CENTRO DE INVESTIGACIONES
EN ÓPTICA, A.C.

VISION-BASED OBSTACLE AVOIDANCE FOR QUADROTOR NAVIGATION

MASTER IN OPTOMECHATRONICS

Advisor:

PhD. Gerardo Ramón Flores Colunga

Student:

Sergio Marcelino Trejo Fuentes

August 2019

León, Guanajuato, México

Contents

Acknowledgments

I would like to thank to the most important person in my life, Ruth, for encouraging me to start this goal and cheering me up during many challenging times. I express thanks to my parents Tere and Marcelino, who that always have been there for supporting me, to my brothers Uli, Jorge, Sandra and Lalo. Thanks to Mr. Gabriel and Celso for being my second family, specially to Mrs. Lety who was very proud of my achievements. I also thank to Dani for all the generous support and friendship provided.

I am very thankful with my advisor Gerardo Flores who guide me to achieve this objective, also thanks to Dr. Luis Valentin and Dr. Carlos Paredes for accepting the revision of my thesis and their valuable suggestions.

I would like to thank my friends Lau, Memo, Axel, Luis, Andres, Alex and Gesem for helping me so many times.

Last and not least, thanks to all the researchers and engineers from CIO who taught me so many wonderful scientific, professional and personal lessons.

Abstract

This thesis presents a vision-based methodology which makes use of a stereo camera rig and a one-dimension LiDAR to estimate free obstacle areas for quadcopter navigation. The presented approach fuses information provided by a depth map from a stereo camera rig, and the sensing distance of the 1D-LiDAR. Once the depth map is filtered with a Weighted Least Squares filter (WLS), the information is fused through a Kalman filter algorithm. To determine if there is a free space large enough for the quadcopter to pass through, this approach marks an area inside the disparity map by using the Kalman Filter output information. The whole process is implemented in an embedded computer Jetson TX2 and coded in the Robotic Operating System (ROS). Experiments demonstrate the effectiveness of this obstacle detection approach.

Publications

"Depth map estimation methodology for free obstacle navigation", **S. Trejo**, K. Martinez, G. Flores. Accepted article in the *International Conference on Unmanned Aircraft Systems 2019 (ICUAS'19)*.

CHAPTER 1

Introduction

Unmanned Aerial Vehicles (UAV) have been used in many military fields for years, like search and rescue operations, disaster relief, monitoring and tracking and border patrol [1]. In the last years UAVs have been gaining popularity on civil and commercial applications such as agriculture, energy, mining, construction, news media and film production. New regulations of the authorities of many countries are restricting the use of UAVs; among the first issues is the safety, it requires a sensor-based detection and avoidance system to be implemented soon [2]. This means that each UAV must have a sense and avoid system in order to have permission to flight. However these last applications demand smaller and agile UAVs to fly at low altitude including inside buildings or around them, exposing these devices to a number of static and dynamic obstacles. In this sense, artificial intelligence and automation is focusing in provide the UAVs a sense and avoid (SAA) technology, which is becoming one of the trendy topics of research in UAVs. There are a number of terminologies to refer this systems where detect

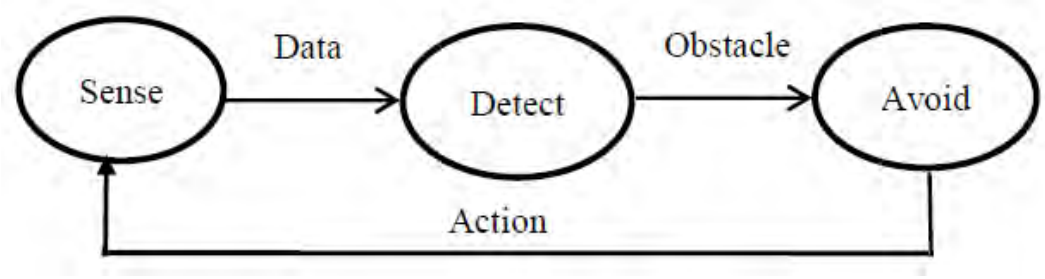


Figure 1.1: Diagram of the taxonomy of sense and avoid [3]

and sense are used to detection and identification of any object from sensory data, and the term avoid represents an avoidance operation from the intruder object. Most researchers call these operations as Sense and Avoid (SAA or S&A) [3], however NASA call them as detect and avoid (DAA), and Federal Aviation Administration (FAA) call them detect, sense, and avoid (DSA) [4]. In this thesis the term SAA is used, because it is the most used term.

In Figure 1.1 is shown a general diagram of a SAA system, where the concept of sense and avoid involves three discrete steps: sense, detect and avoid. The sense stage consists of one or more sensors which perceives the environment surrounding the UAV, with continuous and high updating rates. If the type of sensor is a camera the data will be images, while if a RADAR is used, the data may be the distance and speed of an obstacle. When data comes from two or more sensors, the information can be fused to obtain a better estimation. Detect step consists of a computer with the capability to receive and process all the data coming from the sensors, and then analyse and detect any risk of collision with an obstacle. Particle filters can be used as collision detectors that can estimate the trajectory of an object. Finally, the avoid step consists of a program that regenerates a new path of flight for the UAV if the obstacle is detected and should execute the new path with enough anticipation in order to avoid the collision.

