_i, and moment, M_i, as:

$$T_i = k_F \omega_i^2 \tag{2.13a}$$

$$M_i = k_M \omega_i^2 \tag{2.13b}$$

where k_F and k_M are constants that depend on the air density, and the rotors blade design, such as number of blades, angle of attack and length.

In order for the quadrotor to gain height, the rotors must generate a thrust bigger then the gravitational pull. Using Newton's second law, we can calculate the translational dynamics of the vehicle in the world frame $\{W\}$ as:

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} - {}^{V}_{B}R \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{2.14}$$

where $\ddot{r}$ is the position of the vehicle in the world frame, g is the gravitational acceleration, m is the quadrotor total mass, $T = \sum T_i$, is the total upward thrust, and R is a sequence of rotation in the body fixed frame as Rot(z, ψ), Rot(x, ϕ) and Rot(y, θ), resulting in the following rotation,

$$R = \begin{bmatrix} cos\psi cos\theta - sin\psi sin\phi sin\theta & -sin\psi cos\phi & cos\psi sin\theta + sin\phi sin\psi cos\theta \\ sin\psi cos\theta + cos\psi sin\phi sin\theta & cos\psi cos\phi & sin\psi sin\theta - cos\psi sin\phi cos\theta \\ -cos\phi sin\theta & sin\phi & cos\phi cos\theta \end{bmatrix} \tag{2.15}$$

By changing the rotor thrust in pairs, rotations can be generated on the x-axis and y-axis, which are called, respectively, rolling and pitching torque, and are defined as,

where d is the distance between the source of T and the origin of $\{B\}$.

$$\tau_x = dk_F(\omega_4^2 - \omega_2^2) \tag{2.16a}$$
$$\tau_y = dk_F(\omega_1^2 - \omega_3^2) \tag{2.16b}$$

Those torques exert a reaction on the quadrotor which acts to rotate it. The total reaction torque on the z-axis is,

$$\tau_x = T_1 - T_2 + T_3 - T_4 = k_M(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) \tag{2.17}$$

The difference in signs are due to the difference in rotation direction. This also shows that, by adjusting the rotor speed, a yaw torque can be created.

The rotational acceleration of the quadrotor is given by Euler‘s equation of motion:

$$I\dot{\omega} = -\omega \times I\omega + M \tag{2.18}$$

where I is the 3×3 inertia matrix, ω is the angular velocity vector, and M is the torque applied to the quadrotor as $M = (\tau_x, \tau_y, \tau_z)^t$.

Now, we can describe the forces and moments on the quadrotor in matrix form:

$$\begin{pmatrix} T \\ M \end{pmatrix} = \begin{bmatrix} -k_F & -k_F & -k_F & -k_F \\ 0 & -dk_F & 0 & dk_F \\ dk_F & 0 & -dk_F & 0 \\ k_M & -k_M & k_M & -k_M \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = A \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{2.19}$$

where the matrix A is of full rank if k_f, k_m, $d > 0$ and can be inverted. Doing so, one can find the required rotor speed to apply a specific thrust T and moment M to control the quadrotor.

2.5 Camera Pose Estimation

This section will discuss the basic concept in pose estimation through a method called Perspective-n-Point (PnP), which estimates the pose of a camera given a set of n 3D points in the world frame and its respective correspondence in the image frame. Given $2D \mapsto 2D$ transformation of a point $P = (x, y, 1)$ in the world frame and its correspondence ${}^iP = (x_{im}, y_{im}, 1)$ in the image frame, we have:

$$H^W P = {}^i P, \tag{2.20}$$

where H is a 3×3 homography matrix that maps the transformation from a set of points P to another set of points P' up to a scale. Now, the first question to consider is how many points of correspondence, $P \mapsto P'$, are needed to compute the homography matrix. As a 3×3 matrix, H has 9 entries, but it is defined up to scale, thus the total number of DoF in a $2D$ projective transformation is 8. For each point of correspondence, there are two DoF corresponding to the (x, y) components. Thus, to fully compute the homography matrix we need four $P \mapsto P'$ points of correspondence.

Now, we present a simple linear algorithm for computation, known in the literature as the Direct Linear Transformation (DLT). The $2D \mapsto 2D$ transformation is given by equation ${}^{i}P = H^{W}P$, which can be expanded as:

$$\lambda \begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.21}$$

$$\lambda x_{im} = h_{11}x + h_{12}y + h_{13} \tag{2.22a}$$
$$\lambda y_{im} = h_{21}x + h_{22}y + h_{23} \tag{2.22b}$$
$$\lambda = h_{31}x + h_{32}y + h_{33} \tag{2.22c}$$

where λ is a scalar. By rescaling equations Equation 2.22a and Equation 2.22b by Equation 2.22c we get,

$$\lambda x_{im} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{2.23a}$$
$$\lambda y_{im} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{2.23b}$$

Now we can rearrange it into matrix equation,

$$\begin{pmatrix} a_x \\ b_x \end{pmatrix} h = 0 \tag{2.24}$$

where:

$$\mathbf{a_x} = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx_{im} & yx_{im} & x_{im} \end{bmatrix} \tag{2.25a}$$
$$\mathbf{a_y} = \begin{bmatrix} 0 & 0 & 0 & -x & -y & -1 & xy_{im} & yy_{im} & y_{im} \end{bmatrix} \tag{2.25b}$$
$$\mathbf{h} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{21} & h_{22} & h_{23} & h_{31} & h_{32} & h_{33} \end{bmatrix}^{t} \tag{2.25c}$$

For each of the four points of correspondence, we can generate $a_{x,i}$ and $a_{y,i}$ vectors and concatenate them together, as follows:

$$A = \begin{bmatrix} a_{x,1} \\ a_{y,1} \\ \vdots \\ a_{x,n} \\ a_{y,n} \end{bmatrix} \tag{2.26}$$

Then, the problem will be finding $\mathbf{h}$, such that $A\mathbf{h} = 0$.

Due to noise in the measurement, there might not be a $\mathbf{h}$ such that $A\mathbf{h} = 0$, so we need to introduce an error e such that $A\mathbf{h} = e$. Thus, the goal is to find $\mathbf{h}$, which minimizes the error e. This can be done using a method called Singular Value Decomposition (SVD) of matrices, which is explained in details in (Richard Hartley, 2003).

Given a matrix A, the SVD is a factorization of A as $A = UDV^t$, where U and V are orthogonal matrices, and D is a diagonal matrix with non-negative entries. By using this decomposition, we have that $A = UDV^t$, and then the vector $\mathbf{h}$ will be the last column of V, which can be reshaped to find the homography transformation H.

2.5.1 Planar Transformation

Another possibility of SVD is to compute a planar transformation such as the one in Equation 2.27, where $T = R(\theta)R(-\theta)DR(\theta)$, with D is a diagonal matrix,

$$D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{2.27}$$

By writing SVD of the matrix T, we have $T = UDV = (UV)(UDV) = R(\theta)R(-\theta)DR(\theta)$, resulting in an approximation to the previously presented sequence of rotation.

2.5.2 Extracting Pose from Homography

Now that the homography is known, we must extract the rotation R and translation $\mathbf{t}$ from it. Assuming all the world point lie in the same $z_W = 0$ plane, we have,

$$\begin{bmatrix} x_{im} \\ y_{im} \\ z_{im} \end{bmatrix} = \begin{vmatrix} R & \mathbf{t} \end{vmatrix} \begin{bmatrix} x_W \\ y_W \\ 0 \\ 1 \end{bmatrix} = \frac{1}{z_{im}} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} x_W \\ y_W \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} \begin{bmatrix} x_W \\ y_W \\ 1 \end{bmatrix} \tag{2.28}$$

Figure 13: Deformation by a Planar Transformation

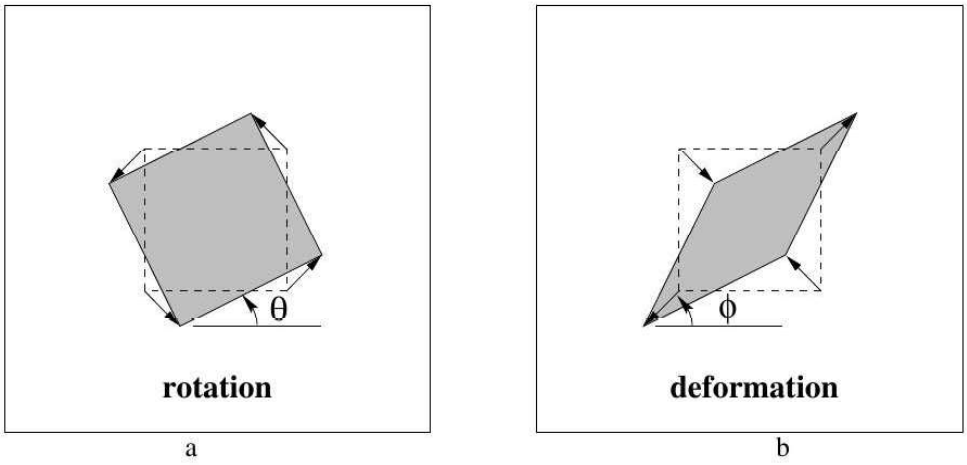


Fig. 2.7. **Distortions arising from a planar affine transformation.** *(a) Rotation by* $\mathtt{R}(\theta)$. *(b) A deformation* $\mathtt{R}(-\phi)\,\mathtt{D}\,\mathtt{R}(\phi)$. *Note, the scaling directions in the deformation are orthogonal.*

Source: (Richard Hartley, 2003)

Because there is noise in the measurement, $\mathbf{h_1}$ and $\mathbf{h_2}$ may not be orthogonal, resulting in something other than a rotation matrix. This problem can be fixed by using SVD, as follows,

$$R^t = \begin{bmatrix} \mathbf{h_1} & \mathbf{h_2} & (\mathbf{h_1} \times \mathbf{h_1}) \end{bmatrix} = UDV^t \tag{2.29}$$

Then we can finally extract the rotation and the translation from the homography using,

$$R = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(UV^t) \end{bmatrix} \tag{2.30}$$

$$\mathbf{t} = \frac{\mathbf{h_3}}{||\mathbf{h_1}||} \tag{2.31}$$

3 TRACTOR BEAM

This chapter discusses the system that will be mounted on the ground robot to assist the UAV in landing on the ground robot. The systems consist in an optical laser and a camera mounted in a two DoF base. The mechanical design and considerations for the base discusses in section 3.1. Lastly, the method for detecting the UAV in the camera image discusses in section 3.2.

3.1 2-DOF Laser Structure

3.1.1 Problem Statement

To target the laser on the UAV, the laser must be mounted on a base. As this base will be mounted on top of the ground robot and the UAV could be positioned in any direction above the ground robot, the base must have a semi-sphere workspace, which can be done with two degrees of freedon (DoF). The first step is to design a 2 DoF support to hold a camera and a laser, with a semi-sphere workspace around the base, with an azimuth range from 0 to 360^o, and an elevation range from 0 to 90^o.

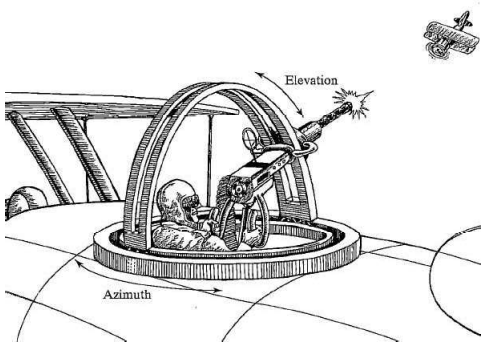
To effectively guide the UAV, the base must also be able to point the laser to the drone, and then towards the goal, in velocity that is possible for the drone to follow, thus allowing guidance.

3.1.2 Literature Review

Recently the interest in unmanned aerial vehicles (UAV) have greatly increased among the civilian, military and scientific community. As a great number of those application requires a camera mounted to the UAV it is to be expected that there is also a great number of researches on gimbals for those cameras. The idea of these gimbals unity is similar to the one needed for this project, but those gimbals our normally Pan Tilt Unity (PTU), like the one from AscTec that is used to keep the camera plane parallel to the ground plane, so it was needed to look elsewhere.

The concept used in World War I biplane rear gunner is essentially what is needed for the laser. As seen in Figure 14, with 2 Dof, azimuth and elevation, the gunner is able to target on the other plane. A good motor arrangement for those Dof is the one used on solar tracker, as described in (ROTH; GEORGIEV; BOUDINOV, 2004) and in details on the book (PRINSLOO; DOBSON,). Inspired on sunflowers, solar tracker is a device that orients a payload towards the sun, as the motor consumes energy and the sun moves slowly thought the day, the system moves in discrete steps. This is a good approach to be done on the design of the camera base.

Figure 14: A World War I biplane rear gunner



(CRAIG, 2004)

Table 1: DC Motor vs Servo Motor vs Stepper Motor ranking table

Motor	Speed	Resolution	Accuracy	Torque	Weight	Power	Cost
DC	1	3	3	1	3	2	2
Servo	2	1	2	2	2	3	3
Stepper	3	2	1	3	1	1	1

3.1.3 Actuator Selection

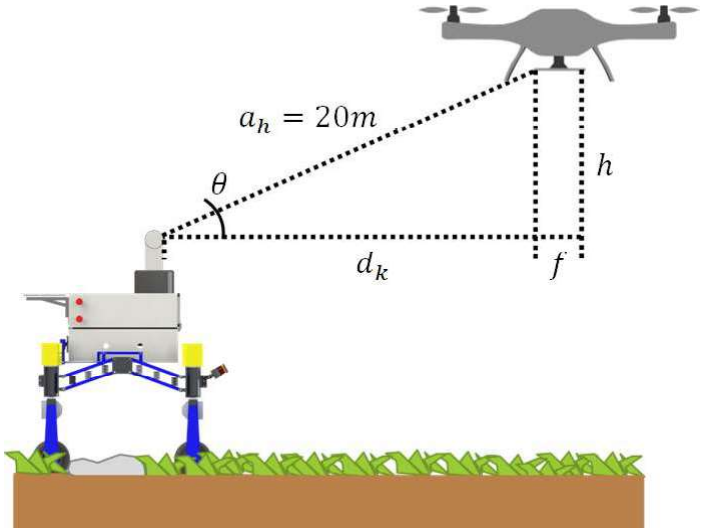
The first step in designing the laser structure is deciding which actuator to use to acquire the 2-DOF needed for this application. To make an informed decision, a research was done on the three possible alternatives, DC motor, servo motor or a stepper motor. By studying the following articles (LACKEY, 2018) and (BURRIS, 2019), checking the application of each motor and deciding of the specifications needed for this application, the Table 1 was assembled. On Table 1, the motors are ranked in each category with a value of 1 for the best and 3 for the worst.

For this application, it is expected for the quadrotor to hold its position, so speed is not a major requirement, which breaks the greatest advantage of using a DC motor, which also need a good encoder for higher resolution and has a lower accuracy. Then, it comes to either a Servo Motor or a Stepper motor.

By looking at Table 1 it is clear that servo motor offers an undeniable performance advantage, they are also used in similar applications, such as camera gimbals. But they are not the best option for this application, as stepper motor has some other advantages.

First, stepper motor offers higher accuracy than servo motor, and the resolution of a stepper motor can greatly increase with the use of micro-stepping, at the cost of holding torque. Servo motors are capable of better speed than stepper motor and with this speed they can use gear boxes to improve the resolution.

Figure 15: Motor Resolution Estimation. Image shows the horizontal movement due a angular step θ_{k-1}



Source: Authors own

In the end, a stepper motor was chosen to be used to acquire the 2-DoF on the laser structure for the following reasons:

1. Stepper motor are more compact and lighter than a servo motor;
2. A servo motor uses rare-earth magnets, greatly increasing its price;
3. Stepper motor are used on 3D printer, another equivalent application, resulting in being widely available on the market, while servo motor are much more sophisticated and hard to purchase;
4. Stepper motor does not require feedback, avoiding the need for motor encoder.

3.1.4 Estimating Required Resolution

Now subsequent to the stepper motor choice as the actuator, the second step should be deciding the required step size. For this application the motor resolution is important because, even the smallest angular movement results in a huge step far away, which means that there could be no possible configuration for the laser to hit the target.

Assuming, as is illustrated on Figure 15, the biggest horizontal movement that the laser dot is able to do, and still hit the target, is the size of the target $f = 50\,mm$, and assuming the equivalent angular movement $\Delta\theta$ is the step size, it is possible to estimate the maximum distance that relates f and $\Delta\theta$.

Now, assuming P_{k-1} is the starting position of the laser spot at a distance a_{k-1} from the laser structure and P_k is a point at a horizontal distance f from P_{k-1}, and a distance d_k from the laser, we have:

$$P_{k-1} \rightarrow a_{k-1}^2 = d_{k-1}^2 + h^2$$
$$P_k \rightarrow a_k^2 = d_k^2 + h^2$$
$$d_k = d_{k-1} - f,\ \theta_k = \theta_{k-1} + \Delta\theta$$

where $\Delta\theta$ is given by the following equation:

$$\Delta\theta = \frac{2\pi}{200(n)} \tag{3.1}$$

Equation 3.1 represents the smallest step given by the stepper motor, for the motor available in the market, 200 full steps per revolution. Using microsteps, n, it is possible to further improve the resolution, which depends on the driver capability. A possible driver allows for 1/128 microsteps, resulting in a maximum resolution of $200 * 128$ steps per revolution.

Now, using the law of the cosine we have that,

$$2d_k^2 + 2h + 2d_k f(-2cos(\frac{2\pi}{200n}))\sqrt{(d_k^2 + h^2)((d_k - f)^2 + h^2} = 0 \tag{3.2}$$

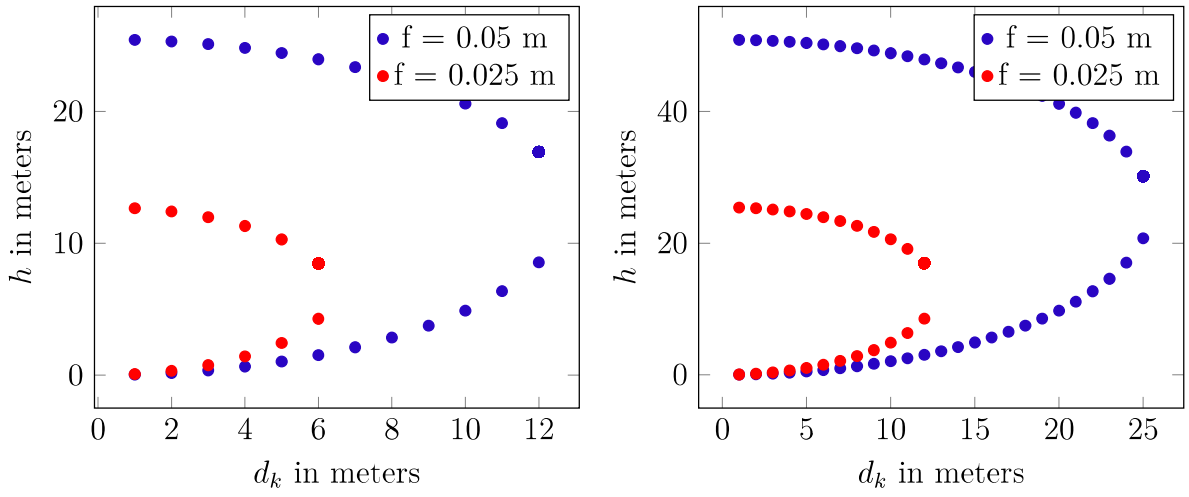
Using Matlab to solve Equation 3.2 for h, with values of d_k between 1 to 100 m for as long as there is a $h \in \mathbb{R}$, and plotting the results on Appendix B for the possible microsteps configuration available on the motor driver, $n = [4, 8, 16, 32, 64, 128]$, and a sensor size $f = 50\,mm$ and half-size, $f = 25\,mm$.

For each amount of microstep on Figure 16 we have a maximum curve in blue, which represents the size of the sensor, and a red curve, which represents half of the sensor size. As long as the quadrotor distance and height from the laser structure is inside the region between the curves, it is possible for the laser to hit the sensor.

By looking at Equation 3.2, at first glance, one could conclude that more microsteps is equivalent to better resolution and then improving the laser range, which is true, but the cost of increasing the microstep is reducing the holding torque, resulting in a mechanical limitation. To avoid this problem, the amount of microstep should be the amount required.

For this application, the quadrotor will fly at a height of around 20 m from the laser structure and should be at a horizontal distance of around 10 m, not farther than 25 m. Therefore, a microstep of 16 or 32 could be enough if the motor is able to hold the torque, to make this decision, it is important to find how impactful is increasing microstep

Figure 16: Horizontal Distance vs Height for comparison between the driver 16 and 32 microsteps configurations



(a) 1:16 Microsteps

(b) 1:32 Microsteps

Source: Authors own

Table 2: Holding torque loss for increasing microsteps

Microsteps/full step	**% holding torque/microstep**
1	100.00
2	70.71
4	38.27
8	19.51
16	9.80
32	4.91
64	2.45
128	1.23

Source: (BUDIMIR, 2003)

on the hold torque to avoid mechanical limitations and to set an acceptable accuracy for each step.

