FEDERAL UNIVERSITY OF SANTA CATARINA
TECHNOLOGY CENTER
AUTOMATION AND SYSTEMS DEPARTMENT
UNDERGRADUATE COURSE IN CONTROL AND AUTOMATION ENGINEERING

Daniel Juchem Regner

Object tracking using a camera gimbal mechanism

Florianópolis
2020

Daniel Juchem Regner

Object tracking using a camera gimbal mechanism

Final report of the subject DAS5511 (Course Final Project) as a Concluding Undergraduate Course of the Undergraduate Course in Control and Automation Engineering at the Federal University of Santa Catarina in Florianópolis.
Supervisor:: Prof. Tiago Loureiro Figaro da Costa Pinto, Dr. Eng.
Co-supervisor:: Prof. Henrique Simas, Dr. Eng.

Florianópolis
2020

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Regner, Daniel
   Object Tracking Using a Gimbal Mechanism / Daniel
Regner ; orientador, Tiago Loureiro Figaro da Costa Pinto,
coorientador, Henrique Simas, 2020.
   87 p.

   Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Centro Tecnológico,
Graduação em Engenharia de Controle e Automação,
Florianópolis, 2020.

   Inclui referências.

   1. Engenharia de Controle e Automação. 2. Sistema de
Controle. 3. Robot Operatyng System ROS. 4. Visão
Computacional. 5. Robótica Móvel. I. Loureiro Figaro da
Costa Pinto, Tiago. II. Simas, Henrique. III. Universidade
Federal de Santa Catarina. Graduação em Engenharia de
Controle e Automação. IV. Título.

Daniel Juchem Regner

Object tracking using a camera gimbal mechanism

This work is dedicated to whom someday believed it was possible

“Everything is possible. The impossible just takes longer.“
(BROWN D., 1998)

ABSTRACT

This work presents a system development for detecting and tracking objects on the camera's field of view coupled to a mechanism of three degrees of freedom called Gimbal. The computer vision technique, You Only Look Once (YOLO), detects an object on image in execution time and communicates between peripherals at Robotic Operating System (ROS) on a remotely piloted aircraft (RPA) processing the data and controling the Gimbal's joint using an on-board computer.

The control system is designed using Gimbal's forward and inverse kinematics mathematical concepts to estimate the angle position maintaining the object centered on image resolution. In order to compare control techniques, a Proportional-Integral linear controllers have been designed to act based on error signal from pixel position to each axis independently.

To refine the algorithm it was used a robotic simulation environment from Gazebo software to test and tune controllers and perform some experiments before starting the practical tests, reducing the probability of failure or damaging the hardware. The hardware used was a set of components provided from only one company, facilitating the connection between aircraft, camera, Gimbal and on-board computer.

The results of simulation and practical experiments validate the theory and allows the mechanism to track the object maintaining it on camera's field of view while the RPA is in motion to inspect the interest area.

Keywords: Gazebo. ROS. RPA. Gimbal. Inverse Kinematic. Computer Vision. YOLO. Track control.

RESUMO

Este trabalho apresenta um sistema de detecção e rastreamento de objetos no campo de visão da câmera acoplado a um mecanismo robótica com três graus de liberdade denominada Gimbal. O processo de detecção de objetos em tempo real usa uma ferramenta de visão computacional chamada YOLO e se comunica entre periféricos com um sistema operacional robótico (ROS) em uma aeronave pilotada remotamente (RPA) usando um computador de bordo para processar os dados.

O sistema de controle é projetado usando os conceitos matemáticos de cinemática direta e inversa do Gimbal para estimar a posição do ângulo e manter o objeto centralizado na resolução da imagem. Para comparar a matemática de controle cinemático inverso, dois controladores lineares Proporcional-Integral foram ajustados para agir com base no sinal de erro da posição do pixel para cada eixo.

Para o estudo, foi utilizado um ambiente de simulação robótica no software Gazebo para testar e ajustar os controladores realizando alguns experimentos antes de utilizá-lo na vida real, reduzindo a probabilidade de falha ou danos ao hardware. O hardware utilizado para o teste é um conjunto de componentes fornecidos por uma única empresa, facilitando a conexão entre aeronave, câmera, Gimbal e computador de bordo.

Os resultados das simulações e experimentos práticos validam a teoria e permitem que a estrutura rastreie o objeto mantendo-o no campo de visão da câmera enquanto o RPA se move para inspecionar todo o equipamento.

Palavras-chave: Gazebo. ROS. VANT. Gimbal. Cinemática Inversa. Visão Computacional. YOLO.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

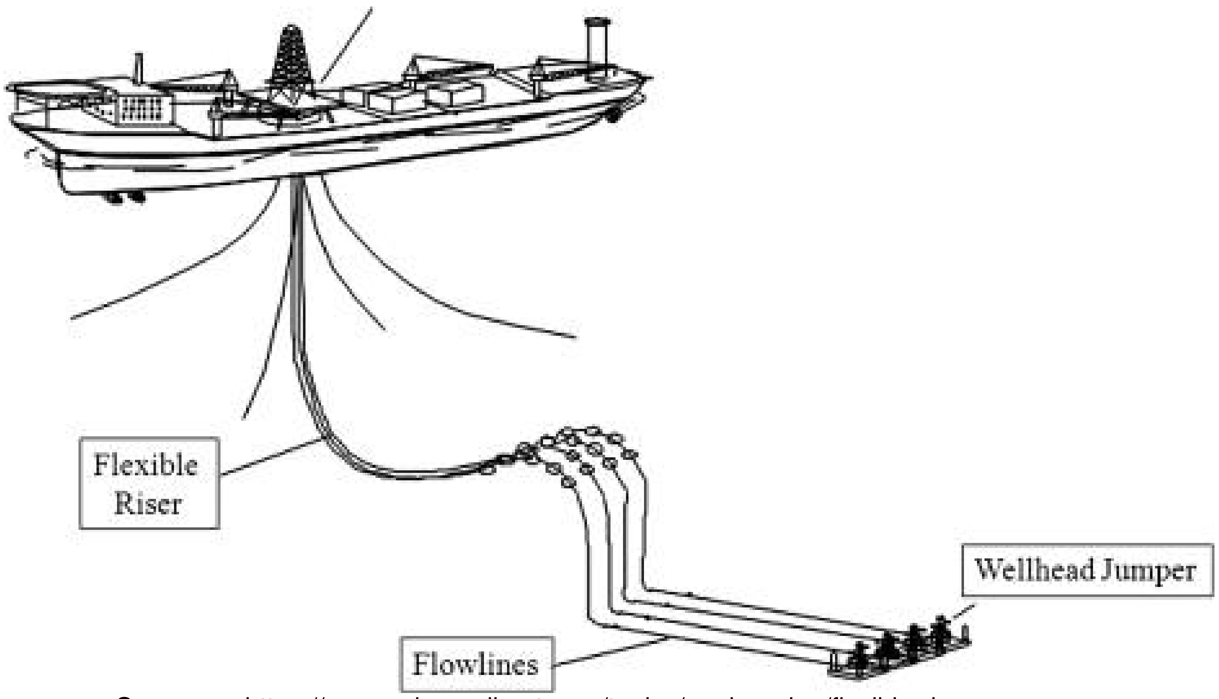
1 INTRODUCTION

Due to a recent growth of microelectronic and computational efficiencies, the use of Remote Piloted Aircraft (RPA) known as Unmanned Autonomous Vehicle (UAV) or informally called *drones*, has evolved quickly in a variety of sectors: military, security, civil engineering, archaeology, agronomy and telecommunication (SHAKHATREH et al., 2018a). Within these activities we can also highlight the use of aerial mapping, rescue operations, traffic monitoring and civil or geophysical infrastructure inspections (JORDAN et al., 2018; KRIDSADA et al., 2016).

Industrial inspections aim to reduce three important factors: human risk, time and cost of work. Previous researches that had been done in this area were mainly focused on electrical system inspection, such as transmission lines, distribution, bridges, buildings and wind turbines. One of the advantages of the use of Remote Piloted Aircraft Systems (RPAs) at these industrial inspections is that they are easier to operate in areas of difficult access. In addition, they can be equipped with different types of instruments, including different types of cameras for high resolution images, thermal images, sensors such as LiDAR (*Light Detection and Ranging*) and radiation. The obtained data can be further processed or viewed in real time remotely by the inspection technician (JORDAN et al., 2018; SHAKHATREH et al., 2018b; SHARIFI et al., 2018).

In 2014, from an international meeting of the OGP (International Association of Oil and Gas Producers), the oil and gas community announced interest in the use of RPAs in three big areas: HSE (health safety environment), security and monitoring installation integrity of structures (MERCURI et al., 2017). Whereas one of the most important structures in the deep seas oil and gas extraction are the flexible pipes, where the *flowlines* are at grade on the seafloor (BAI, Q.; BAI, Y., 2014) and the *Flexible Risers* are the section that makes the connection to the platforms as presented in Figure 1. Being the emerged part of the section the one which this work is focused on.

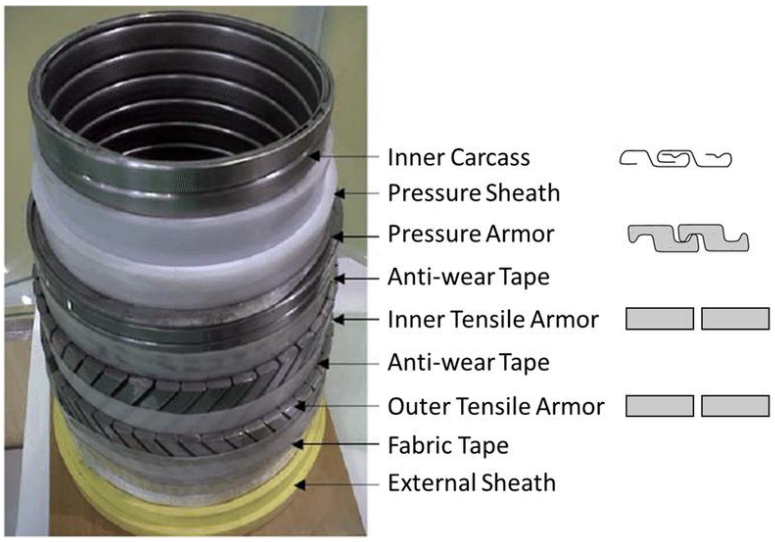
Figure 1 – Sections of a flexible Flow line.



Source – <https://www.sciencedirect.com/topics/engineering/flexible-riser>
Accessed on 10th of July.

Risers are the connection pipes used to transport oil, gas and system cables from the undersea structures to production units above the surface. These pipes are composed of several metallic and plastic layers to protect the whole structure, Figure 2, demanding a periodic maintenance resulted from natural weathering.

Figure 2 – Typical structure of flexible *riser*.



Source – Mosqueira (2017)

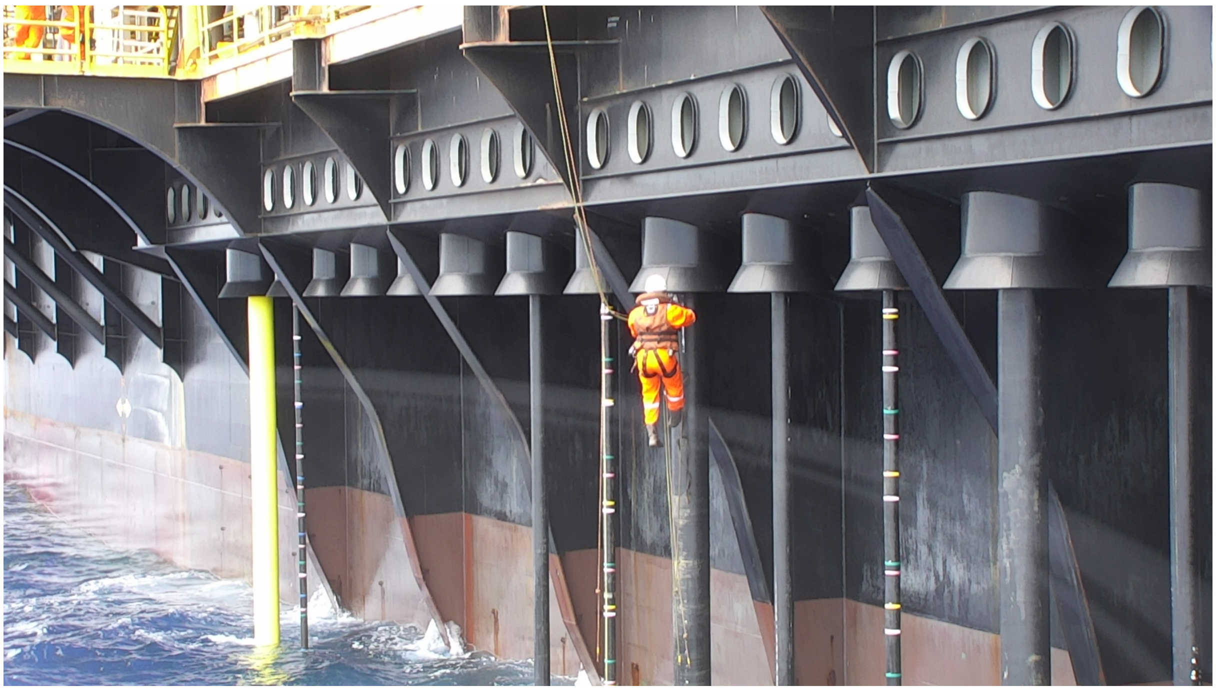
1.1 MOTIVATION

The structure is located in a corrosive environment, with extreme weather conditions such as storms, swirls and even cyclones (WANG et al., 2016), increasing the risk of structural and mechanical failure. Besides the environmental elements, there are other factors that also contribute to riser's structures deterioration, such as the friction from anchoring clandestine fishing ships that can potentially rupture the armor layer of the riser and beyond. (MARINHO et al., 2006).

A riser failure can generate several economic and environmental consequences, demanding a periodic inspection to conserve it's integrity and all the embedded instrumentation systems (WANG et al., 2016).

Riser inspections are traditionally made by industrial climbing. The technician charged to make the inspection needs to climb to the pipe connections and manually take measurements of the diameter and also register images, as shown in Figures 3. This operation requires a huge mobilization of resources, equipment and auxiliary structures, considering that an offshore oil rig typically has 50 risers and as it is a slow process, it takes a day for only one riser to be inspected. The cost implications related to the occupancy of these workers on the vessel, which are limited, results in a high operational cost to the company. Beyond the mobilization, time and value, this is also a high risk operation due to the environment and height of where the work is performed, which can sometimes be more than 20 meters of emerged risers.

Figure 3 – Technical inspection using industrial climb.



Source – Project VANT3D

Based on those inspection problems, an idea was to use RPAs for inspections, allowing the process to work faster and more economically, resulting in a potential option to check the integrity of external structures and instruments, while reducing the security risk, time of inspection and costs. Recently, RPAs have been used to perform the visual inspection of risers, reducing drastically the necessity of a technician to climb every riser, unless the images acquisition detects a fault. RPAs enable the capture of panoramic images with a better framework of risers and instruments, in comparison to technical climbers that make risky maneuvers to get a good angle of image acquisition. To check details from a specific part of the riser the worker has to move himself, which takes time. With RPA it is possible to get high resolution images from a specific part using an objective lens with adequate zoom.

Even with RPAs, climbers are still needed to take manual measurements for risers where a fault has been detected from the image acquisition. Those manual measurements, in most cases, are made in a precarious way, resulting from the difficulties related to the riser length and climber's position. With the goal of reducing the necessity of technical climbers, the project VANT3D aims to develop additional procedures and sensors for the 3D geometric measurement of risers with a minimal error of dimensions, capable of notice minor defects at structures.

To generate a 3D geometric measurement of risers, a minimum quantity of images from different points of view and a certain amount of images overlap are needed, resulting in a big amount of homologous points registered in different images acquired at different angles. To record these images, the RPA needs to cover an extensive riser connection area alongside the vessel, mostly in complicated positions, making it difficult for the pilot to maintain visual contact with the aircraft in an environment with great magnetic interference. As shown in Figure 3, an image taken from one test overseas using a Quadrotor aircraft while the worker was doing an inspection at risers close to a point of view on a balcony on upper left corner of the figure.

Based on this problem, an autonomous control system to detect and track an object can be developed to control the three degree mechanism system, called Gimbal, which the camera is connected to. It will maintain the object tracked and centered on the camera's field of view which will help registering a sufficient number of photos and using it to a three-dimensional reconstruction, helping the pilot to maintain the aircraft under his sight.

1.2 OBJECTIVE

Using RPAs to inspect the emerged part of risers, brings the challenge of using it on a dangerous environment with some strong magnetic field. It can restrict the pilot from taking his sight off the aircraft while looking to a display where the camera acquisition image is shown. The objective of this work is to create an algorithm that process a camera's image in order to detect a specify object and control the gimbal attached to camera to maintain object on camera's field of view and make easier the acquisition image of object to posterior photogrammetry reconstruction.

Inside the major objective of create a gimbal control system able to track an object, it has specific objectives to be achieved. Those goals determine the workflow of the project and present steps of controller construction.

- Detect risers with a trained network for YOLO;
- Develop a simulation environment to generate and analyze the algorithm;
- Communicate aircraft and on-board computer via ROS;
- Process YOLO detection at a main camera of RPAS using a on-board computer;
- Control gimbal position based on object pixel position;
- Compare different track controllers;
- Acquire risers images for a posterior photogrammetry reconstruction.

1.3 WORKFLOW

This report is structured in six chapters. Chapter 1 gives an introduction, motivation and objective to create this work, following Chapter 2 bringing some references to understand the functionality. Chapter 3 show the control track theory to be implemented. Chapter 4 present methods and components used to achieve the goal. Chapter 5 details the tests made on simulation and practical environments, bringing some results. Lastly, Chapter 6 presents the conclusions and aim for future projects.

2 STATE OF ART

This chapter brings some background references to understand the process and systems used, including functionalities, how they work and communicates between robotic peripherals, mathematical description of robotic systems and some concepts of computer vision technology.

2.1 ROBOT OPERATING SYSTEM

The Robot Operating System (ROS) is an open source framework with a diversity of libraries and tools collections to standardize a software communication between robot components as sensors, cameras, actuators on a distributed computing resources. The concept aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms (QUIGLEY et al., 2015; OPEN SOURCE ROBOTICS FOUNDATION, 2010).

Dealing with real-world variations in complex tasks and environments is so difficult that no single individual or institution can hope to build a complete system from scratch. As a result, ROS was created from the ground up to encourage collaborative robotics software development. For example, in the “fetch a item” problem, one organisation might have experts in mapping indoor environments and could contribute on a system for producing indoor maps. Another group might have expertise in using maps to navigate indoor environments. Another group might have discovered a particular computer vision approach to recognising small objects. ROS includes many features specifically designed to simplify this type of large-scale collaboration.

ROS is called as an open-source, Meta-Operating System for robots. Although this is not a dictionary word, it represents a system that has the provides the services expected from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes and package management. It describes a system that performs processes by utilizing virtualization layers between applications and distributed computing resources with different operational systems (YOONSEOK PYO HANCHEOL CHO, 2017; JOSEPH, 2018).

2.1.1 Communication

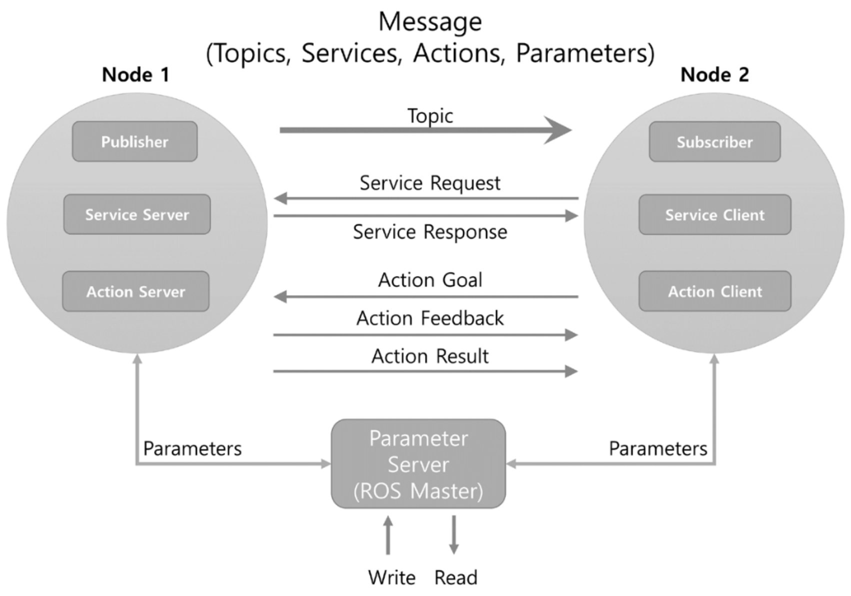
The concept of communication on ROS is based on nodes process where performs computation necessity to exchange information. Nodes are meant to operate as small process. Normally a robot combines a large quantities of nodes combined together into a graph and communicate with one another using streaming topics, services, and Parameter Servers. a robot control system will usually comprise of many nodes. For example, one node performs the acquisition of images from a camera, another node provides a mapping of homologous points from a stereo camera system, another node compute all those points to reconstruct the environment to localize the robot, and so on (OPEN SOURCE ROBOTICS FOUNDATION, 2010; QUIGLEY et al., 2015).

There are three method used among nodes to communicate. A topic provides a unidirectional message transmission/reception; a server which provides a bidirectional message request/response and an action which provides a bidirectional message with an intermediate answer goal/result/feedback. The message communication is illustrated in Figure 4 and summarised in Table 1.

Table 1 – Comparison of communication types.

Type	Features		Description
Topic	Asynchronous	Unidirectional	Continuously exchange of data
Service	Synchronous	Bidirectional	Request processing request and respond in current state
Action	Asynchronous	Bidirectional	To return a intermediate feedback value or when has long response time after request

Figure 4 – Communication between nodes.



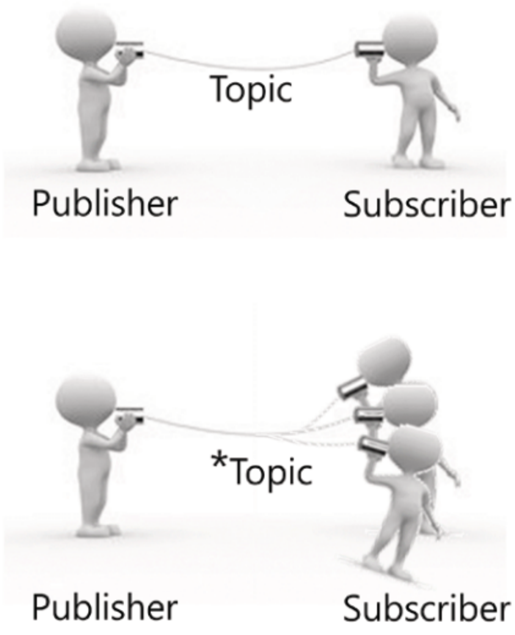
Source – Yoonseok Pyo Hancheol Cho (2017)

2.1.1.1 Topics

Topics are named buses over which nodes to exchange messages. As nodes are characterize a small process of robot, they do not need to know of who they are communicating with, just receive or send data periodically. Publishers are the nodes where constant get information as sensors or nodes that need to send messages to act in another node. Subscriber are the nodes processing some act and do need the information from sensors to perform. They work as anonymous publish/subscribe semantics, which decouples the production of information from its consumption.