The objective of this thesis is to provide an UAV with these three mentioned procedures. For that purpose, the sense step consists of a stereo camera and a 1D-LiDAR,

where the stereo camera data is used to generate a depth map and the 1D-LiDAR is used to measure the distance in front of the UAV. According to the references, the depth map generated by means of stereo correspondence always has an percentage of error in each pixel value, however there are some depth values that does not have a numerical value. The 1D-LiDAR, for its part, also presents some measuring error, e.g. through transparent materials or reflecting surfaces. The detect stage involves the use of Kalman Filter to ameliorate the distance measure in front of the UAV and detect free-obstacle navigation areas with the depth map. The last stage is the avoidance, which decides the best flight trajectory and sends the command to UAV controller to exert the avoidance maneuver.

1.1 Problem Statement

After the study of the state of the art of SAA systems, one of the best methods of obstacle sensing is performed by stereo camera, however all of them have several limitations and problems during sensing and detecting obstacles. One of them is explained below.

Problem

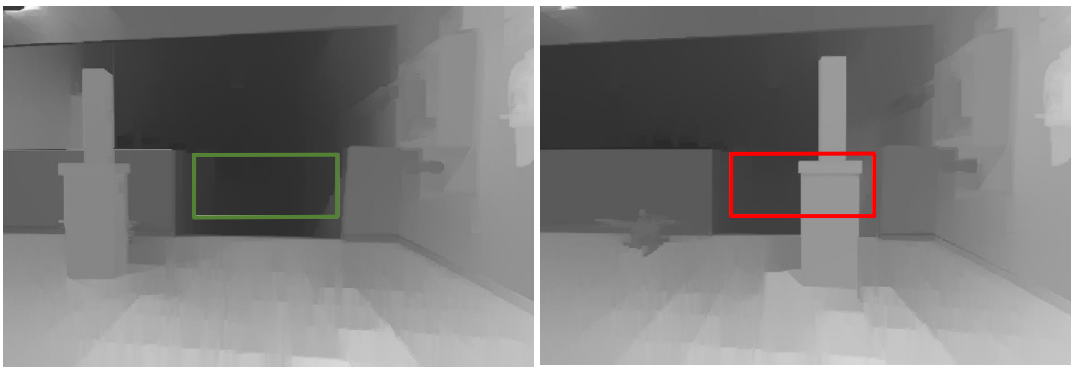
There are several methods for depth map computation presented in the literature, however, many of them generate a disparity map with a great quantity of noise, this noise could interfere with the depth reading. To deal with this problem the most popular solution is the use of expensive computation, which slows down the disparity map generation, and then affecting the efficiency of the obstacle detection. On the other hand, faster algorithms generate a high percentage of bad matching pixels, creating regions in the depth map without real values. It is necessary to find an algorithm with intermediate performance.

To obtain a precise measurement it is possible to fuse depth and the depth measure-

ment from a second sensor. With that in mind, one of the most popular solution is the use of 3D-LiDAR, however this type of sensors are expensive and heavy, making them an option not suitable for small UAVs. Other option are the RGB-D sensors, but they do not work properly under daylight environment.

Solution

For the the detection stage of the system is necessary to ameliorate the depth map using only: a) a stereo camera rig with disparity estimation algorithms free of noise; and b) a simple and low-cost 1D-LiDAR sensor of one dimension. With that aim, the SAA system must determine if there is or not a obstacle in the UAV navigation path. For this purpose, it is calculated a window considering real quadrotor dimensions. Such a window is depicted in the scene captured by the stereo camera rig, a picture of this idea can be seen at Figure 1.2 where the rectangle represents the UAV size to a distance from the camera equal to 2 meters. As it is shown in the Figure 1.2, when there is not any obstacle in the window the rectangle that represents the window area becomes green, indicating that it is a free obstacle path. When an obstacle is present in the window, the rectangle becomes red. The 1D-LiDAR is pointing at the rectangle centroid measuring the depth in that direction. The proposed solution is to fuse the depth measurement in the centroid of the rectangle coming from the stereo camera and depth measurement of the 1D-LIDAR in a KF.



(a) Without obstacle. (b) With obstacle.

Figure 1.2: The (a) image shows a free area where the UAV is able to navigate. Figure (b) represents an obstacle inside the rectangle, which does not allow the UAV to navigate. Observe that the rectangle represents the size of the quadrotor in which the navigation path must be free, therefore the goal is to determine if this area is blocked or not by a potential obstacle.

CHAPTER 2

State of the art

This section addresses the state of the art based on the three stages of a SAA systems. Camera, LiDAR and Stereo Matching are subsections of Sensing stage.

2.1 Sense and Avoid systems

The taxonomy of SAA is presented in Figure 2.1, which consists of three types of architectures. This classification is based on the physical location of information sources and the center of processing or decision, i.e. the on-board implementation, and the ground-based or off-board implementation. There are mixed implementations of these two architectures. [5].

Sensors can be generally classified into two groups: cooperative and non-cooperative. The most widely cooperative method has been used for decades by manned aircraft technology, called Traffic Alert and Collision Avoidance System (TCAS) [6]. An UAV