The article (BUDIMIR, 2003) computes the Table 2, which shows the loss of holding torque due an increase in the number of microsteps per full step. With a microstep of 16, the holding torque would be reduced to 9.8% of its nominal value, while 32 would reduce even more, to 4.9%, possibly resulting in the need of a much bigger motor.

Taking this information in consideration, it was possible to choose the actuator to acquire the 2-DOF needed for this application, and with the goal of reducing the mass of the structure, it was decided to choose a microstep of 16, resulting in a step angle of 0.1125^o, which is enough for this application.

Table 3: Stepper Motor Specifications

Item	**Specifications**
Connection	Bipolar
Holding Torque	4.6 $kgf.cm$
Peak Current	1.4 A
Step Angle	1.8^o
Max. Operation Temperature	80 ^oC
Mass	0.42 kg

3.1.5 Hardware Specifications

The stepper motor chosen to be used on the laser structure is the model *NEMA 23* from the manufacture *AKIYAMA MOTORS*, <https://s3-sa-east-1.amazonaws.com/robocore-lojavirtual/570/dat-I000387_R03.pdf>, (23-05-2018), and a compatible stepper motor driver, *AKDMP5-2.2A*, <https://s3-sa-east-1.amazonaws.com/robocore-lojavirtual/611/man-i005759-rev04.pdf>, (23-05-2018). The relevant technical specifications of the stepper motor are listed in Table 3.

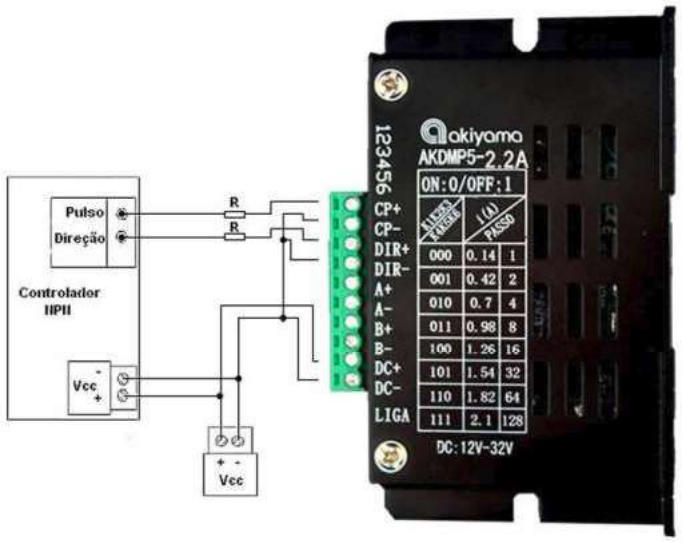
A Raspberry Pi was chosen to move the motor and to compute all the information needed for the laser to hit the sensor underneath the quadrotor. To move the motor, a PWM, Pulse Width Modulation, signal must be sent to the driver with a duration no longer then 5 μs. The number of pulses, on the PWM signal, are the number of steps that the motor will take and the frequency is the speed of the motor. The best motor speed depends on the structure. With this in consideration, the most appropriate speed of the motor was with a signal frequency of 1250 Hz, which was the fastest without any noise or vibration, at any point of the trajectory, on the structure.

The stepper motor driver is connected to the Raspberry Pi GPIO pin, using the PNP configuration, presented in Figure 17 and connected to a $12V$ $10A$ power source. To use the Raspberry Pi at its peak efficiency, it must be feed with $5V$ and $3A$, so a $5V$ voltage regulator was connected directly to the Pi's V_{in} GPIO pin.

Also connected to the raspberry PI, there is a 5 mW laser, and a raspberry camera module v1, whose resolution is 5 Mb.

For the electrical design, solid cables are used for static components, such as the azimuth motor and both driver, and flexible cables are used for dynamic connections, such as all Rapsberry Pi GPIO connections. The size of the flexible cables and the mechanical design also considers that the cables would wind on the structure, allowing a maximum azimuth range of 720^o. Lastly, due the camera connector, the maximum elevation range is around 120^o.

Figure 17: PNP Configuration Connection between each driver, each motor and the Raspberry Pi.



Source: AKDMP5-2.2A Spec Sheet

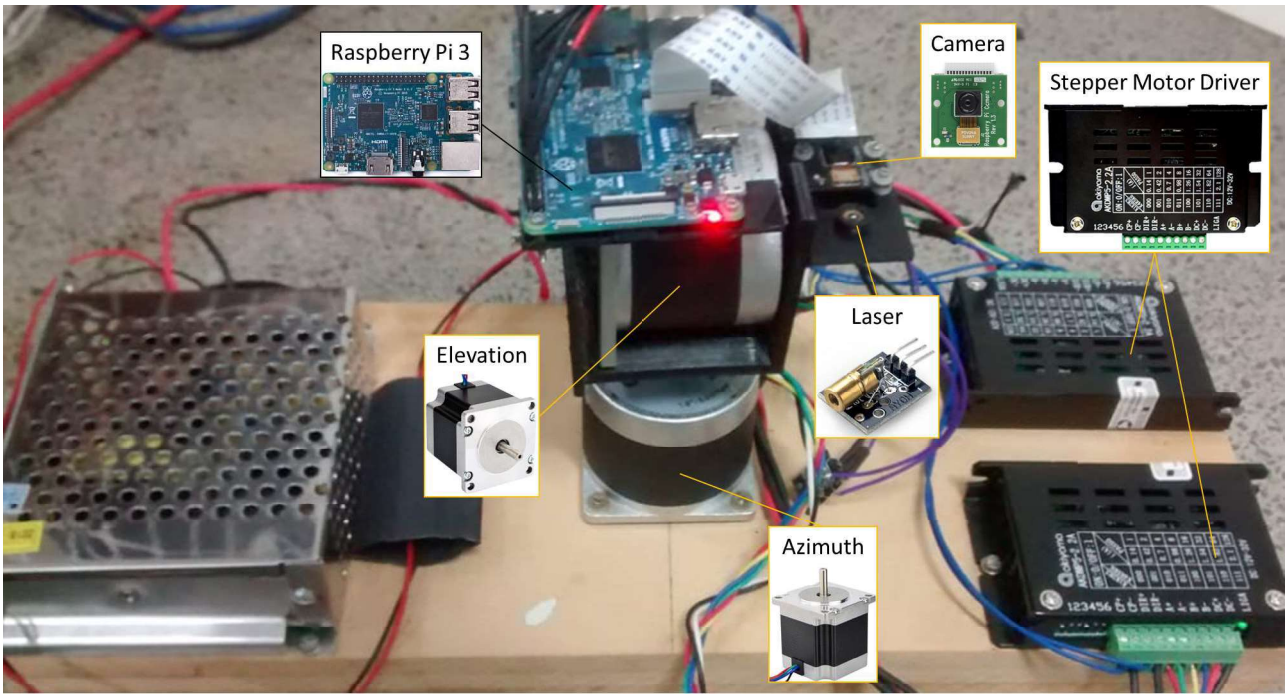
3.1.6 Structural Design

With both stepper motor selected and all the other components specified, it is possible to design the structure that will the laser the 2-DoF needed. The final design has the following specifications:

- Bottom motor, or Motor 1, controls the azimuth of the laser;
- Top motor, Motor 2, controls the elevation;
- Motor 2 CG is positioned on Motor 1 axis of rotation;
- Camera lens is positioned on Motor 2 yz plane;
- Laser is positioned on the camera xz plane, on x-axis;
- There is no motor encoder, position tracking is done by software;
- The elevation is limited from 0^o to 90^o and the azimuth from -360^o to 360^o;
- All structural elements are 3D printed using PLA.

The structure will mount on Mirã, at the edge of the heliport, but installing it would complicate the testing and debugging, then, the structure was first mounted on a wooden

Figure 18: 2 DoF Laser Structure. On the figure, all components are tagged.



Source: Authors own

Table 4: Link parameters of the two-link manipulator

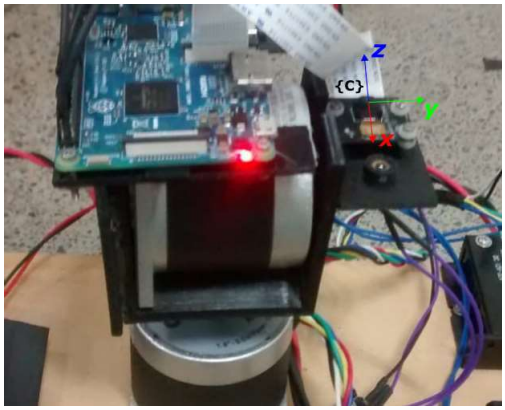
i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	l_1	θ_1
2	-90	0	l_2	θ_2

foundation that can be easily moved. Figure 18 shows a picture of the Laser Structure experimental setup, where all the components are high lighted. Figure 19 presents a close view of the camera with its axis. On Appendix C, there is a collection of the structure CAD images, showing how the motors are positioned, how the structure would be mounted on Mirã and the protection cover that was designed.

3.1.7 Translation Matrix

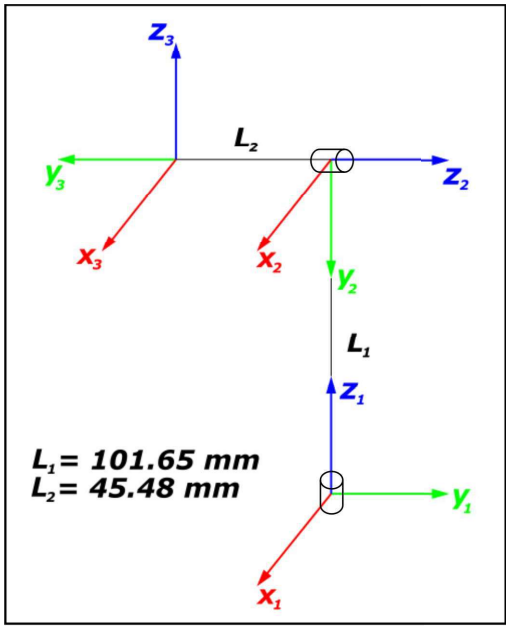
In order to better understand the mechanical structure and the position of the camera at a given angle, the translation matrix was derived following the following the method described in (CRAIG, 2004) Chapter 3. The 2-DOF laser structure can be related to a "RR manipulator" or a 2 rotational joint manipulator. As the elevation is limited from 0^o to 90^o and the azimuth from -360^o to 360^o the workspace will be a semi-sphere around the camera. The manipulator links axis is illustrated on Figure 20 and the link parameters are presented on Table 4.

Figure 19: Laser Structure camera axis



Source: Authors own

Figure 20: RR Manipulator link connection. Figure shows how the stepper motors are connected with eachother, their XYZ axis and illustrates how Table 4 was obtained.



Source: Authors own

Now, using the link parameters on the transformation equation:

$$
{}^{i}_{i-1}\mathrm{T} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}
$$

which result on Equation 3.4 and Equation 3.5.

$$
{}^{0}_{1}\mathrm{T} = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}
$$

$$
{}^{1}_{2}\mathrm{T} = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & l_2 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}
$$

and the combined transformation matrix, Equation 3.6.

$$
{}^{0}_{2}\mathrm{T} = \begin{bmatrix} c_1c_2 & -c_1s_2 & -s_1 & -l_2s_1 \\ s_1c_2 & -s_1s_2 & c_1 & l_2c_1 \\ -s_2 & -c_2 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}
$$

Now, assuming an arbitrary desired point $P = [x, y, z]$, the inverse kinematics can be derived as:

$$
\begin{aligned} \theta_1 &= cos^{-1}(\frac{y}{l_2}) \\ \theta_2 &= cos^{-1}(-R[3,2]) \end{aligned} \tag{3.7}
$$

With both Equation 3.6 and Equation 3.7, it is possible to control the position of the camera in relation to the angle or reciprocally.

3.1.8 Stepper Control Algorithm

With the goal to easily design the interface to move the stepper motor, four functions where develop to facilitate motor control. The algorithm structure is presented on Algorithm 1, but functions' called are hidden for simplification.

To move the stepper with the desired angle, the main program must call the function *angle_2step_count*, that is designed to require a python list as input, *angle*, which consists of the azimuth and the elevation in degree. Because there is no motor

encoder, for the motor to know its position, the algorithm keeps track of every movement and save it on a file. On the first step, the main function will check if the motor position is inside the workspace and if not, correct its movement, this is done through the function *check_limit*.

Once the position is acceptable, the algorithm will make the conversion from degree to number of step *step_count*. This is done rounding to the nearest integer the division between the angle and the step size. Before moving the motor, the number of steps is saved on file. Finally, the PWM command is sent to the driver separately, azimuth (Motor 1) then elevation (Motor 2).

As a note, the motors may move simultaneously, depending on the number of steps, but motor 1 will always start its movement before motor 2.

Algorithm 1 Stepper motor rotation functions

```
1: function angle2step_count(angle)
2:     Input: angle(azimuth, elevation)
3:     Return: step_count
4:
5:     if angle > 0 then
6:         direction = counterclockwise
7:     step_count = abs(round(angle/0.225)))
8: function check_limit(angle)
9:     Input: angle
10:    Return: angle
11:
12:    read(total_angle) from file
13:    future_angle = total_angle + angle
14:    if future_angle[0] >= 360 or future_angle[0] <= -360 then
15:        angle[0] = (angle − 360)%360
16:    if future_angle[1] >= 90 then
17:        angle[1] = 90 − total_angle[1]
18: function save_on_file(step_count, direction)
19:    Input: step_count, direction
20:
21:    read(total_angle) from file
22:    total_angle = total_angle + direction ∗ step_count ∗ 0.225
23:    save(total_angle) on file
24: function run_stepper_motor#(step_count, delay)
25:    Input: step_count, delay
26:
27:    for length(step_count) do
28:        GPIO = high
29:        sleep(delay)
30:        GPIO = low
```

3.2 Drone Detector

3.2.1 Problem Statement

Now that the 2 DoF laser structure is operational, the next problem is how to find the position of the quadrotor on the image to follow it and then hit the underneath sensor with the laser. The used method must be fast enough to locate the quadrotor and to follow it, reliable enough to work under changing light levels, precise enough to target the sensor and hit the sensor with a vector, sending the velocity command to the quadrotor control and lastly, for this application, a Raspberry Pi is used for the computer vision algorithm, then the algorithm must be simple enough to processed by the Pi. With these requirement in mind, the first step was to look the literature for available methods.

3.2.2 Literature Review

The first step is to look on the literature to decide what method to use to find where the quadrotor is. There are many methods to locate the position of an object on an image. A first possibility is by using feducial markers on the quadrotor, but, as the quadrotor also uses feducial markers for landing, as it was explored on (RODRIGUES, 2017) and as described in (OLSON, 2011), the quadrotor could land by it self.

When required to detect objects on images, the trend in computer vision is to use learning algorithm. Multiple algorithms have already been created to count fruits and trees, (RAMOS et al., 2017) and (QURESHI et al., 2017), also being used for ecological counting, (RAMOS et al., 2017) and pedestrian recognition using neural network (HAMMOUDI et al., 2019). On the article (Han et al., 2018) the author describes objectness detection (OD), salient object detection (SOD), and category-specific object detection (COD), their capability, application and robustness. Even those methods require size and light continuity of the object and those that would be reliable in detecting the quadrotor, require heavy computer processing, which the Raspberry Pi 3 can not allow.

For low processing power, the most common methods are using pixel analyst and color recognition. With this in mind, and using the basic of feducial marker algorithm, the author proposes a solution to disguise the quadrotor as a feducial marker. Beside using known formation like a actual feducial marker, and taking advantage that now in Brazil, all UAV are required to have LEDs, (ANAC, 2017). The solution designed is to install four green LEDs underneath each motor of the quadrotor, resulting in a square shape. The algorithm will locate this LED, estimate its center, representing the position of the sensor, and then will estimate the rotation that the stepper motor should do in order for the center of the LED be located on the center of the image. This approach is similar to the one in (NAGATA et al., 2017), where the author uses a known shape of IR LEDs to find the position of entertainment balloons for trajectory control.

Figure 21: LEDs arrangement for testing the Drone Detector Algorithm

Source: Authors own

3.3 Proposed Solution

The first step was to choose what LED to use. The color green was chosen because between the basic RGB image color (red, green and blue), green is the most suitable for this application, as the laser used is red and blue is the color of the sky, which would be the background as the camera is looking to the quadrotor upwards. The next step was planning a experimental setup. Taking advantage of the Vicon system on our Lab, the experimental setup was built, consisting of a "X" arrangement, where the LEDs were positioned on the same distance as they would on the quadrotor. Each LED is connected to a 1 $kOhm$ resistor and the arrangement is powered in parallel with a 5V, 2200 mAh power bank (MultiLaser).

This arrangement works as a model of the quadrotor and it will be used to test and improve all drone detector algorithms. The model is presented on Figure 21.

The algorithm for finding the quadrotor will be discussed in the following section in two stages. The first stage is to use computer vision to find the position, on the image, of the four LEDs, filtering outliers. The second stage is to compute the center of the points and estimate the angular motion to move the center of the points to the center of the image, following the quadrotor.

3.4 Detector Algorithm

This stage consists of finding the position, in pixel, of each LED. Previously, a paper was submitted on (SOLER et al., 2018), where the algorithm would simply use a single threshold filter. The filter was optimized to be very strict, clearing all outliers. As the project developed, this filter presented to be susceptible to ambient light, filtering inliers as well, being this a common problem in computer vision that can be solved by using a less strict filter, resulting in outliers, and adjusting the camera exposure.

To fix the problems with the first version of the algorithm, it was decided to go on a different route, besides using strict filter to avoid outliers, the goal now was to allow both inliers and outliers and then filter out the outliers. With this in mind, the detector part of the algorithm will be explained as follow.

3.4.1 Check White Light

The first part of the algorithm is a method to reduce the susceptibility to ambient light, by adjusting the camera exposure parameters in an attempt to keep all frames in the same light levels. In photography, by adjusting camera exposure, one can take multiple pictures in different light levels that look the same, which is exactly the goal here.

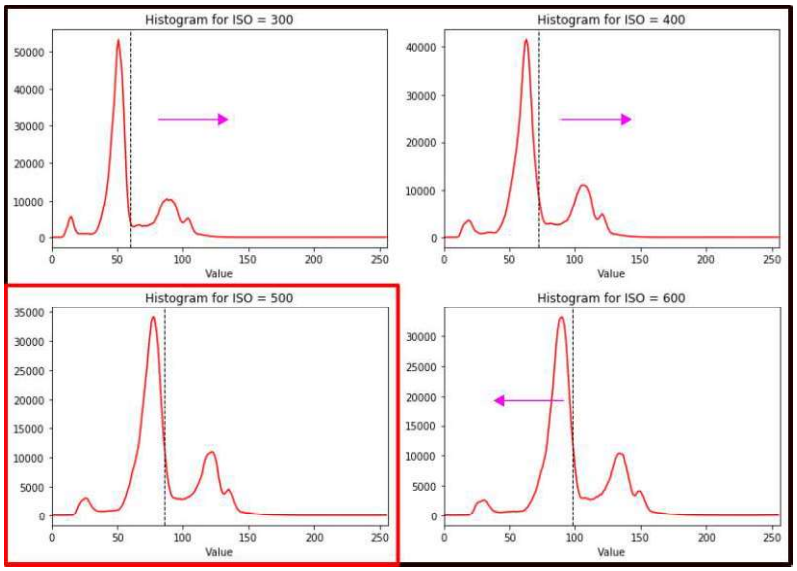
As explained on (CAMBRIDGE. . . , a) and (CAMBRIDGE. . . , b), camera exposure is controlled by three setting, shutter speed, which controls the duration of the exposure. Aperture, how much light can enter. Lastly, ISO speed controls the sensitivity to light of the sensor, any combination of this setting can result in a image with the same light levels. With this knowledge and taking in consideration the camera limitation, it was decided to adjust the ISO Speed and leave the remaining setting on a fixed level. For the Rasp Camera, the available ISO values are 100, 200, 300,..., 800.

As will be described next, the threshold filter was optimized with a series of image, and as the optimization occurs, the values for exposure setting were also acquired, then the final fixed value of shutter speed and aperture was registered and the algorithm uses that ISO Speed as a initial value.

Now, with these values, the algorithm will set the initial ISO Speed, convert the image to the HSV model, where V, stand for Value, which is the brightness of the color, (TECH-FAQ. . . ,) and then, it will calculate the color histogram, once it is stabilized, the algorithm will compare the most frequent region with the values registered on the optimization, if it is either higher or lower, will reduce and increase respectively the camera ISO speed repeating until it is closer. And finally, resulting in an image with the same light level as the ones used to calibrate the threshold filter.