The structure publisher/subscriber is the most common method to exchange data in a distributed system and can be used in multiples nodes at same time as shown in Figure 5. Before nodes start to transmit data over topics, they must advertise the topic name and type of message that are going to be sent. Topics are unidirectional and remain connected to continuously send or receive messages, it is suitable for sensor data that requires messages to be published periodically. Topics works with multiple subscribers receiving messages from a publisher and vice versa (JOSEPH, 2018; YOONSEOK PYO HANCHEOL CHO, 2017).

Figure 5 – Topic message communication. *Topic not only allow 1:1 Publisher / Subscriber communication, but also supports 1:N, N:1 and N:N.



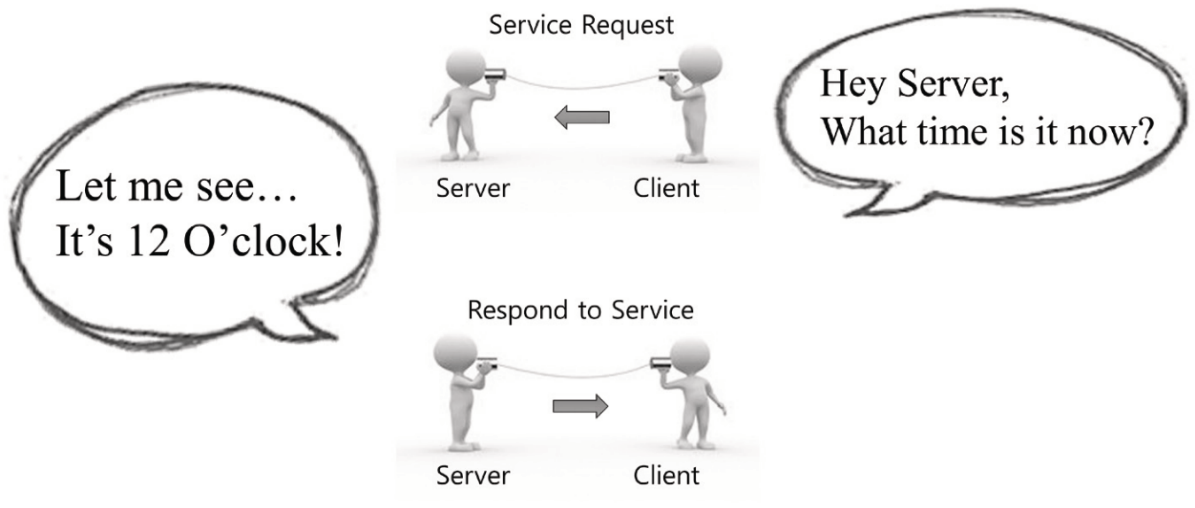
Source – Yoonseok Pyo Hancheol Cho (2017).

2.1.1.2 Services

Services are another way to pass data between nodes in ROS. Services are just synchronous Remote Procedure Calls (RPC), They allow one node to call a function that executes in another node as a request and reply analogy. A provide ROS node offers a service under a string name, and a client calls the service by sending the request message and awaiting the reply.

A service consists of a service server that responds only when there is a request and a service client that can send requests as well as receiving responses. Unlike the topic, services are one time message communications, when the request and response of the service are completed, the connection between nodes will be disconnected. Where the client requests the server for the current time, the server will check the time and respond, after responding the connection will be closed (QUIGLEY et al., 2015).

Figure 6 – Server message communication.



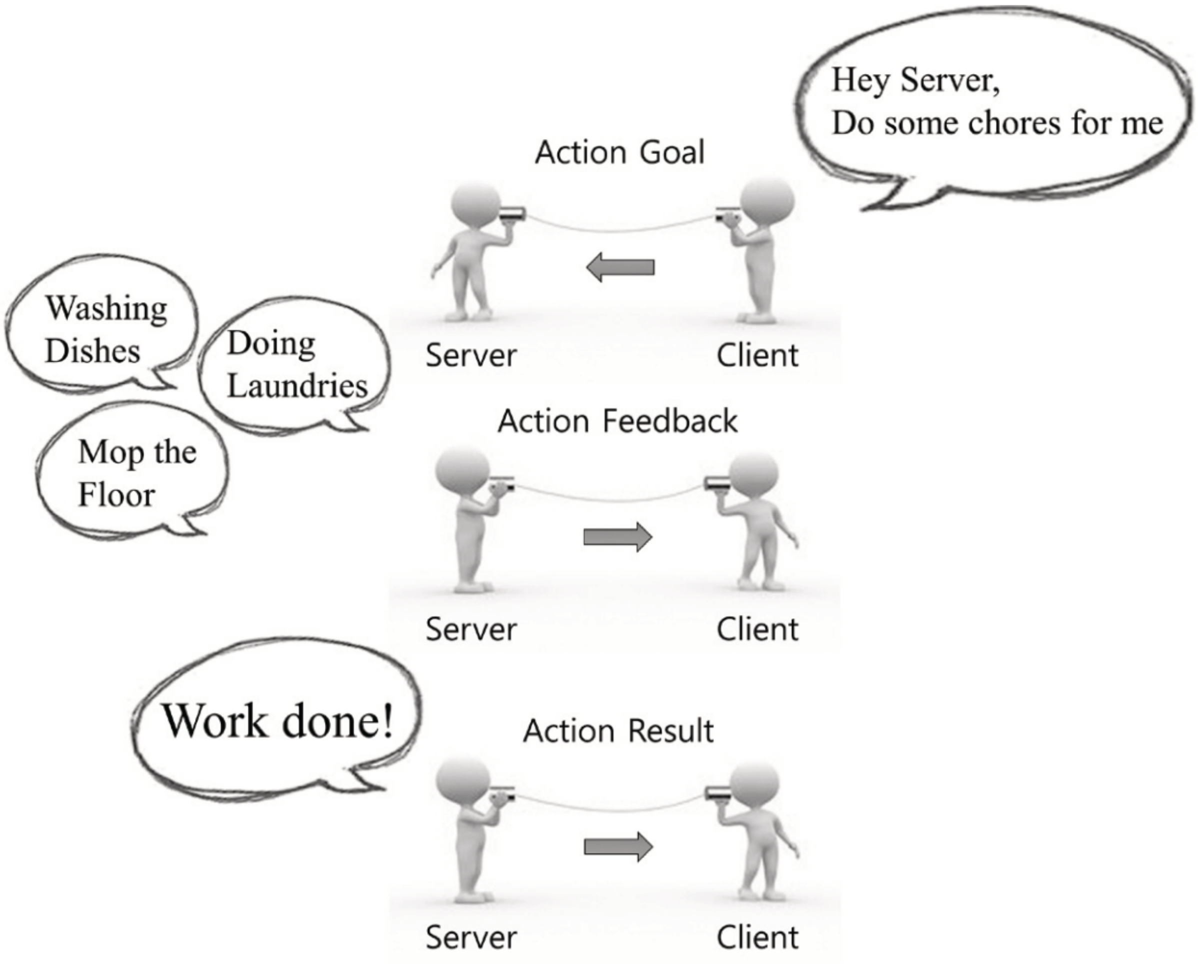
Source – Yoonseok Pyo Hancheol Cho (2017)

2.1.1.3 Actions

ROS actions are the best way to implement interfaces to time-extended, goal oriented behaviors. While services are synchronous, actions are asynchronous. Similar to the request and reply of a service, an action uses a goal to initiate a behavior and sends a result when the behavior is complete. Different than services, action uses feedback to provide updates on the behavior's progress toward the goal and also allows for goals to be canceled. Actions are themselves implemented using topics. An action is essentially a higher-level protocol that specifies how a set of topics (goal, result, feedback, etc.) should be used in combination (JOSEPH, 2015).

In example, Figure 7, if a client sets a home-cleaning task as a goal to the server, the server informs the progress along the task sending feedback messages of its progress until finally sending a message advertising the end of the work. Unlike the service, the action is often used to command complex robot tasks such as canceling transmitted goals while the operation is in progress.

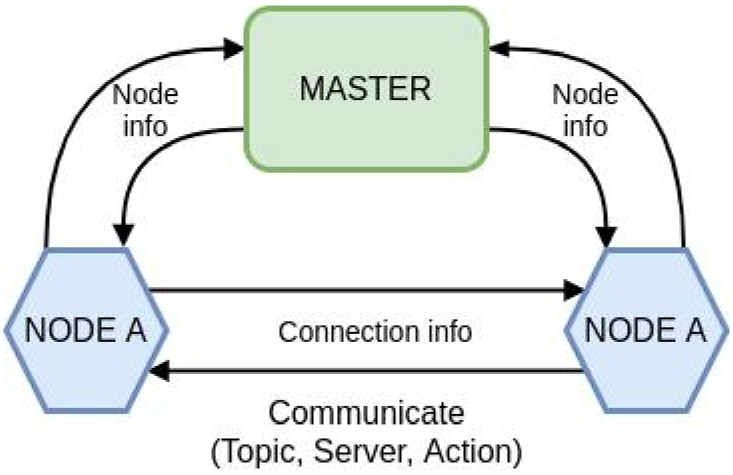
Figure 7 – Action message communication.



Source – Yoonseok Pyo Hancheol Cho (2017)

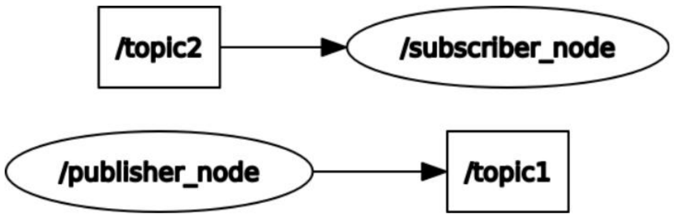
All methods of exchange messages describe previously, publisher; subscriber; client and server from service and client and service from actions can be implemented in separate nodes. In order to exchange messages among these nodes, the connection has to be established first with the help of a master. A master acts like a name server as it keeps names of nodes, topics and action as well as the Uniform Resource Identifier (URI) address, port number and parameters. In other words, nodes register their own information with the master upon launch, and acquire relative information of other nodes from the master. Then each node directly connects to each other to perform message communication.

Figure 8 – Message Communication



ROS have a package to visualize in graph form the communication between nodes using topics. A node is represented using a ellipse and a topic is represented by a rectangle, using arrows to identify who is publishing the topic and who is subscribing to acquire some data. Figure 9 shown an example.

Figure 9 – ROS graph example.



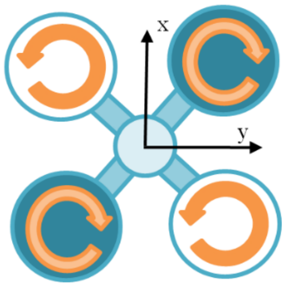
2.2 QUADROTOR TECHNOLOGY

Quadrotor helicopters are an emerging rotor craft concept for unmanned aerial vehicle platforms. It consists of a vehicle of four rotors, with two pairs of counter rotating fixed pitch rotors. Due to some unique abilities such as high manoeuvrability, small size, and easy control, they have been widely used for industrial inspections, military surveillance and homeland security (HOFFMANN et al., 2007; POUNDS et al., 2006).

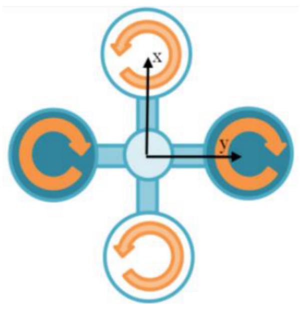
Quadrotors are agile vehicles controlled by the rotational speed of the four rotors. The position of rotor arrangements relative to the body coordinate system has two different types of configurations: the "x" configuration and the " + " configuration shown in Figure 10 (NOROUZI GHAZBI et al., 2016; NOORDIN et al., 2017).

Figure 10 – Quadrotors type of configurations

(a) Cross Configuration



(b) Plus configuration

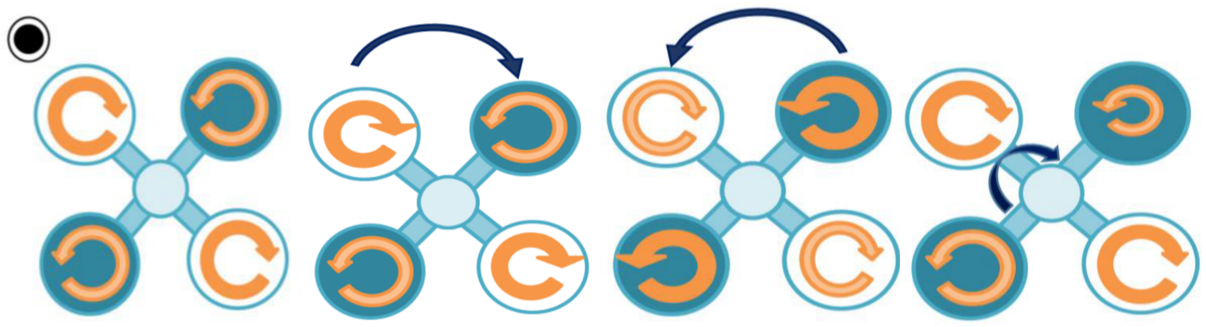


Source – Norouzi Ghazbi et al. (2016).

The aircraft has six degrees of freedom (*DoF*), three translational and three rotational movements. The fly mechanism of the vehicle is simple. Adjusting the angular velocity of each rotor in relation to the other three is possible by rotating on three axis. However, even if the aircraft has six *DoFs*, it is not possible to use them independently, changing orientations in relation to the X and Y axes result in a translation under other axes. As a example, changing the angle around X axis will result in a translation along Y axis (ZHANG, Xiaodong et al., 2014).

Using the X configure, Figure 10a, if half the rotors rotate in a different direction to the other half with the same speed, there is no variation in the yaw angle as shown in Figure 11a. Difference in speed between the two pair motors creates a move variation on yaw angle, Figure 11b and 11c. A different speed in opposite motors creates a net roll or pitch, Figure 11d. To rotate along roll movement requires changing angular velocity from parallel pairs of rotors along the *X* axis, resulting in an offset on the *Y* axis. For clockwise movement, the velocity is increased in the front rotors while decreasing in the rear rotors. The same is done to change the pitch angle but changing pairs of rotors along the Y axis, and generating a movement along *X* axis. with left and right motions (TERWILLIGER et al., 2017; ZHANG, Xiaodong et al., 2014).

Figure 11 – Quadrotors movement.



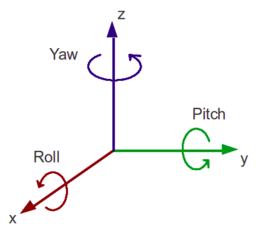
Source – Norouzi Ghazbi et al. (2016).

2.2.1 Mathematical Model

An important aspect of mathematical modelling of quadrotors aircraft is the coordinate system in use. As presented before, the structure has two common configurations, + or *x* configurations. This present work will be defining the mathematical model for a *x* configuration, similar to the hardware available.

It is convenient to define a body-fixed coordinate system $[X\ Y\ Z]^T$ in the aircraft. In aeronautic applications, quantities such as acceleration, velocities and angular rates are often, or at least partially, measured in relation to the aircraft (REIZENSTEIN, 2017). The body fixed coordinate system can be used for relations. This is commonly a North-East-Down coordinate system, with the X axis pointing fore, the Y axis pointing starboard and Z axis pointing to the keel of the craft (NELSON, 1998). To relate the orientation of the local coordinate system to a global one, the quadrotor aircraft orientation can be defined by three Euler angles along its coordinate system. Which are roll (Φ), pitch (Θ), yaw (Ψ), evaluated via sequent rotation around each one of the inertial axes $[X\ Y\ Z]^T$ respectively, as shown in Figure 12.

Figure 12 – Euler angles.



Source – Prabha and Thottungal (2016).

As described in (NELSON, 1998), three rotations needed to transform the global coordinate system $[X_E\ Y_E\ Z_E]^T$ to the local system $[X_Q\ Y_Q\ Z_Q]^T$. Each rotation results in a new coordinate system, to determine the orientation of the body frame *Q* and global frame *E* must rotate the consecutively coordinates system yaw Ψ, pitch Θ and roll Φ. This rotations matrix multiplied results in matrix *R*, equation (1).

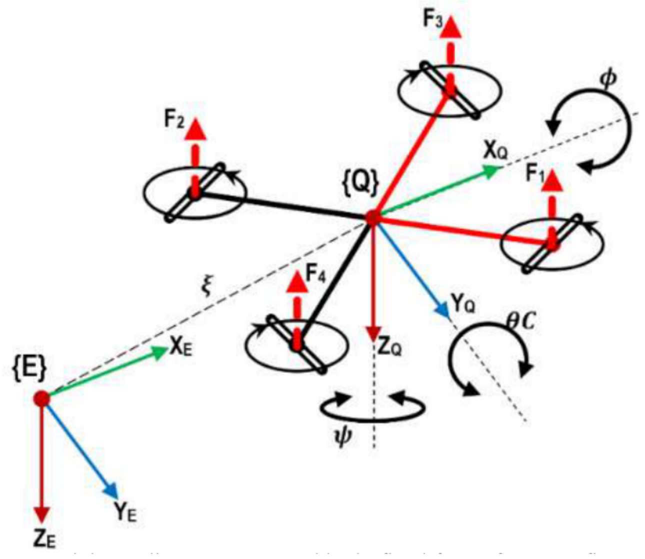
$$R = \begin{bmatrix} C_\Psi\, C_\Theta & C_\Psi\, S_\Theta\, S_\Phi - S_\Psi\, C_\Phi & C_\Psi\, C_\Phi\, S_\Theta + S_\Psi\, S_\Phi \\ S_\Psi\, C_\Theta & S_\Psi\, S_\Theta\, S_\Phi + C_\Psi\, C_\Phi & S_\Psi\, C_\Phi\, S_\Theta - C_\Psi\, S_\Phi \\ -S_\Theta & C_\Theta\, S_\Phi & C_\Theta\, C_\Phi \end{bmatrix} \tag{1}$$

where:

$$\begin{aligned} S_\Phi &= sin(\Phi), & C_\Phi &= cos(\Phi) \\ S_\Theta &= sin(\Theta), & C_\Theta &= cos(\Theta) \\ S_\Psi &= sin(\Psi), & C_\Psi &= cos(\Psi) \end{aligned}$$

After multiplying the rotation matrix R on coordinate frames E, the result is represented by Figure 13, assuming the aircraft has a rigid body and is symmetric with respect to the XY-axis.

Figure 13 – Global coordinate system *E* and Quadrotor aircfart coordinate system *Q* from a X configuration body.



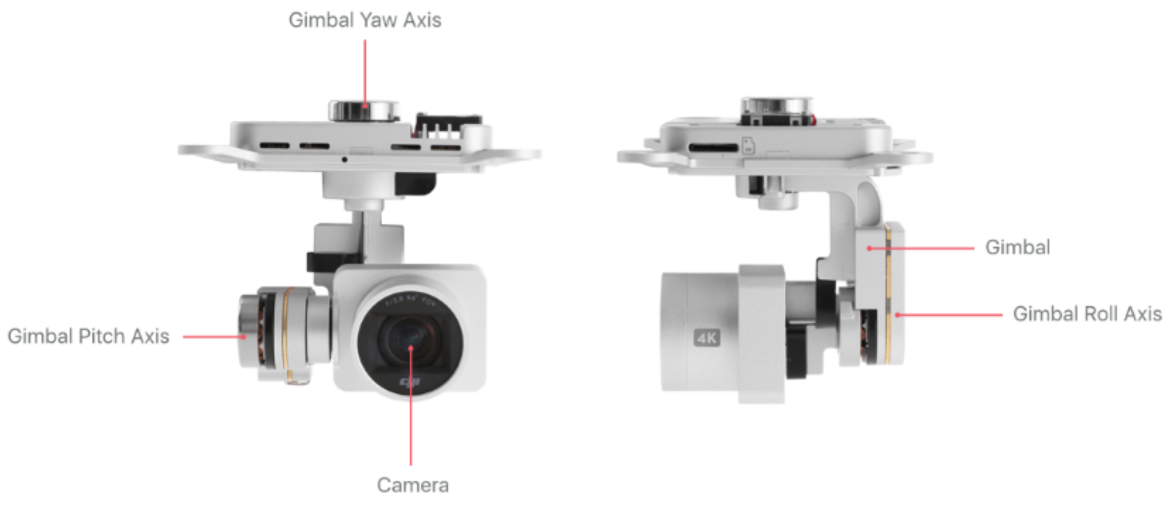
Source – Noordin et al. (2017)

Understanding the operation of a quadrotor aircraft and mathematical behavior to describe its positioning, one can study the payloads coupled to the aircraft and its mathematical model based on the coordinate system described in Figure 13.

2.3 GIMBAL

The gimbal mechanism used for the control the camera position is a mechanical device which is designed using the rings mounted on axes at right angles to each other. The objects presented in unstable environments are arranged in a stable position using this mechanical device and rejects disturbances such as motor friction, unbalanced aerodynamics, spring torque forces and structure vibration of any type (JAKOBSEN; JOHNSON, 2005). A traditional use for a Gimbal mechanism is to stabilize camera images in RPA as shown in Figure 14.

Figure 14 – Example of a Gimbal mechanism.



Source – `https://developer.dji.com/mobile-sdk/documentation/introduction/component-guide-gimbal.html`
Accessed: July 23th.

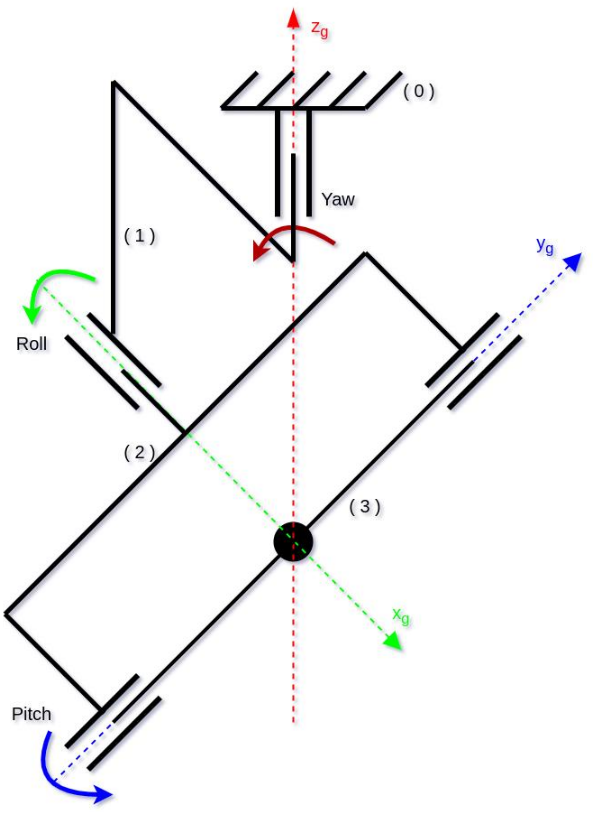
If the camera positioning is not compensated or stabilized, it produces shakes in the video capture, blurred images, and failure in object tracking and so on during aerial photography or autonomous target tracking etc (RAJESH; KAVITHA, 2016). The gimbal described is a 3-axis gimbal which is mounted above or under the body of the aircraft and has a individual controller to each gimbal actuators to movement of the gimbal axis angles yaw, roll and pitch.

2.3.1 Mathematical Modelling

The 3-axis gimbal consists of three revolute joints and it has yaw-roll-pitch axis representation. Here Ψ_g, Θ_g and Φ_g represent yaw-roll-pitch angles. The schematic diagram of gimbal kinematics with 3 revolute joints is shown in Figure 15 (RAJESH; KAVITHA, 2016; KULKARNI; MOHANTY, 2013).