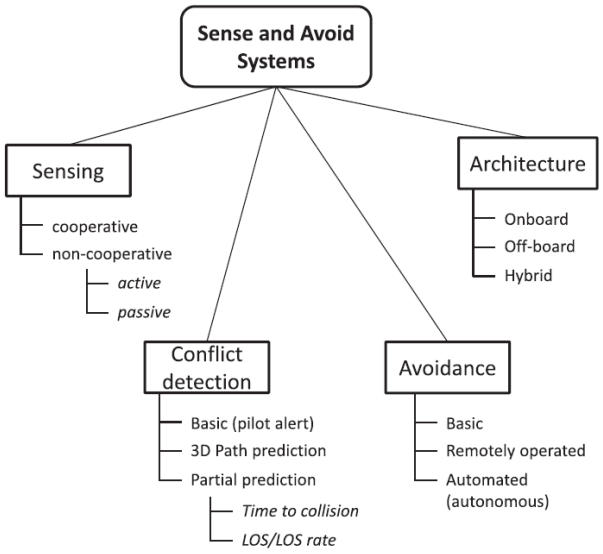


Figure 2.1: Basic taxonomy of SAA systems [3]

equipped with TCAS can communicate with other UAVs equiped with TCAS and then avoid collisions using the information providing from a ground-based radar device. A more recent method is the Automatic Dependent Surveillance (ADS-B) [7] which broadcasts the identification, position (latitude and longitude), altitude and velocity of the aircraft to other UAS near the area; the information comes from on-board GPS devices. The non-cooperative technologies do not need other aerial vehicles to be equipped with the same devices. Non-cooperative sensors examples are the acoustic systems and infrared sensor (Figure 2.2), radar, light detection and ranging (LiDAR) and electro-optical (EO) systems (Figure 2.3). These sensors can be classified into two groups: passive sensors and active sensors. The active sensors emit a type of energy signal towards the objects and then detects the reflected signal with a receptor; the passive sensors, for their part, do not emit any kind of energy.

The cameras can be categorized according to operating wavelength, e.g. Ultra Violet, Infrared and visible spectrum, each one of which have special properties. There are some variations of cameras that are capable of measure depth Figure 2.4, commonly

(a) Long Range IR Distance Sensor by Sharp [8].

(b) UM30 Ultrasonic Sensor by Sick [9].

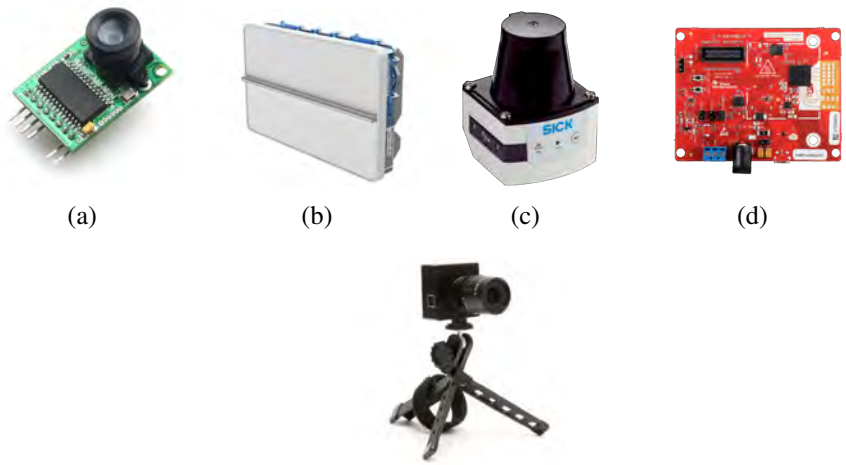
(c) SF02 Laser Rangefinder by Lightware [10].

Figure 2.2: Examples of one-dimensional sensors (a),(b) and (c), all of them active type sensors.

using the human-visible spectrum of the light, like the stereo camera which measure depth using two displaced images and the mathematical model of the cameras arrangement.

The Time of Flight (or Arrival) cameras (TOF) is a active type camera that works by measuring the round trip time of an artificial light signal provided by laser or LED. There are some other special cameras, like the multispectral and hyperspectral cameras that serves for examination and measurement purposes. Event-based cameras are novel, bio-inspired visual sensors, where each pixel output is asynchronous and independent when local intensity changes, called ”events”, and recently, this type of camera has been used by D. Falanga et al. in their work [11] for fast-moving obstacle detection.

The LiDAR is an active light-based sensor that measures distance using laser pulses. A Laser Range Finder (LRF) is a single device that measures distance in 1D. If the LRF is rotated by a servo motor, the type of sensor is 2D-LiDAR. If more than one LRF is attached to the rotating axis at a different angle with respect to the axis, then this arrangement corresponds to a 3D-LiDAR. There is a solid-state type of LiDAR that do not have any moving part, and works by a laser beam steering system that are projected to the environment and then an array structure of cell that detects the laser beam and



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

(e)

Figure 2.3: Examples of two-dimensional sensors (a) Mini Module Camera Shield with OV2640 by ArduCAM [12], b) EchoFlight airborne radar by Echodyne [13], c) 2D LiDAR sensor TiM561 by Sick [14], d) mmWave Radar AWR1443 by Texas Instruments [15], and e) Dynamic Vision Sensor by Inivation [16]

obtain the depth measurement.

When the sensor of the SAA system is a stereo camera, the first issue to solve is to choose an algorithm of stereo matching, which is crucial to obtain a good disparity map. In the website Middlebury Stereo Evaluation - Version 2 [20] there is a list of more than 150 stereo matching algorithms ranked according to the average percent of bad pixels, obtained from the relation between the computed disparity map and the ground truth. However, the best reference and to compare stereo matching algorithms is the version 3 website of the aforementioned evaluation site [21] which is based on the paper of D. Scharstein [22] where several parameters of stereo matching are defined with the purpose of comparison. In this list we find the latest stereo matching algorithms and the most accurate are predominantly performed by neural networks or superpixels methods or a mixture of them. Among the more accurate methods is [23], which proposes an algorithm based on superpixels labeling of an image and then applying a bilayer matching cost where a neural network compare similarity between layers.

(a) BlasterX Senz3D by Creative Labs [17].

(b) ZED camera by Stereolabs [18].

(c) Intel Realsense D435 by Intel [19].