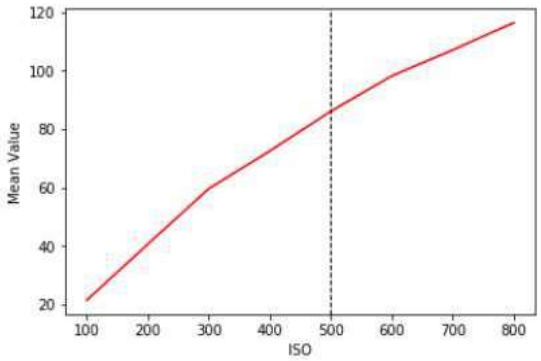
The histogram of a set of multiple images is plotted on Figure 22. The images are presented on Appendix D along all the remaining histogram plots. During multiple testing,

Figure 22: Histogram comparison for multiple ISO value. ISO value ranging in increments of 100 from 300 to 600, the algorithm is optimized to work around a value of 90, close to the one on ISO = 500.



Source: Authors own.

Figure 23: Change of Value with increasing ISO.



Source: Authors own.

the threshold filter was optimized to work at a Value of 90. On Figure 22, the algorithm accepted the ISO 500 with a value of 86.01 and rejected both ISO 400 with a value of 72.44 and ISO 600 with a value of 98.10. An analysis of the change of the mean of the images Values with the increase of ISO speed is plotted on Figure 23.

3.4.2 Threshold Filter

The first step in the detection algorithm is a threshold filter that at the same way as before, but, to avoid filtering inliers, the threshold filter was made less strict.

To make the filter less strict, the threshold filter needed to be calibrated. From the first attempt on the filter, which was published on the article (SOLER et al., 2018), the main problem noted was its susceptibility to ambient light, resulting in an unreliable filter. With this in mind, the threshold filter calibration was done alongside with the camera ambient light adjustment.

The process to calibrate the filter is as follows, at a certain light level and camera adjustment, 2 pictures were taken of the model at a close proximity to the camera, then the model was moved further, and another 2 pictures where taken. This process was repeated with different ISO values. The same process was repeated later in the day, resulting in a collection of 32 images.

Following the image data collection, for each image, a single pixel for each of the four specific LEDs were manually taken. The point selection had a few rules:

A) One point for each LED, hence four per image;

B) Points should be on the green region, avoiding the white center.

The reason for (A) is to make the process faster and (B) is used as a second thresholder filter and will be discussed later. After points in all images were taken, the following step was to estimate the mean and standard deviation of the HSV values on every image, with changing light level. The result for the range of HSV values of the LED on the image is $mean = [70, 80, 120]$ and $std = [10, 20, 30]$.

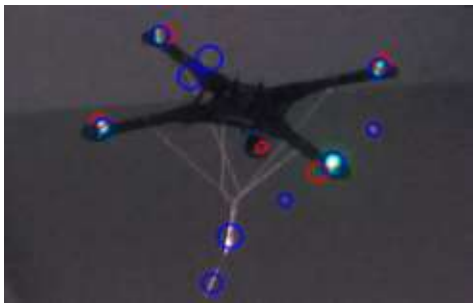
With the filter parameters setted, the threshold filter can be used by the algorithm, once it filtered the image, resulting in a binary image, the algorithm will locate the center of the blobs on the image, if the blob‘s area is not bigger than $100\,px$. At the end, the position and radius of each blob will be known, but, as discussed, some might be outliers that need further filtering. An example of an output is presented on Figure 24, note multiple outliers from both threshold filter that will be filtered.

3.4.3 White Light Filter

As stated above, to improve the detection algorithm, multiple filters are used. A second threshold filter is used in conjunction with the colored filter above, the goal being to remove the outliers acquired due to making the filter less strict.

A light source directed to a camera overshots the sensor, resulting in a region with RGB value of (255, 255, 255). This is true for the LED used on the project, where it

Figure 24: Threshhold Filter Close View. The image highlight the quadrotor, the red circle represents the threshold filter and the blue circle the white light filter. On this frame one can see multiples outliers that need to be filtered.



Source: Authors own

consists of a green ring formation with a white core. Then, the second threshold filtering used is simply filter only those white regions. The remaining is the same as the previous algorithm, which will acquire the position and radius of the blobs.

By returning once again to the image collection used to calibrate the filter, it was estimated the following:

1. The LED has elliptical structure;
2. The LED consists of a green ring around a white center;
3. The total radius is approximately 1.2 times the core radius.

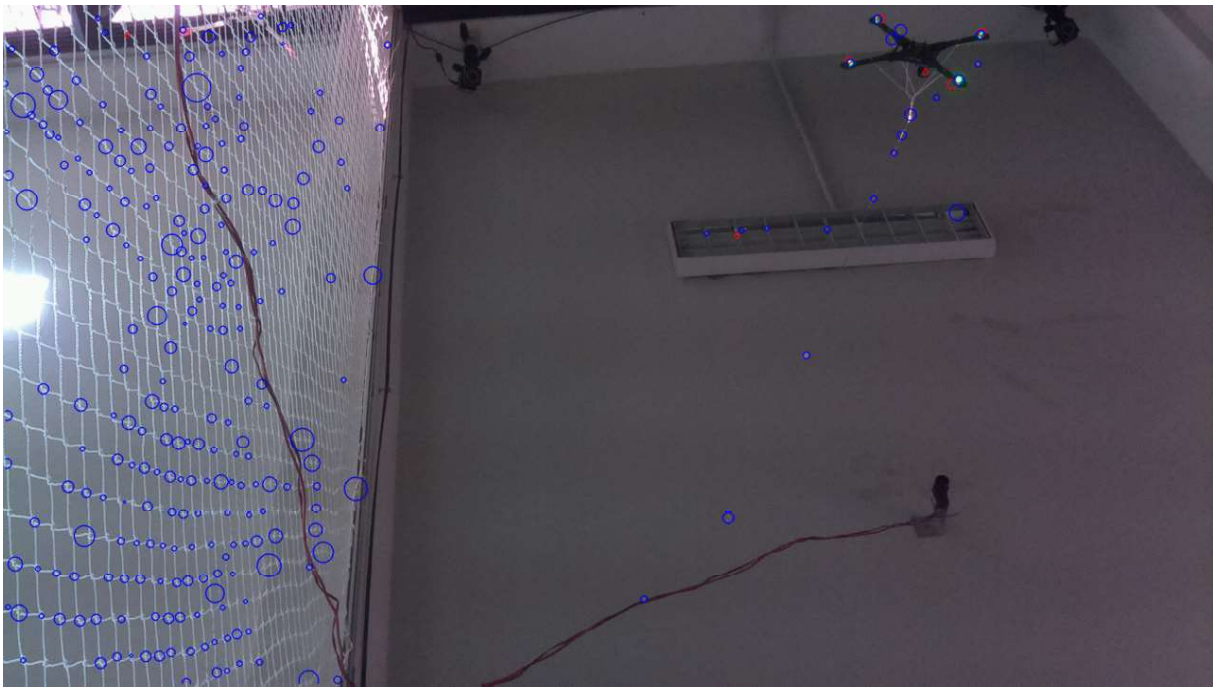
With this known, the result of both threshold filter where compared and the points that match the requirement where found as inliers. This is demonstrated as the cyan full circles on Figure 26.

3.4.4 Metric Filter

To avoid false positives, the algorithm will check the relationship between the points. False positives may come in three different ways. In the first, the outlier could come from an unknown far away source, resulting in a point in great distance between the others points. After some testing, this far away points can be filtered by limiting the distance to $100\,px$.

The second type of remaining outliers is the ones that are close to the points, but would result in huge error for the algorithm. The source of this point are usually the reflection of the green led on a nearby medium, the laboratory walls for example. This is solved by obtaining the standard deviation of the distances between the points and

Figure 25: Threshhold Filter Full View. Image shows the full range of view. The red circle represents the threshold filter and the blue circle the white light filter. On this frame one can see multiples outliers that need to be filtered especially on the white light filter.



Source: Authors own

keeping track of it for five iterations. Whenever the change on the standard deviation is higher than 20%, the algorithm ignores the iteration and removes the first element on the line, in case the target moved.

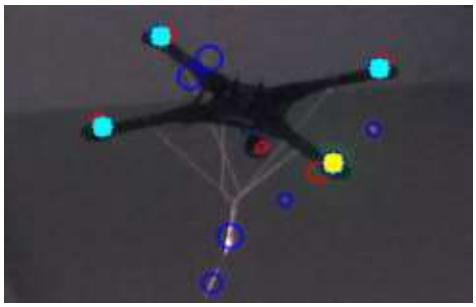
Finally, the last type of outliers are positioned alongside of the inliers, which could be either a reflection of the LED on the model frame or a single LED that was divided in two blobs. As this case does not result in a huge error for the algorithm, those outliers are acceptable.

3.4.5 Expand Points

To include the acceptable outliers unfiltered by the previous filter, the algorithm will accept a maximum of 5 inliers before estimating the rotation for the motors. If at this point, the algorithm has less than 5 inliers, the algorithm will review each individual points by both the threshold filter and the white light filter and compute the distance between every inliers. The algorithm will accept as a new inlier as long as each distance are:

1. Greater than half the distance between each inlier (not too close);

Figure 26: Proximity and expanded points. The image highlights the quadrotor, the cyan circles are inliers obtained through the proximity filter by both threshold filters. The yellow circle is obtained by the distance of the available inliers (expand points).



Source: Authors own.

2. Lower than the distance between each inliers, times $2^{0.5} = 1.21$ (not too far).

With this filter, it is possible to obtain the remaining points. Once again this is presented for a single frame on Figure 26. With this, the algorithm demonstrated itself to be more capable of detecting the quadrotor then the previous version.

3.5 Rotation Estimation

With the inliers found on the image, the next step on the algorithm is to estimate the rotation for which the stepper motor need to taken for the center of the model to be positioned on the center of the image.

The method used on article (SOLER et al., 2018) was to find the homography matrix and estimate the distance to the target. This method has the advantage to also find the distance to the model, which could be used to better control the motor speed, but, this problem was resolved with the increase in data acquisition rate of the sensor.

Even though finding the distance can still be helpfull with the improved sensor, as it will be explained on subsection 4.1.4, the previous method had the following problems:

A High computation needed;

B Heavy dependency on found inliers;

C The order of the inliers must match the model;

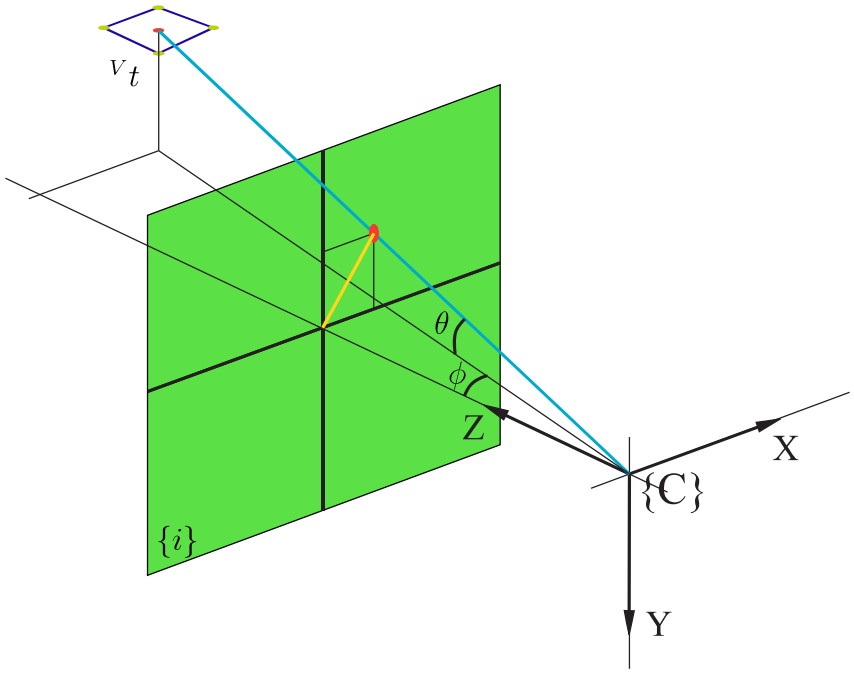
D Measurement in meters, which need to be converted to degrees, increasing the error.

With this in mind a new approach was taken. Based on the camera model described on section 2.2, it is possible to find the angular rotation to move a point to the center

of the image with just the camera calibration values, which is done in a very reliable way (ZHANG, 2002), and the distance in pixel between the point and the center. This is illustrated on Figure 27.

This new approach is computationally much lighter than the previous one (A), also because it only need a single point on the image, it depends a little less on reliable the detection algorithm, resulting in a more robust algorithm, (B) and (C). Lastly, this approach has a higher resolution, as is only dependent on how good the camera calibration was done, which is often very reliable as it is done with very robust algorithm and with no time constraint.

Figure 27: The Pixel Plane with azimuth and elevation



Source: Authors own. Changed from Figure 8

3.5.1 Rotation Algorithm

Before estimating the stepper motor rotation, the algorithm must first find the center of the found inliers, which represent the place on the model where the sensor is positioned. To do so, the algorithm will simply compute the mean of the (X, Y) coordinates on the image, in pixels.

The method will be very reliable when the algorithm finds the four LEDs, actually resulting in the center of the model. But, sometimes a few LEDs can malfunction or be blocked by the sunlight, resulting in less than four points. For tree points, the solution is to estimate the mean just like before, resulting in a small error, but a more robust algorithm. When there are only two points, the algorithm will include the last center

position obtained as an inliers and will compute the mean like before. To reduce the error when using fewer points, the solution is to acquire five estimation and then check if the standard deviation is lower than two degrees, ten times the motor resolution.

Finally, with the center of the quadrotor model, the rotation that the motor must take to reach the center of the image can be computed by the Equation 3.8, as shown on Figure 27. On the equation, $atan2$ is used to distinguish between quadrants, and the calibration parameters u_0, v_0, f_x and f_y to increase the algorithm resolution.

$$\begin{aligned} \theta &= atan2(\frac{Y - v_0}{f_y}) \\ \phi &= atan2(\frac{X - u_0}{f_x}) \end{aligned} \tag{3.8}$$

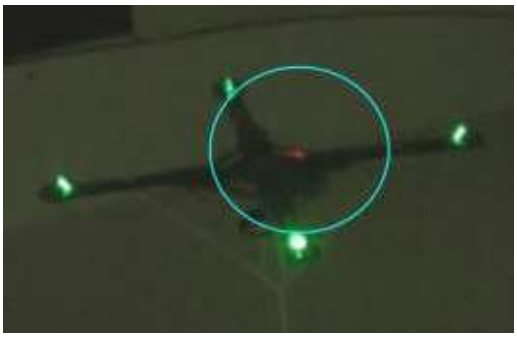
3.6 Results

The article (SOLER et al., 2018) presents the first version of the drone detector algorithm. The algorithm was capable of detecting and estimating the distance between the Tractor Beam and the quadrotor. The article concludes with a mean position error of $mean(x, y, z) = (0.2093, 0.0906, 0.2674)\ m$, to compare with the new version of the algorithm, this values must be converted to degree, which would reflect the command for the stepper motor that the older version was going to send. The result is $elevation = 18.71^o$ and $azimuth = 23.41^o$. This value could follow the quadrotor, which is the goal of the articles, but are not acceptable to target a laser on the sensor.

To test the capability of the new version of the drone detector algorithm, an experimental test was setted. On this experiment, the model of the quadrotor is positioned on seven known points on a horizontal rope beneath the tractor beam until the other side of the room, around $2\,m$ long. The model is moved between the points and the algorithm is expected to estimate the required rotation and execute the rotation. At each iteration, two pictures are taken before and after the motor rotation is executed. During the test 376 pictures where taken referring to 94 points.

To estimate the resolution of the algorithm, for each of the 376 pictures taken after the movement, the actual center of the model is manually taken, resulting in a point coordinate (M_x, M_y). As two pictures refer to the same movement, this is used as an "error in selection". With the coordinates (M_x, M_y), the error on the motor movement will be the distance between the point to the center of the camera image (u_0, v_0), as $e = (M_x - u_0, M_y - v_0)$ in pixels. The error is then converted to degree by using equation

Figure 28: Tractor Beam Laser on the Model. Laser hitting the model on the position where the sensor is installed.



Source: Authors own

Equation 3.9.

$$\begin{aligned} e_\theta &= atan2(\frac{e_x}{f_y}) \\ e_\phi &= atan2(\frac{e_y}{f_x}) \end{aligned} \tag{3.9}$$

By using this method for every frame, the resolution of the Drone Detector Algorithm is $elevation = 4.96^o$ and $azimuth = 0.27^o$, with a standard deviation of 1.03 and 0.51 degrees. The algorithm only failed to detect the model when it is towards a huge light source, such as the room LED light tubes or the actual sun. This also happen with the human eye when trying to see a object in the direction of the sun.

Lastly Figure 28 illustrate the capability of the algorithm to hit the quadrotor as seen by the red dot on the middle of the cyan circle.

4 SENSING AND CONTROL

In this section we explore the method in which the quadrotor senses the laser and reacts to its commands. On section 4.1, two methods for laser sensing will be explored and latter in the section they will be compared to decide the most appropriate one. Then, with the position of the spot laser on the UAV known, section 4.2 will discuss the method for controlling the quadrotor controller with the information from the sensor.

4.1 Sensing

4.1.1 Literature Review

A laser beam can not be seen when it is traveling on a transparent medium, such as the atmosphere, but the projection of the laser can be seen on a media. For example, a laser can be seen when directly projected on a camera CCD, but this is not recommended, as it easily overshots the sensor and could even damage it. Based on how movie theaters project light into a screen, a possible approach to find the position of the laser dot is to create a screen for the laser and use a camera to find its position. This approach will be described in subsection 4.1.2.

As explained in (CORKE, 2011), a camera light sensor consist of an array of luminance- and chrominance-sensitive over which is an array of red, green and blue filters, invented by (Bryce E. Bayer, 1975). Each one of these photosensors is a pixel on the image. For this application, there is no need for the photosensor to be as small as seen on a traditional camera, and as the laser will be direct on the sensor, there is no need for chrominance sensitive and color filter, resulting in a simplified camera. This approach will be described in subsection 4.1.4 and it is inspired on how a CMOS camera scans the pixels of the image. The schematics for the first version of the sensor was done by Richard Mascarin from, *Laboratório de Instrumentação Eletrônica* (LIEPO) and the construction of the sensor was done in collaboration with Clara Louzada.

4.1.2 Laser Screen

The goal of the laser screen is to receive the command from the Tractor Beam. This is done by estimating the trajectory of the laser on the screen, resulting in a vector which the quadrotor will acknowledge as a velocity vector.

4.1.2.1 Screen Design

The main component of the laser screen is the actual screen which serves as a projection media for the laser beam and it should result in a clear laser spot on the image

that will be recorded by a camera. The first step was deciding the guidelines for the design, which are:

- The laser screen must be a parallel plane in front of the camera plane;
- The screen material must be able to refract light;
- The screen material must be opaque enough to avoid glowing;
- The screen material and the support must have some form of mechanical resistance so it does not break in flight.

4.1.2.2 Support

The first design consideration was the distance from the screen to the camera lens. As this distance also dictates the size of the screen, it is important that it is as big as possible.

To avoid mechanical changes on the Pelican structure and with the goal to keep the Pelican PTU (Pan Tilt Unity), which is capable of keeping the camera plane parallel to the ground plane, the available distance between the camera lens and the ground is $70\,mm$. Thus, using a gap of $20\,mm$ from the screen and the ground, the distance between the camera lens and the screen will be $50\,mm$, which was adjusted to be the camera focus.

4.1.2.3 Screen

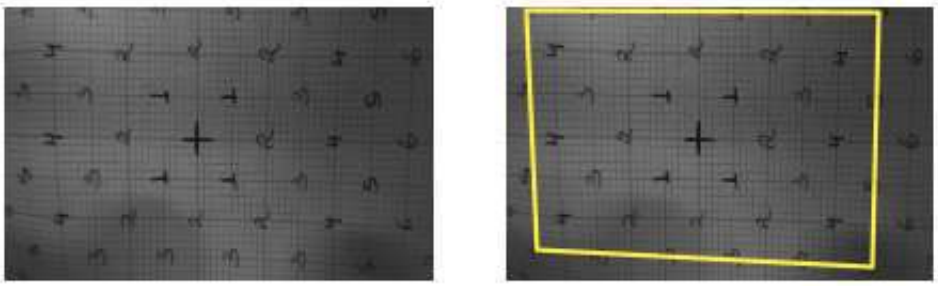
Setting the focus of the camera to the screen, we acquire a sharper image of the laser dot, and by using the camera model equations we can estimate the screen size using the projective transformation Equation 2.5. To further, an experimental test using a graph paper was done, as shown in the Figure 29. As the camera uses a fish-eye lens, the size of the sensor should be the less distorted region, which result in a 50x30 mm screen.

To check what material is best to be used on the laser screen, two types of paper were tested, a tracing paper and a printer paper (Chamex Office). By testing both on the camera support, the conclusion was that the printer paper is better for the laser screen, because it is more opaque than the tracing paper, so it absorbed some of the light, reducing the amount of noise on the image.

4.1.2.4 Laser Position Estimation

This section will briefly discuss the laser screen algorithm, Algorithm 2. The goal of the algorithm is to find, in a fast and reliable way, the position in which the Tractor Beam's laser hits the screen and, with enough point, estimates a velocity vector. The velocity vector estimation is the same as in the final Algorithm subsection 4.2.1. Then, at this point of the discussion, only the laser spot position estimation will be covered.