The three axis mechanism system intersected on the center of camera field of view compensates for all the angular movements of the hull it is attached to and the general principle of the axes arrangement assures it is able to avoid the gimbal lock state during its normal operation. The configuration, yaw (Φ_g) over pitch (Θ_g) over roll (Ψ_g), allows up to 90 degrees of roll or pitch movement by the aircraft before a gimbal lock occurs, which of course is highly unlikely to ever happen (TIIMUS; TAMRE, 2010).

Figure 15 – Gimbal Kinematics with 3 revolute joints.



Source – Rajesh and Kavitha (2016)

To find the Gimbal forward kinematics is used with a three-dimension rotation matrix in each axis. The transformation from body (0) to body (3) is presented below, where body 0 is the RPA.

The rotation matrix of yaw axis from the frame of body (0) to the frame of body (1) is:

$$R_1^0 = \begin{bmatrix} cos(\Phi_g) & -sin(\Phi_g) & 0 \\ sin(\Phi_g) & cos(\Phi_g) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The rotation matrix of roll axis from the frame of body (1) and the frame of body (2) is

$$R_2^1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\Psi_g) & -sin(\Psi_g) \\ 0 & sin(\Psi_g) & cos(\Psi_g) \end{bmatrix} \quad (3)$$

The rotation matrix of pitch axis from the frame of body (2) and the frame of body (3) is

$$R_3^2 = \begin{bmatrix} cos(\Theta_g) & 0 & sin(\Theta_g) \\ 0 & 1 & 0 \\ -sin(\Theta_g) & 0 & cos(\Theta_g) \end{bmatrix} \quad (4)$$

The total rotation matrix between the base frame (0) and frame of body (3) is

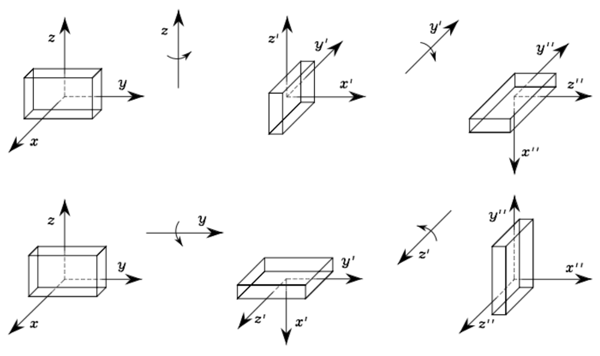
$$R_3^0 = R_1^0 \, R_2^1 \, R_3^2 \quad (5)$$

$$R_3^0 = \begin{bmatrix} C_{\Phi_g} C_{\Theta_g} - S_{\Phi_g} S_{\Psi_g} S_{\Theta_g} & -C_{\Psi_g} S_{\Phi_g} & C_{\Phi_g} S_{\Theta_g} + C_{\Theta_g} S_{\Phi_g} S_{\Psi_g} \\ C_{\Theta_g} S_{\Phi_g} + C_{\Phi_g} S_{\Psi_g} S_{\Theta_g} & C_{\Phi_g} C_{\Psi_g} & S_{\Phi_g} S_{\Theta_g} - C_{\Phi_g} C_{\Theta_g} S_{\Psi_g} \\ -C_{\Psi_g} S_{\Theta_g} & S_{\Psi_g} & C_{\Psi_g} C_{\Theta_g} \end{bmatrix} \quad (6)$$

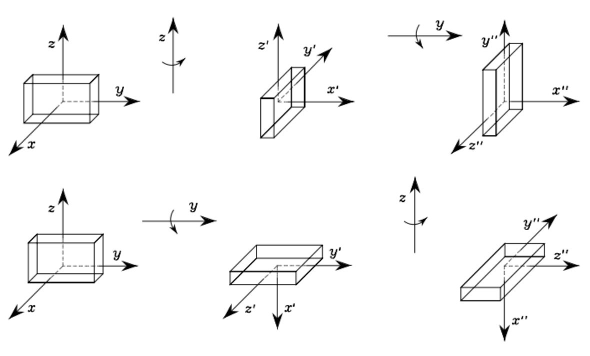
Since this matrix (6) refers to a successive rotations referred to the current coordinate frame for each rotation. it is possible specify by constantly referring them to the initial frame and represent on a rotation matrix the position of body after rotates over a initial fixed frame. As described in (SICILIANO et al., 2009) a multiplication of rotations matrix described in a fixed frame is represented as the final frame rotation multiplied until represents the initial frame as equation (8) present.

It can be stated that composition of successive rotations with respect to a fixed frame is obtained by pre multiplication of the single rotation matrices in the order of the given sequence of rotations. An important issue of composition of rotations is that the matrix product is not commutative. In view of this, it can be concluded that two rotations in general do not commute and its composition depends on the order of the single rotations. Figure 16 presents a example of rotation along a current frame and fixed frame.

Figure 16 – Example of rotation on current and fixed frame.



(a) Rotation around current frame.



(b) Rotation aro fixed frame.

$$\overline{R_3^0} = R_3^2\, R_2^1\, R_1^0 \tag{7}$$

$$\overline{R_3^0} = \begin{bmatrix} C_{\Phi_g}C_{\Theta_g} + S_{\Phi_g}S_{\Psi_g}S_{\Theta_g} & C_{\Theta_g}S_{\Phi_g}S_{\Psi_g} - C_{\Phi_g}S_{\Theta_g} & C_{\Psi_g}S_{\Theta_g} \\ C_{\Psi_g}S_{\Phi_g} & C_{\Phi_g}C_{\Psi_g} & -S_{\Psi_g} \\ C_{\Phi_g}S_{\Psi_g}S_{\Theta_g} - C_{\Psi_g}S_{\Theta_g} & C_{\Theta_g}C_{\Phi_g}S_{\Psi_g} + S_{\Phi_g}S_{\Theta_g} & C_{\Psi_g}C_{\Theta_g} \end{bmatrix} \tag{8}$$

Through equation (8) of forward kinematic from Gimbal mechanism it is possible to set the angles based where the camera's field of view is orientated based to a fixed coordinate system. Using this result to posterior develop a control algorithm to track the detected specify objects based on where the camera is orientated.

2.4 COMPUTER VISION

Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and in general, dealing with the extraction of high-dimensional data from the real world in order to produce numerical or symbolic information in the forms of decisions (BALLARD; BROWN, 1982; KLETTE, 2014; HUANG, 1997).

> Why is vision so difficult? In part, it is because vision is an inverse problem, in which we seek to recover some unknowns given insufficient information to fully specify the solution. We must therefore resort to physics-based and probabilistic models to disambiguate between potential solutions. However, modeling the visual world in all of its rich complexity is far more difficult than, say, modeling the vocal tract that produces spoken sounds (HUTTENLOCHER, 2004, p. 3).

The actions or decisions that computer vision attempts to make based on camera data are performed in the context of a specific purpose or task. We may want to remove noise or damage from an image so that our security system will issue an alert if someone tries to climb a fence or because we need a monitoring system that counts how many people cross through an area in an amusement park. Vision software for robots that wander through office buildings will employ different strategies than vision software for stationary security cameras because the two systems have significantly different contexts and objectives. As a general rule: the more constrained a computer vision context is, the more we can rely on those constraints to simplify the problem and the more reliable our final solution will be.

2.4.1 OpenCV

OpenCV is a open-source image process library created in a initiative of Intel research to advance CPU-intensive applications. It is aimed at providing the basic tools needed to solve computer vision problems (INTEL®, 2000). In some cases, high-level functionalities in the library will be sufficient to solve the more complex problems in computer vision. Even when this is not the case, basic components in the library are complete enough to enable creation of a complete solution of your own to almost any computer vision problem. In the early days of OpenCV, the goals of the project were described (BRADSKI; KAEHLER, 2009) as:

- Advance vision research by not only open but also optimized code for basic vision infrastructure.
- Disseminate vision knowledge by providing a common infrastructure that developers could build on.

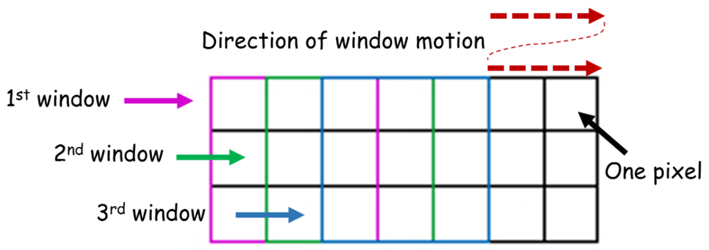
- Advance vision-based commercial applications by making portable, performance-optimized code available for free.

2.4.2 Object Detection

Object detection is one of the most complex branches of computer vision in terms of executing two important tasks at the same time and in time for execution. The first task is to determine where the objects are located in a given image (object localization) and which category each object belongs to (object classification) (ZHAO et al., 2019). So the workflow of traditional object detection models can be mainly divided into three stages: informative region selection, feature extraction and classification. All deep-learning methods have their own unique approach of defining each of the three parts to determine their computational speed and accuracy (LECUN et al., 2015; KEMAJOU et al., 2019).

- **Informative Region selection:** as different objects may appear in any position in the image and have different aspect ratios or size, it is necessary to scan the whole image with a multi-scale sliding window. Although this strategy can find all the possible positions of the object, it demands a strong computation effort and produces many redundant windows. If only a fixed number of sliding window templates are applied, it can produce a unsatisfactory result. Figure 17 present the method.

Figure 17 – Sliding window method.



Source – Xin Zhang et al. (2020)

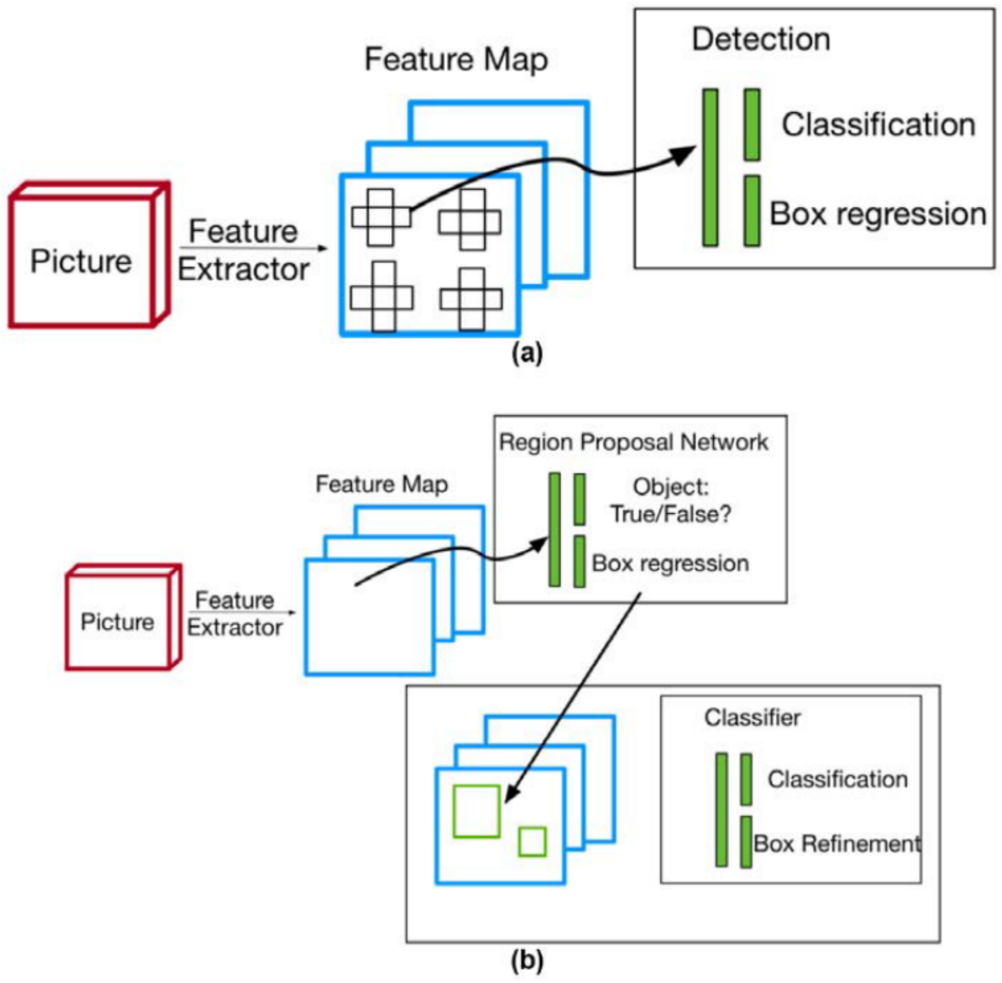
- **Feature extraction:** to recognize different objects, it is necessary extract visual features which can provide a semantic and robust representation. SIFT (MOREL; YU, 2008) HOG (LINDGREEN; LINDGREEN, A., 2004) and Haar-like (LIENHART; MAYDT, 2002) are representative ones. This is due to the fact that these features can produce representations associated with complex cells in the human brain (LOWE, 2004). However, due to the diversity of appearances, illumination conditions and backgrounds, it is difficult to manually design a robust feature descriptor.

- **Classification:** a classification is needed to distinguish a target object from all the other categories and to make the representations more hierarchical, semantic and informative for visual recognition (LECUN et al., 2015).

The different object detection models can be classified into two categories: one-stage and two-stage detectors (KEMAJOU et al., 2019). The two-stage detector model has an external proposal generator that generates anchor boxes that might contain objects from feature maps using, for example, a sliding window technique. It relies on the external proposal generator. A one-stage detector is a single-shot detector which applies small convolutional layers on top of texture extractors using neural network.

Figure 18 shows a schematic description of the two types of detectors. Although the one-stage detector saves computational time, it suffers accuracy loss, particularly for small objects. The one-stage detector evaluates thousands of candidates on every image, but only a few actually contain objects, which causes the model to spend most of its time identifying background objects, impeding its accuracy and efficiency

Figure 18 – Schematic plot for (a) one-stage detector and (b) two-stage detector.



Source – Kemajou et al. (2019)

Different methods have been developed with the objective of maximizing computational optimization effort, some methods are presents in the following sections.

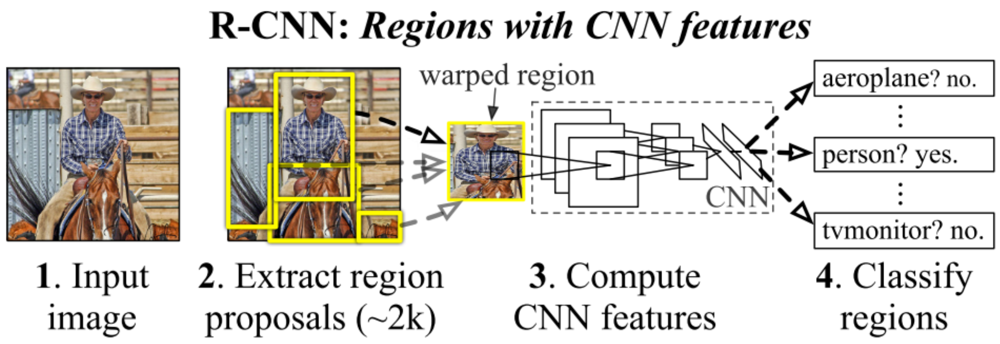
2.4.2.1 R-CNN

Regions with convolutional neural networks (R-CNN) combine restricted proposal region values with CNNs presented by (GIRSHICK et al., 2014). This method generates around 2000 category independent region proposals for the input image, extracts a fixed-length feature vector from each proposal using a CNN, and then classifies each region with category-specific linear support vector machines (SVM).

R-CNN object detection systems consist of three modules. The first generates category-independent region proposals. These proposals define the set of candidate detection available to our detector. The second module is a large convolutional neural network that extracts a fixed-length feature vector from each region. The third module is a set of class- specific linear SVMs.

Figure 19 presents an overview of the described method in four stages. Stage 1 input of image, following the extract of around 2000 bottom-up region proposals in stage 2 and computes features for each proposal using a large convolutional neural network (CNN) at stage 3. At the last stage it classifies each region using class-specific linear SVMs (GIRSHICK et al., 2014).

Figure 19 – Object detection system overview.



Source – Girshick et al. (2014)

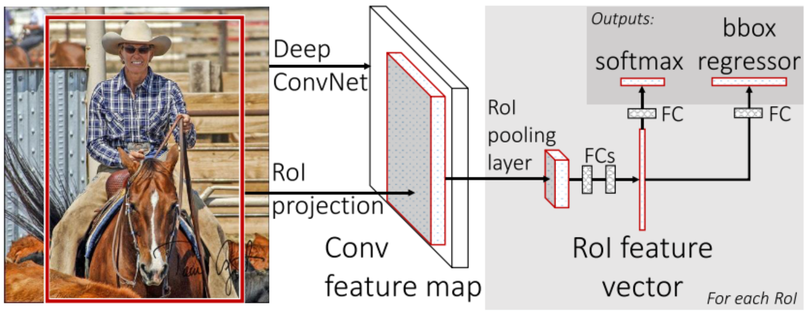
Even this method results in a good accuracy and computational optimization to detect a object in a image. R-CNN still takes a huge amount of time to train the network as you would have to classify approximately 2000 region proposals per image, not being able to be implemented in a real time process due to a long time to process each frame.

2.4.2.2 Fast R-CNN

To solve the problems of R-CNN method, (GIRSHICK, 2015) presents a fast R-CNN method. The approach is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, it feeds the input image to the CNN to generate a convolutional feature map. From the convolutional feature map, it identifies the region of proposals and warps them into squares and by using a Region of Interest (ROI) pooling layer it reshapes them into a fixed size so that it can be fed into a fully connected layer. From the ROI feature vector, it uses a softmax layer to predict the class of the proposed region and also the offset values for the bounding box.

The reason “Fast R-CNN” is faster than R-CNN is because you don’t have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.

Figure 20 – Fast R-CNN architecture.



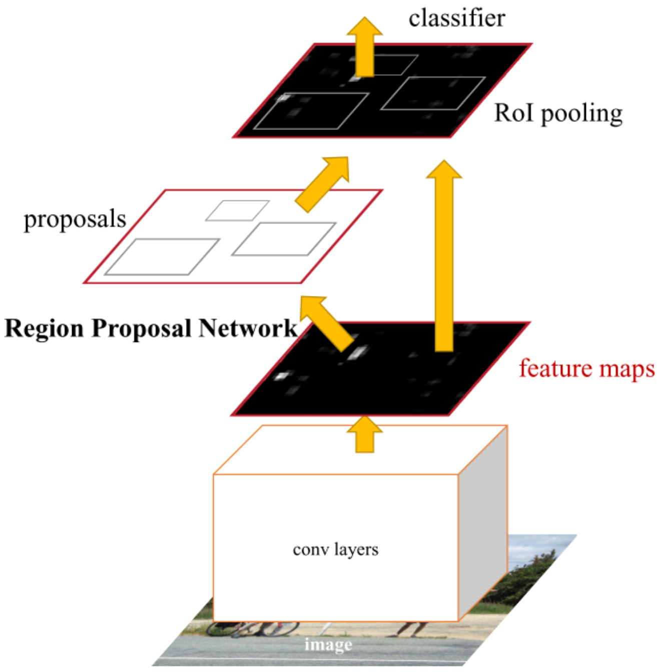
Source – Girshick (2015)

An input image and multiple ROI are input into a fully convolutional network. Each ROI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers. The network has two output vectors per ROI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss (GIRSHICK, 2015).

2.4.2.3 Faster R-CNN

Both of the above algorithms(R-CNN & Fast R-CNN) uses selective search to find out the region proposals. Selective search is a slow and time-consuming process affecting the performance of the network. Therefore (REN et al., 2017) came up with an object detection algorithm that eliminates the selective search algorithm and lets the network learn the region proposals.

Figure 21 – Faster R-CNN is a single, unified network for object detection. The RPN module serves as the ‘attention’ of this unified network.



Source – Ren et al. (2017)

,

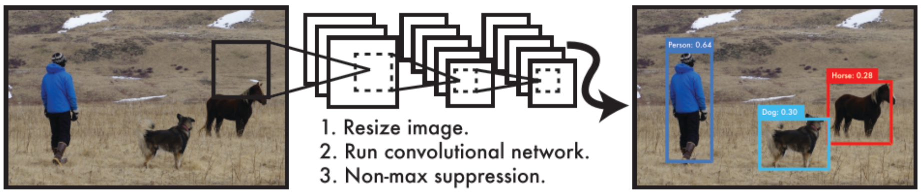
Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using a ROI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes (REN et al., 2017).

2.4.2.4 YOLO - *you only look once*

All of the previous object detection algorithms use regions to localize the object within the image. The network does not look at the complete image, but parts of the images which have high probabilities of containing the object. YOLO or You Only Look Once is an object detection algorithm much different from the region based algorithms seen above. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

YOLO is refreshingly simple: see Figure 22. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection. Processing images with YOLO is simple and straightforward (REDMON et al., 2016).

Figure 22 – The YOLO Detection System. YOLO (1) resizes the input image to 448x448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.



Source – Redmon et al. (2016)

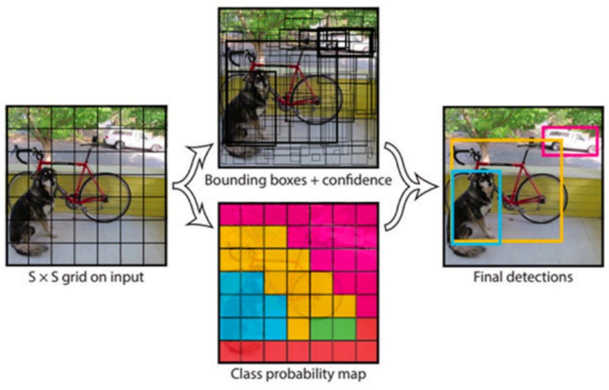
YOLO has orders of magnitude faster than other object detection algorithms. The limitation of YOLO algorithm is that it struggles with small objects within the image, for example it might have difficulties in detecting a flock of birds. This is due to the spatial constraints of the algorithm.

This method works by dividing the input image into an S x S grid and if the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts.

Formally we define confidence as $Pr(Object) * IOU_{pred}^{truth}$. If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image.

Figure 23 – System models detection as a regression problem.

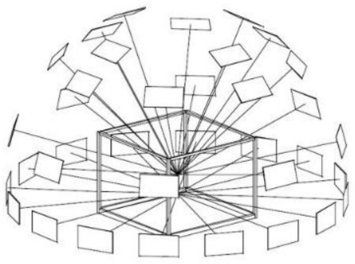


Source – Redmon et al. (2016)

2.5 PHOTOGRAMMETRY

Photogrammetry is a science that encompasses different methods of three-dimensional object measurement through a set of two-dimensional images recorded of the object with different positions and orientations (LUHMANN et al., 2014; U. RAUTENBERG, 2002). Figure 24 presents a good configuration of images acquisition of a specify object with sufficient overlap between images to generate enough homologous points capable to recreate it in a three dimension space.

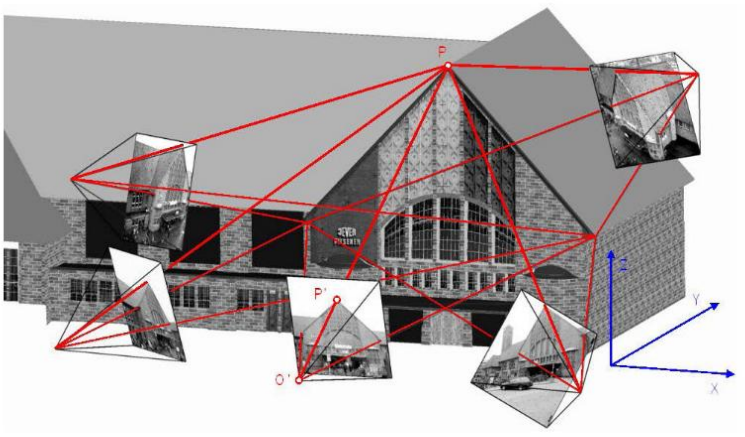
Figure 24 – Good configuration of image acquisition to photogrammetry.