Figure 2.4: Examples of three-dimensional sensors, (a) a time-of-flight camera, of active type, (b) is of passive type, and (c) is a hybrid of both stereo and time-of-flight camera.

This kind of approach reduces the disparity map noise but the computation time increases significantly. Meanwhile H. Hirschmller [24] proposes the Semi-Global Block Matching (SGBM) method, that works with less precission but faster, so this method was adopted in this thesis to generate the disparity map.

Sensor fusion proposes a great opportunity to defeat the physical limitations of the sensing systems. It works by combining the information from a variety of different sensors to give a robust and complete description of the rapidly changing environment. Each sensor has different features, depending of the approach, they can be strengths or weaknesses, as stated by J. A. Jackson et al. [25], so the sensor fusion provides the possibility of combine different technologies in order to obtain an improved estimated measurement. Since this extra processing increases the latency of sensing, it is important to use the optimal one for each situation. S. Ramasamy and R. Sabatini in their work [26] show a simulation studio of sensor fusion combining a number of non-cooperative sensors. The algorithm uses Kalman filter tracking to every sensor, then it is combined and compared to observation, where it can verify the robustness of the algorithm to solve issues, like limited information or loss of information.

K.R. Sharma et al. in their publication [27] fuse with a KF three distance sensors in order to obtain the distance and orientation with respect to a wall. K. Park et al. in their work [28] presents a high-precision depth map using a high cost 3D-LiDAR,

however, cost of implementation is considerably higher than the approach presented in this paper. G. Fasano et al. in their work [29] implements sensor fusion to improve the collision detection using the information of radar and electro-optical sensors, reducing the false detection, increasing the detection range and reducing the computational cost. Similarly, J. Kim and Y. K. Kwag [30] fuses the data of synthetic aperture radar and electro-optical sensor, resulting in a reduction of false alarm between 14% and 22% in target detection per imaging area compared to the case of a single sensor. Another experiment done in the work [31] R. Rambabu et al. with infrared and ultrasonic sensor reduced the reading noise that delivered an accurate range estimation. H. R. Song et al. proposes in their work [32] the fusion of RGBD and 2D-LiDAR for tracking purposes. Roopa et al. [33] fuse images using Kalman Filter (KF) to get more information about the localization of a target, this approach is applied to different cameras and different localization.

The main goal of the collision detection algorithm is to decide if a collision will happen using the information from cooperative or non cooperative sensors, and it must be efficient in order to make the path planning accurate. Efficiency in obstacle detection requires fast updating rate and low false alarm detection [34]. J. J. Ruz et al. in [35] and the work of B. Vanek [36] discussed Markov decision process (MDP) as a handy in situations where there is uncertainty and is optimal for collision avoidance algorithms like numerical optimization technique and dynamic programming technique, however the disadvantage is that it is computationally slow. In the work [37] Strobel and Schwarzbach uses a worst case approach, and the algorithm estimates all the possible trajectories of a intruder using the current trajectory, position and speed to detect collisions when the trajectory of the UAV meets with the obstacles trajectory, the problem is that it consumes long time to calculate the possible trajectories of the intruders, but it is more accurate than most algorithms. The work of A. Zarandy et al. [38] considers the actual trajectories to predict the future trajectory, then calculate the minimum

distance between the UAV and the intruder, if this distance is less than a predetermined threshold value, the UAV must must execute an avoidance maneuver. There are also detection algorithms that depends according to the type of sensor, for instance A. Stanoev et al. [39] establish a threshold in the depth map where the close objects are white and labeled as obstacles and the farther ones are black and then ignored; if the robot moves quickly, the threshold decreases. In some cases it is necessary to differentiate obstacles over a flat surface, in this case it is useful to implement V-disparity maps [40] which is a function of the disparity map, that accumulates the disparities of the horizontal line into the v-disparity function, where the abscissa corresponds to the number of disparities. This approach can be used in vehicle navigation on a road. B. Lopez [41] proposes a perception and planning approach that significantly reduces the computation time using instantaneous perception for obstacle avoidance. Aman [42] proposes a methodology to fuse ultrasonic sensor measurement and depth map from time-of-flight camera sensor. M. ki et al. [43] propose a framework which implements a stereo camera and a 2D-LiDAR on an UAV, however the sensor is the only obstacle detector, and the camera is just used for monitoring.

When an obstacle is confirmed by the collision detection algorithm, the obstacle avoidance algorithms must create a new path of flight to avoid collision. This algorithm must be supported by path planning. Some examples of implementation of mixed integer linear programming, is described in the publication [44] of Zuqiang et al. and in the paper [45] of A. L. Smith, which consists in divide the problem of path planning in parts, taking in count as main constrains the minimum time and minimum fuel consumption. A dynamic programming algorithm is implemented by N. E. Smith et al. in [46], and E. J. Forsmo et al. in [47] which consists in divide the path into sections, considering the minimum time as constraint, these sections are connected by nodes and then finds the shortest path among the possible ones that reach to the next node, and then repeating this process it is possible to find the shortest path between an

initial node to a final node. There is an optimization of particle swarm, discussed by C. Lopez-Franco et al. [48] which works by a finite number of iterations, which reaches to an optimal solution. This works fine in environments where it is important to consider the terrain and obstacles in the ground. A widely used method is the potential field approach, discussed in [49] by F. Rehmatullah and J. Kelly, which consists in considering the entire space as potential field, by assigning equal charges to the UAV and the potential obstacles, causing repulsion forces according to the position and velocity of the UAV and objects. However this algorithm must know the complete state of the intruders to work efficiently. In the work of B. Du and S. Liu [50], and the work of C. Fu [51] the avoidance is performed by fuzzy logic controllers, which works using linguistic variables as inputs and outputs. In [52] T. Liao developed a searching algorithm called A*, and like dynamic programming, uses as constraint a cost function and distance constraint. This algorithm provides different paths that passes through some nodes, between the starting point and the final point. It works well in environment with static obstacles, but not good enough with dynamic obstacles.