Figure 29: Laser Screen Size Estimation



Source: Authors own

Algorithm 2 Position of Laser Dot on Screen

Input: **ROS::subscribe**(*camera_image*)
Return: **ROS::publish**(*keypoint*)

1: $hsv\_image =$ **CV::RGB2HSV**$(camera\_image)$
2: $filter\_image =$ **cv::inRange**(hsv_image)
3: $dilate\_image =$ **cv::dilate**$(filter\_image, closing\_size)$
4: $closing\_image =$ **cv::erode**$(dilate\_image, closing\_size)$
5: $keypoints =$ **cv::SimpleBlobDetector**$(closing\_image, params)$

First, the algorithm will subscribe to the ROS node *image_rect* acquiring the image *camera_image*, which is converted to the HSV model. The image is filtered using OpenCV *inRange* function, using a threshold between bright white $(0, 0, 255)$ and bright red $(0, 80, 255)$. Even with this filter, the resulting image still has a lot of noise. To improve the image, the threshold filter is followed by a closing process, using a closing radius of $5px$, resulting in a clear blob.

With a clear binary image, it is possible to use OpenCV *SimpleBlobDetector* function to estimate the center of the blob. The blob is expected to have varying sizes but it retains a convex shape, so the function parameters are settedalso to look for something close of an ellipse.

Once the center of the laser spot is known, the algorithm will keep collecting centers for multiple frame, estimate the velocity vector, and send it to the quadrotor control, attracting the quadrotor to the Tractor Beam.

4.1.3 Laser Screen Alternative

The laser screen was mounted on the quadrotor and then it was validated on flight. The laser screen was successful in reading the laser direction as a velocity vector and was able to send a valid command to the quadrotor control algorithm. A short video of this can be found on the extra resources. The final conclusion was that the laser screen was a

good validation tool, but it demonstrated the following flaws:

- Demands the use of the camera, which could be used for autonomous landing;
- High computational cost, possibly lagging other applications;
- Subject to upwards and downwards light sources;
- Low acquisition time, losing many points.

With this validation two main problem were noted. First, during flight, depending on the position of the sun in relation to the quadrotor, the sun could completely illuminate the screen, neutralizing light from the laser. Second, with the low data acquisition, if the laser moved too fast, or the quadrotor was too far away, the sensor would not acquire any reading, due to the fact that on those situations, the laser spot velocity on the screen plane was much faster than the data acquisition rate.

Another huge problem on the laser screen is the fact that it uses a camera that could be used for position estimation, for pose estimation (WANG; OLSON, 2016) or in monocular visual odometry (FORSTER; PIZZOLI; SCARAMUZZA, 2014), resulting in the need of a second camera, which will increase the payload for the MAV and finally reducing flight time.

Then, it was decided that an alternative was in need. The first step was to estimate the sensor acquisition rate. On Appendix E is presented this estimation, which assumes the max distance between the quadrotor and the laser is $20\,m$, and concludes that the sensor acquisition rate should be $200\,Hz$.

A way to acquire the target acquisition rate is to reduce the image resolution and to do this, it was decided to arrange photo detector sensors in a matrix with a fixed distance between each other, resulting in a very low resolution "camera". By using threshold filter hardware, it is also possible to filter ambient light, solving another laser screen problem.

Before designing the sensor, the following questions needed to be answered:

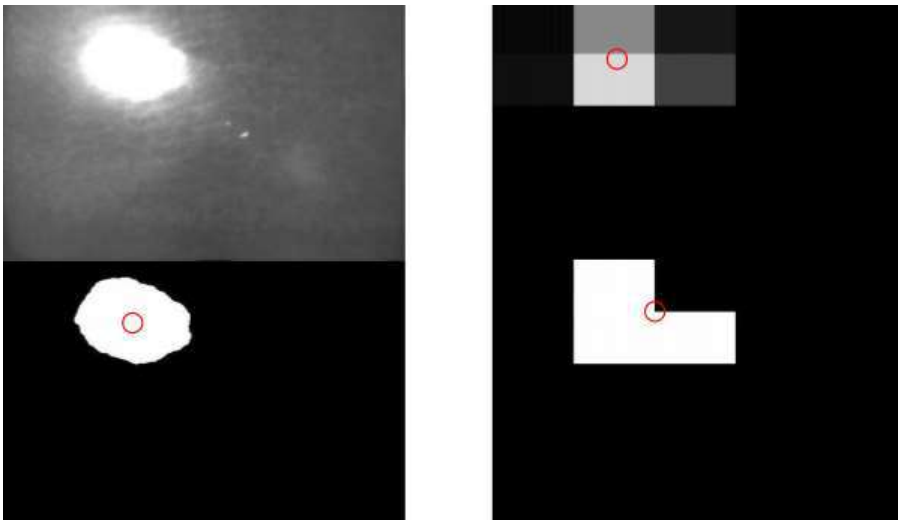
(A) How much the loss of resolution influence the precision?

(B) How many photo detectors are enough for this application?

(C) Is it possible to use threshold filter on the sensor hardware for speed?

An experiment was designed to answer those questions. The experiment consists in setting the laser screen upward and to use a laser to hit the screen from $1\,m$ away and in as many orientation as possible. In total, the experiment resulted in a sample size of 578

Figure 30: Comparison of the "blob" output. Top left: Original HSV image; Botton left: Filtered HSV image; Top right: Grayscale 5x5 Image ; Bottom right: Binary 5x5 image



Source: Authors own

Table 5: Comparison Results

Image	**Accurancy**	**Mean (x, y) [mm]**	**STD (x, y) [mm]**
Grayscale	0%	(-0.09 -0.15)	(0.95 0.48)
Binary	2.08%	(-0.12 -0.12)	(1.80 0.95)

Mean and standard deviation for a 5×5 image using the binary and grayscale method for a screen of 50 mm width (x) and 30 mm height (y).

images. The sensor is first planned to be a 5×5 matrix, so all of the images will be scaled down from a 752×480 to 5×5, to simulate a sensor output.

During this experiment, it will be assumed that the laser screen algorithm outputs the true center position, which is in full resolution, and used to compare with the scaled down resolution, answering Question (A). The algorithm is the same as described on Algorithm 2, but the center is estimated using a weighted average and then converted from px to mm by multiplying by the inverse intrinsic matrix, $mm = K^{-1}$, for better visualization. On the scaled down image, before estimating the blob position, the image is adjusted using the Opencv *resize()* function. This is illustrated on Figure 30.

To simulate a binary sensor reading, and answering Question C, after scaling down the image, it was filtered for an index higher than 0.2, resulting in the bottom right image on Figure 30. Finally, the mean and the standard deviation (STD) were obtained for each of the methods. The result is presented on Table 5 and illustrated on Appendix F.

Table 6: Number of Photodetector Result

Size	Mean (x, y) [mm]	STD (x, y) [mm]
3×3	(0.14 -0.27)	(2.66 1.31)
4×4	(-0.06 -0.20)	(1.58 0.71)
5×5	(-0.09 -0.15)	(0.95 0.48)
6×6	(-0.12 -0.13)	(0.58 0.37)

As the biggest standard deviation for the grayscale image is 3.2% and for the binary 3.6%, the photo detector matrix can be used, closing Question A.

To estimate the impact on using a bigger or smaller sensor, the same method was done for a 3×3, 4×4 and 6×6 grayscale image. The results are presented and illustrated on Appendix G.

As the maximum standard deviation of the 6×6 matrix is 1.2% of the sensor size, it answers Question B. Even if the sensor does not reach the 200 Hz rate, the algorithm can get faster by start using the digital input instead of the analog pins and still get an acceptable precision.

4.1.4 Photo Detectors Sensor

The goal in designing the new sensor is to solve the flaws of the laser screen discussed above. Now, the following is a list of what capabilities the sensor must have:

(A) Less consumption of the quadrotor CPU;

(B) Data acquisition rate of 200 Hz;

(C) Capable of negating ambient light;

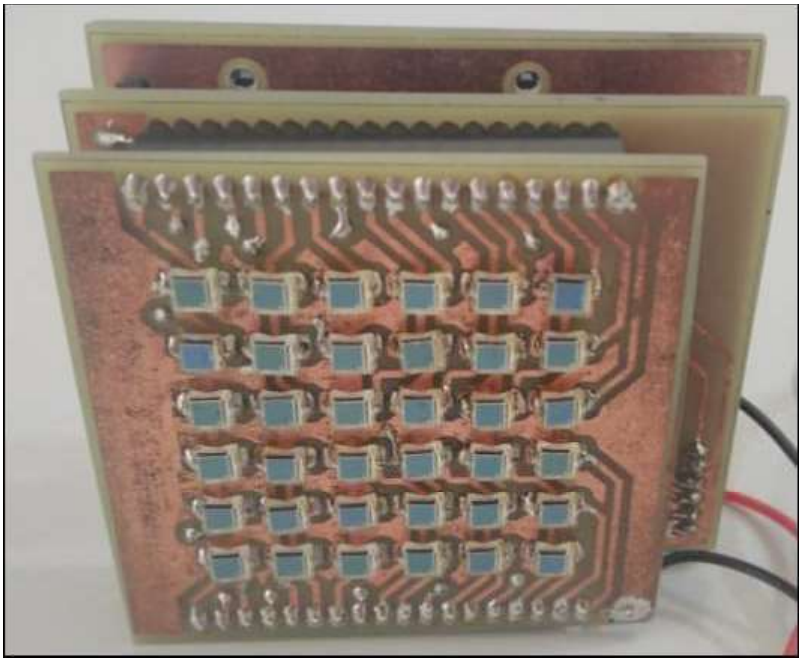
(D) Compact and lightweight.

The photo detector sensor has gone through two versions, the first being a 5×5 matrix, which was designed by Richard Mascarin. Appendix A presents both the schematics and a picture of the sensor. The first version was assembled by the author and Clara Louzada and can be found on Appendix H. The final version was expanded to a 6×6 matrix and built in a more compact way. This version will be described next and the schematics can be found on Appendix I. The second version was designed by the author and Clara Louzada. A picture of the actual sensor is presented on Figure 31.

4.1.4.1 Eletronic Design

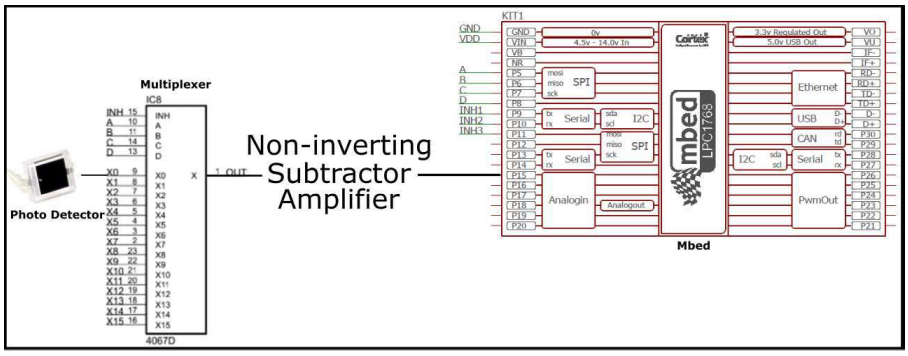
As described in subsection 4.1.3, the sensor is a 6×6 matrix of photo detector, arranged in a fixed distance between each other. A Mbed *LPC1768* microcontroller will

Figure 31: Picture of the Second Version of the Sensor.



Source: Authors own

Figure 32: Simplified Photo Detector Sensor Circuit.



Source: Authors own

read the voltage difference in each photo detector, estimate the position of the laser spot, and then send it to the quadrotor, solving **(A)**. A simplified version of the circuit is presented on Figure 32, detailed circuit including resistor values and a tagged picture of the actual sensor are presented Appendix I.

As the voltage difference on a photo detector when it is disturbed by light is in the order of $300\,mV$, the circuit must include an amplifier. But, the voltage difference on a sunny day, without the laser, is around $100\,mV$. So, by using a subtractor amplifier circuit,

which output the difference both inputs, it is possible to filter the ambient light and also make sure only the laser disturbance is being amplified, these solve **(C)**. A potentiometer is used to change the ambient light offset when needed.

To switch between all photo detector, the circuit includes three multiplexers, each controlling twelve photo detectors. The multiplexer used is the *"74HC4067 16-channel analog multiplexer"*, which is a 16 : 1 multiplexer switch. The switch features 4 digital inputs (S_0, S_1, S_2, S_3), sixteen independent input (Y_n), one common output (Z) and an enable/disable input (H).

The Mbed *LPC1768* can switch through the photo detector by sending to the digital inputs the binary correspondence of 0 to 15, and then reading the common output of the multiplexer on the microcontroller analog input.

On the first version of the photo detector sensor, the voltage difference was amplified before the switch, resulting in the need of six amplifier circuits. This is improved, by placing the switch before the amplifier circuit and also connecting all three multiplexer outputs, resulting in a single amplifier circuit. The multiplexer can be disabled, resulting in infinite impedance.

Lastly, to solve **(D)**, the sensor is built in three layers, as seen by the roman characters on Figure 33. The sensor was manufactured by the author and Clara Louzada on *Laboratório de Instrumentação Eletrônica*.

4.1.4.2 Algorithm

One of the capabilities of the photo detector sensor discussed before was "(A) Less consumption of the quadrotor CPU", which is done by using another microcontroler, the mbed LPC1768, to handle reading and estimating the center of the laser spot. Once it is computed, the data is sent to the quadrotor via serial communication.

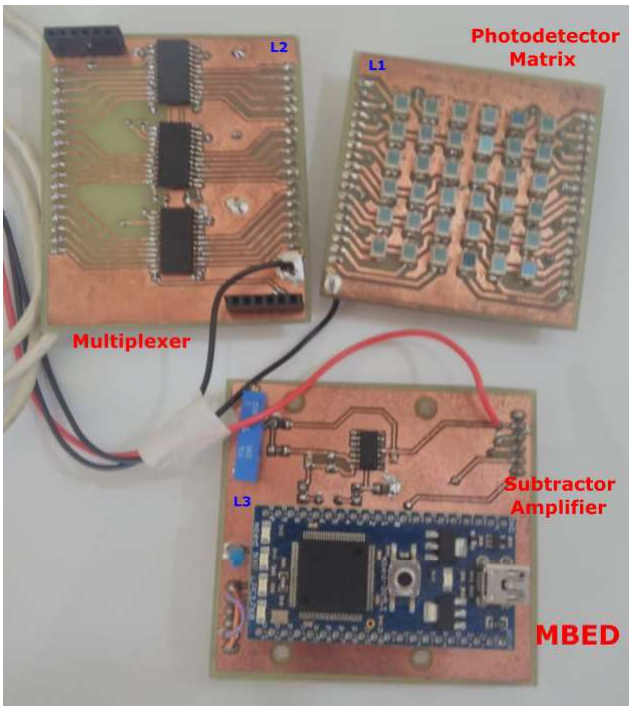
The algorithm was designed to reach 200 Hz data acquisition, to do so, the baud rate was setted to 115200, which is the maximum possible. The structure is presented on Algorithm 3.

The first step on the algorithm is to enable the GPIO input/output. There are three multiplexer enable port H_n, setted as GPIO output, one multiplexer common output Z, setted as analog input. Now, for the four digital switch input S_n, the first PORT, (PORT 0.6), was setted to output.

With lessons learned from the first version of the sensor, it was found that it is much faster to write the bits to the GPIO port. Then, during the electronic design the GPIO pins chosen as output were sequentials, those are PIN 8, 7, 6, 5 which are PORT 0.6, 0.7, 0.8, 0.9, respectively, refered on Appendix B.

Another feature of the new sensor is that, the first two lines of photo detector are

Figure 33: Picture of the sensor, with tagged components. The layers were separated for presentation and are tagged for presentation.

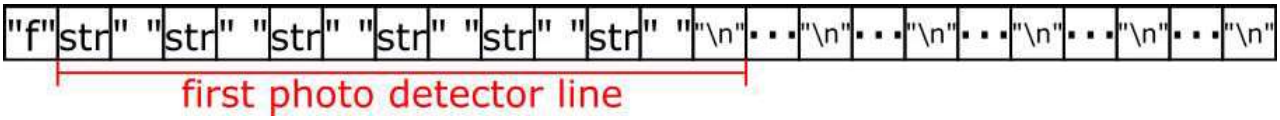


Source: Authors own

Algorithm 3 MBED: Photo detector sensor interface

Enable_$GPIO_{out}(S_n)$ = Pin 5, 6, 7, 8
Enable_$GPIO_{out}(H_n)$ = Pin 9, 10, 11
Enable_$GPIO_{in}(Z)$ = Pin 15
for $H_n = 1$ *to* 3 **do**
    for $i = 1$ *to* 12 **do**
        S_n = **Write**_**Binary**(i)
        Sleep_**us**(20)
        Matrix[H,i] = **Read**(Z)
        Clear(S_n)
function CASE 1: FLIGHT
    Center = **Get**_**Center**(Matrix[H,i])
    Serial_**Write**(Center)
function CASE 2: USER
    Message = **Build**_**Message**(Matrix[H,i])
    Serial_**Write**(Message)

Figure 34: MBED String Structure for User Visualization



Source: Authors own

connected to multiplexer 1, the following two are connected to multiplexer 2 on the same switch and so on. Then, starting on line 5, the algorithm will map each individual photo detector and save the output on its corresponding position, which is also done on a CMOS Camera, as described by (CORKE, 2011).

After reading each photo detector, the algorithm has two available possibilities. On Case 1, the algorithm will estimate the position of the laser spot using Equation 4.1:

$$\frac{\sum(p_i(x,y) * V_i)}{\sum V_i} \tag{4.1}$$

for a photo detector i and where $p_i(x,y)$ is its x and y position and V_i its voltage difference, allowing for the quadrotor to receive messages from the Tractor Beam.

On Case 2, it is expected that the mbed is communicating with a computer, with the goal of allowing an user to visualize the sensor. On line 15, the algorithm will build a string with the structure presented on Figure 34.

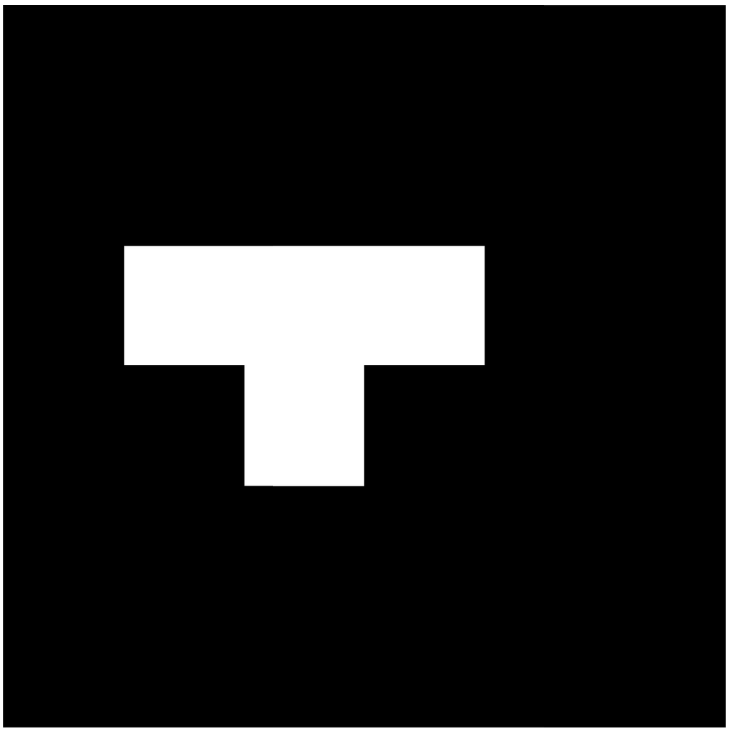
The structure of the interface, which is running on the user computer, is presented on Algorithm 4.

Algorithm 4 User visualization

```
1: Serial_Read(Message)
2: Check_Message()
3: if Message = OK then
4:     Image = Convert_to_Image(Message)
5:     CV::resize(Image)
6:     Show(Image)
```

On line 3, the algorithm will check if the message is 78 characters long and if it starts with a f. If this is true, it will convert to an image format and scale by 100, resulting in a 600×600 image. Figure 35 presents an example of an output from the sensor. The sensor will obtain multiples similar reading when a laser spot is present. On the extra component is presented multiples reading at a frequency of 200 Hz of a square being drawn on the sensor.

Figure 35: Sensor output image representation. Sensor output of the binary number 000000 000000 011100 001000 000000 000000 or 7372800 in decimal.