Source – U. Rautenberg (2002)

The concept of photogrammetry is a measurement technique which uses central projection imaging as its fundamental mathematical model. Figure 25 presents a object reconstruction using a set of images, it shapes and position are determined by reconstructing bundles of rays in which, for each camera, each image point P', together with the corresponding perspective center O', defines the spatial direction of the ray to the corresponding object point P. This is possible as long as the geometry of the image on the camera and the location of the image system in the object space are known, all image rays can be defined in the 3D object space (LUHMANN et al., 2014; LINDER, 2016).

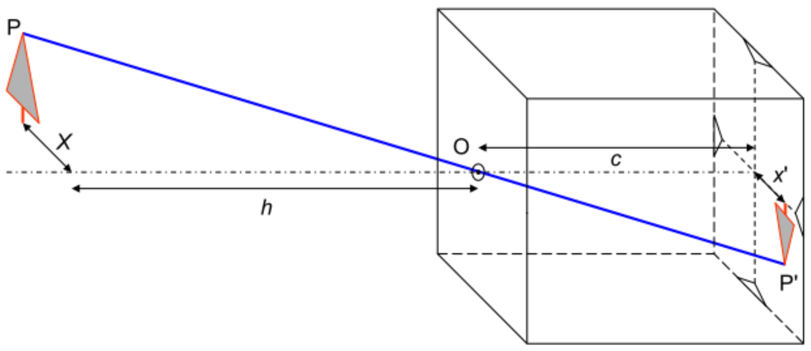
Figure 25 – Principle of photogrammetric measurement



Source – Luhmann et al. (2014)

Using the model of the pinhole camera, Figure 26, knowing the spatial location of the perspective center *O*, where all image rays pass. The interior orientation defines the position of the perspective centre relative to a reference system fixed in the image coordinate system, as well as departures from the ideal central projection. All is based on internal parameter *c* which defines the distance between center *O* and projection plane.

Figure 26 – Pinhole camera model.



Source – Luhmann et al. (2014)

After a brief understand of some basic concepts of kinematics of a quadrotor aircraft and the Gimbal mechanical structure to stabilize cameras, computer vision and communication between sensors and actuators using Robot Operating System (ROS). It is possible to generate a control system to the Gimbal to track an object, maintaining the object in the camera's field of view to acquire images without loosing object and posterior three-dimensional reconstruction. The next chapter will present the concept of gimbal's track algorithm and methodology used to validate it.

3 GIMBAL CONTROL SYSTEM

This chapter presents the proposal control algorithm to maintain detected object tracked when on gimbal mechanism. The following sections is described the software structure needed, two control theory to maintain the object centered on image to posterior comparation and finishing presenting the workflow of data to structure a feedback system.

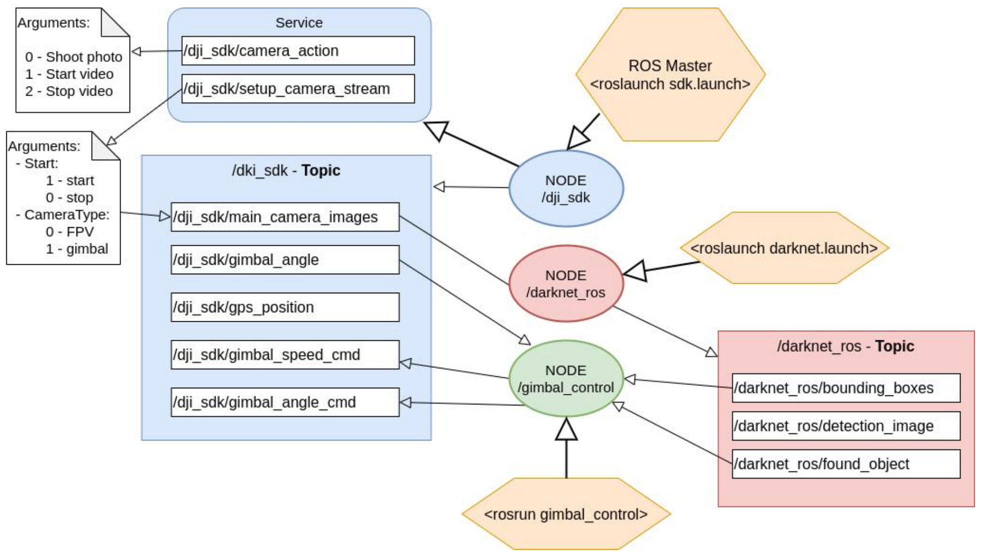
3.1 SOFTWARE STRUCTURE

Understanding how operates a quadrotor aircraft, the gimbal mechanism to stabilize and its mathematical concept, Communicating between payloads and sensor using an on-board computer with Robotic Operational System. It is possible to develop a node capable of tracking an object detected using computer vision and determine the angles positions from gimbal mechanism to maintain it centred on image using information provided from ROS topics when aircraft is powered on.

To initialize the *gimbal_control* node it is necessary activate the *dji_sdk* node, providing RPAs sensors, and payload communication, such as gimbal angles position and posterior start *darknet_ros* node to perform YOLO detection over camera image topic. After initialized *dji_sdk* and *darknet_ros* will provide enough data to begin the track control. To initialize the gimbal's camera demands a service called */dji_sdk/setup_camera_stream* is necessary to open a ROS topic from image registered from camera.

When gimbal's camera service is requested, the image data is presented at topic */dji_sdk/main_*
camera_images, and will be possible to initialize node *darknet_ros* to YOLO process images and detect objects. The *darknet_ros* node will subscribe the camera's image topic and publish the topics of found objects and bounding boxes position of each object. Figure 27 shows the workflow of commands and its provided topics necessary to initialize *gimbal_control*.

Figure 27 – ROS topics necessary to *gimbal_control* node and its respectively origin.



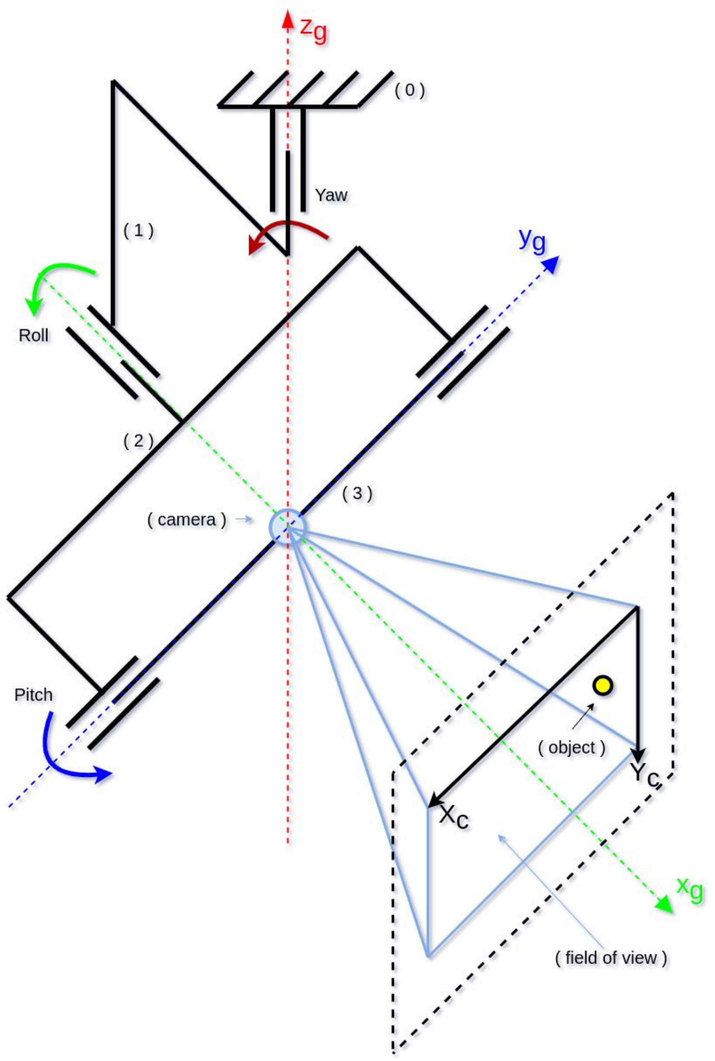
After all data from sensors and YOLO image process, it is possible to initialize the *gimbal_control* node to control the gimbal's angles position control based on pixel error signal processed from detection of desired object from */darknet/bounding_boxes* topic and a reference value, center of image resolution. Following sections will explain two theories of control to maintain object centered using information of pixel position.

3.1.1 Gimbal Inverse Kinematic

Based on gimbal coordinate system G, described on section 2.3.1, and its axis orientation as Figure 15. This section present the mathematical of gimbal's inverse kinematic equations to be used as a control theory, using a camera positioned at the origin of mechanism coordinate system G with the camera's field of view orientation. Figure 28 presents the gimbal mechanism and the coordinate system attached and at its origin, the camera and the initial orientation of the camera's field of view along X_g axis. As the center of mass from the camera is centered at origin, it will not have a translation movement, just rotation. Along the end of the camera's field of view, at a nominal distance called dx, the desired object is detected. When transformed the camera's field of view to the image resolution processed, a new coordinate system of camera image called C is created. This coordinate system is the pixel matrix of image and the object is detected over the topic */darknet_ros/bounding_boxes* in pixel.

To track the object detected at camera's image is necessary only two degrees of freedom at gimbal mechanism. The angle roll, at X_g axis, makes no impact in changing spatial position of camera's field of view, it will just change the image orientation from landscape to portrait. To simplify the process, roll angle is held at 0 degrees position.

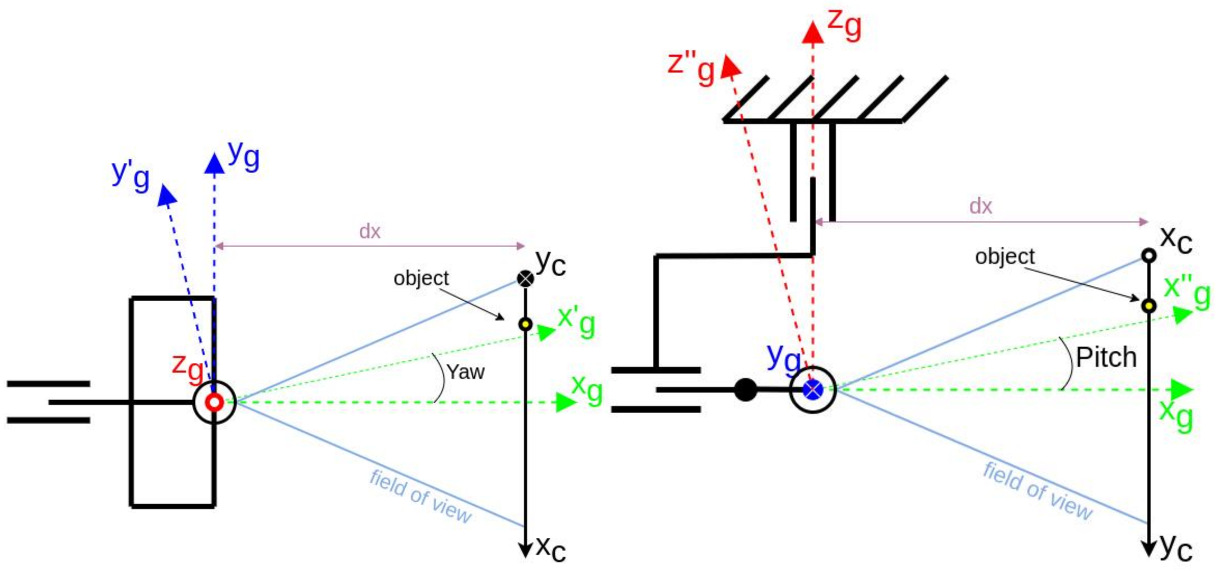
Figure 28 – Gimbal mechanism with a camera's field of view and image coordinate system.



The displacement of the camera's field of view is given by the yaw (Φ_g) and pitch (Θ_g) angles of the gimbal, along axis Z_g and Y_g respectively. Figure 29 present the axis orientation of gimbal *G* and image *C* at gimbal planes X-Z and X-Y. The camera image coordinate system *C* is dislocated from gimbals origins at a distance *dx* along axis X_g. This parameter will represent the distance from object to the origin of gimbal mechanism and allows to set of an initial spatial position of gimbal kinematics as $[dx\,0\,0]^T$.

To move the center of image along X_c axis to center the object on image acquired, the gimbal has to rotate the angle yaw (Φ_g) at Z_g axis. This movement gives a new gimbal coordinate position $[X_g'\; Y_g'\; Z_g]^T$, presented on Figure 29a. Same occurs to camera image axis Y_c when we rotate pitch angle at Y_g (Θ_g) axis, result in a new position $[X_g''\; Y_g\; Z_g'']^T$, as Figure 29b.

Figure 29 – Gimbal coordinate planes.



(a) XY Plane. (b) XZ Plane.

Estimating the distance between the object and camera as parameter *dx*, the equation (8) from the forward kinematic of the gimbal, where represent a fixed coordinate system, allow the deduct the inverse kinematic formulas that represent the angles of gimbal from the spatial position $[X\ Y\ Z]^T$. As the gimbal joint roll Ψ_g is defined equal to 0^o, the rotate matrix is expressed on equation (9).

$$\overline{R_3^0} = \begin{bmatrix} C_{\Phi_g} C_{\Theta_g} & -C_{\Phi_g} S_{\Theta_g} & S_{\Theta_g} \\ S_{\Phi_g} & C_{\Phi_g} & 0 \\ -C_{\Phi_g} S_{\Theta_g} & S_{\Phi_g} S_{\Theta_g} & C_{\Theta_g} \end{bmatrix} \tag{9}$$

Multiplying the equation (9) to the initial position $[dx\,0\,0]^T$, result in three equations to express the new $[X_g\ Y_g\ Z_g]^T$ position.

$$\begin{bmatrix} X_g \\ Y_g \\ Z_g \end{bmatrix} = \begin{bmatrix} C_{\Phi_g} C_{\Theta_g} & -C_{\Phi_g} S_{\Theta_g} & S_{\Theta_g} \\ S_{\Phi_g} & C_{\Phi_g} & 0 \\ -C_{\Phi_g} S_{\Theta_g} & S_{\Phi_g} S_{\Theta_g} & C_{\Theta_g} \end{bmatrix} \begin{bmatrix} dx \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} X_g \\ Y_g \\ Z_g \end{bmatrix} = \begin{bmatrix} C_{\Phi_g} C_{\Theta_g} dx \\ S_{\Phi_g} dx \\ -C_{\Phi_g} S_{\Theta_g} dx \end{bmatrix} \tag{10}$$

where derive two equations for the Y_g and Z_g provided from image position:

$$sin(\Phi_g) = \frac{Y_g}{dx} \tag{11}$$

$$sin(\Theta_g) = \frac{-Z_g}{dx\, cos(\Phi_g)} \tag{12}$$

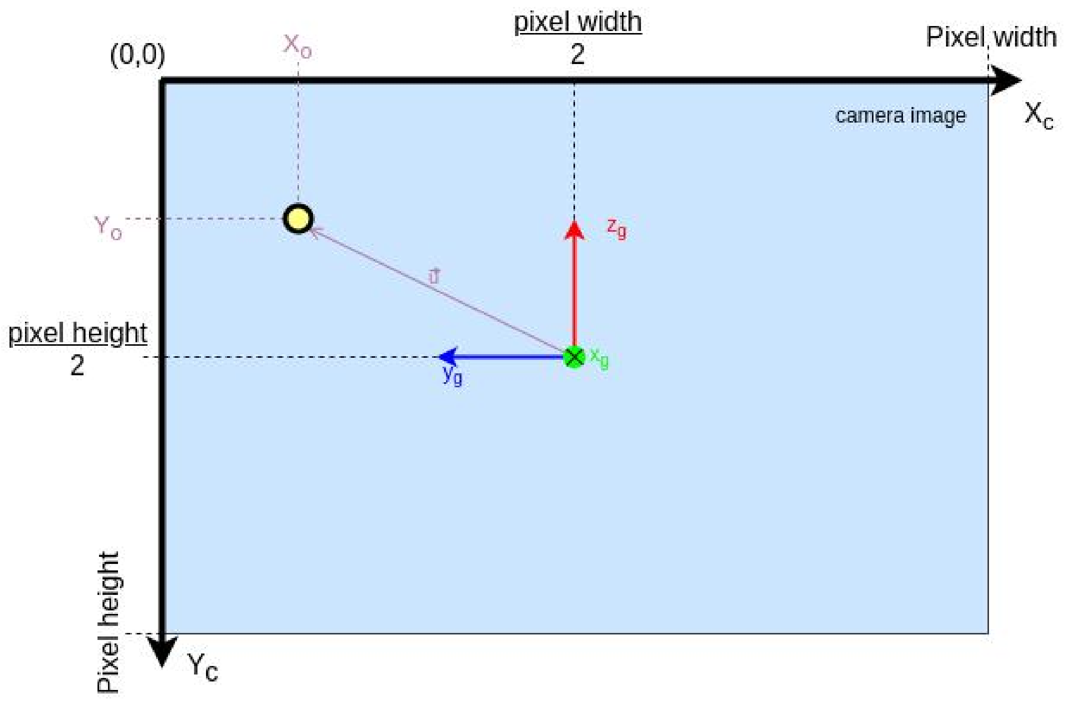
representing the new angle's position as:

$$\Phi_g = arcsin\left(\frac{Y_g}{dx}\right) \tag{13}$$

$$\Theta_g = arcsin\left(\frac{-Z_g}{dx\, cos(\Phi_g)}\right) \tag{14}$$

After deducting the equations of gimbal inverse kinematic, (13) and (14), it is necessary determine the values of Z_g and Y_g to estimate the new angles to maintain the object centered of the camera image. Being the only object position data is in pixel resulted from image detection using YOLO, it has to be converted to a value in meters. Figure 30 presents the camera image coordinate system C with the position of gimbal coordinate system G along the image view transcript at position $(\frac{pixel\ height}{2}, \frac{pixel\ width}{2})$ and the object O detected at position (X_o, Y_o).

Figure 30 – Gimbal coordinate system G represented over the camera image coordinate system C.



Vector $\overrightarrow{u}$ determine the movement of Gimbal coordinate system to maintain the object centered at the image acquired. The values of Y_g and Z_g based on object position can be represented as equations (15) and (16). It is the difference between object position and center of image, reference values, multiplied by a parameter called Ground Sample Distance (GSD), as a sensitivity parameter, to convert the values from pixel to meters as well. The value of GSD is determined by the initial distance parameter *dx*. Equations (15) and (16) also can be used as the pixel error signal to to control system.

$$Z_g = \left(\frac{pixel\,width}{2} - X_o\right) GSD \tag{15}$$

$$Y_g = \left(\frac{pixel\,height}{2} - Y_o\right) GSD \tag{16}$$

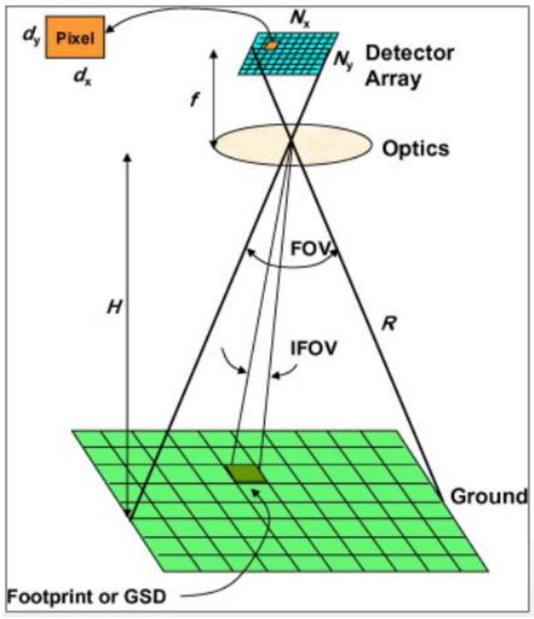
In physics and geosciences, the term spatial resolution refers to the precision of a measurement with respect to space, or the real dimension that represents a pixel of the image (SRIVANTAVA et al., 2016). It can be defined as GSD, representing the distance from ground to air or space by measuring the distance between pixel centers on the ground on aerial photogrammetry inspections. GSD is the area of object or ground represented in one pixel, it is possible to know the spatial resolution from the distance of an object, focal distance and pixel resolution of a camera. Figure 31 exemplify a GSD. It is possible to determine GSD based on pixel dimension multiplied by the distance of object divided to focal length, equation (17).

$$GSD = \frac{Pixel\,Dimension\,x\,Distance}{Focal\,Length} \tag{17}$$

Also Ground Sample Distance can be express using the object distance parameters (*dx*), horizontal angle viewing (AoV_h) resulted of lens focal length, and the pixel resolution. equation (18).

$$GSD = \frac{2\,tan(\frac{AoV_h}{2})\,dx}{width\ resolution} \tag{18}$$

Figure 31 – Ground Sample Distance or spatial resolution.



Source – Zheng and Tidrow (2009)

3.1.2 Proportional-Integral Control

As the position control resulted from gimbal's inverse kinematic has two equations coupled to determine new angles position, is proposed a proportional-integral-derivative (PID) controller to compare the perform, being one of the most common control algorithm. This proposal referred to two independent controls for each axis of image resolution based on pixel's error signal (ÅSTRÖM; HÄGGLUND, 1995).

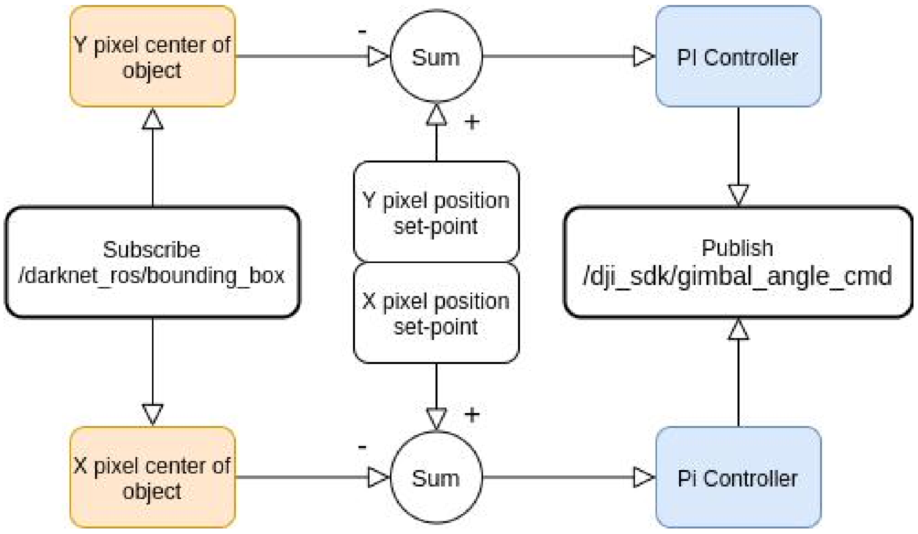
As pixel position reference being the center of image resolution, the error signals are represented from (15) and (16), multiplied to GSD as a sensibility parameter. The controller receive the error signal as an input to return new angle's position. Integral part of algorithm will guarantee the reference segment and reject of disturbances resulted of movement of aircraft. The feedback systems are presented in Figure 32, where the data are received and published on ROS topics. A time variant PID equation used to implement algorithm is represented by equation (19) (MICHAEL A. JOHNSON MOHAMMAD H., 2007).

$$PID(t) = K_C\, e(t) + \frac{K_C}{T_i}\int_0^t e(t)dt + K_C\, T_d\, \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{19}$$

The controller is traditional tuned using a derivative part of error's signal. However, for this system a derivative part is not necessary to achieve the desired specifications, also this part of controllers amplify high frequency noise. As the YOLO detection process the neural network frame by frame from image's topic, it generates an imprecise position of object reflected at published data from bounding boxes topic. The main objective is reduce noise effect the controller is just tuned without the derivative part, resulting at equation (20).

$$PI(t) = K_C\, e(t) + \frac{K_C}{T_i} \int_0^t e(t)dt \tag{20}$$

Figure 32 – PI Controller schematic.