CHAPTER 3

Theory Fundamentals

In this section it is briefly described the basic concepts of the Quadrotor Model, Stereo Matching which in turn is divided in some topics to make it easy to review the concepts needed for the methodology,and finally followed by Kalman Filter.

3.1 Quad-rotor model

By definition [53] an UAV or unmanned aircraft is just a part of a Unmanned Aircraft System (UAS), which comprises:

- Control Station (CS), where is located the system operators and their interfaces with the rest of the system.
- Aircraft carrying the payload.
- System communication between the CS and the aircraft.

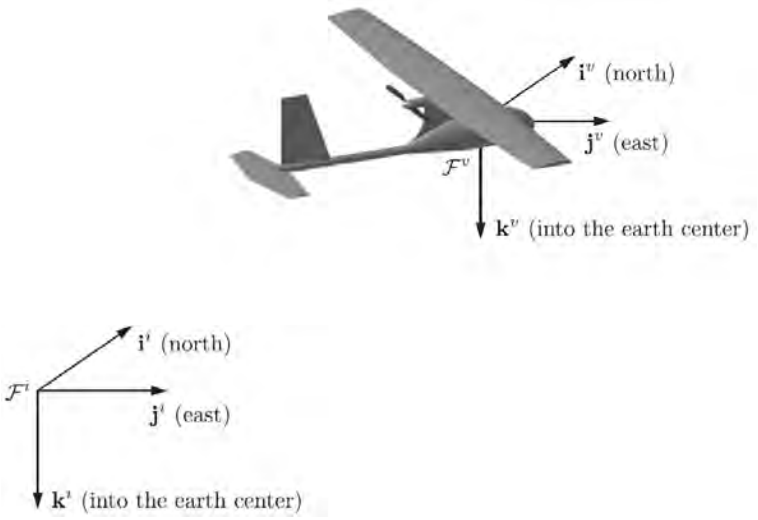


Figure 3.1: Left-Down: inertial frame and its axis i^i, j^i and k^i. Up-right: vehicle frame and its axis i^v, j^v and k^v. [54]

- Support equipment like maintenance and transport items.

It must not be confused with radio-controlled model aircrafts or with drones. A model aircraft it is used for sport and it must be within sight of the operator. A drone usually is launched with a pre-programmed mission to accomplish and can fly out of sight of the operator, after the mission is carried out it returns to the base, but it does not have any intelligent behaviour. An UAS can have some degree of automatic intelligence.

3.1.1 Reference frames

In this section it is defined three reference frames: the inertial frame, vehicle frame and body frames.

The Inertial frame $\mathbb{F}^i$ has its frame origin fixed to a home location on the Earth. The name of the axes are north, east and down, which have the unit axis i^i, j^i and k^i respectively. The down axis is pointing at the center of the Earth, as can be seen in the Figure 3.1.

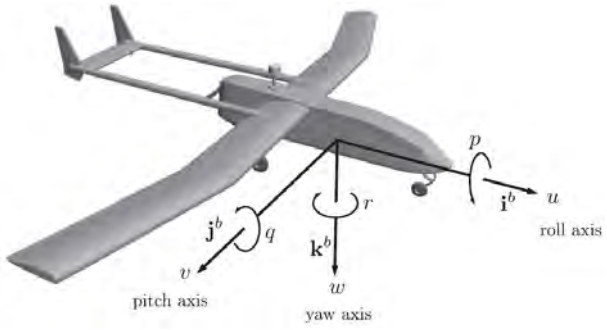


Figure 3.2: Body frame, the i^i-axis is also known as roll axis, j^i-axis as pitch axis and k^i-axis as yaw axis [54].

The Vehicle frame $\mathbb{F}^v$ is a frame whose origin is located in the center of mass of the vehicle, however, its axis points are aligned with the inertial frame, see Figure 3.1.

The body frame $\mathbb{F}_b$ has its origin in the center of mass of the vehicle, but the axis are located after a sequence of rotation in the angles ψ, θ and ϕ. The i^b-axis points out the nose of the airframe, j^b-axis points out the right wing and k^b-axis points out the belly, see Figure 3.2.

A quadrotor or quadcopter is a rotary-wing type UAV of four engines attached in the extremities of the device and usually holds in the middle a controller and a battery. The Figure 3.3 shows six types of quadcopters.

3.1.2 Euler Angles

As the name says, they were introduced by Leonard Euler to describe the orientation of a rigid body in a 3-dimensional Euclidean space. They are typically denoted by

$$\begin{aligned} \phi &\in [-\pi, \pi] \\ \theta &\in [\tfrac{\pi}{2}, \tfrac{\pi}{2}] \\ \psi &\in [-\pi, \pi] \end{aligned} \tag{3.1}$$