Source: Authors own

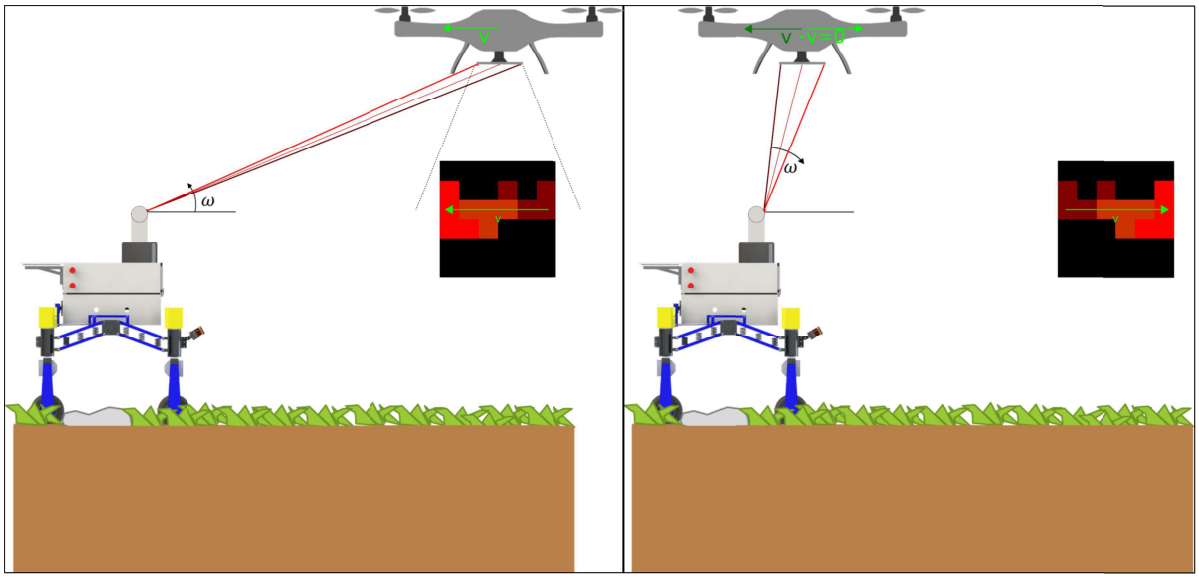
4.2 Sensor Interface

As explained before, with the proposed sensor it is possible to estimate the position where the laser dot hits the sensor. With this information, the next problem is to decide what is the best method to use this information with in order for the quadrotor to reach the ground station.

As explained in section A.2, the *asctec_mav_framework*, which is the ROS package to control our quadrotor, has three possible modes of operation, and after some research done on the subject it was possible to identify that both, the position control and velocity control, can be used to control the UAV and make it react to inputs from the laser dot sensor.

The position control operation method can be used by interpreting the position of the laser dot, relative to the center of the sensor, as waypoints for the UAV to follow. This is inspired on the Carrot chasing path following control method, as described in (TABATABAEI et al., 2015), where the UAV would follow waypoint commands, just like a donkey would follow a carrot. The problem of this approach is that if the laser fails to hit the UAV, there will be no waypoint command, resulting in the UAV just holding its

Figure 36: Interface between laser and quadrotor. Interaction between the laser and the quadrotor. The laser will move towards itself to attract the quadrotor, once it arrived close enough, will move on the opposite direction, resulting in a command to stop.



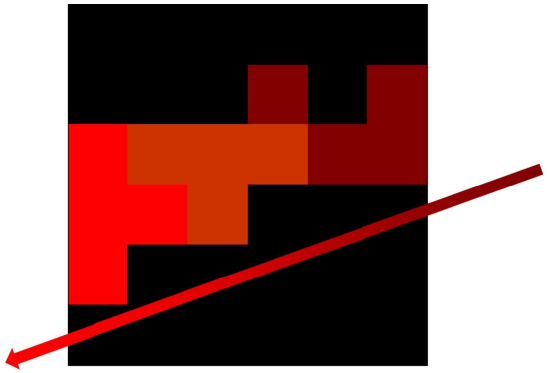
Source: Authors own

position.

The velocity control method works the same as using the radio controller and could be used in two distinct ways. The first possible use would work in a similar way to the position control. Instead of interpreting the input of the laser sensor as waypoints, they would be interpreted as velocity commands. This works in a the same as a potential fields, for obstacle avoidance, (CJ Taylor,) and the vector field path following control method (MEENAKSHISUNDARAM; GUNDAPPA; KANTH, 2011), where the velocity vector would grow as the laser dot is farther from the center of the sensor.

By continuous testing, it was found that using an exact point of the laser on the screen would require an extremely high precision of the detector algorithm, which also reflect in a loss of range. With this problem in mind, the second form in which the velocity control method could be used is introduced. By acquiring multiples points from the laser displacement on the sensor, it is possible to estimate the direction where the laser is moving, which can then be interpreted as a velocity vector for the quadrotor to follow. The quadrotor would remain moving in that direction until an opposite vector is sent by the laser. This method is illustrated in Figure 36.

Figure 37: Velocity vector of multiple sensor reading. The sensor will acquire multiple reading, then proceed to estimate the velocity vector, represented by the arrow. The gradient represent the direction of the vector, from older, darker red, to newer, bright red, points



Source: Authors own

4.2.1 Velocity Vector

As described above, the goal of the laser interface algorithm is to interpret multiples points of the laser on the sensor until it is possible to acquire a vector. This is done by keep a communication with the sensor at a baud rate of 115200 bps and at every message, decode the sensor array into a numpy matrix. In the case the matrix is not all zeroes, there is a laser dot on the sensor, the algorithm will acquire all points coordinates. Sometimes, the algorithm will acquire more than one point, which means that the laser spot extend on multiples photodetectors.

Once the algorithm found multiple matches, it will proceed by estimating a straight line that better represents all of the points. The direction of the vector is acquired by looking at the position of the newer and older points, finally having a velocity vector. The sensor has the same coordinate frame as the quadrotor, which means that this velocity vector can be sent without change to *asctec_mav_framework*.

Once the detector algorithm notices that the quadrotor is close, by inspecting the raise on the radius of the points, it will send an opposite vector, then, once the algorithm notice this, it will command the quadrotor to stop and start the landing algorithm.

Figure 37 provides a example of the sensor estimating the velocity vector. The arrow represents the velocity vector and the gradient the direction towards the Tractor Beam.

5 CONCLUSIONS

The goal of this chapter is to conclude this document, entitled, 2-DOF Laser Structure for Autonomous External Quadrotor Guidance, and to clarify what was achieved. First, section 5.1 will review the content of this work, and then, section 5.2 will clarify what can still be done on this project and suggest improvements for the each component.

5.1 Review

The problem of attracting a UAV to land on a ground vehicle, without relying on information acquired by the UAV itself is a complex problem with multiple requirements and limitations. To simplify, this problem was divided into two approaches. The device installed on the ground vehicle, known as Mirã and the one installed on the aerial vehicle, every component of this work was tested and validated, demonstrated to be capable of completing its respective goal.

Rural regions tend to have bad GPS connection resulting in the possibility of a position gap between the two robots, making autonomous landing impossible, because both robots would never be close enough for the landing algorithm to work. As a complementary method to solve this problem, this work focus on designing a 2 DoF structure capable of hitting an UAV with an optical laser and by doing so, effectively guiding it back to the ground robot.

The structure, also known as Tractor Beam, is made of 3D PVC printed parts, resulting in a light frame. The 2 DoF, azimuth and elevation, were acquired using two stepper motor adjusted for 1:16 microstepping, resulting in an excellent motor resolution and very precise movement. The electrical design was made to allow for motor movement without pushing or crushing the cables. The structure carries the laser and a RGB camera, which is used to locate the UAV, follow it, and send the return command when needed. A Raspberry Pi 3 is used to control the motor and to process the computer vision algorithm. The designed computer vision algorithm relies on the green LEDs installed on the UAV frame. The algorithm has brightness adjustment regulation to solve the problem of changing ambient light. The algorithm uses a light filter along a color filter for initial segmentation. To remove outliers from the filter, the algorithm uses two metric filters, one to check if the distance of the points is consistent to previous measurement. The second metric filter will enter in effect once the algorithm has estimated the center of more than one LED frames, attempting to acquire multiples values with low standard deviation.

With a reliable LED center, which is the same as the center of the quadrotor, the algorithm will use the focus of the camera and camera calibration parameters to find the

motor rotation needed to displace the center of the quadrotor to the center of the image. This method demonstrated to be reliable in following the quadrotor and centering the laser underneath, where the sensor is positioned reaching a resolution of $elevation = 4.96^o$ and $azimuth = 0.27^o$, with a standard deviation of 1.03 and 0.51 degrees.

With a system capable of shooting a laser on the UAV, the problem is that the UAV most be capable to sensing and interpreting it. For this, a $0.5 \times 0.5\,m$ sensor was designed combining 36 photodetectors into a 6×6 matrix, forming a simplified camera. This sensor demonstrated to be capable of scanning all photodectors and sending the reading at a rate of $200\,Hz$, capable of reading the laser beam movement from $20\,m$ away of the source. This is a major improvement compared to a actual camera, which has a rate of $30\,fps = 30\,Hz$, as well as reducing the computation needed on the quadrotor.

To interpret the measurement of the sensor, an algorithm was designed on the UAV, where it first decodes the message received from the sensor and converts it to a binary number that refers to each specific photodector, or pixel. With multiple readings, the algorithm estimates the vector being draw by the laser dot on the sensor using linear approximation of the combination of points. The direction of the vector is obtained by the older and newer points, resulting in a velocity vector. This velocity vector can be sent to the UAV control algorithm just the same as a joystick command. The UAV proceed to follow this vector until the algorithm notices the rise in the distance of the points, which will command the laser to send a vector on the opposite way, commanding the UAV to stop. Finnaly, the UAV will be close enough to the ground robot for vision based landing algorithm to work on a reliable way.

5.2 Future Work

Possible improvements on the components of this work are as follow:

- **Sensor Characterization and Optimization:** During this work, the sensor was tested but not fully characterized. Plans for fully characterizing the sensor are being done to publish in a paper. Characterization could improve the capability of the sensor, resulting in even faster acquisition;

- **Photodector Voltage feeding:** On this application, the photodector received a small voltage of around 0.5V. According to the datasheet, for high speed application, such as communication, the photodectors circuit receives a voltage of 5V. This could greatly improve the sensor range of reading;

- **Photodector Array Improvement:** Some modifications on the photedetector array are being considered. The first change is to replace the manufacturer level

work to a final level work, where the circuit is built on by a hired company. This will allow for better space optimization;

- **Drone Detector Improvement:** Other methods can still be explored to improve the rate of the drone detector algorithm. A possibility for better filtering is using more LEDs, possibly of another color, such as purple, resulting in less points needed for the metric filter;
- **Tractor Beam Installation:** The structure is currently installed on a structure made of wood, which is done because both Mirã and the structure itself needed testing. After both tests are done, the structure should be installed on Mirã;
- **Protection installation:** As the goal of this system is to be used on rural regions, a protection against sand and particles was designed, but still needs final assembling;
- **Aerial Vehicle Test:** Due to problems with the aerial vehicle, only the laser screen method of the sensor was tested, and even with reduced capability, and it demonstrated to work. The structure was tested on our Lab using the quadrotor model, but still need to be tested outside on an actual quadrotor.

BIBLIOGRAPHY

ACHTELIK, M. et al. Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. In: **Proceedings - IEEE International Conference on Robotics and Automation**. IEEE, 2011. p. 3056–3063. ISBN 9781612843865. ISSN 10504729. Disponível em: <http://ieeexplore.ieee.org/document/5980343/>.

ANAC. REQUISITOS GERAIS PARA AERONAVES NÃO TRIPULADAS DE USO CIVIL. **REGULAMENTO BRASILEIRO DA AVIAÇÃO CIVIL ESPECIAL**, n. RBAC-E nº 94, 2017. Disponível em: <https://www.anac.gov.br/assuntos/legislacao/legislacao-1/rbha-e-rbac/rbac/rbac-e-94-emd-00/@@display-file/arquivo{_}norma/RBACE94EMD00.pdfhttps://www.anac.gov.br/assuntos/paginas-tematicas/dro>.

Andrew A. FrankMasahiko Nakamura. **Laser radar for a vehicle lateral guidance system**. 1991. Disponível em: <https://patents.google.com/patent/US5202742A/en>.

ASCTEC. **AscTec Pelican : Ascending Technologies**. Disponível em: <http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-pelican/>. Acesso em: 2018-02-05.

BECHAR, A.; VIGNEAULT, C. Agricultural robots for field operations: Concepts and components. **Biosystems Engineering**, v. 149, p. 94–111, 2016. ISSN 15375110.

Bryce E. Bayer. **Color imaging array**. US Grant, 1975. Disponível em: <https://patents.google.com/patent/US3971065>.

BUDIMIR, M. **Microstepping myths**. 2003. Disponível em: <http://www.machinedesign.com/archive/microstepping-myths>. Acesso em: 2018-05-02.

BURRIS, M. **Choosing Between Stepper Motors or Servo Motors**. 2019. Disponível em: <https://www.lifewire.com/stepper-motor-vs-servo-motors-selecting-a-motor-818841>.

CAMBRIDGE camera-exposure. Disponível em: <https://www.cambridgeincolour.com/tutorials/camera-exposure.htm>.

CAMBRIDGE image-noise. Disponível em: <https://www.cambridgeincolour.com/tutorials/image-noise.htm>.

CJ Taylor. **Coursera | Robotics: Computational Motion Planning - University of Pennsylvania**. Disponível em: <https://www.coursera.org/learn/robotics-motion-planning/home/info>.

CONROY, J. et al. Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. **Autonomous Robots**, v. 27, n. 3, p. 189, 2009. ISSN 1573-7527. Disponível em: <https://doi.org/10.1007/s10514-009-9140-0>.

CORKE, P. **Robotics, vision and control: fundamental algorithms in MATLAB®. Vol. 73. Springer Science & Business Media**. [S.l.: s.n.], 2011. 329–340 p. ISSN 1610-7438. ISBN 3540221085.

CRAIG, J. J. Introduction to Robotics: Mechanics and Control 3rd. **Prentice Hall**, v. 1, n. 3, p. 408, 2004. ISSN 0885-9000.

FAA GPS Product Team, W. J. H. T. C. **Global Positioning System (GPS) Standard Positioning Service (SPS) Performance Analysis Report 1284 Maryland Avenue SW**. [S.l.], 2014. 1–61 p. Disponível em: <http://www.nstb.tc.faa.gov/reports/PAN89{_}0415.>

FAESSLER, M. et al. Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle. **Journal of Field Robotics**, v. 33, n. 4, p. 431–450. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21581>.

FAO. The State of the World's Land and Water Resources for Food and Agriculture. Managing Systems at Risk. **Lancet**, v. 2, n. 7929, p. 285, 2011. ISSN 0140-6736.

FORSTER, C.; PIZZOLI, M.; SCARAMUZZA, D. SVO: Fast semi-direct monocular visual odometry. In: **Proceedings - IEEE International Conference on Robotics and Automation**. [s.n.], 2014. p. 15–22. ISBN 9781479936847. ISSN 10504729. Disponível em: <http://rpg.ifi.uzh.ch/docs/ICRA14{_}Forster.>

FORSTER, C. et al. SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems. **IEEE Transactions on Robotics**, v. 33, n. 2, 2017. ISSN 15523098.

Fraundorfer, F. et al. Vision-based autonomous mapping and exploration using a quadrotor mav. In: **2012 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2012. p. 4557–4564. ISSN 2153-0866.

HAMMOUDI, K. et al. Computing Multi-purpose Image-Based Descriptors for Object Detection: Powerfulness of LBP and Its Variants. In: YANG, X.-S. et al. (Ed.). **Third International Congress on Information and Communication Technology**. Singapore: Springer Singapore, 2019. p. 983–991. ISBN 978-981-13-1165-9.

Han, J. et al. Advanced deep-learning techniques for salient and category-specific object detection: A survey. **IEEE Signal Processing Magazine**, v. 35, n. 1, p. 84–100, Jan 2018. ISSN 1053-5888.

HIGUTI, V. A. H. et al. Cobem-2017-1647 robotic platform with passive suspension to be applied on soybean crops: initial results. **24th ABCM International Congress of Mechanical Engineering**, p. 10, 2017.

IMECHE. Global food waste not, want not. **Institute of Mechanical Engineers**, p. 31, 2013. Disponível em: <http://www.imeche.org/docs/default-source/reports/Global{_}Food{_}Report.pdf?sf>. Acesso em: 2018-04-07.

James Bowman, P. M. **camera_calibration - ROS Wiki**. Disponível em: <http://wiki.ros.org/camera{_}calibrat>. Acesso em: 2018-05-05.

Johnson, A. E.; Miguel San Martin, A. Motion estimation from laser ranging for autonomous comet landing. In: **Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)**. [S.l.: s.n.], 2000. v. 1, p. 132–138 vol.1. ISSN 1050-4729.

JOSEPH, L. **Mastering ROS for Robotics Programming**. [S.l.: s.n.], 2015. v. 64. 451 p. ISSN 0717-6163. ISBN 9781783551798.

J.Y.BOUGUET. **Camera Calibration Toolbox for Matlab**. 2014. Disponível em: <http://www.vision.caltech.edu/bouguetj/calib{_}d>. Acesso em: 2018-02-05.

KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. **Journal of Basic Engineering**, v. 82, n. 1, p. 35, 1960. ISSN 00219223. Disponível em: <http://fluidsengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1430402>.

KELLY, A. Growing Food , Growing Problems. p. 1–5, 2012. Disponível em: <http://knowledge.wharton.upenn.edu/article/growing-food-growing-problems/>.

Kostas Daniilidis, J. S. **Coursera | Robotics: Perception**. 2017. 1–4 p. Disponível em: <https://www.coursera.org/learn/robotics-perception/home/welcome>.

LACKEY, B. **What's the Difference Between Servo and Stepper Motors?** 2018. Disponível em: <https://www.machinedesign.com/motion-control/what-s-difference-between-servo-and-stepper-motors>.

Larenas, M.C, Magalhães, D.V. and Milori, D. Sistema de enfoque automático para espectroscopia de plasma induzido por laserT. In: **In Proceedings of IX Reunion Ibero-americana de óptica y XII reunión Ibero-americana de óptica, lasers y aplicaciones – RIAO/OPTILLA2016**. Pucón, Chile: In: REUNIÃO IBEROAMERICANA DE ÓPTICA, 9.; ENCUENTRO LATINOAMERICANO DE ÓPTICA, LASERES Y APLICACIONES, 12., 2017., 2016. Disponível em: <https://www.alice.cnptia.embrapa.br/handle/doc/1079751>.

Larenas, M.C, Milori, D.M.B.P. and Magalhães, D. Efeito da compactação do solo sobre o nível da sinal obtida por espectrometria de emissão óptica com plasma inducido por laser (LIBS). In: **In Proceedings of 2° Simpósio do programa de pós-graduação em engenharia mecânica da EESC-USP – SiPGEM2017**. São Carlos-SP, Brazil: [s.n.], 2016. p. 2016.

LEHNERT, C. et al. Autonomous Sweet Pepper Harvesting for Protected Cropping Systems. **IEEE Robotics and Automation Letters**, v. 2, n. 2, p. 872–879, 2017. ISSN 23773766.

LYNEN, S. et al. A robust and modular multi-sensor fusion approach applied to MAV navigation. In: **IEEE International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2013. p. 3923–3929. ISBN 9781467363587. ISSN 21530858.

MEENAKSHISUNDARAM, V. S.; GUNDAPPA, V. K.; KANTH, B. S. Vector field guidance for path following of MAVs in three dimensions for variable altitude maneuvers. **International Journal of Micro Air Vehicles**, v. 2, n. 4, p. 255–265, 2011. ISSN 1756-8293.

MENENDEZ-APONTE, P. et al. Software and hardware architectures in cooperative aerial and ground robots for agricultural disease detection. **Proceedings - 2016 International Conference on Collaboration Technologies and Systems, CTS 2016**, p. 354–358, 2016.

MONTGOMERY, J. et al. The jet propulsion laboratory autonomous helicopter testbed: A platform for planetary exploration technology research and development. **J. Field Robotics**, v. 23, n. 2000, p. 245–267, 2006. ISSN 14746670.

NAGATA, H. et al. Development of entertainment balloon robot system for an indoor event venue. **Artificial Life and Robotics**, Springer Japan, v. 0, n. 0, p. 1–8, 2017. ISSN 16147456. Disponível em: <http://dx.doi.org/10.1007/s10015-017-0419-5>.

OH, K. H.; AHN, H. S. Extended Kalman filter with multi-frequency reference data for quadrotor navigation. **ICCAS 2015 - 2015 15th International Conference on Control, Automation and Systems, Proceedings**, n. Iccas, p. 201–206, 2015.

OLSON, E. AprilTag: A robust and flexible visual fiducial system. **Proceedings - IEEE International Conference on Robotics and Automation**, p. 3400–3407, 2011. ISSN 10504729.

Population Reference Bureau. **2017 World Population Data Sheet**. 2017. 20 p. Disponível em: <https://www.prb.org/2017-world-population-data-sheet>. Acesso em: 2018-04-07.

PRINSLOO, G.; DOBSON, R. **Solar Tracking, doi = 10.13140/RG.2.1.4265.6329/1, institution = SolarBooks, isbn = 97890365338671, year = 2015**. [S.l.: s.n.]. 1–542 p.

QURESHI, W. S. et al. Machine vision for counting fruit on mango tree canopies. **Precision Agriculture**, v. 18, n. 2, p. 224–244, 2017. ISSN 1573-1618. Disponível em: <https://doi.org/10.1007/s11119-016-9458-5>.