Y pixel center of object
Sum
PI Controller
Subscribe /darknet_ros/bounding_box
Y pixel position set-point
X pixel position set-point
Publish /dji_sdk/gimbal_angle_cmd
X pixel center of object
Sum
Pi Controller
-
+
+
-

Proportional-Integral controllers have been tuned in order to reach the new references as fast as possible without a overshoot bigger than 5%. To adjust the controllers were used a dynamic response from data resulted of pixel position topic. It is using the equation of a PI controller in a Z-transform discrete equation (21). In mathematics and signal processing, the Z-transform converts a discrete-time signal, which is a sequence of real or complex numbers, into a complex frequency-domain representation. It can be considered as a discrete-time equivalent of the Laplace transform. This similarity is explored in the theory of time-scale calculus (LATHI, 2009; ATTAR, 2006; PETALE, 2018).

$$PI = K_C \frac{z - z_0}{z - 1} \tag{21}$$

To convert (21) to discrete sample time is necessary to shift backwards on z sample time multiplying it to z^-1. It will result on equation (22) based of Z-transform concept that the power of z represents which sample time to use (LATHI, 2009).

$$
\begin{aligned}
U(z) &= PI\, e(z) \\
U(z) &= K_C \tfrac{z-z_0}{z-1}\, e(z) \\
(z-1)U(z) &= Kc(z-z_0)\, e(z) \times \tfrac{1}{z} \\
(1-\tfrac{1}{z})U(z) &= Kc(1-\tfrac{z_0}{z})\, e(z) \\
u(k) &= K_C\,[e(k) - z_0\, e(k-1)] + u(k-1)
\end{aligned} \tag{22}
$$

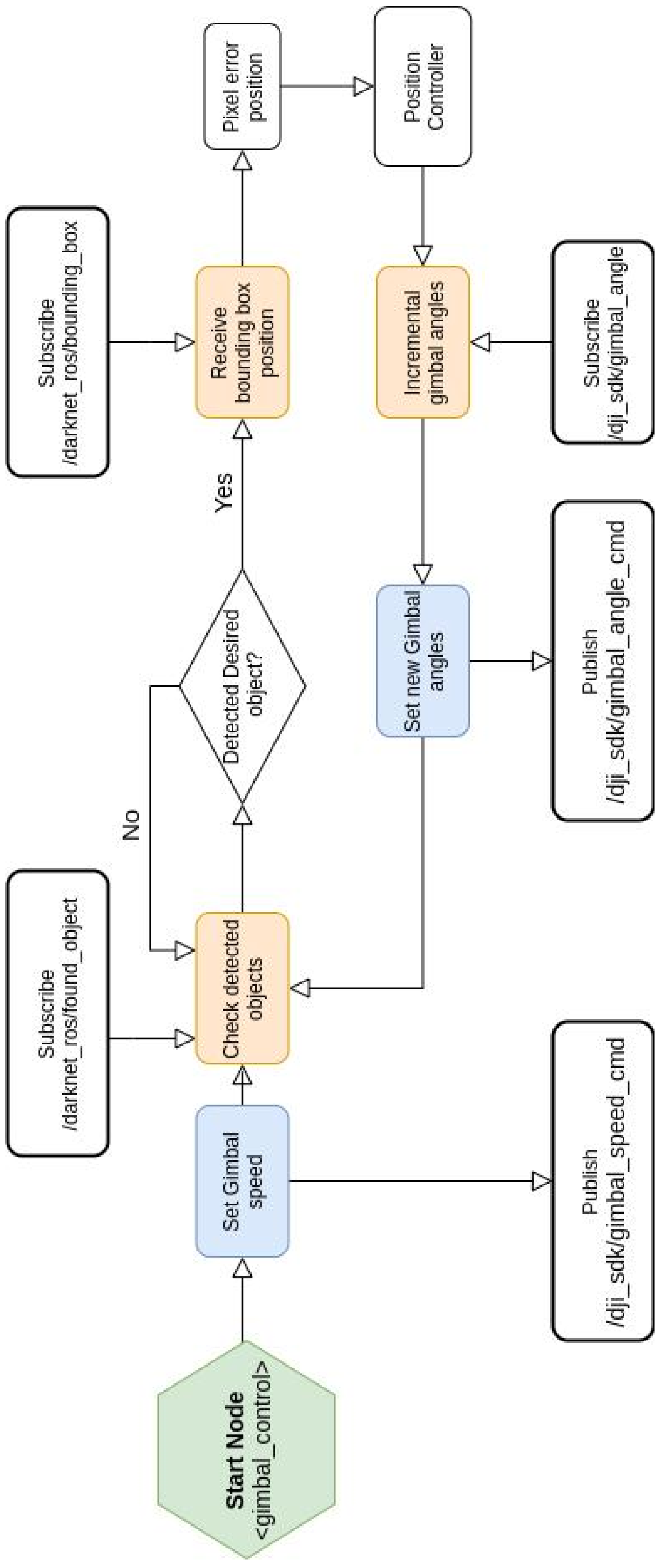
3.1.3 Node gimbal_control

The node *gimbal_control* works as a process, subscribing data topics to compute new gimbal angles. Data from *dji_sdk* and *darknet_ros* nodes are processed and determined the new values, publishing the speed of gimbal joints and new angles computed.

The main concept of this node is described at Figure 33. The process started defining the gimbal joint speed, publishing it to the topic */dji_sdk/gimbal_speed_cmd*, and type of object to be tracked. Once initialized, the node will subscribe to */darknet_ros/found_object* topic and receive the numbers and names of detected objects, checking if the desired one is detected.

After process detected desired object, will subscribe to */darknet_ros/bounding_boxes* topic to get pixel position of that and fed the information on controllers structure, generating the new angles position to the system. Those new values of angle position will be added to actual positions provided from topic */dji_sdk/gimbal_angle* as the mode defined (incremental or absolute). At end will publish on topic */dji_sdk/gimbal_angle_cmd* and the algorithm will check if the same object is still detected, closing the feedback system with error zero resulted of bringing system back to initial position $[dx\, 0\, 0]^T$.

Figure 33 – Node *gimbal_control* sequence.


Start Node
<gimbal_control>
Set Gimbal speed
Publish /dji_sdk/gimbal_speed_cmd
Subscribe /darknet_ros/found_object
Check detected objects
No
Detected Desired object?
Yes
Subscribe /darknet_ros/bounding_box
Receive bounding box position
Pixel error position
Position Controller
Incremental gimbal angles
Subscribe /dji_sdk/gimbal_angle
Set new Gimbal angles
Publish /dji_sdk/gimbal_angle_cmd

4 METHODOLOGY

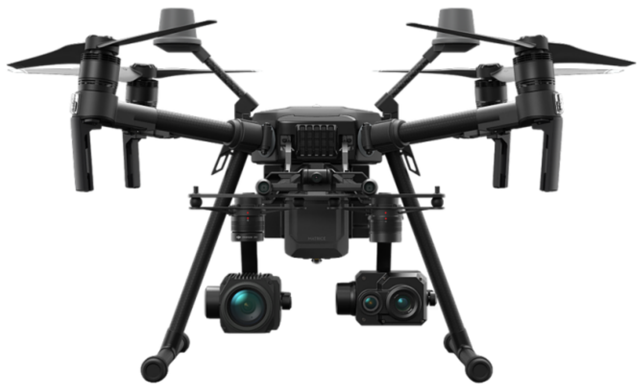
This chapter explain the materials and tools used for development, all equipment provided by the research project of VANT3D from Labmetro[1] at Federal University of Santa Catarina.

4.1 DJI MATRICE M200 V2 RTK

The Matrice 200 v2 RTK is a powerful aerial imaging system with class-leading agility and speed. This aircraft is mostly used in industrial inspections and has been tested on field for visual inspection and acquired some images from a offshore riser balcony and it is robust enough to work in strong magnetic interference fields resulting from all metallic structure.

This Matrice 200 series is a versatile platform, it comes with new a feature of obstacle avoidance with an anti-collision beacon. In addition to this new function, there are additional accessories such as a Real Time Kinematic (RTK) mobile station which improves the relative accuracy with centimeter-level precision positioning data and real-time differential corrections. The aircraft version is shown in Figure 34.

Figure 34 – Matrice M200 v2 RTK.



Source – https://www.dji.com/matrice-200-series-v2. Accessed: May 10th

Real-time kinematic (RTK) positioning is a satellite navigation technique used to enhance the precision of position data derived from satellite-based positioning systems (global navigation satellite systems, GNSS). Reducing the position uncertainty helps in the three-dimensional reconstruction of the object when using the photos acquired with aircraft and the RTK systems.

[1] Laboratory of metrology

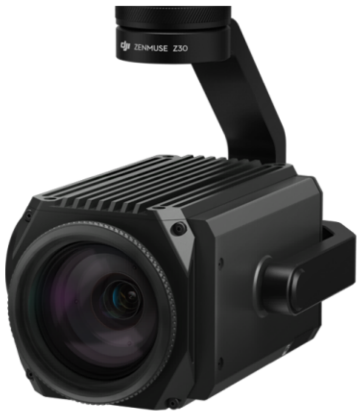
In addition, it has a on-board SDK Compatibility, allowing the construction of powerful automated drone applications, and flight controllers. Using an on-board computer where you can build customized solutions using all peripherals components. Although this is a new feature, it has the common payload as gimbal mechanism, cameras and sensors all integrated with ROS.

4.1.1 Camera Gimbal Mechanism

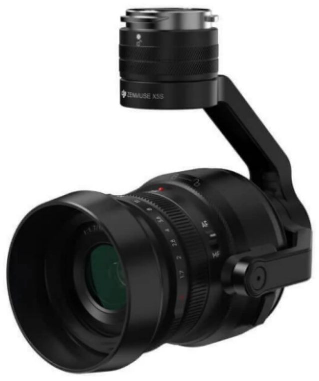
Matrice series is compatible with camera gimbal mechanism developed by DJI to integrate with vehicles and all operational systems behind it, with a rough low-level controller capable in adverse conditions.

For testing, the DJI M200 v2 RTK has two camera gimbal compatibles, allowing tests to decide the best camera configuration for the purpose of photogrammetric inspection of risers. The cameras are: Zenmuse Z30, Figure 35a and Zenmuse X5S, Figure 35b. Table 2 presents the technical specification for both cameras, where it is possible to realize the Z30 has poor resolution images but a great optical zoom range, while the X5S has a good resolution image but is restricted to a fixed focal length lens.

Figure 35 – DJI gimbal cameras.



(a) Zenmuse Z30.



(b) Zenmuse X5S.

Source – https://www.dji.com/zenmuse-z30/; https://www.dji.com/zenmuse-x5s/. Accessed: April 10th.

Comparing the performance of both camera gimbal mechanisms, to the photogrammetry inspection purpose it is clear that a Zenmuse X5S provides good quality resolution and images save in different formats than JPEG. It compresses image data by reducing sections of images to blocks of pixels or “tiles.” JPEG compression has the unfortunate side effect of being permanent. Beyond resolution and image format, a idea of using a fixed focal length helps for a posterior reconstruction. Although for visual inspection Zenmuse Z30 allows with a 30x optical zoom a better look on surfaces damages.

Table 2 – Specifications of available camera gimbals.

	Zenmuse Z30	Zenmuse X5S
Sensor	CMOS, 1/2.8"	CMOS, 4/3"
Pixel Resolution	1920x1080	5280x3956
Lens	30x zoom	15-45 mm
Photo Format	JPEG	DNG & JPEG
Weight	556g	461g
Yaw Controllable Range	$\pm 320^o$	$\pm 320^o$
Pitch Controllable Range	$+30^o$/–120^o	$+40^o$/–120^o
Roll Controllable Range		$\pm 20^o$
Yaw Mechanical Range	$\pm 330^o$	$\pm 330^o$
Pitch Mechanical Range	$+50^o$/–140^o	$+50^o$/–140^o
Roll Mechanical Range	$+90^o$	$+90^o$/–50^o
Yaw Controllable Speed	$\pm 180^o/s$	$\pm 270^o/s$
Pitch Controllable Speed	$\pm 180^o/s$	$\pm 180^o/s$
Roll Controllable Speed		$\pm 180^o/s$
Angle position feedback	$\pm 0,01^o$	$\pm 0,01^o$

Source – `https://www.dji.com/zenmuse-z30/`; `https://www.dji.com/zenmuse-x5s/`. Accessed: April 10th.

4.2 DJI MANIFOLD 2

Compatible with DJI series Matrice 200 and flight controllers, embodies a powerful processor integrated with on-board SDK developer platform allowing to create personal autonomous applications. Manifold 2, Figure 36, provides two configurations to better fit your needs. Between the two configurations, presented in Table 3, it has a model Computer Process Unit (CPU) with Intel Core i7-8550U most used for real-time data analysis and autonomous flight and a model Graphic Process Unit (GPU) from NVidia Jetson TX2 mostly used for artificial intelligence algorithms, image processing and detection of objects.

Figure 36 – DJI Manifold 2.

Source – `https://www.dji.com/manifold-2`. Accessed: April 14th.

Table 3 – Manifold 2 Specifications.

Model	Manifold 2-G	Manifold 2-C
Weight	≈ 230g	≈ 205g
Processor	NVidia Jetson TX2	Intel Core i7-8550U
Memory	8 GB 128 bit, DD4 1333 MHz	8 GB 128 bit, DD4 2400 MHz
eMMC	28 GB available	N/A
SATA-SSD	128 GB	256 GB
Network	1000 Mbps Ethernet	1000 Mbps Ethernet
I/O	CAN, UART, I2C, SPI	UART
Power	3-25 W	5-60 W

Source – `https://www.dji.com/manifold-2/specs`. Accessed: April 15th

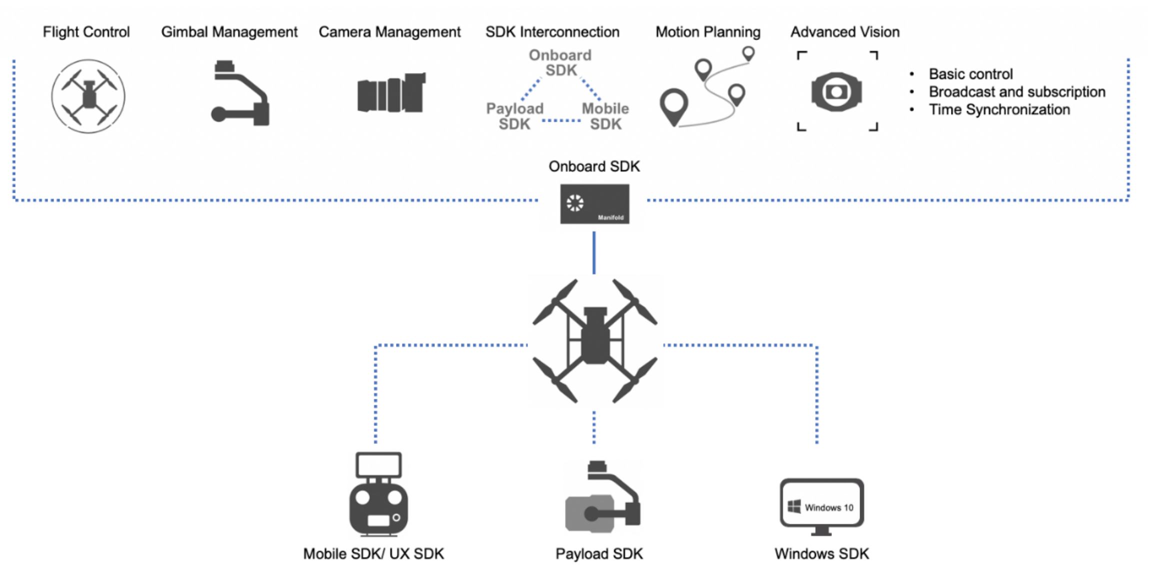
VANT3D is a research project of three-dimensional optical inspection, it purchased the Manifold 2-G for image processing and process neural networks for object detection. Any citation of Manifold 2 ahead in this document is explicitly talking about GPU configuration. Using OS Linux in Manifold 2 makes it possible to use the API of on-board SDK for ROS, granting all benefits of communication and exchanging data from ROS.

4.2.1 Communication

The on-board computer communicates to the DJI aircraft through on-board SDK (OSDK) platform. It is a development toolkit for developing applications, which could run on the on-board computer. According to the software logic and algorithm framework of OSDK, it is possible to perform actions such as Automated Flight, Payload Control And Video Image Analysis. Figure 37 presents the external device connections and on-board features using OSDK.

The communication is given by an universal asynchronous receiver-transmitter (UART), it is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. Full connection to communicate is established via USB cable from an on-board computer and RPA to receive and transmit data. Figure 38 shows the connection made between Manifold and aircraft with features of Advance Sensing and transmission of images from gimbal's camera. A USB cable is needed to transfer any type of image data due to a limitation of UART connection around 500 KB/s and an USB 3.0 bus transfer approximately 625 MB/s.

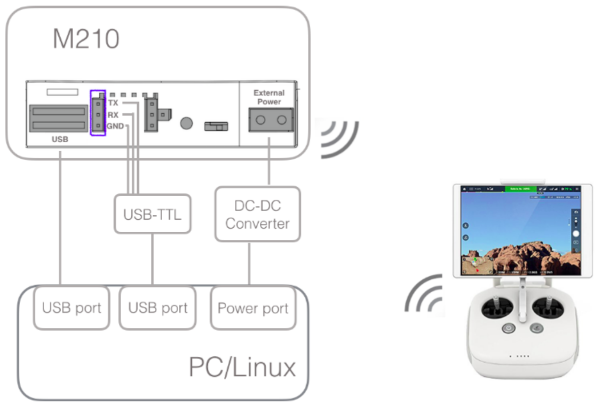
Figure 37 – Device connections and features from on-board SDK.



Source – https://developer.dji.com/document/1be2b992-5862-4eb5-819b-dd6036f76133. Accessed: July 23rd

Advanced Sensing is a combination of forward and downward stereo camera with a Vision Positioning System (VPS) who's gives the precise hovering and collision avoidance capabilities, even without satellite positioning support. The stereo system also allows it to brake instantly and hover when joystick controls are released, and have several additional data of object depth and detection. This feature has a capability of detect objects and determine its distance from aircraft. (DJI, 2019)

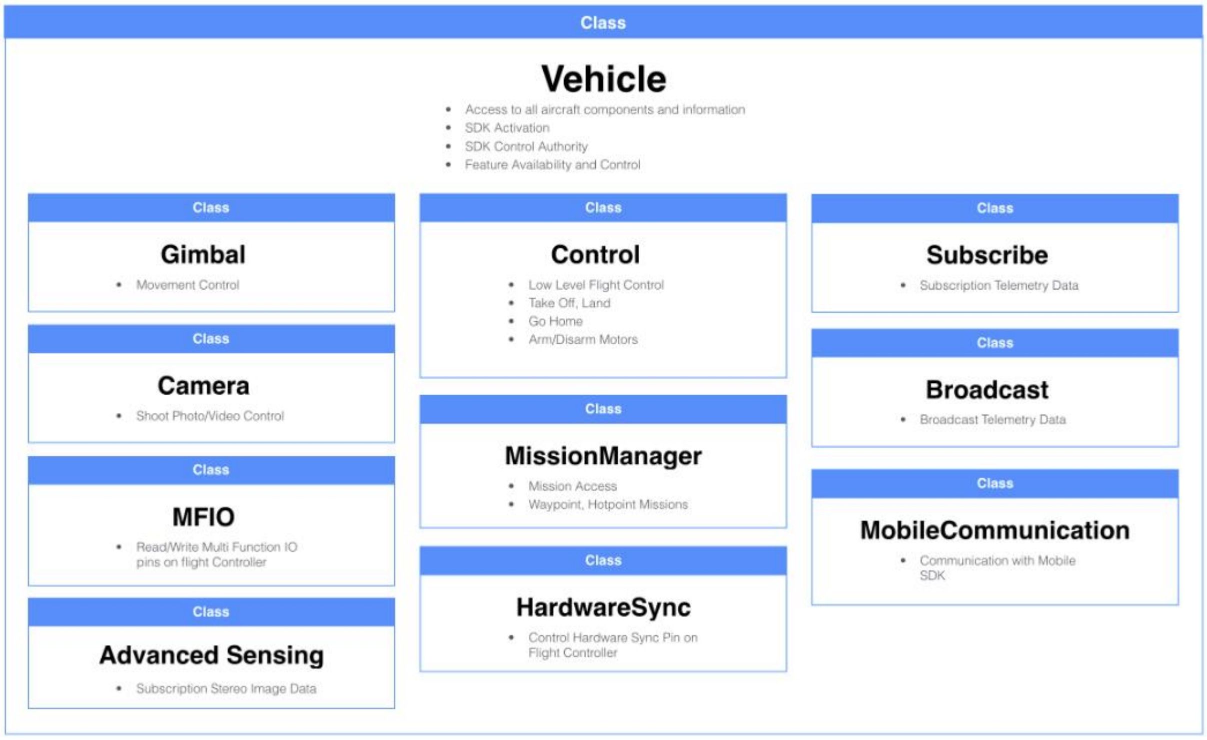
Figure 38 – M200 + PC/Linux machine with a communication via USB to image data.



Source – https://developer.dji.com/on-board-sdk/documentation. Accessed: May 5th

After the connection between aircraft and on-board computer is established, the Application Programming Interface (API) hierarchy classes of on-board SDK, shown in Figure 39, has a class to each application accessing the DJI product capabilities at the main Vehicle class in the on-board SDK. To develop a control in joint gimbal, it is possible to access its data and set references to the joint controllers using the Gimbal class.

Figure 39 – Classes of on-board SDK.