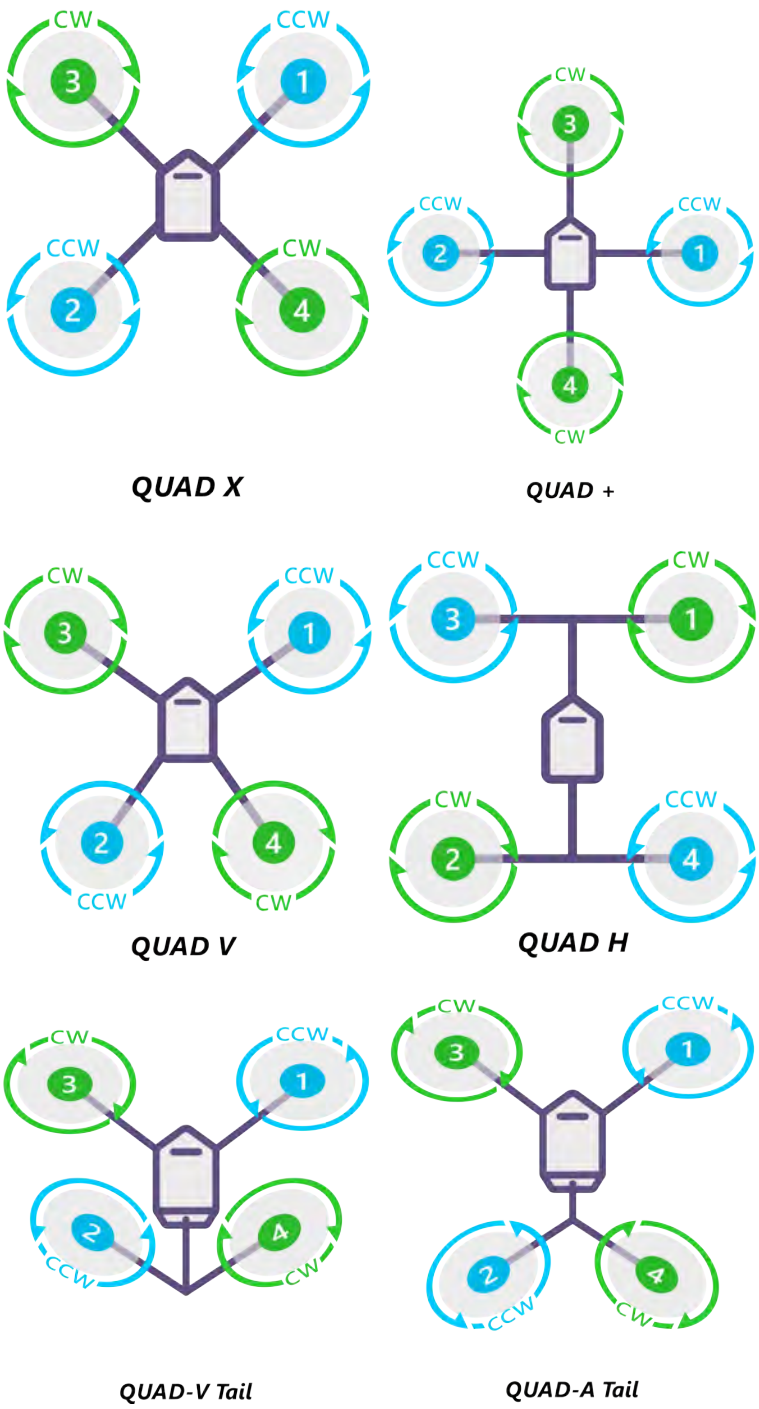


Figure 3.3: Type of quadcopters. The green propellers spin in Clockwise direction (CW), while the blue propellers spin in Counter-Clockwise direction (CCW) [55]

These angles can be applied in a sequence to move any body in any orientation. These angles are applied by the rotation matrices

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi) & c(\phi) \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix} \tag{3.2}$$

$$R_z(\psi) = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where the functions $c(\phi) = cos(\phi)$, $s(\phi) = sin(\phi)$ also applies to the angles θ and ψ. The rotation of the body reference system with respect to the inertial reference is defined with the rotation matrix $\mathbf{R}_{zyx}(\phi,\theta,\psi) \in SO(3)$

$$\mathbf{R}_{zyx}(\phi,\theta,\psi) = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)$$

$$= \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \tag{3.3}$$

3.1.3 Kinematics

Let $[p_n, p_e, p_d,]^T$ be the vector that is composed of the linear and angular positions in the inertial frame, and $[u, v, w, p, q, r]^T$ is the vector of the linear and angular velocities

of the body frame.

$$\xi = \begin{bmatrix} p_n \\ p_e \\ p_d \end{bmatrix} \quad \eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$$

$$\mathbf{v} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad \omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

The values of (u, v, w) are the projection of the inertial velocity components. Also, the translational position of the MAV is commonly referred from the inertial frame. The relation of the translational velocity and position is done by

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{bmatrix} = \mathbf{R}_b^v \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{bmatrix} = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.4}$$

The angular rates (p, q, r) are not the simple derivative of the angular rates (ϕ, θ, ψ) because they are in two different reference frames. The angular rates are defined in the body frame $\mathbf{F}^b$. The angular positions are defined in three different frames when each rotation transformation was applied to the vehicle frame to take it to the body frame. The angular rates in body frame is expressed in terms of the derivatives of the angular

positions as follows

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \mathbf{R}(\phi)\mathbf{R}(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & c(\phi) & -c(\theta)s(\phi) \\ 0 & -s(\phi) & c(\theta)c(\phi) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Inverting the expression

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi)/c(\theta) & c(\phi)/c(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.5}$$

where $t(\theta) = tan(\theta)$.