RAMOS, P. et al. Automatic fruit count on coffee branches using computer vision. **Computers and Electronics in Agriculture**, v. 137, p. 9 – 22, 2017. ISSN 0168-1699. Disponível em: <http://www.sciencedirect.com/science/article/pii/S016816991630922X>.

REGGIANI, L.; POLITECNICO, D. Drones in Agriculture Applications. p. 1–9, 2015.

Richard Hartley, A. Z. **Multiple View Geometry**. [S.l.: s.n.], 2003. v. 53. 1689–1699 p. ISSN 1098-6596. ISBN 9788578110796.

RODRIGUES, R. T. **Vision-based Autonomous Landing of mini-UAV with Quadrotor Configuration**. 2016. Tese (Qualifição de Mestrado) — Universidade de São Paulo, 2016.

RODRIGUES, R. T. **Aterrissagem Autônoma Baseada em Visão para Mini Quadrirrotor**. 2017. Tese (Dissertação de Mestrado) — Universidade de São Paulo, 2017.

ROTH, P.; GEORGIEV, A.; BOUDINOV, H. Design and construction of a system for sun-tracking. **Renewable Energy**, Pergamon, v. 29, n. 3, p. 393–402, mar 2004. ISSN 09601481. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0960148103001964>.

SANKARAN, S. et al. **Low-altitude, high-resolution aerial imaging systems for row and field crop phenotyping: A review**. 2015.

Santos Junior, D. et al. Espectrometria De Emissão Óptica Com Plasma Induzido Por Laser (Libs) **-. Revista Analytica**, n. August, p. 10, 2006.

SOLER, D. et al. **AUTONOMOUS QUADROTOR-TRACKING CAMERA ALGORITHM**. 2018. Disponível em: <http://soac.eesc.usp.br/index.php/SiPGEM/iiisipgem/paper/view/1346/812>.

Stephan M. Weiss. **Vision Based Navigation for Micro Helicopters**. 2012. Tese (Doctoral Thesis) — ETH ZURICH, 2012. Disponível em: <http://publications.asl.ethz.ch/files/weiss12versatile.pdf>.

Stephan Weiss, M. A. **ethzasl_sensor_fusion - ROS Wiki**. 2012. Disponível em: <http://wiki.ros.org/ethzasl{_}sensor{_}>. Acesso em: 2018-04-03.

TABATABAEI, S. A. H. et al. Three dimensional fuzzy carrot-chasing path following algorithm for fixed-wing vehicles. **International Conference on Robotics and Mechatronics, ICROM 2015**, p. 784–788, 2015.

TEACH, R. W. D. F. R. L. **Laser guidance system for crop spraying aircraft**. 1978. Disponível em: <https://patents.google.com/patent/US4225226A/en>.

TECH-FAQ hsv. Disponível em: <http://www.tech-faq.com/hsv.html>.

Vijay Kumar. **Coursera | Robotics: Aerial Robotics**. Disponível em: <https://www.coursera.org/learn/robotics-flight/home/welcome>.

WANG, J.; OLSON, E. AprilTag 2: Efficient and robust fiducial detection. **IEEE International Conference on Intelligent Robots and Systems**, v. 2016-Novem, p. 4193–4198, 2016. ISSN 21530866.

WEISS, S. **asctec_mav_framework - ROS Wiki**. 2011. Disponível em: <http://wiki.ros.org/asctec{_}mav{_}fra>. Acesso em: 2018-04-02.

WELCH, G.; BISHOP, G. An Introduction to the Kalman Filter. **In Practice**, v. 7, n. 1, p. 1–16, 2006. ISSN 10069313. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.79.6578{&}rep=rep1{&}ty>.

WORLDFOODSUMMIT. **WATER AND AGRICULTURE**. Disponível em: <http://www.fao.org/WorldFoodSummit/sideevents/papers/Y6899E.htm>. Acesso em: 2018-04-07.

ZHANG, Z. A Flexible New Technique for Camera Calibration (Technical Report). **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 22, n. 11, p. 1330–1334, 2002. ISSN 01628828.

APPENDIX A – HARDWARE

This chapter has the goal to introduce the UAV used in this work. On subsection A.1.1, the AscTec Pelican hardware and specifications are presented, introducing its components. The framework used to control the UAV are discussed in section A.2 and, based on (WELCH; BISHOP, 2006), the Kalman Filter will be introduced in **??** to better discuss framework pose prediction.

A.1 Pelican

The Ascending Technologies, now part of the Intel, has developed multiple research and professional multi-rotor aircrafts. The quadrotor used in this dissertation is the AscTec Pelican. It has the highest payload among all the AscTec UAVs and its tower design enables easy customization and adaptation. The Pelican frame is built out of laminated carbon fiber into plywood, resulting in a robust and lightweight frame.

The custom Pelican used in the dissertation is equipped with a AscTec Autopilot, Odroid XU4, mvBlueFOX 2, Pan Tilt Unity (PTU), IMU, pressure sensor, GPS and a radio link for pilot teleoperation. Figure 38 shows a picture of the AscTec Pelican from the manufacture website.

The AscTec Autopilot contains two ARM-7 (Advanced RISC Machines) processors. The low level processor (LLP), runs the priority tasks including IMU data fusion and a

Figure 38: AscTec Pelican

Source: <http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-pelican/>, 23-04-2018

Table 7: AscTec Pelican Technical Data

UAV Type	**Pelican**
Max. takeoff weight	1.65 kg
Max. payload	0.65 kg
Max. airspeed	16 m/s
Max. climb rate	8 m/s
Max. thrust	36 N

robust attitude controller. Users are not granted access to the LLP, however they do have access to the high level processor (HLP), which can be coded in C language and have communication channel to the LLP using Asctec Communication Interface (ACI) protocol.

Data from the IMU (accelerometers, gyroscopes and compass) and a pressure sensor are sent to the LLP, which runs an attitude data fusion filter at 1 kHz. A GPS provides absolute position to the HLP.

The ODROID-XU4 is a 2 Ghz, Octa core onboard computer. It is extremely powerful and, energy efficient and, runs Ubuntu 16.04 on a 16 Gb high rate micro SD card, is equipped with two USB 3.0 and one USB 2.0 port. Communication to the HLP/LLP processors is carried out through a serial port. Communication with ground station is done through a 3.0 USB Wi-Fi Hub using a *ssh* connection.

The quadrotor is also equipped with a camera, Matrix Vision mvBlueFox 2.0, which is a compact 8 Mpixels camera with a resolution of 752x480. The camera is mounted on a Pan Tilt Unit (PTU), which adjusts the camera to keep the camera xy plane parallel to the vehicle plane by countering the vehicle rotations.

Table 7 shows performance limitation of the quadrotor extracted from the manufacturer's website (ASCTEC,).

A.1.1 Sensor Performance

Our AscTec Pelican has been used by many other students during the years, e.g. on (RODRIGUES, 2016), the Pelican sensors were tested. Since the sensors have not been changed it is possible to assume that such results are still valid. The test was conducted after sunset in a clear night sky. The procedure was as follows: the quadrotor was left in turned on for about 10 minutes to stabilize, then turned the rotors for about half speed and started recording the data for another 10 minutes. The conclusion of the test was that the Pelican's GPS error is $\pm$3.5m for x coordinate and $\pm$7.0m for y coordinate.

Withi this, it is possible to conclude that the quadrotor could leave the Rover and then return to a position where it would not find the landing spot again.

A.2 Asctec MAV Framework

To establish a connection between ROS and the HLP on the Asctec Autopilot we use the ROS package asctec_mav_framework. This Ros package creates an interface that allows for quadrotor autonomous control and sensor data acquisition through a set of ROS nodes. The package has three modes of operation for the HLP, direct acceleration control, GPS Based Velocity Control, and Position Control on the HLP. All those methods are explained on the package wiki page (WEISS, 2011). For this work, only the position control was used. At any point during flight or in case of fail, control can be toggle to the LLP, allowing operator control of the quadrotor.

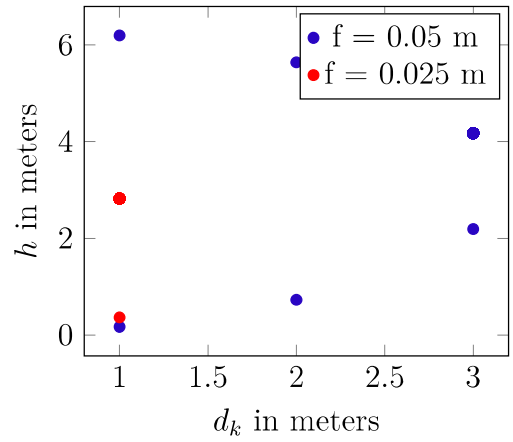
The position control mode is explained in (ACHTELIK et al., 2011), where the authors use pose estimation from visual SLAM and IMU. Then, this information is sent to the AscTec autopilot via ROS where the LLP provides reliable attitude control and the position control is implemented on the HLP. By sending desired position commands through the interface, the position controller on the HLP, that relies on nonlinear dynamic inversion and PD controllers, models the quadrotor translation dynamics at 1 kHz.

For the position controller to work properly, fast data fusion must be sent to the LLP and HLP. This is done using another ROS package, *ethzasl_sensor_fusion* (Stephan Weiss, 2012). This package contains a sensor fusion framework designed to fuse data from a IMU and a camera, performing visual odometry, based on an Extended Kalman Filter for vehicle pose estimation.

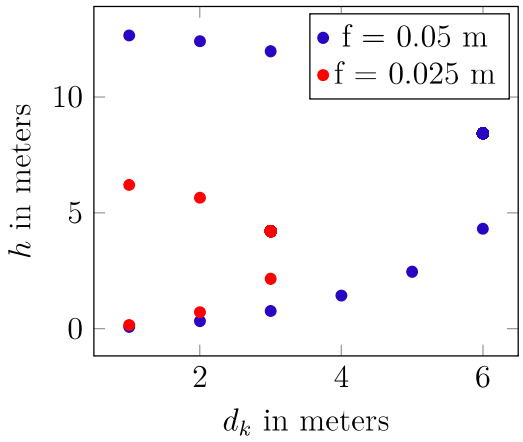
In (LYNEN et al., 2013) and on his Phd Thesis (Stephan M. Weiss, 2012), the author shows all the calculation involved for state estimation and prediction used on the EKF, which can receive sensor pose estimation at rates as low as 10 Hz, and publishs predictions at 100 Hz with acceptable error. In the article, the author also introduce his approach for multi sensor fusion, enabling the package to be used for an unlimited number of sensors.

APPENDIX B – HORIZONTAL DISTANCE VS HEIGHT

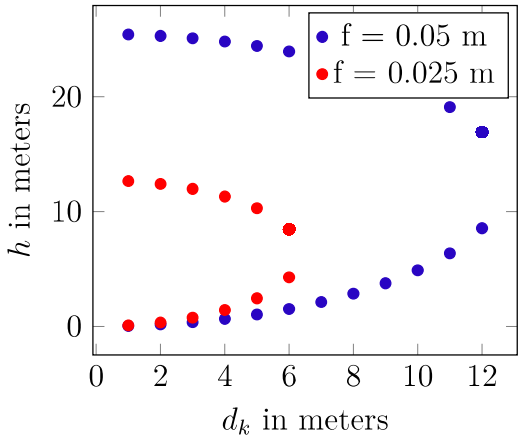
Figure 39: Horizontal Distance vs Height for comparison between the all driver microsteps configurations



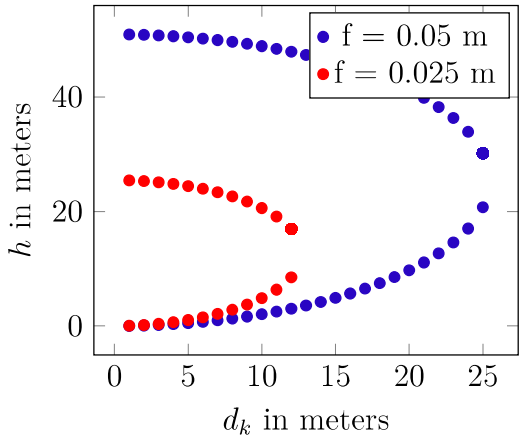
(a) 1:4 Microsteps



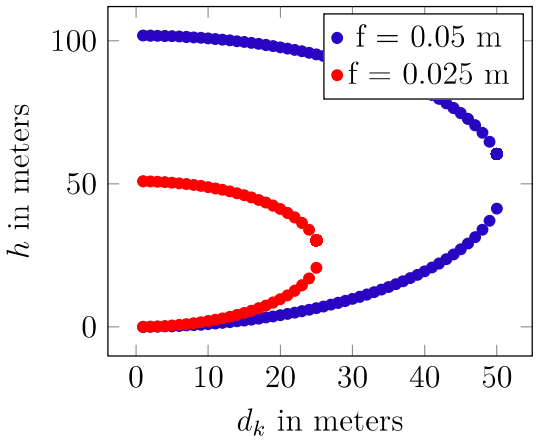
(b) 1:8 Microsteps



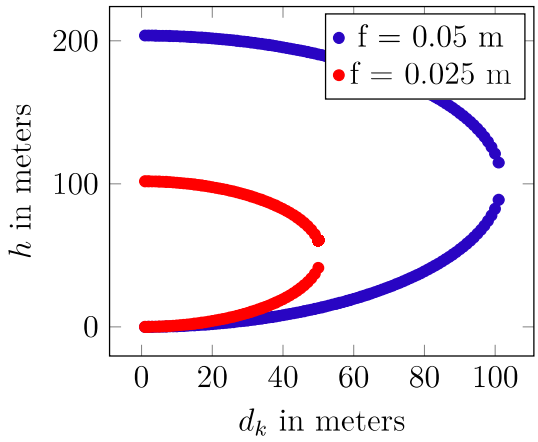
(c) 1:16 Microsteps



(d) 1:32 Microsteps



(e) 1:64 Microsteps



(f) 1:128 Microsteps

Source: Authors own

APPENDIX C – 2-DOF LASER STRUCTURE CAD DRAWINGS

Figure 40: Laser Structure Isometric View

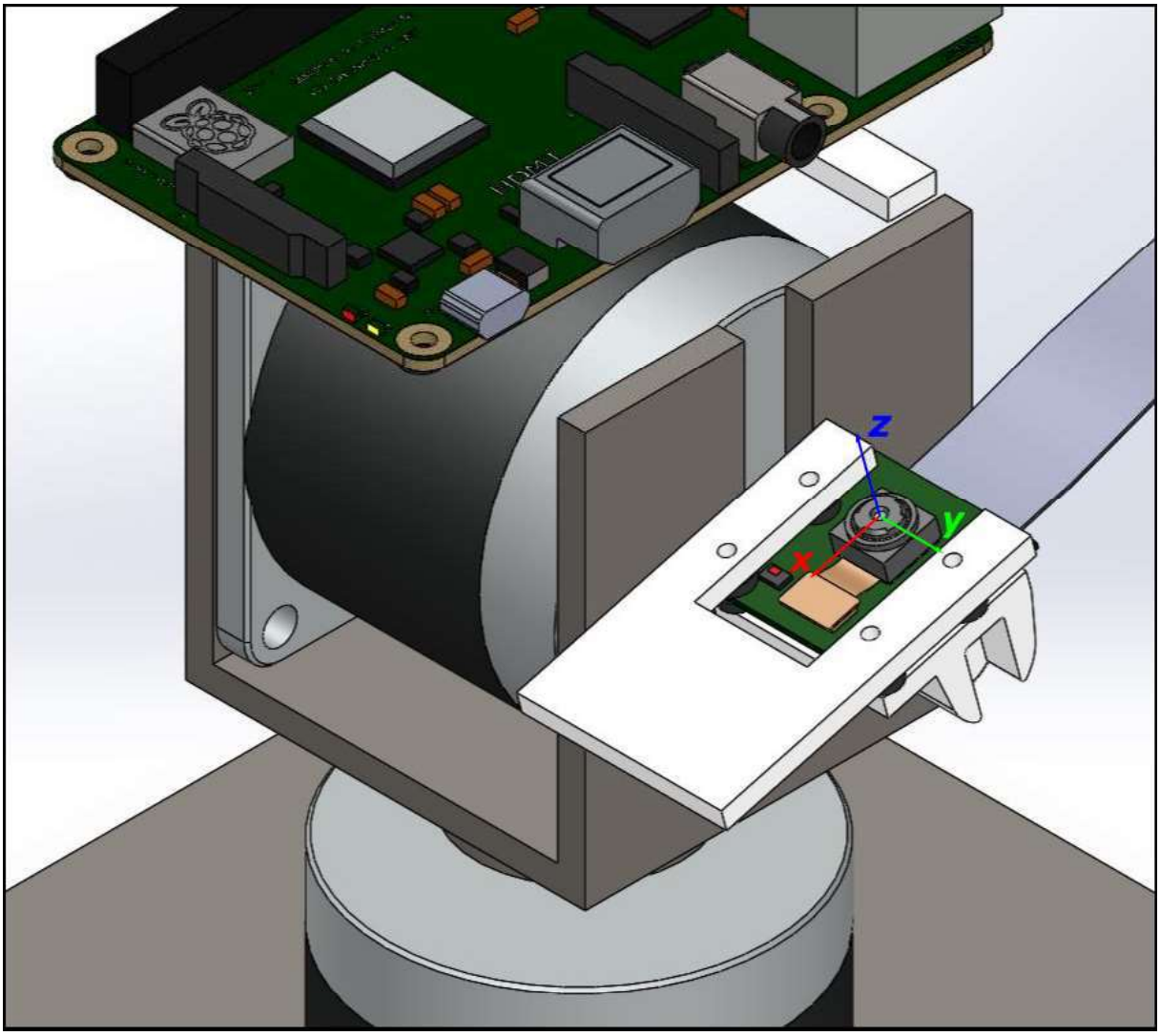


Figure 41: Laser Structure Isometric View with protection

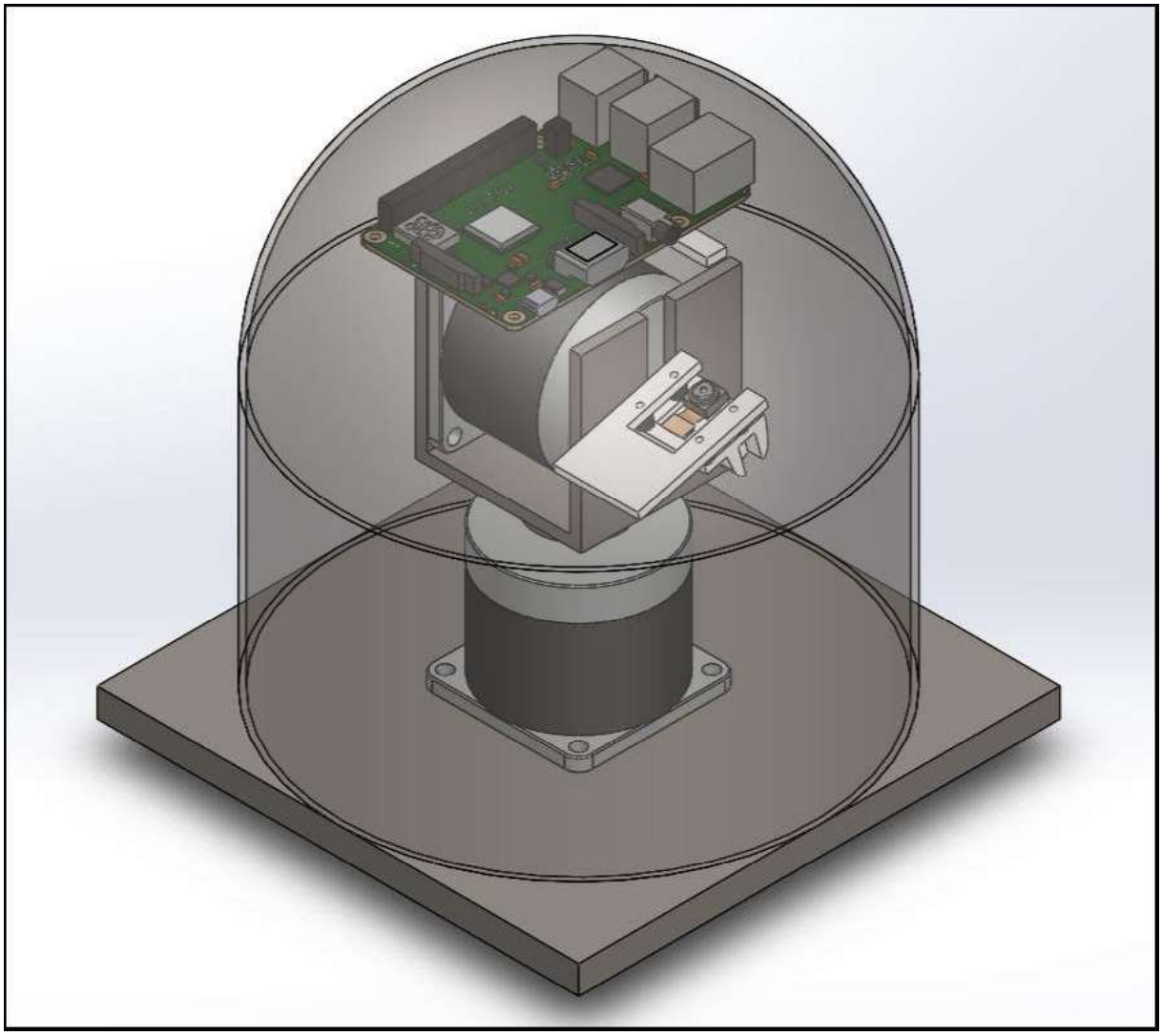


Figure 42: Stepper Motor Axis. Note that Motor 2 CG is on Motor 1 z axis and the camera axis is parallel to Motor 2 yz plane.