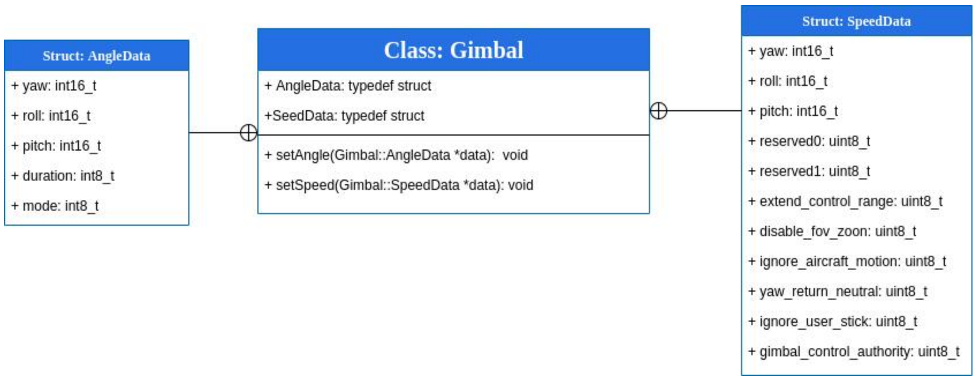

Class
Vehicle
Access to all aircraft components and information
SDK Activation
SDK Control Authority
Feature Availability and Control
Class
Gimbal
Movement Control
Class
Camera
Shoot Photo/Video Control
Class
MFIO
Read/Write Multi Function IO pins on flight Controller
Class
Advanced Sensing
Subscription Stereo Image Data
Class
Control
Low Level Flight Control
Take Off, Land
Go Home
Arm/Disarm Motors
Class
MissionManager
Mission Access
Waypoint, Hotpoint Missions
Class
HardwareSync
Control Hardware Sync Pin on Flight Controller
Class
Subscribe
Subscription Telemetry Data
Class
Broadcast
Broadcast Telemetry Data
Class
MobileCommunication
Communication with Mobile SDK

Source – `https://developer.dji.com/on-board-sdk/documentation`. Accessed: May 5th

4.2.1.1 Gimbal Class

The Gimbal class inside the main class Vehicle allows the use of two public functions implemented, *setAngle* function defining mode and angles for the gimbal and *setSpeed* function, defining rate of change for the gimbal angles, Figure 40.

Figure 40 – Class Gimbal of on-board SDK API


Struct: AngleData
+ yaw: int16_t
+ roll: int16_t
+ pitch: int16_t
+ duration: int8_t
+ mode: int8_t
Class: Gimbal
+ AngleData: typedef struct
+SeedData: typedef struct
+ setAngle(Gimbal::AngleData *data): void
+ setSpeed(Gimbal::SpeedData *data): void
Struct: SpeedData
+ yaw: int16_t
+ roll: int16_t
+ pitch: int16_t
+ reserved0: uint8_t
+ reserved1: uint8_t
+ extend_control_range: uint8_t
+ disable_fov_zoon: uint8_t
+ ignore_aircraft_motion: uint8_t
+ yaw_return_neutral: uint8_t
+ ignore_user_stick: uint8_t
+ gimbal_control_authority: uint8_t

Source – `https://developer.dji.com/onboard-api-reference/classDJI_1_1OSDK_1_1Gimbal.html`. Accessed: May 5th

A struct *AngleData* defines the angle roll, pitch and yaw in a 0.1^o resolution; the duration for the gimbal to reach the commended position, for example value 20 takes 2 second; and a mode of use detailed at Table 4. *SpeedData* defines the angular velocity of gimbal joints and activates or deactivates the rest of the parameters of struct.

Table 4 – AngleData struct parameters mode.

Bit #:	Set to 0:	Set to 1:
bit 0	Incremental control, the angle reference is the current Gimbal location	Absolute control, the angle reference is related to config. in DJI Go App
bit 1	Gimbal will follow the command in Yaw	Gimbal will maintain position in Yaw
bit 2	Roll invalid bit, the same as bit[1]	Roll invalid bit, same as bit[1]
bit 3	Pitch invalid bit, the same as bit[1]	Pitch invalid bit, same as bit[1]
bit[4:7]	bit[4:7]: reserved, set to be 0	

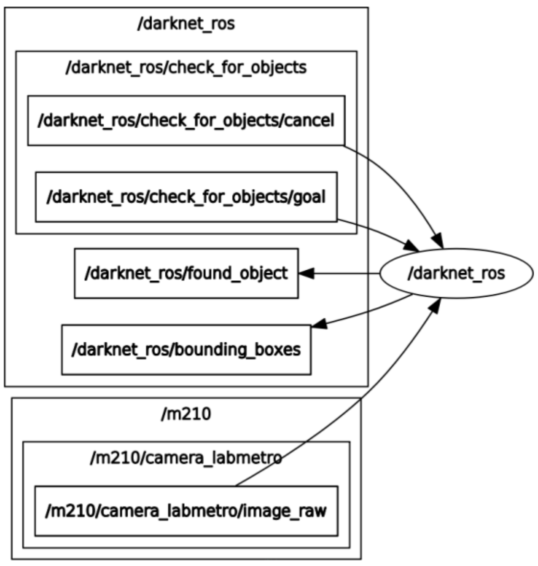
Source – `https://developer.dji.com/on-board-api-reference/structDJI_1_1OSDK_1_1Gimbal_1_1AngleData.html`. Accessed: May 8th

4.3 REAL TIME OBJECT DETECT PROCESS

You only look once (YOLO) is a state-of-the-art, real-time object detection system developed to run individually, searching in stream data entrance of an object in dataset using a pre-trained model of the convolutional neural network, enables it to detect pre-trained classes.

Associating the ROS environment and YOLO algorithm, (BJELONIC, 2016) created a ROS package capable of integrating the algorithm with the robotic platform, publishing topics of objects detected and their respective bounding boxes in two ROS topics. The workflow of node darknet is shown in Figure 41, the node subscribes to the image_raw topic of gimbal camera and publishes a topic with a list of objects detected, pixel position of bounding boxes and a image topic with the bounding boxes.

Figure 41 – Darknet ROS node and its topics.


/darknet_ros
/darknet_ros/check_for_objects
/darknet_ros/check_for_objects/cancel
/darknet_ros/check_for_objects/goal
/darknet_ros/found_object
/darknet_ros
/darknet_ros/bounding_boxes
/m210
/m210/camera_labmetro
/m210/camera_labmetro/image_raw

The bounding box topic published a message, created by the *darknet_ros* package (BJELONIC, 2016), from the node bringing information of position in pixel from the detected object. Topic's message structured of probability value, minimum and maximum value of position (X, Y), the identification number and class of the object, Table 5.

Table 5 – Bounding Box message struct.

Parameter	Type
Header	header
Header	image_header
x_{min}	int64
y_{min}	int64
x_{max}	int64
y_{max}	int64
id	int16
string	Class

Source – Bjelonic (2016)

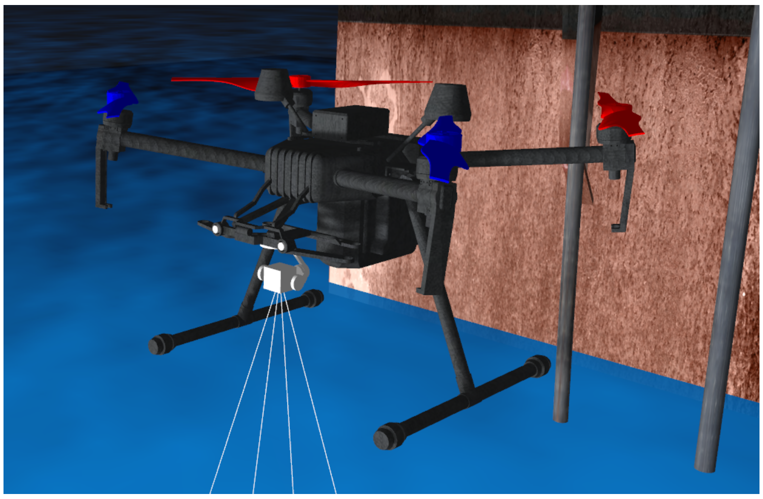
4.4 GAZEBO SIMULATION ENVIRONMENT

To debug and syntonize the proposed node without damage or creating software problems in the equipment, it has been developed using the simulation environment from software Gazebo.

Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. While similar to game engines, Gazebo offers physics simulation at a much higher degree of fidelity, a suite of sensors, and interfaces for both users and programs. It is typically used to test robotic algorithms, design robots and perform regression testing with realistic scenarios (OPEN SOURCE ROBOTICS FOUNDATION, 2014).

The simulation environment consists of a CAD model representation of the aircraft Matrice 200 v2 using a *rotors_simulation* package (FURRER et al., 2016). RotorS is a Micro Air Vehicle (MAV) gazebo simulator. It provides some multi-rotor models such as the AscTec (GURDAN et al., 2007) Pelican or Firefly, four and six rotors respectively. The simulator is not limited for the use with theses multi-copters, it allows some modification of parameters to use the AscTec Pelican quadrotor controller at the CAD model of Matrice 200 v2. Figure 42 shows the RPA Matrice 200 in the Gazebo environment. All controllers developed in *rotors_simulation* are communicate using ROS.

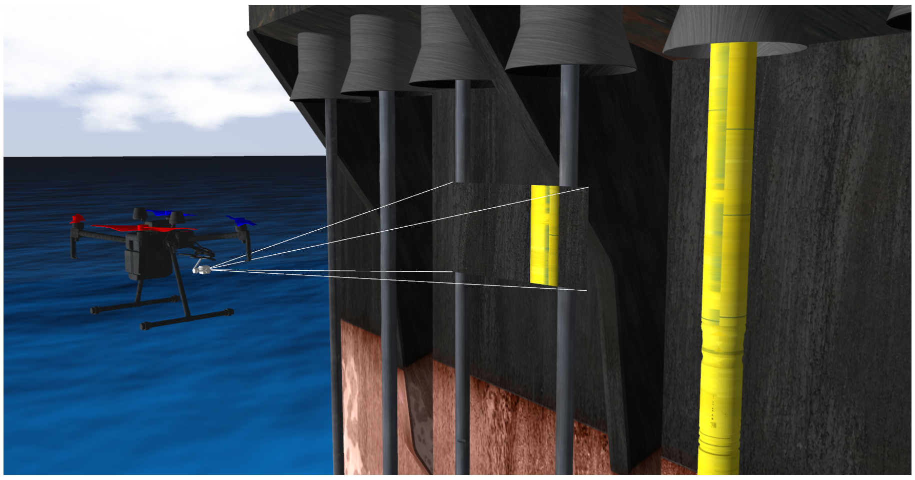
Figure 42 – Matrice 200 v2 at Gazebo simulation.



Source – Project VANT3D.

Some studies developed at VANT3D research project aimed the construction of a simulation environment to test camera configurations, trajectory of acquisition and algorithms to control it autonomously. Figure 43 presents the environment created to Gazebo, a section from the risers balcony on a offshore oil-rig, using RPA model M200 with a Gimbal mechanism developed for acquisition tests.

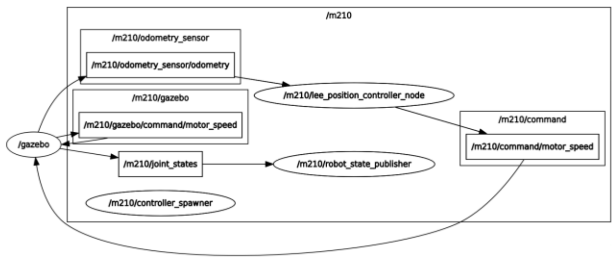
Figure 43 – VANT3D Gazebo environment.



Source – Project VANT3D.

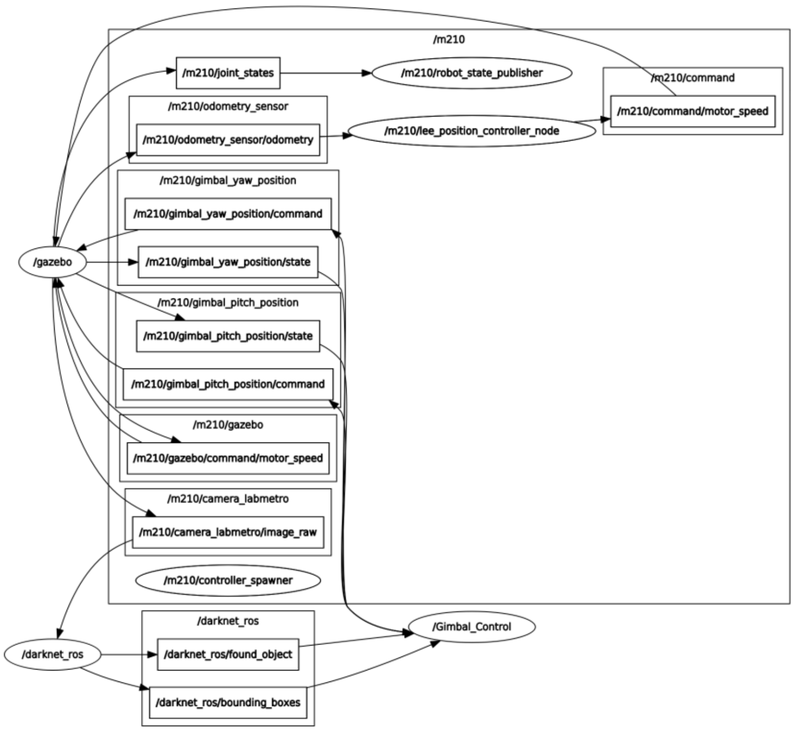
As package *rotors_simulation* is the base of the environment, its quadrotors position controllers works as a receiving data from odometry sensors to maintain flying. Figure 44 present the ROS graphs of nodes and topics necessary to simulate the aircraft, where its presented the node */gazebo* opening the topics necessary to make the aircraft fly.

Figure 44 – ROS graph of RPA M200 on Gazebo environment.


/m210
/m210/odometry_sensor
/m210/odometry_sensor/odometry
/m210/gazebo
/m210/gazebo/command/motor_speed
/m210/lee_position_controller_node
/m210/command
/m210/command/motor_speed
/gazebo
/m210/joint_states
/m210/robot_state_publisher
/m210/controller_spawner

Launching the *darknet_ros* node and the *control_gimbal* developed for this project, the total graph of nodes and topics is shown in Figure 45. As a simulation have some slight differences than a real life, we do not have the gimbal topic provided as DJI documentation, to contour this problem it was developed a generic gimbal mechanism where it is possible to control each joint individually. This problem results in subscribe to three different topics to acquire the actual state of gimbal joints and publish separately the new desired values, different than presented on section 4.2.1.1 where it has a topic structured for all joints.

Figure 45 – ROS Graph with RPA, darknet_ros and gimbal_control on Gazebo environment.


/m210
/m210/joint_states
/m210/robot_state_publisher
/m210/command
/m210/command/motor_speed
/m210/odometry_sensor
/m210/odometry_sensor/odometry
/m210/lee_position_controller_node
/m210/gimbal_yaw_position
/m210/gimbal_yaw_position/command
/gazebo
/m210/gimbal_yaw_position/state
/m210/gimbal_pitch_position
/m210/gimbal_pitch_position/state
/m210/gimbal_pitch_position/command
/m210/gazebo
/m210/gazebo/command/motor_speed
/m210/camera_labmetro
/m210/camera_labmetro/image_raw
/m210/controller_spawner
/darknet_ros
/Gimbal_Control
/darknet_ros
/darknet_ros/found_object
/darknet_ros/bounding_boxes

Defined all components used to create this track control is necessary to do some experiments to analyze results and check the constrains of it. Experiments were started on simulation environments to refine and tune the algorithm before introduce on real components, reducing the probability of damage equipment.

5 RESULTS

This chapter brings the experiments made to validate the Gimbal Control System using the Gazebo simulation environment created to perform different equipment configurations and syntonize it. It also brings an experimental test made on practical environment and its slightly differences and results.

5.1 GAZEBO ENVIRONMENT

The Gazebo simulation environment is processed on a desktop computer with specifications presented on Table 6. It has a strong GPU to process all graphic needed from simulation environment and neural networks from computer vision used. From its GPU process also resulted in a good frame rate to work detect objects using YOLO.

Table 6 – Specifications of computer used at simulation environmentt.

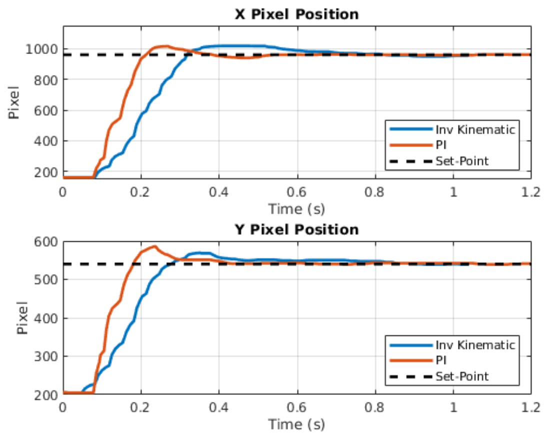
Desktop Computer	
Processor	AMD Ryzem 5 2600 64-Bit 3400 MHz 6 CPU cores and 12 threads
MotherBoard	Asus Prime B350M-K
GPU	NVIDIA GTX 1080 8 GB DDR5X @ 10 Gbps
Memory	32 GB DDR4
Storage	SSD 240 GB 970 EVO M2

5.1.1 Tune Controllers

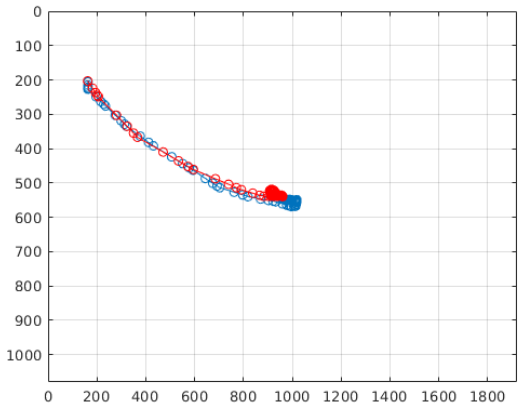
The controllers were tuned with a low-pass filter to reduce variance of position measurements from *darknet_ros*, YOLO node. The Gimbal inverse kinematic controller also uses one low-pass filter on angle's equations from inverse kinematic mathematics to minimize major variations. The PI controllers are tuned as referred on section 3.1.2, as fast as possible without overshoot, resulting in equation (23). Occasionally both controllers had the same setting time and slightly overshoot signal. Figure 46a present the step response of each controller axis, and Figure 46b shown the pixel position of the object in the image resolution for both controllers techniques.

$$u(k) = 0.8\,[e(k) - 0.92\,e(k-1)] + u(k-1) \tag{23}$$

Figure 46 – Step response of controllers PI and Inverse Kinematic.



(a) Step response over time of controllers



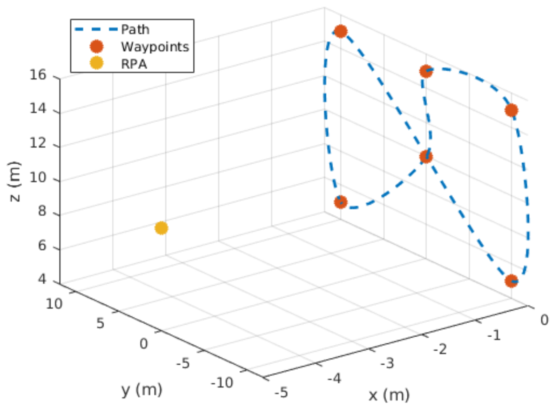
(b) Object position in image resolution

To develop tests under the tracking algorithm of a specific object, a scenario was created in Gazebo with different objects in moving in a predefined trajectory. Figure 47 present the pathway of objects with its way-points and the position of RPA, static at $[-5\ 0\ 10]^T$ meters, while performing the tracking. Table 7 present the position and sequence of the object path. Object has a constant velocity of 2 m/s moving from way-point zero to three, from three to four the velocity changes to 3 m/s and moving from four to seven the velocity is 6 m/s.

Table 7 – Object way-point path.

Way-point	x (m)	y (m)	z (m)
1	0	0	10
2	0	10	5
3	0	10	15
4	0	-10	5
5	0	-10	15
6	0	0	15
7	0	0	10

Figure 47 – Object pathway.



For the record of data it is used the parameters expressed in Table 8, where is the resolution of a Zenmuse-Z30 and a angle of horizontal field of view similar to a 45 mm focal length. With the Zenmuse Z30 resolution the darknet process image detection with a frame rate around 30-35 FPS. As it is unknown the exact distance of object from aircraft and being a nominal distance between aircraft and risers balcony as five meters. The *dx* value is set smaller than a normal distance of objects to check the parameters robustness from controllers. Figure 48 present the experiment on simulation environment with YOLO detecting the desired object, it was used a aeroplane from pre trained data set, to initiate the tests over track algorithm.

Table 8 – Parameters of simulation environment.

Parameter	Value
Aircraft	M210 V2
Camera	Zenmuse-Z30
Resolution	1920 x 1080 (2 MP)
distance object	5 meters
CNN	YOLO V3
Rate	30 FPS
AoV_h	22.56^o
Parameter *dx*	1 meter
GSD	0.208 mm/px
Sample Time	50 milliseconds

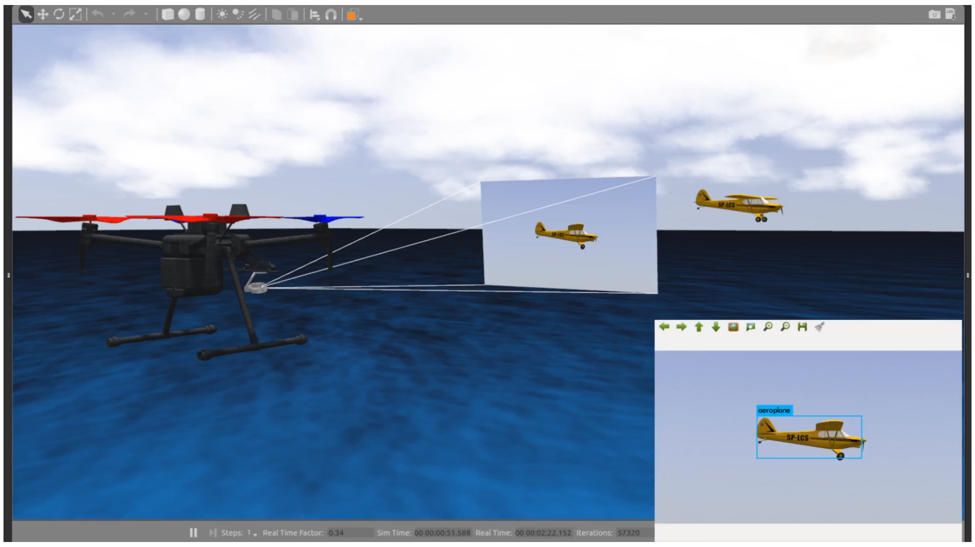
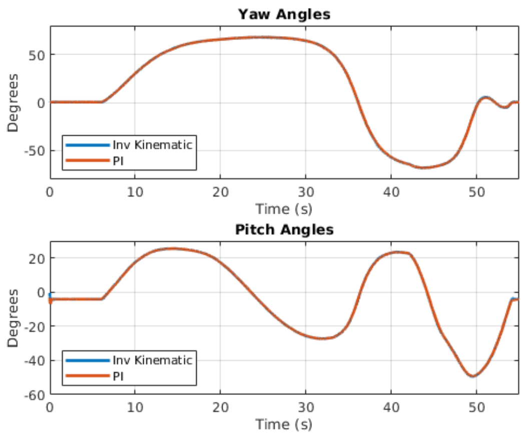


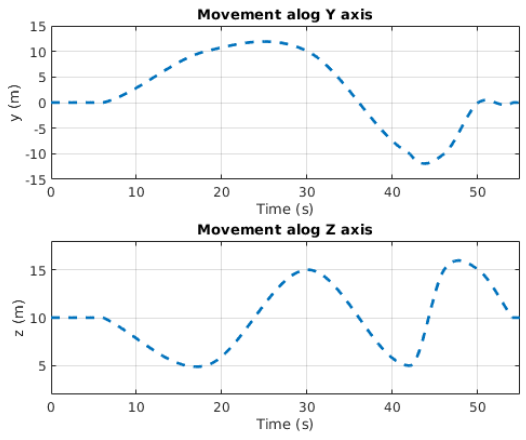
Figure 48 – Camera detection an aeroplane used as an object to be tracked

The movement of the angles yaw and pitch from Gimbal correspond, respectively, to the position variation of the world coordinate axis Y and Z and the image coordinate axis X_c and $-Y_c$. The variation of *pitch* angle is opposite to object variation resulted of direction of positive angles being downward. As Figure 49a present the angles of both controllers tracking the movement of ball shown in Figure 49b, it is noticeable that both controllers perform the same dynamic movement of the angles for tracking.