3.1.4 Dynamics

Now using the Newton's second law it is going to be derived the translational and rotational dynamic models. For this the motion of the body must be referenced to the inertial frame, which in this case is the ground. It is defined the vector $\mathbf{V}_b^g = (u, v, w)^T$ as the velocity of the body frame with respect to the ground (inertial) frame. Applying the Newton's second law it is obtained that

$$\mathbf{f} = m\frac{d\mathbf{V}_b}{dt_i} \tag{3.6}$$

where m is the mass of the body, the $\frac{d}{dt_i}$ is the time derivative with respect to the inertial frame and $\mathbf{f}$ is the sum of all the forces actuating on the body, such as gravity, aerodynamic forces and the forces exerted by the propellers. It is important to take in count that the aerodynamic forces depends on the speed of the surrounding air, in the case of the wind is not present, however the wind is almost always present in some degree, so the velocity of the aircraft relative to the surrounding air is the airspeed vector, denoted

$\mathbf{V}_a$, the ground speed vector $\mathbf{V}_g$ is the velocity vector of the airframe with respect to the inertial frame, and $\mathbf{V}_w$ is the vector velocity of the wind, with respect to the inertial frame. So the relationship between these three vector is

$$\mathbf{V}_a = \mathbf{V}_g - \mathbf{V}_w$$

It is necessary to express the derivative of $\mathbf{V}_g$ of (3.6) as a function of the derivative in the body frame and the angular velocity, for this it is necessary to use the derivative of a momentum vector. Let's say that the vector $\mathbf{p}$ is moving in body frame but not rotating with respect to the inertial frame, i.e.

$$\mathbf{p} = p_x i^b + p_y j^b + p_z k^b$$

where i^b, j^b and k^b are the unit vector of $\mathbf{p}$. Then, deriving the vector $\mathbf{p}$ in the inertial frame is

$$\frac{d}{dt_i}\mathbf{p} = \dot{p}_x i^b + \dot{p}_y j^b + \dot{p}_z k^b + p_x \frac{d}{dt_i} i^b + p_y \frac{d}{dt_i} j^b + p_z \frac{d}{dt_i} k^b \tag{3.7}$$

where the first three terms are the change of $\mathbf{p}$ seen from the rotating body frame itself. The next three terms are the change in $\mathbf{p}$ seen from the inertial frame. The derivative of the components i^b, j^b and k^b are

$$\dot{i}^b = \omega_{b/i} \times i^b$$
$$\dot{j}^b = \omega_{b/i} \times j^b$$
$$\dot{k}^b = \omega_{b/i} \times k^b$$

where $\omega_{b/i}$ is the angular velocity of the body with respect to the inertial frame. Then rewriting the terms

$$p_x \dot{i}^b + p_y \dot{j}^b + p_z \dot{k}^b = p_x(\omega_{b/i} \times i^b) + p_y(\omega_{b/i} \times j^b) + p_z(\omega_{b/i} \times k^b) = \omega_{b/i} \times \mathbf{p} \tag{3.8}$$

With the Equation (3.7) and Equation (3.8) it can be obtained

$$\frac{d}{dt_i}\mathbf{p} = \frac{d}{dt_b}\mathbf{p} + \omega_{b/i} \times \mathbf{p} \tag{3.9}$$

If it is assume that the mass is constant in the equation $\mathbf{p} = m\mathbf{V}_g$, the mass can be removed from Equation (3.9) and obtain

$$\frac{d}{dt_i}\mathbf{V}_g = \frac{d}{dt_b}\mathbf{V}_g + \omega_{b/i} \times \mathbf{V}_g \tag{3.10}$$

Now substituting the Equation (3.10) in Equation (3.6) the equation results

$$\mathbf{f} = m\left(\frac{d}{dt_b}\mathbf{V}_g + \omega_{b/i} \times \mathbf{V}_g\right) \tag{3.11}$$

Due to the fact that $\dot{V}_g$ is expressed in the body frame the following is defined as

$$\begin{aligned}
\dot{V}_g &= (u, v, w)^T \\
\omega_g &= (p, q, r)^T \\
\mathbf{f} &= (f_x, f_y, f_z)^T \\
\tfrac{d}{dt_i}\mathbf{V}_g &= (\dot{u}, \dot{v}, \dot{w})^T
\end{aligned}$$

Rearranging and then expanding the cross product of Equation (3.11)

$$\frac{d}{dt_b}\mathbf{V}_g = \frac{\mathbf{f}}{m} - \omega_{b/i} \times \mathbf{V}_g$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m}\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{3.12}$$

In this way the Equation (3.12) is the translational motion model of the quadcopter. It remains the rotational motion model, and using the Newton's second law

$$\frac{d\mathbf{h}}{dt_i} = \mathbf{m} \tag{3.13}$$

where $\mathbf{h}$ is the angular momentum, and $\mathbf{m}$ is the sum of all moments applied on the body. Using the expanded Equation (3.7) to substitute the derivative of angular momentum

$$\frac{d}{dt_i}\mathbf{h}_g = \frac{d}{dt_b}\mathbf{h}_g + \omega_{b/i} \times \mathbf{h}_g = \mathbf{m} \tag{3.14}$$

The angular momentum for a rigid body is defined as

$$\mathbf{h} = \mathbf{J}\omega_{b/i} \tag{3.15}$$

where $\mathbf{J}$ is the inertia matrix whose components are given by

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_x & -\mathbf{J}_{xy} & -\mathbf{J}_{xz} \\ -\mathbf{J}_{yx} & \mathbf{J}_y & -\mathbf{J}_{yz} \\ -\mathbf{J}_{zx} & -\mathbf{J}_{zy} & \mathbf{J}_z \end{bmatrix} \tag{3.16}$$

$$\begin{aligned} \mathbf{J}_x &= \int (y^2 + z^2) d\mathrm{m} \\ \mathbf{J}_y &= \int (x^2 + z^2) d\mathrm{m} \\ \mathbf{J}_z &= \int (x^2 + y^2) d\mathrm{m} \\ \mathbf{J}_{xy} &= \mathbf{J}_{yx} = \int xy d\mathrm{m} \\ \mathbf{J}_{xz} &= \mathbf{J}_{zx} = \int xz d\mathrm{m} \\ \mathbf{J}_{yz} &= \mathbf{J}_{zy} = \int yz d\mathrm{m} \end{aligned}$$