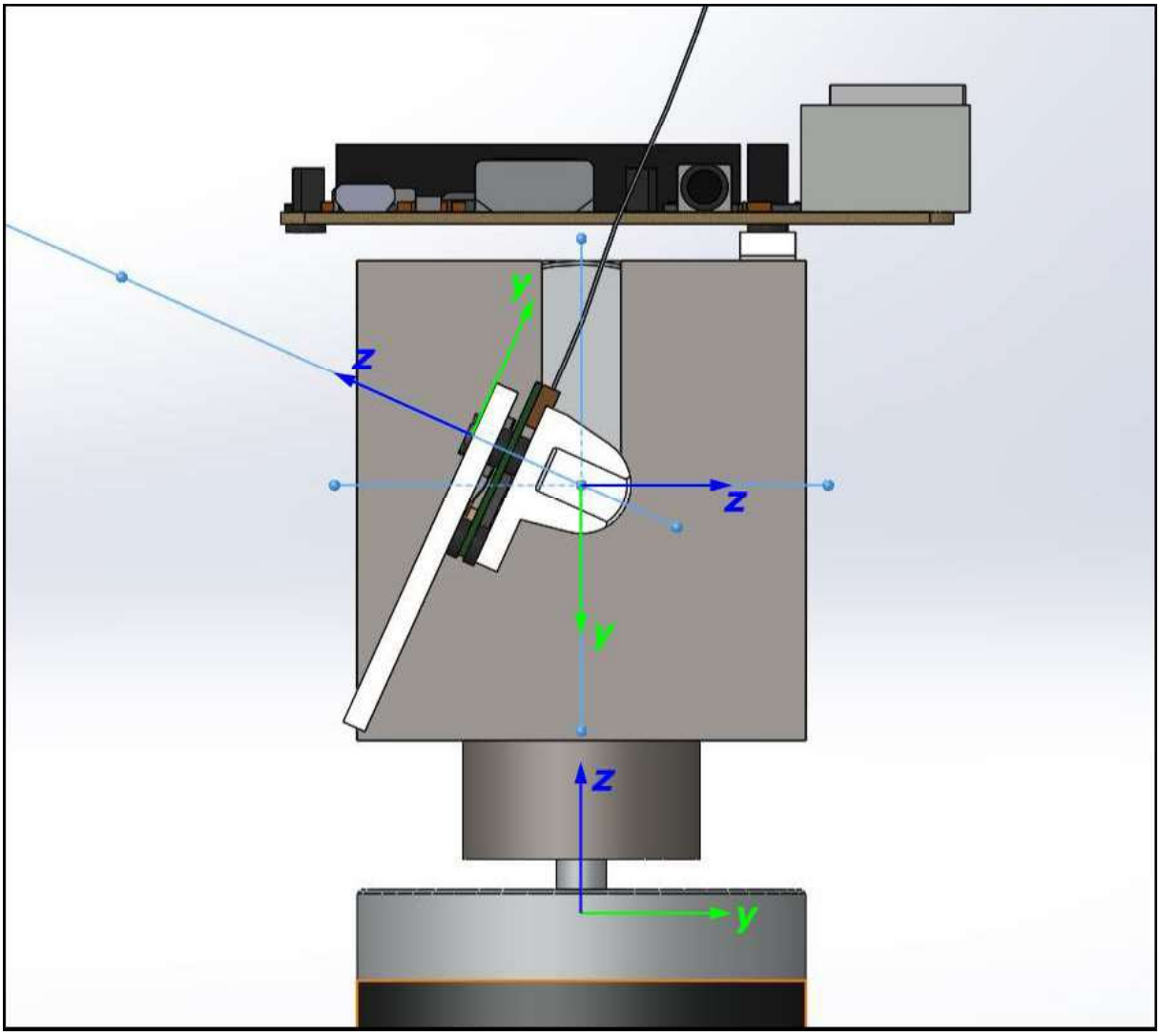
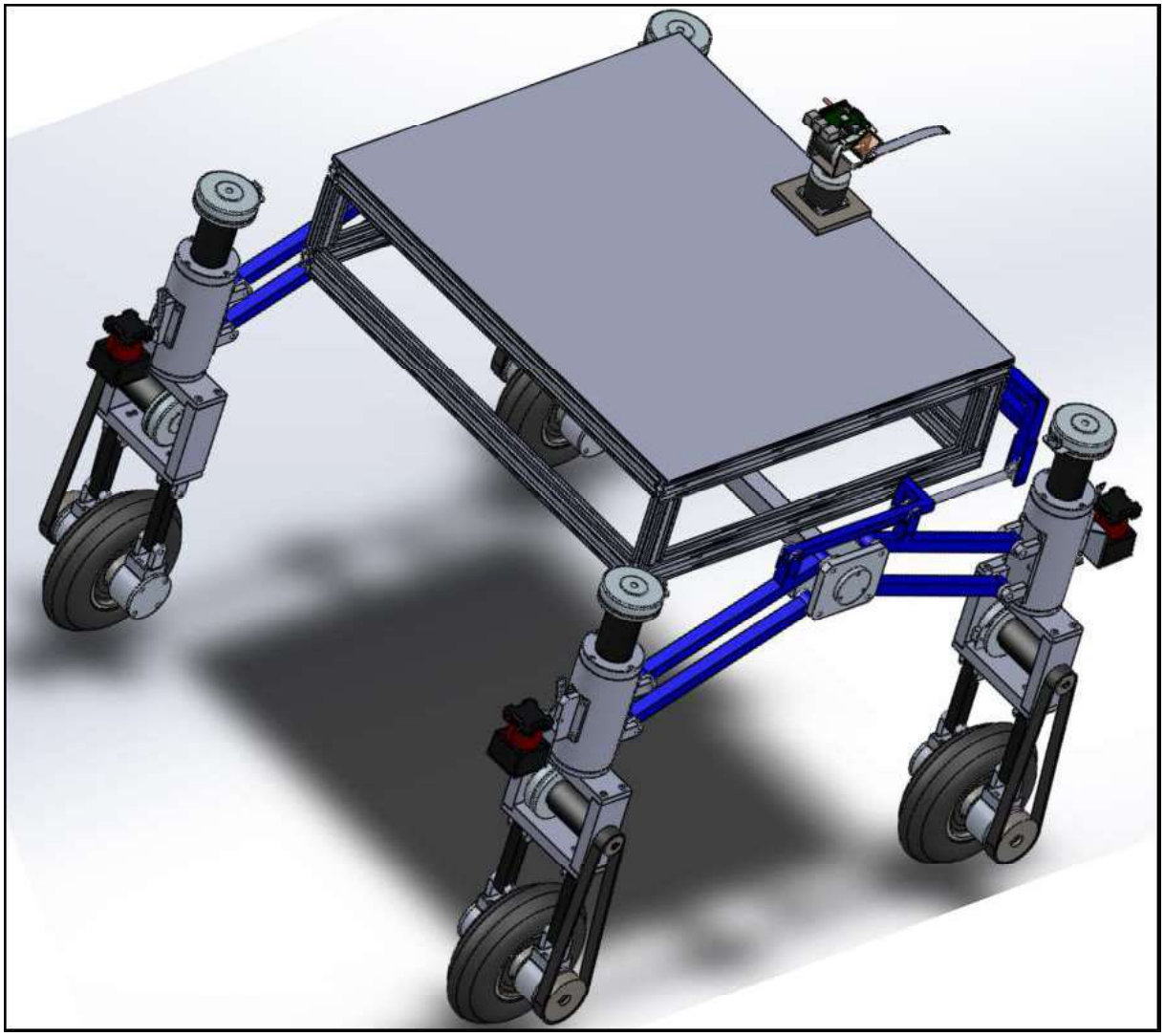


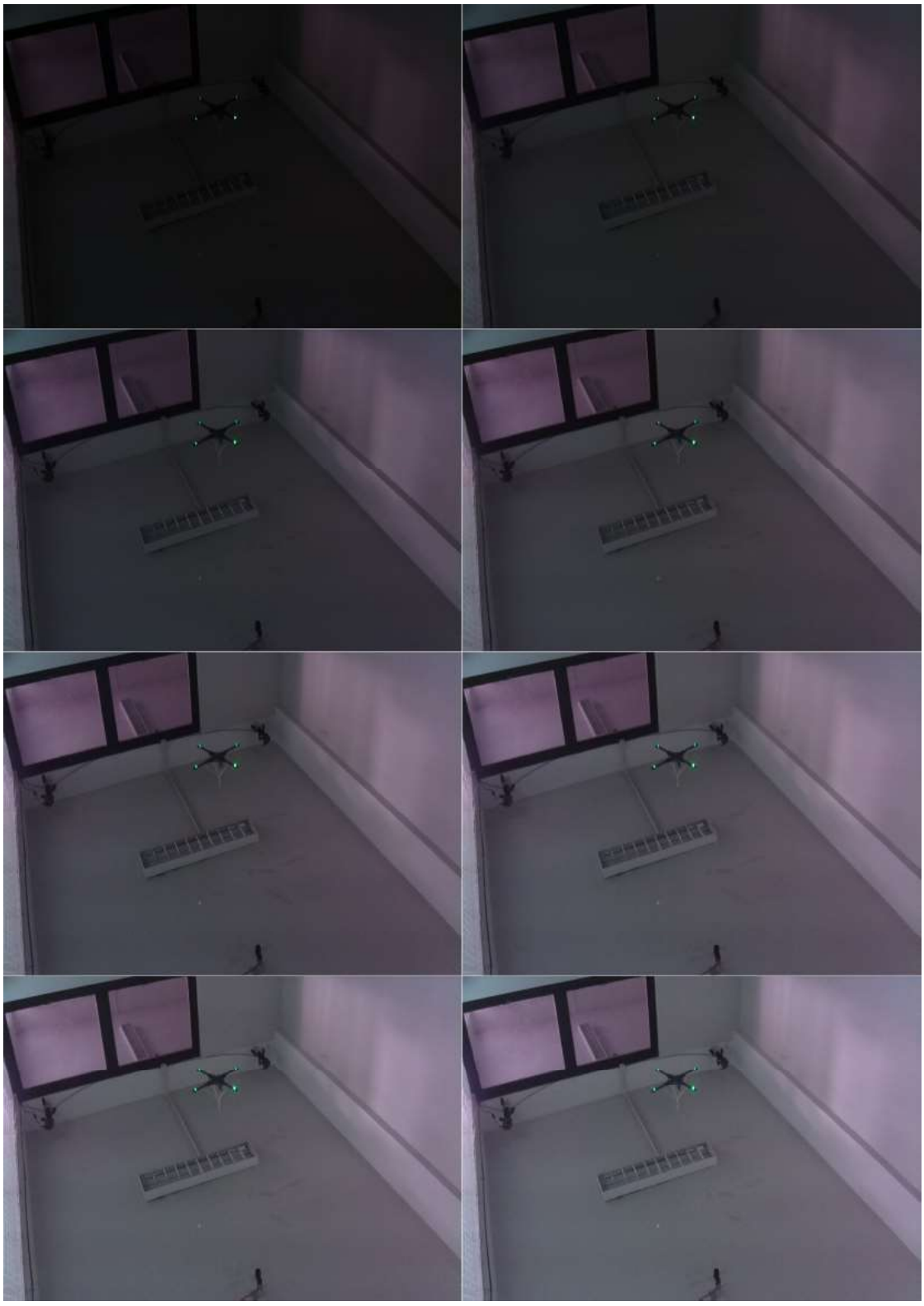
Figure 43: Laser Structure mounted on Mirã



Source: Authors own

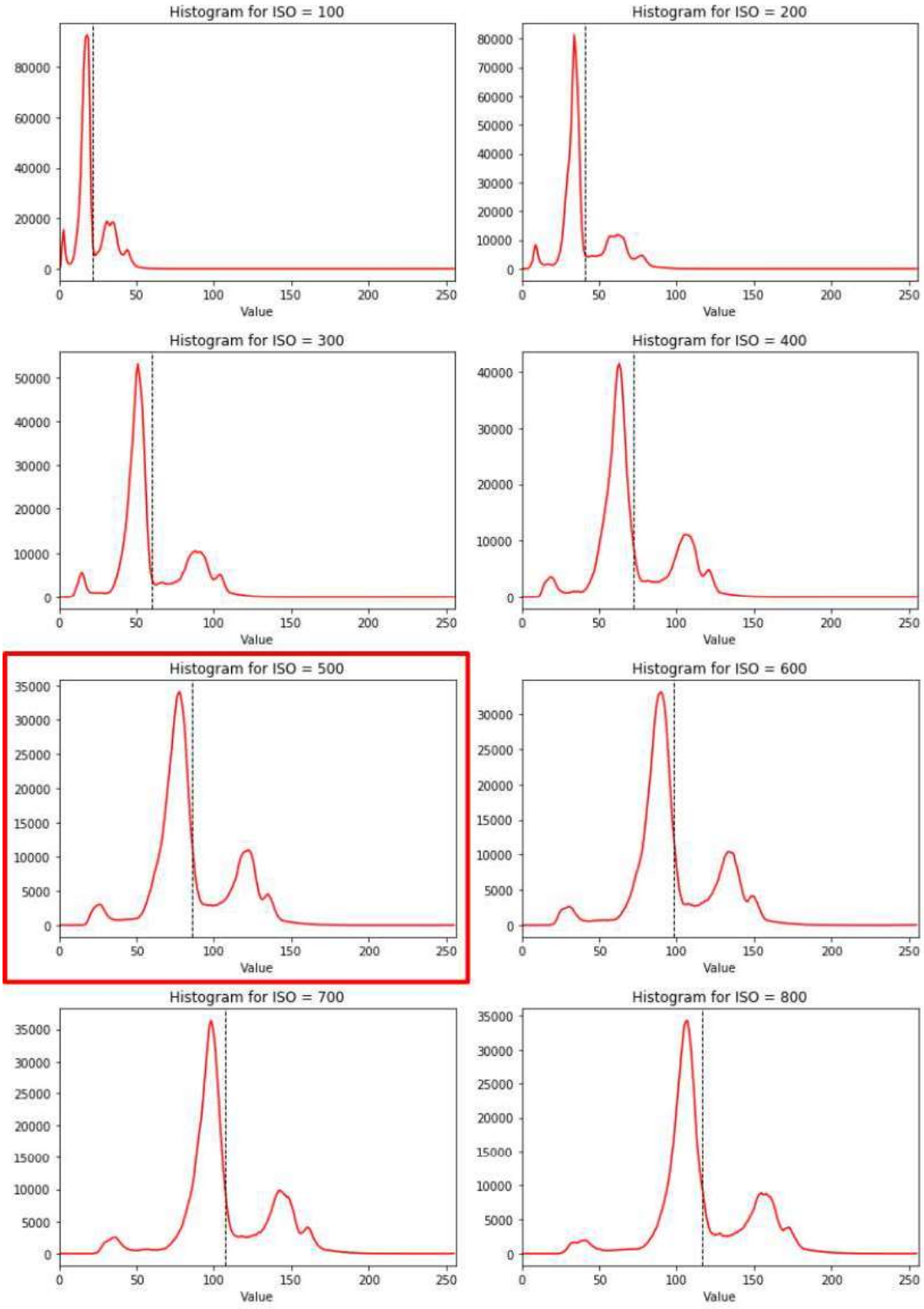
APPENDIX D – WHITE LIGHT FILTER HISTOGRAM COMPARISON

Figure 44: Histogram comparison for multiple ISO value.



ISO value ranging in increments of 100 from, top left: ISO = 100 until bottom right: ISO = 800.

Figure 45: Histogram plot for multiple ISO value.



APPENDIX E – SENSOR ACQUISITION RATE ESTIMATION

The laser structure must send the velocity command in a way that is possible for the sensor to read. This is a particular problem because, as described on subsection 3.1.5, the stepper motor is rotating in a fixed 1250 *steps per second* velocity, which is equivalent to $23,43\,rpm$. The problem is, as the distance between the quadrotor and the laser raises, the laser dot velocity on the sensor also raises. What this means is that, with the sensor data acquisition is not high enough the sensor could get any data from the laser. This is a problem observed during the laser screen validation, and described on subsection 4.1.3.

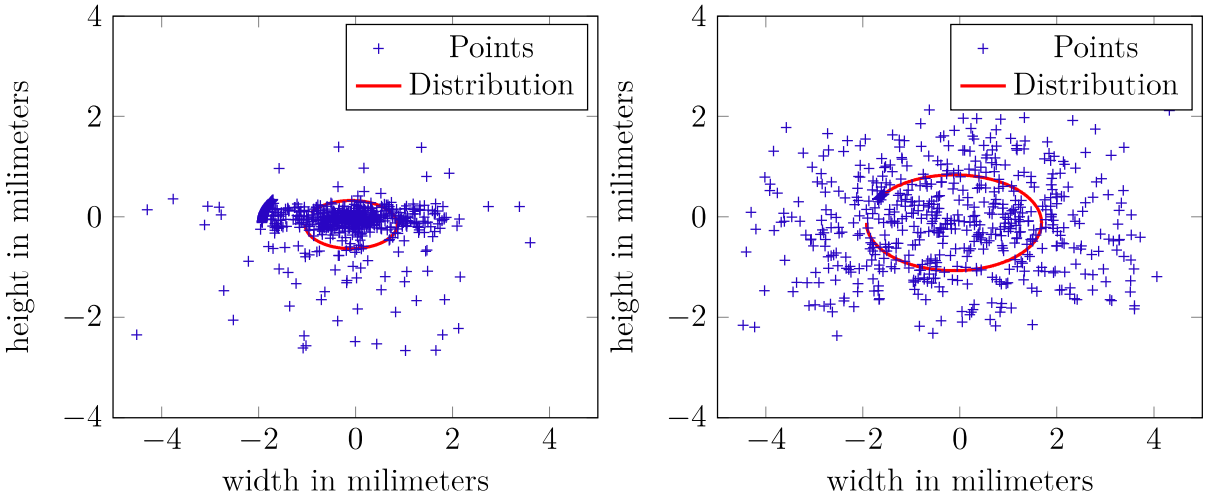
To design the final version of the sensor, the data acquisition frequency was first estimated. Assuming the laser horizontal velocity v_h can be obtained using the following equation,

$$v_h = a_h * \left(\frac{2 * \pi * \omega_M}{60}\right) * cos(\theta) \tag{E.1}$$

where at a given point in time, a_h is the distance between the laser and the sensor, ω_M the motor rotation, or $23,43\,rpm$, and θ the angle.

To estimate the sensor acquisition, assuming for the equation above, $a_h = 20\,m$ and $\theta = 30^o$, the laser dot will travel at around $v_h = 42.5\,m/s$. Now, with the size of the sensor $f = 50\,mm$, the laser spot will pass the sensor at a frequency of $85\,Hz$, two points are needed to estimate a velocity vector, which result in a frequency of $170\,Hz$. Finally, considering a safety threshold, the sensor acquisition rate should be designed to be at least $200\,Hz$.