Figure 49 – Test of tracking object.



(a) Angle position along time.



(b) Objecting movement on plane along time.

5.1.2 Controller Comparison

After tuned the controllers to reach the reference in a small time minor overshoot and capable to track bring error signal to zero, Figures 50 and 51 presents the pixel position controllers, when tracking the object movement presented on Figure 47, respectively PI and Inverse Kinematic controllers. It is evident when increasing the speed of the object, the controller takes a long time to track with zero error, however even in speed upper than 6 m/s, still possible to keep the object on image field of view.

Figure 50 – PI controller.

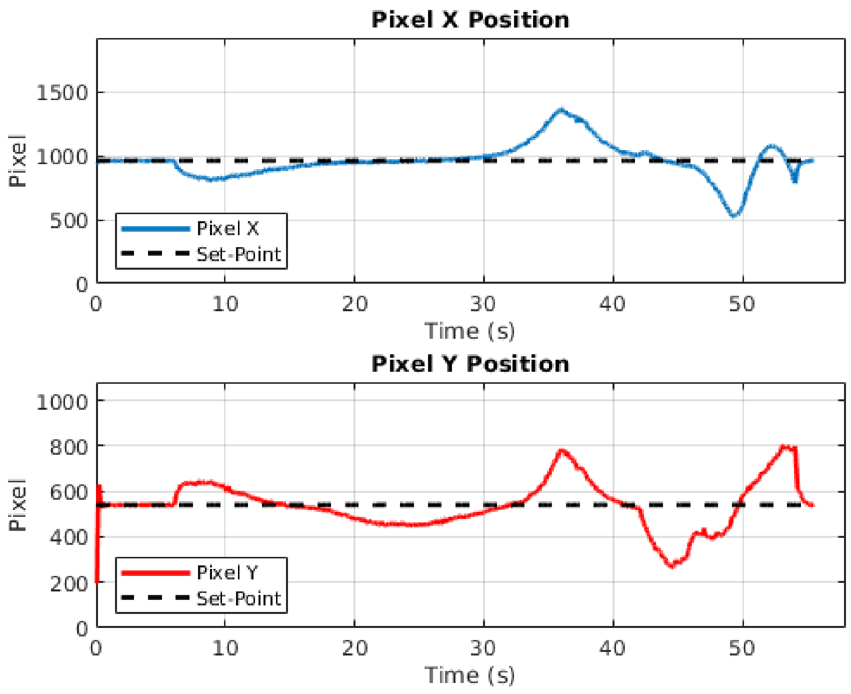
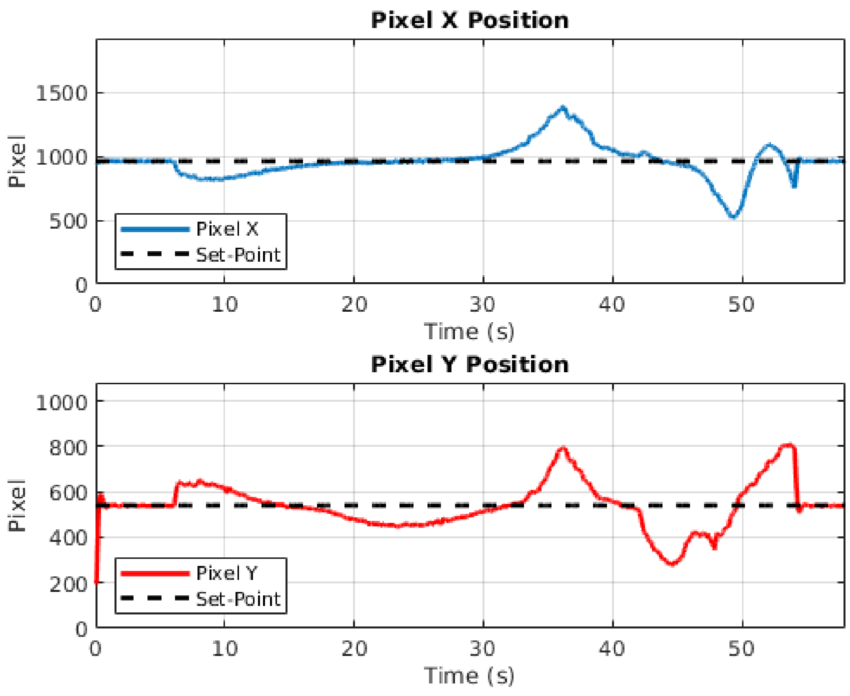
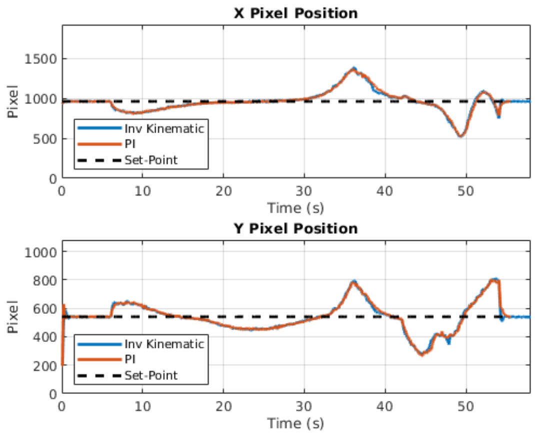


Figure 51 – Inverse Kinematic controller.

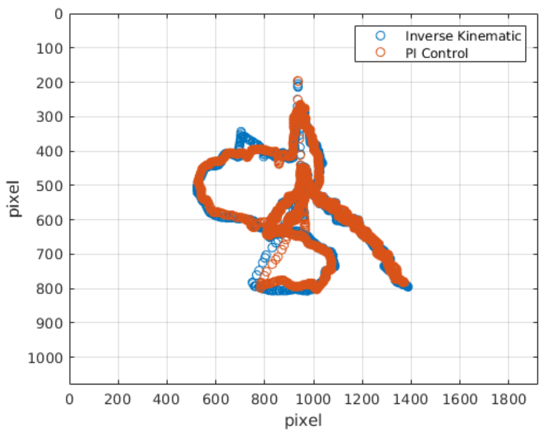


Comparing the results of controllers, Figure 52a, they have a similar dynamic and errors, even when the object increase the speed. Figure 52b shows the object position at camera's image resolution when the controllers are activated on experiments. They obtained a good result tracking the object from five meters away and the parameter (dx) as one meter, resulting a GSD smaller than the real value.

Figure 52 – Compare between PI and Inverse Kinematic controllers.



(a) Pixel position



(b) Object position at image.

To check the parameter robustness of controllers it has been made a few tests using the same object pathway described above but in different distances between object and RPA. The principal parameter of those controllers is the GSD resulted of knowing the distance of the object, the focal length (f) and the image resolution. Camera Zenmuse Z30 allows the system knows the optical zoom set at mobile SDK interface, Zenmuse X5S just has fixed lens zoom where is changed on land.

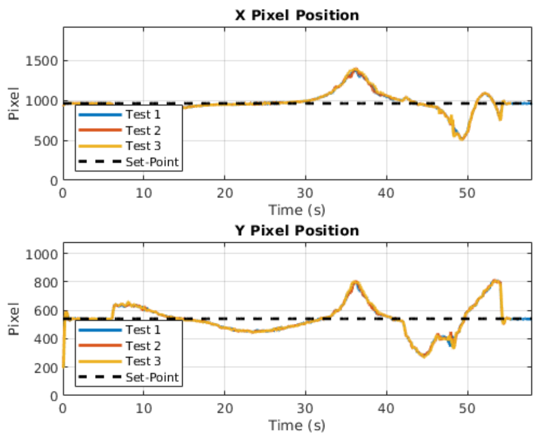
Three tests were made with different value of dx parameter, resulting in a different GSD presented on Table 9. GSD is a parameter that multiply the pixel error transforming the error variable in meters. Unfortunately, these tests were only possible on the inverse kinematic controller, the PI controller was tuned to the GSD parameter in order of 0.2 mm/px and becomes unstable when obtaining parameters greater than 0.5 mm/px. This can be explained from a transport delay, visible on Figure 46, turning the system unstable to gain variation when using a PI controller.

Table 9 – Parameter Tests.

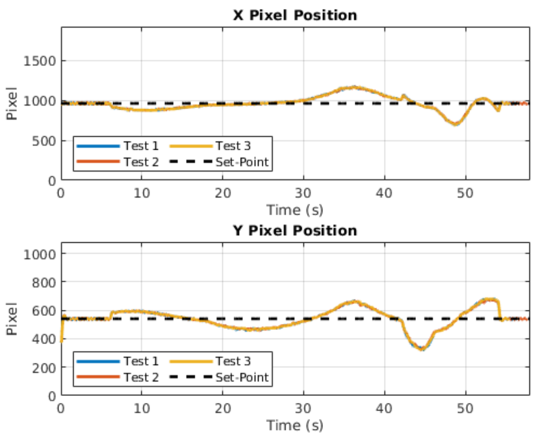
Test	Distance (dx)	H. AoV	H. FoV	GSD
1	1 m	22.56^o	0.3989 m	0.208 mm/px
2	5 m	22.56^o	1.9457 m	1.039 mm/px
3	10 m	22.56^o	3.9892 m	2.078 mm/px

After the tests have been developed, the results are shown in the Figure 53, where the robustness for these parameter variations is visible at different distances from the object, even with large GSD variations the controller maintains the same response dynamics.

Figure 53 – Robustness of dx parameter at Inverse Kinematic controller.



(a) Tests @ 5 meters.



(b) Tests @ 10 meters.

5.2 ON-BOARD COMPUTER

To validate the track system to the RPA it has been used an on-board computer different than the one specified previously on section 4.2 to not interfere different researches related to the scope of the VANT3D. After tuned the tracker controller the package generated is ready to add on workspace of Manifold 2G.

The computer tested was the on-board NVIDIA Jetson TX2, as Table 10 present the specifications. To process YOLO the most important component is Graphics processing unit (GPU) and a good RAM memory capable to do all mathematics from trained CNNs, fortunately Manifold 2G has been developed over Jetson TX2 platform and all GPU system and RAM memory are the same.

Table 10 – Jetson TX 2 Specifications.

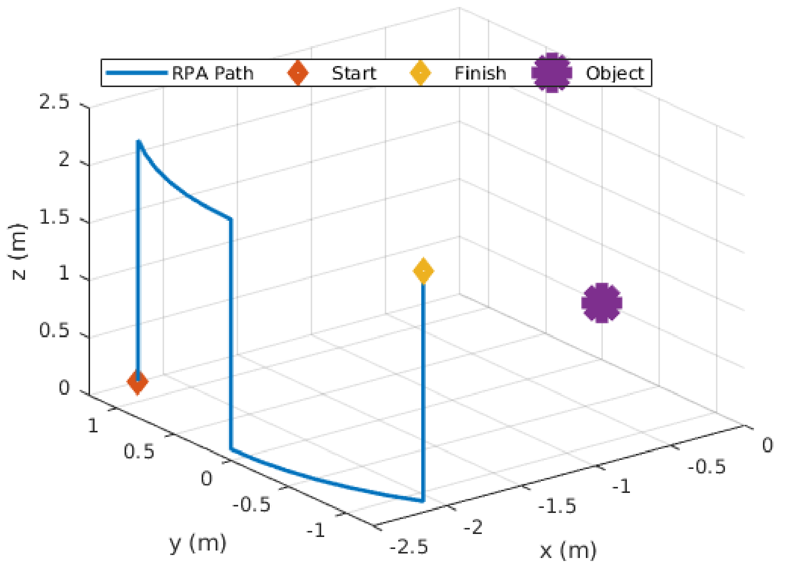
NVIDIA Jetson TX2	
Processor	Dual-Core NVIDIA Denver 1.5 64-Bit CPU and Quad-Core ARM® Cortex®-A57 MPCore
GPU	256-core NVIDIA Pascal^{TM} GPU architecture with 256 NVIDIA CUDA cores
Memory	8GB 128-bit LPDDR4, 1866 MHz
Network	10/100/1000 BASE-T Ethernet, WLAN
Storage	32GB eMMC 5.1
I/O	CAN, UART, I2C, SPI
Power	7.5 - 15 W

Source – https://developer.nvidia.com/embedded/develop/hardware. Accessed: July 15th

Different than tests made on Gazebo environment, section 5.1, were the object was moving on a specific trajectory, this tests were done moving the drone manually around a static object, simulating a photogrammetry pathway. Recording the position and orientation of RPAS by ROS topics when initialized the aircraft via ROS from *sdk.launch* node.

Tests have been moving manually the aircraft in a specify pathway described in Figure 54 where it simulates a semi-circle path with 2.5 meters radius and three positions to change the height of aircraft. This points are in -30, 0 and 30 degrees at 2.5 meters distance from object.

Figure 54 – Aircraft semi-circle pathway to tests for inverse kinematic and PI approaches.



The parameters of experiments are presented on Table 11, where mostly are similar to Gazebo simulation experiments. Most significant changes are camera resolution that decrease to 1280 x 720 pixels resulted from default configurations of image acquisition at DJI ROS API, this caused a drop in frame rate to 5 FPS due to the difference of GPU at computer used to simulate and on-board devices. To get a better FPS it has been reduced the image's input size at CNN of YOLO to 192x192 pixel, originally 416x416, to get achieve a FPS capable to maintain real time track. Unfortunately this change does not allow precisely detection for small objects.

Although this differences, the algorithm used on simulation is the same as used on real life, just fitting to ROS topic's name from DJI SDK and the publisher type of message to Gimbal's mechanism.

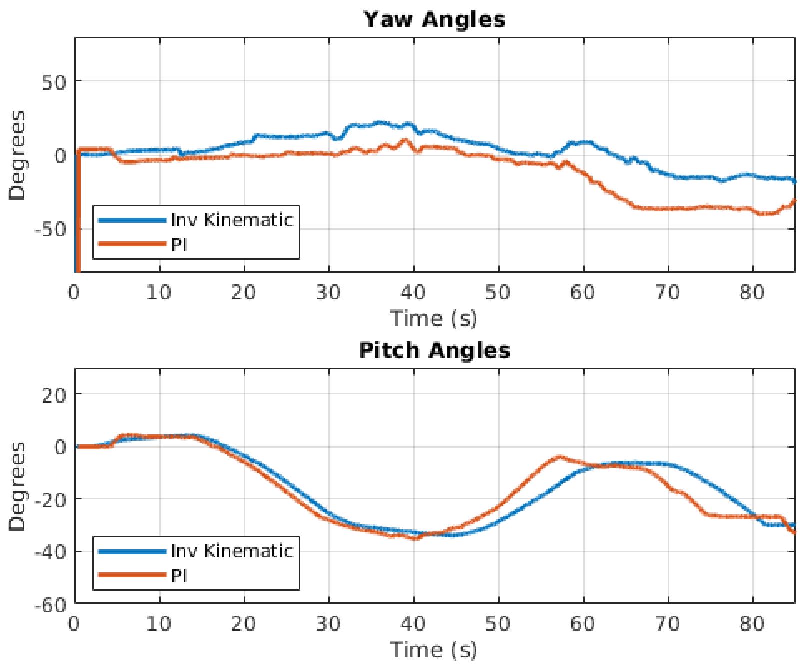
Table 11 – Parameters of practical tests.

Parameter	Value
Aircraft	M210 V2
Camera	Zenmuse-Z30
Resolution	1280 x 720 (1 MP)
distance object	3 meters
CNN	YOLO V3
Rate	15 FPS
AoV_h	22.56^o
Parameter dx	1 meter
GSD	0.208 mm/px
Sample Time	50 milliseconds

5.2.1 Controller Comparison

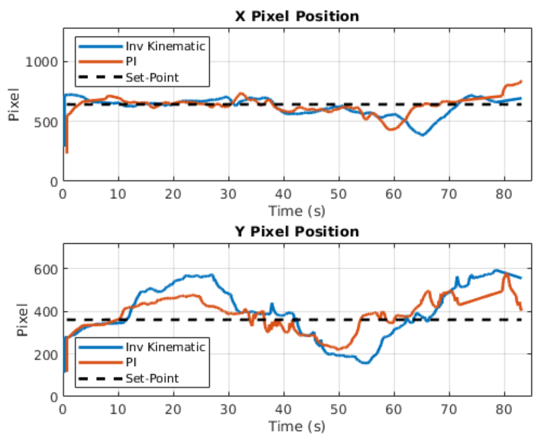
Tests carried out with the movement of the aircraft around a fixed object as shown in Figure 54, the results of both track controller referred on gimbal's angles are presented in Figure 55. Both controllers have similar dynamics to maintain the object centered at image resolution, inverse kinematic approach has a smoother control, different than PI approach where it is possible to set the dynamic of controller to reject disturbances faster. The angles yaw and pitch have similar performance, at pitch angle the PI approach has an earlier response resulting better pixel position control. Yaw angles have slightly differences caused of variation of RPA yaw angle while performing tests.

Figure 55 – Angle Position along time.

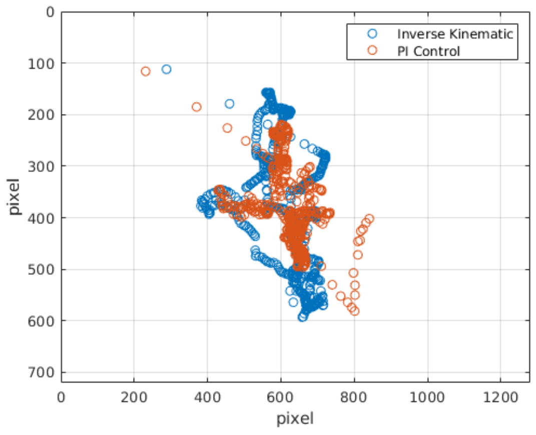


Comparing the object position over the image resolution, Figure 56b, controllers maintain the object under the center region, varying height image position due to movements of semi-circle path, also is possible to notice that the PI controller region is smaller. Comparing both controllers, Figure 56a, is clear visible the faster dynamic of PI over disturbances rejections as RPA movements. Inverse Kinematic controller has a smoother response but allowing major variations of position before control, oscillating around desire reference. As inverse kinematic control how control requires computational effort to compute the trigonometry equations allows the object to move away from the center of the image before determine new angles.

Figure 56 – Compare between PI and Inverse Kinematic controllers.



(a) Pixel position.



(b) Object position at image.

However, it was possible to track a detected object through pre-trained neural networks for YOLO, Figure 57 and 58 present the track result while the RPA is moving on its pathway. The gimbal mechanism were able to maintain the object tracked even while aircraft is varying its position.

Figure 57 – RPA starting a vertical movement while tracking the object

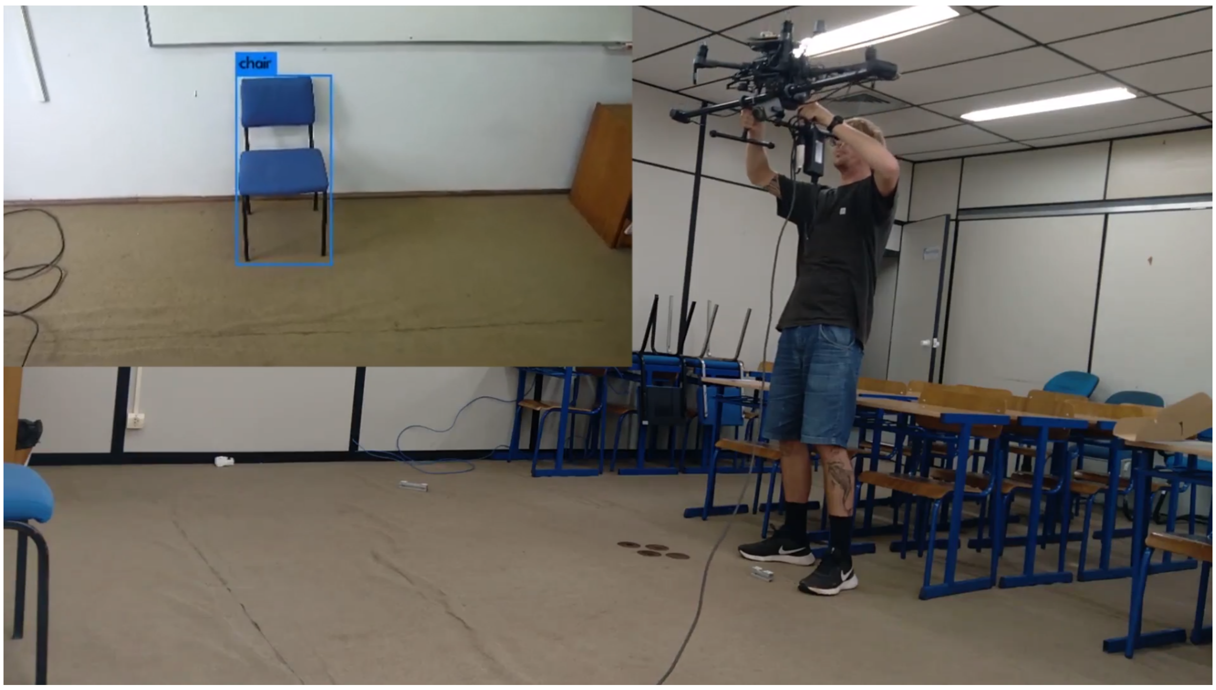
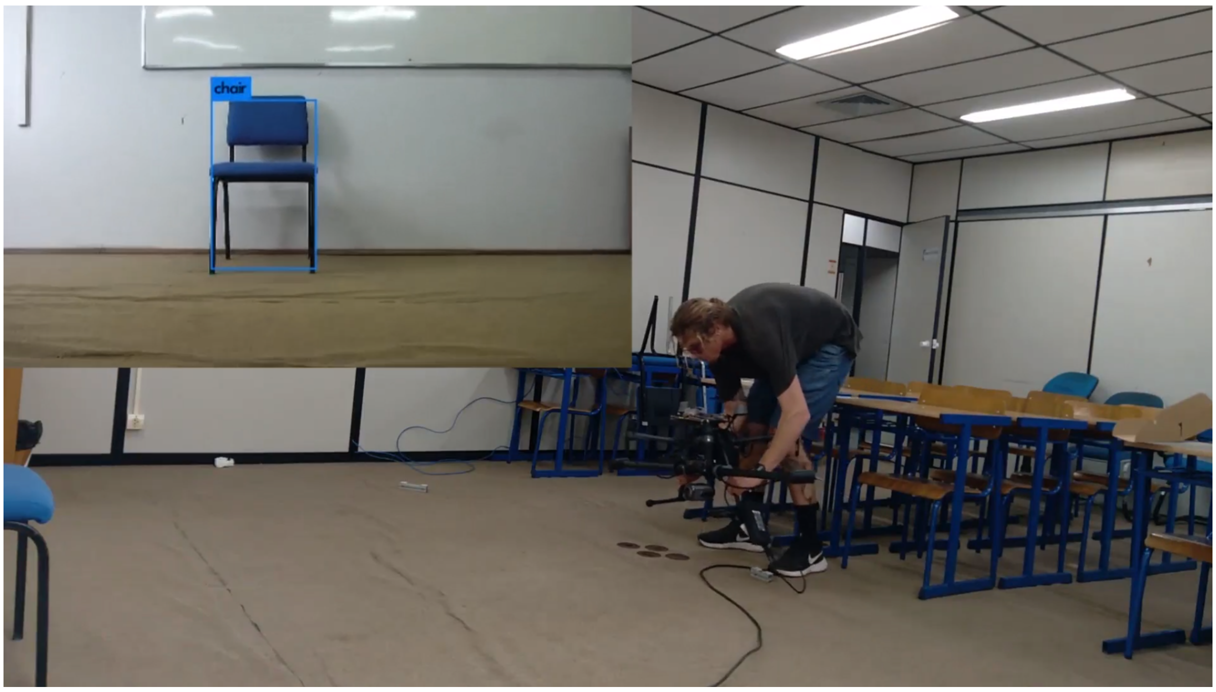


Figure 58 – RPA finishing the vertical movement while tracking the object



The controllers are also presented separately at Figure 59 and Figure 60. X pixel axis are mostly of the time rejecting variation of aircraft, but Y axis suffered more to reject disturbances of a semi-circle path.

Figure 59 – Pixel position of Inverse Kinematic controller.

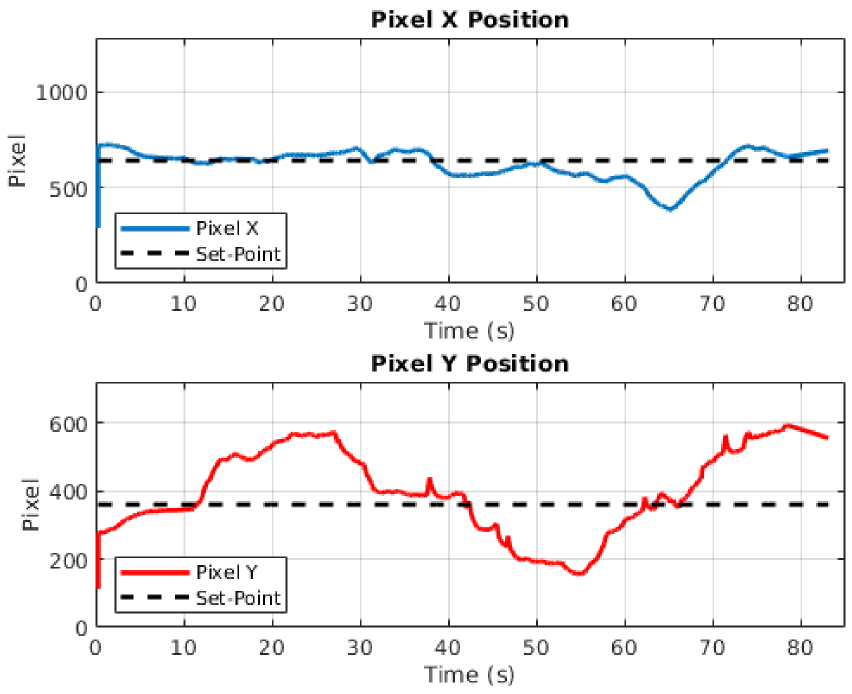
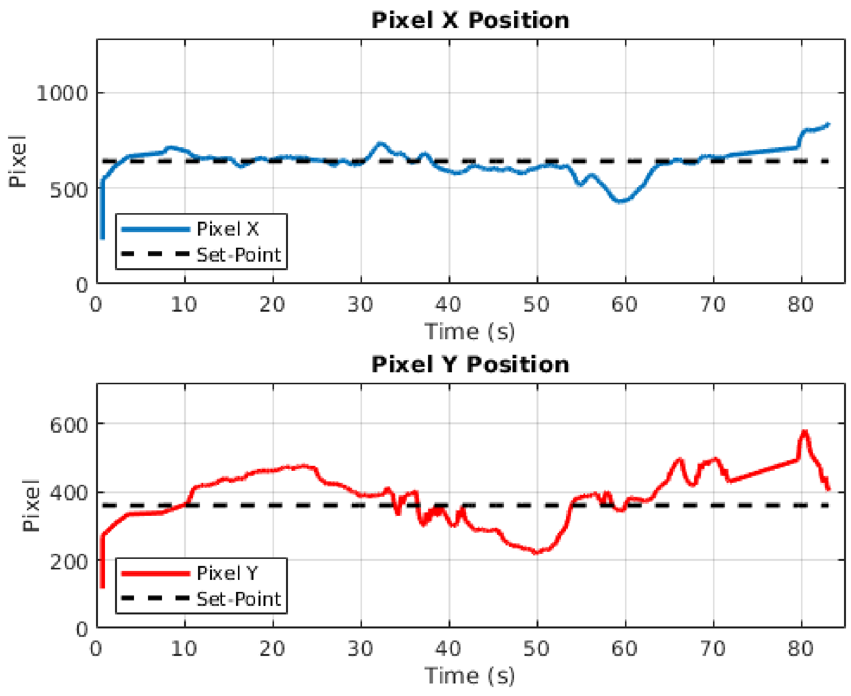


Figure 60 – Pixel position of PI controller.



Through the results obtained from simulation and practical tests, the algorithm is validated and able to track an object while the aircraft is moving around the object to acquire images for inspection and posterior photogrammetry. At simulation environment was possible to refine and test the algorithm and reduce the necessity of test and fix problem embed using an on-board computer coupled on aircraft. Unfortunately due to reduction of input size of YOLO CNN the algorithm is capable to detect mainly large objects as the desired one, a riser.

6 CONCLUSION AND PROSPECTS

This work presented the process of an object track algorithm using ROS system to exchange messages from peripherals as sensors and motors on a Gimbal mechanism, used to freely move a camera coupled on a mobile robot, where at this work was used a drone, to acquire images from different points of view, rejecting some robot movement disturbances over the camera. Was developed an algorithm using data from ROS packages of image process using YOLO to determine new angle's position of revolute joints of Gimbal to maintain detected object centered at camera's field of view.

The algorithm proved to be functional in simulation environment, with good results controlling angles based on the pixel position of object detected at camera's image. From tests were possible to validate the theory when processed using a strong GPU and CPU, making it possible to track even in velocities higher than 20 km/h between object and RPA.

As practical tests were able to communicate between aircraft and sensors to onboard computer via ROS, controlling gimbal's position therefore processing YOLO on main camera attached on Gimbal mechanism over Matrice v2. The control of gimbal's position is based on object's pixel position from camera's field of view to acquire images and posterior reconstruct over photogrammetry process.

The results presented on document were satisfactory, even in simulation as in practical environment. Using a pre-trained dataset of CNNs from YOLO capable to detect conventional objects and using camera Zenmuse-Z30 from DJI the algorithm was able to maintain the object in camera's field of view, but it was necessary to reduce the input image size to reach a good frame rate able to track in execution time. Unfortunately, this reduction of input size made detection of small objects inaccurate.

As the goals of maintaining an object at the camera's field of view to acquire images for photogrammetry, rejecting disturbances resulted from movement of the aircraft controlled remotely from a pilot. In a risk zone where loose the RPA from eyes represent a high risk of accidents, this algorithm helps the pilot to control it without the necessity to check if the object is in a good position at camera's image.

For following works, a good point to be reached is the computational effort necessary to perform a real time track from small objects, increasing the input image size without loosing performance using a camera with bigger resolution as a DJI Zenmuse X5S to detect risers from a pre trained data-set developed at VANT3D research project.

REFERENCES

ÅSTRÖM, Karl Johan; HÄGGLUND, Tore. **PID controllers: theory, design, and tuning**. v. 2. [S.l.: s.n.], 1995.

ATTAR, Refaat El. **Lecture notes on Z-Transform**. [S.l.]: lulu.com, 2006. P. 84. ISBN 141161979X.

BAI, Qiang; BAI, Yong. Flexible Pipe. In: SUBSEA Pipeline Design, Analysis, and Installation. [S.l.]: Elsevier, 2014. P. 559–578. ISBN 9780123868886. DOI: 10.1016/B978-0-12-386888-6.00024-9.

BALLARD, Dana H.; BROWN, Christopher M. **Computer Vision**. [S.l.]: Prentice Hall 1982, 1982. P. 539. ISBN 9780131653160.

BJELONIC, Marko. **YOLO ROS: Real-Time Object Detection for ROS**. [S.l.: s.n.], 2016. https://github.com/leggedrobotics/darknet_ros.

BRADSKI, Garry; KAEHLER, Adrian. **Learning OpenCV**. Ed. by Mike Loukides. [S.l.]: O'Reilly, 2009. v. 16, p. 543. ISBN 9780596516130. DOI: 10.1109/mra.2009.933612.

DJI. **OnBoard SDK Documentation**. [S.l.: s.n.], 2019. Available from: https://developer.dji.com/onboard-sdk/documentation. Visited on:

FURRER, Fadri; BURRI, Michael; ACHTELIK, Markus; SIEGWART, Roland. Robot Operating System (ROS): The Complete Reference (Volume 1). In: ed. by Anis Koubaa. Cham: Springer International Publishing, 2016. RotorS—A Modular Gazebo MAV Simulator Framework, p. 595–625. ISBN 978-3-319-26054-9. DOI: 10.1007/978-3-319-26054-9_23. Available from: http://dx.doi.org/10.1007/978-3-319-26054-9_23.

GIRSHICK, Ross. Fast R-CNN. **Proc. IEEE Int. Conf. Comput. Vis.**, 2015 International Conference on Computer Vision, ICCV 2015, p. 1440–1448, 2015. ISSN 15505499. DOI: 10.1109/ICCV.2015.169. arXiv: 1504.08083.

GIRSHICK, Ross; DONAHUE, Jeff; DARRELL, Trevor; MALIK, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. **Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.**, p. 580–587, 2014. ISSN 10636919. DOI: 10.1109/CVPR.2014.81. arXiv: 1311.2524.

GURDAN, Daniel; STUMPF, Jan; DOTH, Klaus-Michael; ACHTELIK, Michael. **Ascending Technologies**. [S.l.: s.n.], 2007. Available from: www.asctec.de/. Visited on: 5 May 2020.

HOFFMANN, Gabriel M.; HUANG, Haomiao; WASLANDER, Steven L.; TOMLIN, Claire J. Quadrotor helicopter flight dynamics and control: Theory and experiment. **Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference 2007**, v. 2, May, p. 1670–1689, 2007. DOI: 10.2514/6.2007-6461.

HUANG, T S. Computer Vision: Evolution and Promise. **19th Cern Sch. Comput.**, p. 21–25, 1997. DOI: 10.5170/CERN-1996-008.21.

HUTTENLOCHER, Daniel. Computer vision. **Comput. Sci. Handbook, Second Ed.**, p. 43–1–43–23, 2004. ISSN 1340-5551. DOI: 10.4324/9780429042522-10.

INTEL®. **OpenCV Library**. [S.l.: s.n.], 2000. Available from: https://docs.opencv.org/. Visited on:

JAKOBSEN, Ole C.; JOHNSON, Eric N. Control architecture for a UAV-mounted pan/tilt/roll camera gimbal. **Collect. Tech. Pap. - InfoTech Aerosp. Adv. Contemp. Aerosp. Technol. Their Integr.**, v. 4, September 2005, p. 2170–2179, 2005. DOI: 10.2514/6.2005-7145.

JORDAN, Sophie; MOORE, Julian; HOVET, Sierra; BOX, John; PERRY, Jason; KIRSCHE, Kevin; LEWIS, Dexter; TSE, Zion Tsz Ho. State-of-the-art technologies for UAV inspections. **IET Radar, Sonar & Navigation**, v. 12, n. 2, p. 151–164, 2018. ISSN 1751-8784. DOI: 10.1049/iet-rsn.2017.0251.

JOSEPH, Lentin. **Mastering ROS for Robotics Programming**. [S.l.: s.n.], 2015. v. 22, p. 137–141. ISBN 9781783551798. DOI: 10.1007/s13398-014-0173-7.2. arXiv: arXiv:1011.1669v3.

JOSEPH, Lentin. **Robot Operating System (ROS) for Absolute Beginners**. 1. ed. [S.l.]: Apress, Berkeley, CA, 2018. P. 282. ISBN 978-1-4842-3405-1. DOI: 10.1007/978-1-4842-3405-1.

KEMAJOU, Vanessa Ndonhong; BAO, Anqi; GERMAIN, Olivier. Wellbore schematics to structured data using artificial intelligence tools. **Proc. Annu. Offshore Technol. Conf.**, 2019-May, May, 2019. ISSN 01603663. DOI: `10.4043/29490-ms`.

KLETTE, Reinhard. **Concise Computer Vision**. [S.l.]: Springer International Publishing, 2014. ISBN 978-1-4471-6319-0. DOI: `10.1007/978-1-4471-6320-6`. Available from: `http://link.springer.com/10.1007/978-1-4471-6320-6`.

KRIDSADA, L.; CHATCHAI, La.; MANOP, C.; THANA, S. Sustainability Through the Use of Unmanned Aerial Vehicle for Aerial Plant Inspection. **Offshore Technology Conference Asia**, 2016. DOI: `10.4043/26576-MS`.

KULKARNI, Sudhanwa; MOHANTY, Akash. Analysis of the Gimbaled Platform for the Three Degrees of Freedom Using Differential Equations. May, p. 31–34, 2013.

LATHI, B. P. **Linear Systems and Signals**. 2nd. USA: Oxford University Press, Inc., 2009. ISBN 0195392566.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015. ISSN 14764687. DOI: `10.1038/nature14539`.

LIENHART, Rainer; MAYDT, Jochen. An Extended Set of Haar-like Features for Rapid Object Detection, p. 900–903, 2002.

LINDER, Wilfried. **Digital Photogrammetry**. 4. ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016. P. 209. ISBN 978-3-662-50462-8. DOI: `10.1007/978-3-662-50463-5`. Available from: `http://link.springer.com/10.1007/978-3-662-50463-5`.

LINDGREEN, A; LINDGREEN, A. Corruption and unethical behavior: report on a set of Danish guidelines. **J. Bus. Ethics**, v. 51, n. 1, p. 31–39, 2004. ISSN 0167-4544. DOI: `10.1023/B`.

LOWE, David G. Distinctive Image Features from Scale-Invariant Keypoints. **Int. J. Comput. Vis.**, v. 60, n. 2, p. 91–110, 2004. ISSN 1573-1405. DOI: `10.1023/B:VISI.0000029664.99615.94`. Available from: `https://doi.org/10.1023/B:VISI.0000029664.99615.94`.

LUHMANN, Thomas; ROBSON, Stuart; KYLE, Stephen; BOEHM, Jan. **Close-Range Photogrammetry and 3D Imaging**. Ed. by Thomas Luhmann, Stuart Robson, Stephen Kyle and Jan Boehm. 2. ed. Germany: De Gruyter, 2014. P. 708. ISBN 9783110302691.

MARINHO, Mauro G.; SANTOS, Joilson M. dos; CARNEVAL, Ricardo de O. Integrity Assessment and Repair Techniques of Flexible Risers. In: VOLUME 4: Terry Jones Pipeline Technology; Ocean Space Utilization; CFD and VIV Symposium. [S.l.]: ASME, 2006. P. 253–260. DOI: 10.1115/OMAE2006-92467.

MERCURI, S M; FISICARO, A; TRAMONTANO, V. UAV the Impact & Influence in the O&G. **Offshore Mediterranean Conference and Exhibition**, p. 1–8, 2017.

MICHAEL A. JOHNSON MOHAMMAD H. **Michael A. Johnson Mohammad H. Moradi PID Control New Identification and Design Methods**. [S.l.: s.n.], 2007. P. 863–866. ISBN 9781852337025.

MOREL, Jm; YU, G. On the consistency of the SIFT Method. **Prepr. C.**, p. 1–17, 2008. Available from: http://scholar.google.com/scholar?hl=en%7B%5C&%7DbtnG=Search%7B%5C&%7Dq=intitle:On+the+consistency+of+the+SIFT+Method%7B%5C#%7D1.

MOSQUEIRA, Cristiano. **FINITE ELEMENT ANALYSIS OF FLEXIBLE PIPES: BENDING COMBINED WITH TENSILE LOAD**. Oct. 2017. PhD thesis. DOI: 10.13140/RG.2.2.20576.17927.

NELSON, Robert C. **Flight Stability**. [S.l.: s.n.], 1998. P. 1–452. ISBN 0070462739.

NOORDIN, Aminurrashid; BASRI, Mohd Ariffanan Mohd; MOHAMED, Zaharuddin; ABIDIN, Amar Faiz Zainal. Modelling and PSO fine-tuned PID control of quadrotor UAV. **International Journal on Advanced Science, Engineering and Information Technology**, v. 7, n. 4, p. 1367–1373, 2017. ISSN 24606952. DOI: 10.18517/ijaseit.7.4.3141.

NOROUZI GHAZBI, S.; AGHLI, Y.; ALIMOHAMMADI, M.; AKBARI, A. A. Quadrotors unmanned aerial vehicles: A review. **International Journal on Smart Sensing and Intelligent Systems**, v. 9, n. 1, p. 309–333, 2016. ISSN 11785608. DOI: 10.21307/ijssis-2017-872.

OPEN SOURCE ROBOTICS FOUNDATION. **Gazebo Simulator**. [S.l.: s.n.], 2014. Available from: `http://gazebosim.org/`. Visited on: 1 May 2020.

OPEN SOURCE ROBOTICS FOUNDATION. **Wiki ROS**. [S.l.: s.n.], 2010. Available from: `http://wiki.ros.org`.

PETALE, M. D. **Z-Transform: Theory & Solved Examples**. 4. ed. [S.l.]: Amazon Digital Services LLC, 2018. P. 92. ISBN 1980778043, 9781980778042.

POUNDS, Paul; MAHONY, Robert; CORKE, Peter. Modelling and control of a quad-rotor robot. **Proceedings of the 2006 Australasian Conference on Robotics and Automation, ACRA 2006**, 2006.

PRABHA, M L; THOTTUNGAL, Rani. Modeling and simulation of X-quadcopter control. **International Journal for Research in Applied Science & Engineering Technology (IJRASET)**, v. 4, n. 4, p. 282–287, 2016. Available from: `www.ijraset.com`.

QUIGLEY, Morgan; GERKEY, Brian; SMART, William D. **Programming Robots with ROS A Practical Introduction to the Robot Operating System**. [S.l.: s.n.], 2015. v. 53, p. 559. ISBN 9788578110796. DOI: `10.1017/CBO9781107415324.004`. arXiv: `arXiv:1011.1669v3`.

RAJESH, R. J.; KAVITHA, P. Camera gimbal stabilization using conventional PID controller and evolutionary algorithms. **IEEE Int. Conf. Comput. Commun. Control. IC4 2015**, November 2018, 2016. DOI: `10.1109/IC4.2015.7375580`.

REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross; FARHADI, Ali. You only look once: Unified, real-time object detection. **Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.**, 2016-Decem, p. 779–788, 2016. ISSN 10636919. DOI: `10.1109/CVPR.2016.91`.

REIZENSTEIN, Axel. **Position and Trajectory Control of a Quadcopter Using PID and LQ Controllers**. 2017. S. 1–452. PhD thesis – Linköping University. ISBN 9780552157629.

REN, Shaoqing; HE, Kaiming; GIRSHICK, Ross; SUN, Jian. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. **IEEE Trans. Pattern**

Anal. Mach. Intell., v. 39, n. 6, p. 1137–1149, 2017. ISSN 01628828. DOI: 10.1109/TPAMI.2016.2577031. arXiv: 1506.01497.

SHAKHATREH, Hazim; KHREISHAH, Abdallah; KHALIL, Issa. Indoor Mobile Coverage Problem Using UAVs. **IEEE Systems Journal**, v. 12, n. 4, p. 3837–3848, 2018a. ISSN 19379234. DOI: 10.1109/JSYST.2018.2824802. arXiv: 1705.09771. Available from: https://ieeexplore.ieee.org/abstract/document/8352733.

SHAKHATREH, Hazim; SAWALMEH, Ahmad; AL-FUQAHA, Ala; DOU, Zuochao; ALMAITA, Eyad; KHALIL, Issa; OTHMAN, Noor Shamsiah; KHREISHAH, Abdallah; GUIZANI, Mohsen. Unmanned Aerial Vehicles: A Survey on Civil Applications and Key Research Challenges. **IEE Access**, p. 1–58, 2018b. arXiv: 1805.00881.

SHARIFI, Mostafa; CHEN, Xiao Qi; PRETTY, Christopher; CLUCAS, Don; CABON-LUNEL, Erwan. Modelling and simulation of a non-holonomic omnidirectional mobile robot for offline programming and system performance analysis. **Simulation Modelling Practice and Theory**, v. 87, May 2017, p. 155–169, 2018. ISSN 1569190X. DOI: 10.1016/j.simpat.2018.06.005.

SICILIANO, Bruno; SCIAVICCO, Lorenzo; VILLANI, Luigi; ORIOLO, Giuseppe. **Robotics: Modeling, Planning, and Control**. [S.l.]: Springer, 2009. v. 16, p. 618. ISBN 9781846286414. DOI: 10.1109/MRA.2009.934833.

SRIVANTAVA, Prashant; PETROPOULOS, George; KERR, Y. H. **Satellite Soil Moisture Retrieval**. 1. ed. [S.l.]: Elsevier, 2016. P. 440. ISBN 9780128033890.

TERWILLIGER, Brent; ISON, David C; ROBBINS, John; VINCENZI, Dennis. **Small Unmanned Aircraft Systems Guide: Exploring Designs, Operations, Regulations, and Economics**. [S.l.]: Aviation Supplies & Academics, Inc., 2017. ISBN 161954394X.

TIIMUS, K.; TAMRE, M. Camera gimbal control system for unmanned platforms. **Proc. Int. Conf. DAAAM Balt.**, p. 202–207, 2010. ISSN 23466138.

U. RAUTENBERG, M. Wiggenhagen. Abnahme und Überwachung photogrammetrischer Messsysteme nach VDI-2634. **Photogrammetrie Fernerkundung Geoinformation**, p. 117–124, 2002.

WANG, Chunguang; SHANKAR, Krishnakumar; MOROZOV, Evgeny V. Tailored design of top-tensioned composite risers for deep-water applications using three different approaches. **Advances in Mechanical Engineering**, v. 9, n. 1, p. 1–18, 2016. ISSN 16878140. DOI: `10.1177/1687814016684271`.

YOONSEOK PYO HANCHEOL CHO, Leon Jung. **ROS Robot Programming**. 1. ed. [S.l.]: ROBOTIS Co.,Ltd, 2017. P. 487. ISBN 9791196230715. DOI: `10.1109/PROC.1983.12681`.

ZHANG, Xiaodong; LI, Xiaoli; WANG, Kang; LU, Yanjun. A survey of modelling and identification of quadrotor robot. **Abstract and Applied Analysis**, v. 2014, 2014. ISSN 16870409. DOI: `10.1155/2014/320526`.

ZHANG, Xin; HAN, Liangxiu; HAN, Lianghao; ZHU, Liang. How well do deep learning-based methods for land cover classification and object detection perform on high resolution remote sensing imagery? **Remote Sens.**, v. 12, n. 3, p. 1–29, 2020. ISSN 20724292. DOI: `10.3390/rs12030417`.

ZHAO, Zhong Qiu; ZHENG, Peng; XU, Shou Tao; WU, Xindong. Object Detection with Deep Learning: A Review. **IEEE Trans. Neural Networks Learn. Syst.**, v. 30, n. 11, p. 3212–3232, 2019. ISSN 21622388. DOI: `10.1109/TNNLS.2018.2876865`. arXiv: `1807.05511`.

ZHENG, Lucy; TIDROW, Meimei. Analyses of infrared focal plane array figure of merit and its impact on sensor system trades. **Infrared Physics & Technology - INFRARED PHYS TECHNOL**, v. 52, p. 408–411, Nov. 2009. DOI: `10.1016/j.infrared.2009.08.001`.