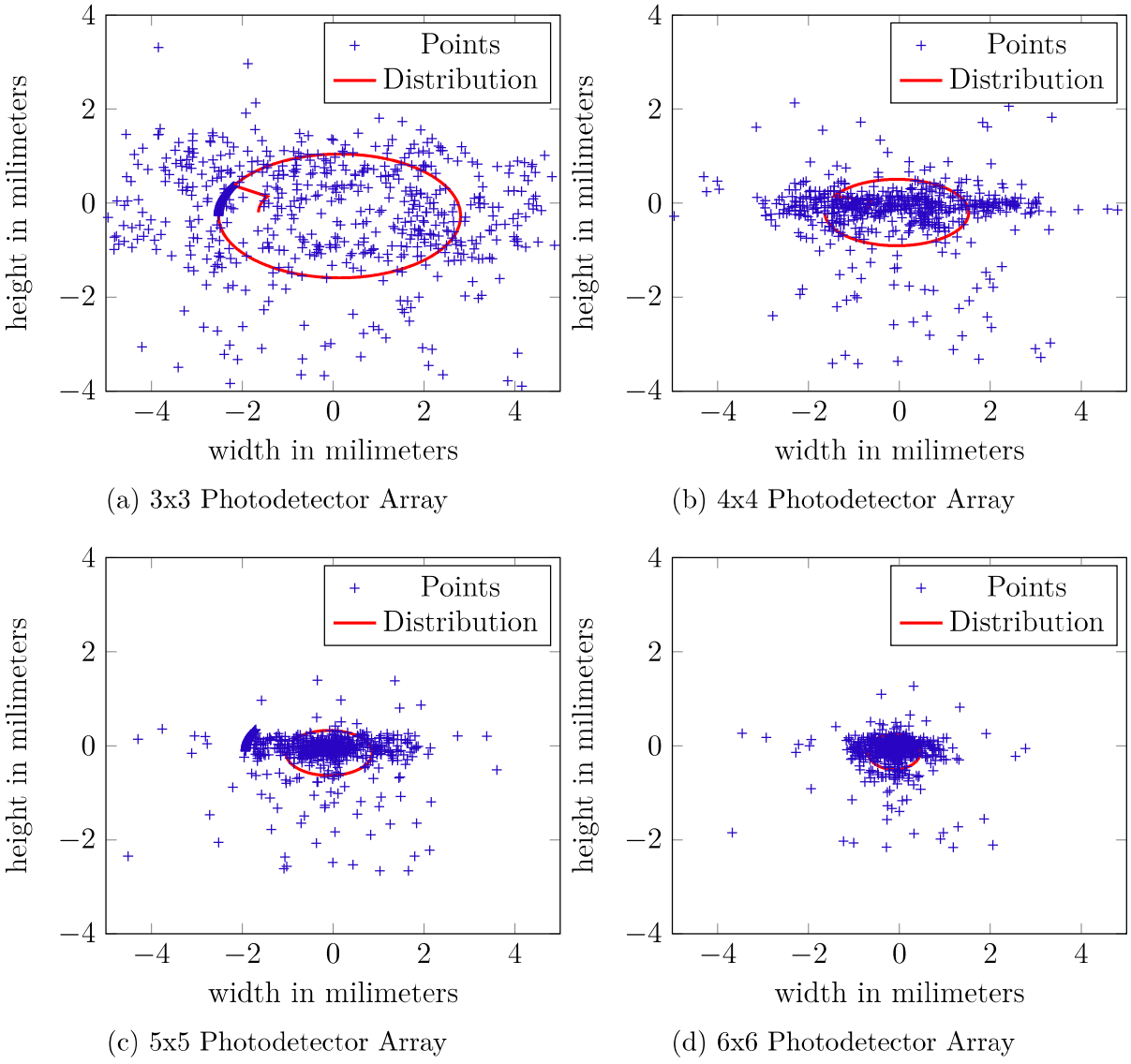
APPENDIX F – SENSOR ERROR ESTIMATION DUE SCALING



(a) Grayscale (b) Binary

Error distribution comparison for two estimation methods, grayscale and binary, for a screen of 50mm width and 30mm height.

APPENDIX G – SENSOR SIZE COMPARISON



(a) 3x3 Photodetector Array

(b) 4x4 Photodetector Array

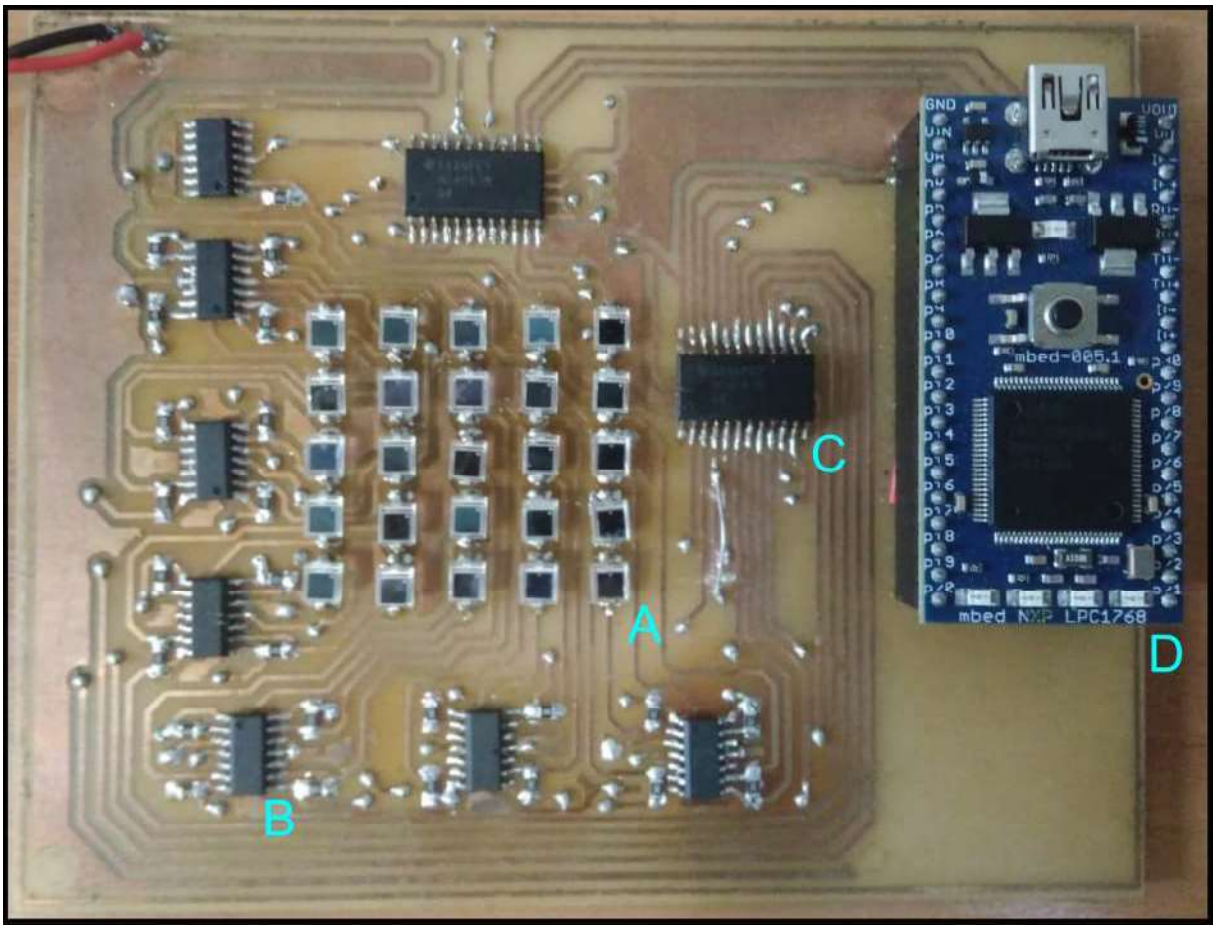
(c) 5x5 Photodetector Array

(d) 6x6 Photodetector Array

Error distribution comparison for four photodectector arrays size, for a sensor of 50mm width and 30mm height.

APPENDIX H – PHOTO DETECTOR SENSOR: VERSION 1

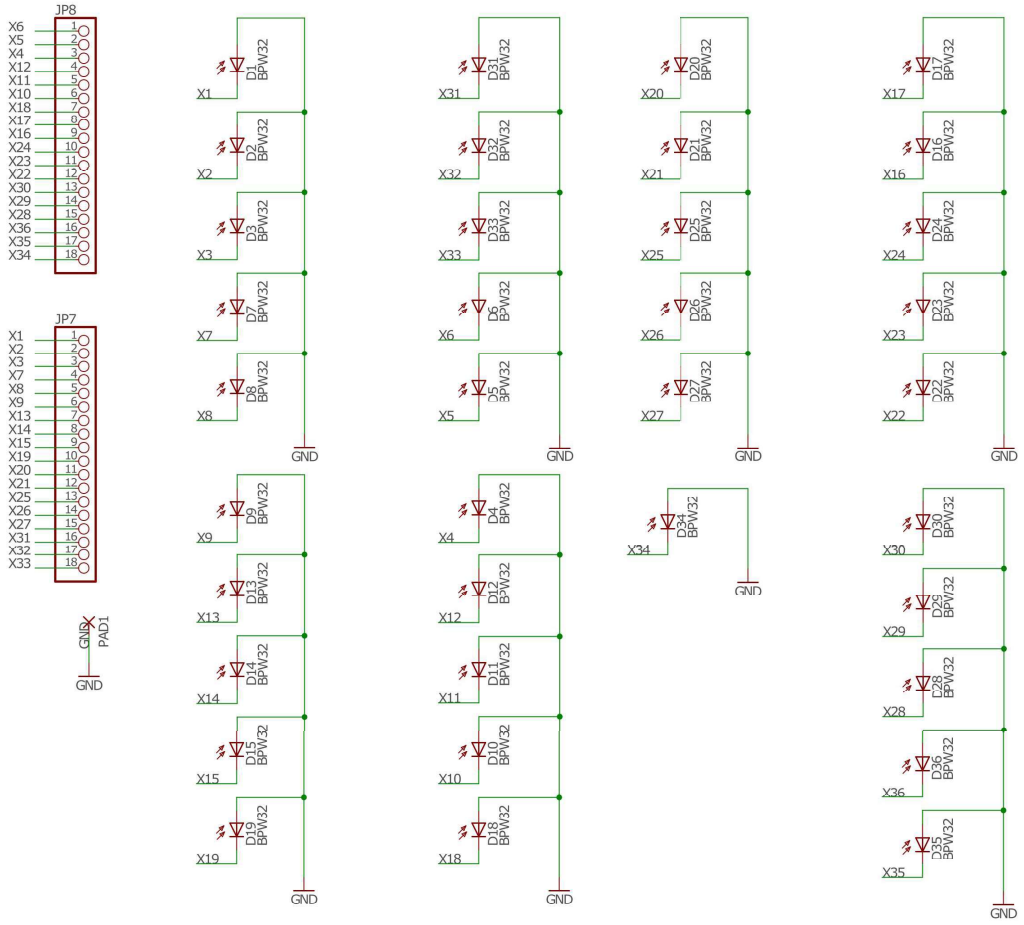
Figure 48: Picture of the first version of the photo detector sensor. The components on the circuit are: (A) Photo Detector; (B) Op-Amps; (C) Multiplexer; (D) Mbed.



Source: Authors own

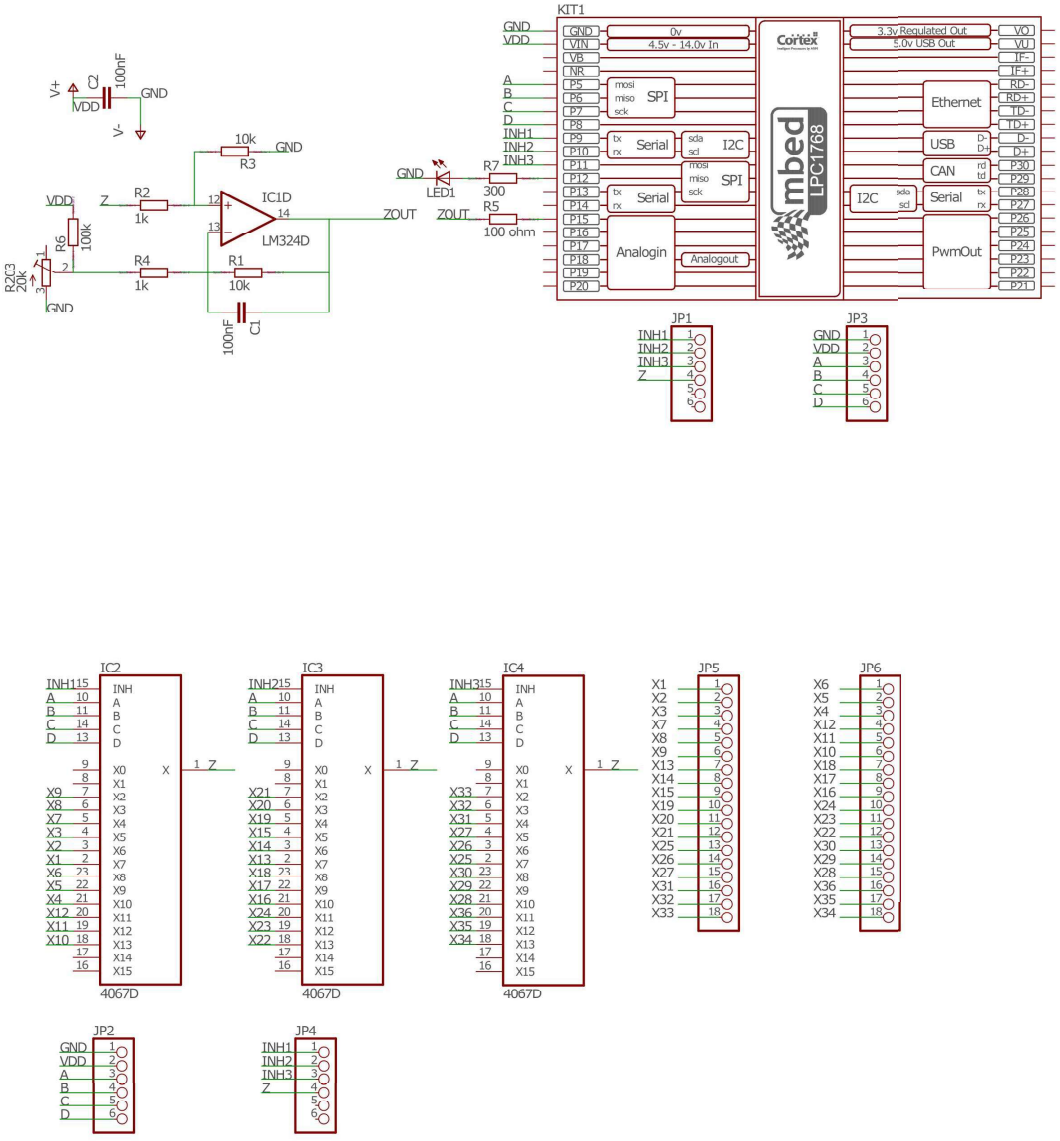
APPENDIX I – PHOTO DETECTOR SENSOR: VERSION 2

Figure 49: Schematics of both the mbed and the multiplexer of the second version of the sensor



Source: Authors own

Figure 50: Schematics of both the mbed and the multiplexer of the second version of the sensor

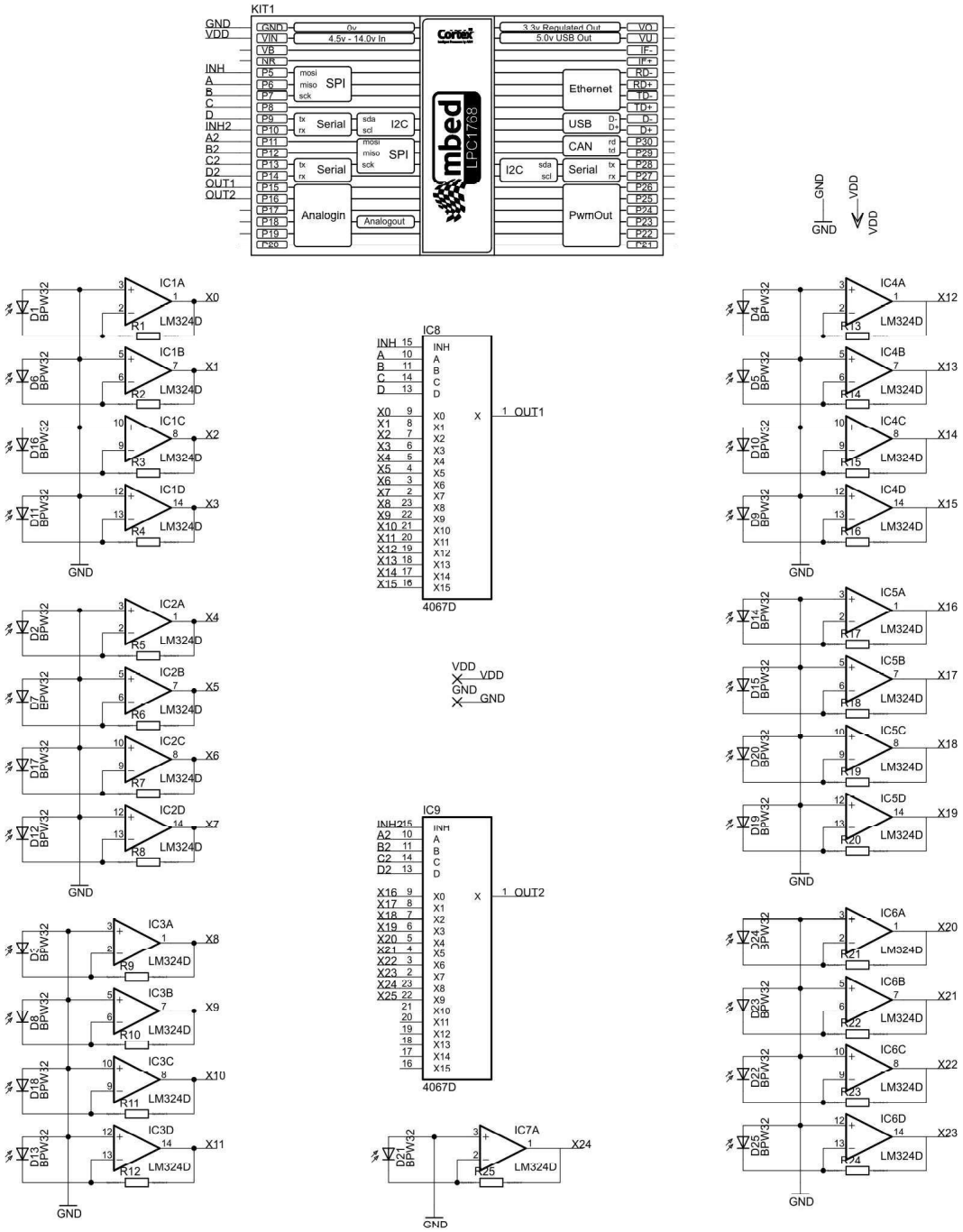


Source: Authors own

Annex

ANNEX A – PHOTODETECTOR MATRIX V1 SCHEMATICS

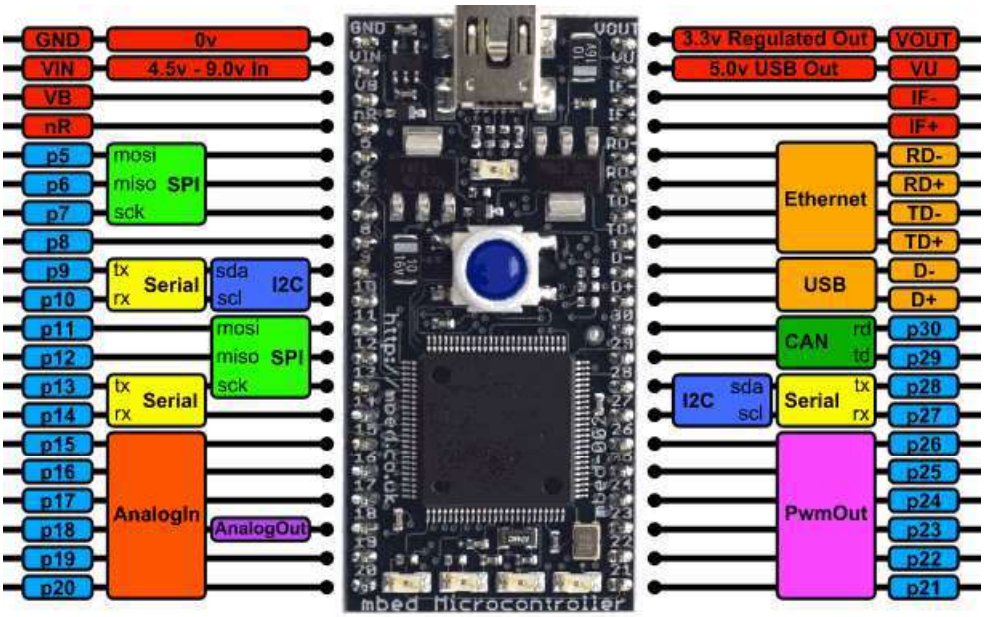
Figure 51: Schematics



Source: Richard Mascarin, Laboratório de Instrumentação Eletrônica (Liepo) <http://cepof.ifsc.usp.br/inovacao/LIEPO-Port>

ANNEX B – MBED LPC 1768 REFERENCE

Figure 52: Mbed LPC 1768 pin port names.



Source: <https://www.mbed.com/en/platform/mbed-os/> 25/04/2019

Figure 53: Mbed LPC 1768 pin port names.

mbed	LPC1768	PORT
P5	76	P0.9
P6	77	P0.8
P7	78	P0.7
P8	79	P0.6
P9	46	P0.0
P10	47	P0.1
P11	60	P0.18
P12	61	P0.17
P13	62	P0.15
P14	63	P0.16
P15	9	P0.23
P16	8	P0.24
P17	7	P0.25
P18	6	P0.26
P19	21	P1.30
P20	20	P1.31
P21	68	P2.5
P22	69	P2.4
P23	70	P2.3
P24	73	P2.2
P25	74	P2.1
P26	75	P2.0
P27	49	P0.11
P28	48	P0.10
P29	80	P0.5
P30	81	P0.4
LED1	32	P1.18
LED2	34	P1.20
LED3	35	P1.21
LED4	37	P1.23

Source: <https://www.mbed.com/en/platform/mbed-os/> 25/04/2019