However, the body of a quadcopter is usually symmetric around the plane spanned by i_b and k_b, and the plane spanned by j_b and k_b, since the angular momentums cancelled

in each side of the planes, so the inertia elements $\mathbf{J}_xy = \mathbf{J}_yz = \mathbf{J}_xz = 0$, in this way, the inertial matrix in Equation (3.16) is

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_x & 0 & 0 \\ 0 & \mathbf{J}_y & 0 \\ 0 & 0 & \mathbf{J}_z \end{bmatrix}$$

Substituting the definition of the angular momentum Equation (3.15) in Equation (3.17) results

$$\mathbf{J}\frac{d}{dt_b}\omega_{b/i} + \omega_{b/i} \times \mathbf{J}\omega_{b/i} = \mathbf{m} \tag{3.17}$$

On the other hand

$$\frac{d}{dt_b}\omega_{b/i} = \dot{\omega}_{b/i} = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \tag{3.18}$$

Solving for $\dot{\omega}_{b/i}$ in Equation (3.17)

$$\dot{\omega}_{b/i} = \mathbf{J}^{-1}\left[-\omega_{b/i} \times (\mathbf{J}\omega_{b/i}) + \mathbf{m}\right] \tag{3.19}$$

Where $\mathbf{J}^{-1}$ is

$$\mathbf{J}^{-1} = \begin{bmatrix} \frac{1}{\mathbf{J}_x} & 0 & 0 \\ 0 & \frac{1}{\mathbf{J}_y} & 0 \\ 0 & 0 & \frac{1}{\mathbf{J}_z} \end{bmatrix}$$

and $\mathbf{m} = (l, m, n)^T$ is the moment vector. Then rewriting the Equation (3.19) in component form

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{\mathbf{J}_x} & 0 & 0 \\ 0 & \frac{1}{\mathbf{J}_y} & 0 \\ 0 & 0 & \frac{1}{\mathbf{J}_z} \end{bmatrix} \left[\begin{pmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{pmatrix} \begin{pmatrix} \mathbf{J}_x & 0 & 0 \\ 0 & \mathbf{J}_y & 0 \\ 0 & 0 & \mathbf{J}_z \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} + \begin{pmatrix} l \\ m \\ n \end{pmatrix} \right]$$

$$= \begin{bmatrix} \left(\frac{J_y - J_z}{J_x}\right) qr + \frac{l}{J_x} \\ \left(\frac{J_z - J_x}{J_x}\right) pr + \frac{m}{J_y} \\ \left(\frac{J_x - J_y}{J_z}\right) pq + \frac{n}{J_z} \end{bmatrix} \tag{3.20}$$

In this manner the set of equation that models the quadcopter is complete, in [54] the derivation of the formulas are more explained in details. In summary Equations (3.4), (3.5), (3.12) and (3.20)

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{bmatrix} = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.4}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi)/c(\theta) & c(\phi)/c(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.5}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{3.12}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \left(\frac{J_y - J_z}{J_x}\right) qr \\ \left(\frac{J_z - J_x}{J_x}\right) pr \\ \left(\frac{J_x - J_y}{J_z}\right) pq \end{bmatrix} + \begin{bmatrix} \frac{l}{J_x} \\ \frac{m}{J_y} \\ \frac{n}{J_z} \end{bmatrix} \tag{3.20}$$

3.2 Stereo Matching

Stereo matching or stereo correspondence, in its analytical approach, is an active research topic which requires a solid background of epipolar geometry before go in depth. After this subsection the next subsection addresses about stereo matching, that depend-

ing on the approach it is supported in different mathematical fields.

3.2.1 Epipolar geometry

Epipolar geometry is basically the projective geometry that involves two views. Almost all the analytical stereo matching algorithms is based on epipolar geometry, which is of course the approach used for this work.

Pinhole camera

Almost all the work done about stereo vision has been done considering the simplest camera model, the pinhole camera. This kind of model shows a good approximation to the real construction of a modern camera with lenses due to the similarity with the CCD sensors.

Basically what a camera does is to project the rays of light coming from the 3D points to an internal plane of the camera, named the image plane. The Figure 3.4 shows the geometrical configuration of the pinhole camera where the distance between the camera center and the image plane is the focal distance f. Observe that the point in the 3D world $\mathbf{X} = (x, y, z)^T$ is mapped to the point $\mathbf{x} = (u, v)^T$ in the image plane by a line joining the center of the projection to the point $\mathbf{X}$. This mapping is represented as

$$(x, y, z)^T \mapsto (fx/z, fy/z)^T \tag{3.21}$$

This mapping can be performed by $\mathbf{x} = \mathbf{PX}$ where $\mathbf{P}$ is a 3×4 matrix which can be expressed with

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \tag{3.22}$$

where $\mathbf{K}$ is a 3×3 matrix called camera calibration matrix, and it contains the internal parameters of the camera; the $\mathbf{R}|\mathbf{t}$ matrix is a 3×4 rotation matrix, where $\mathbf{R}$ and $\mathbf{t}$

pin

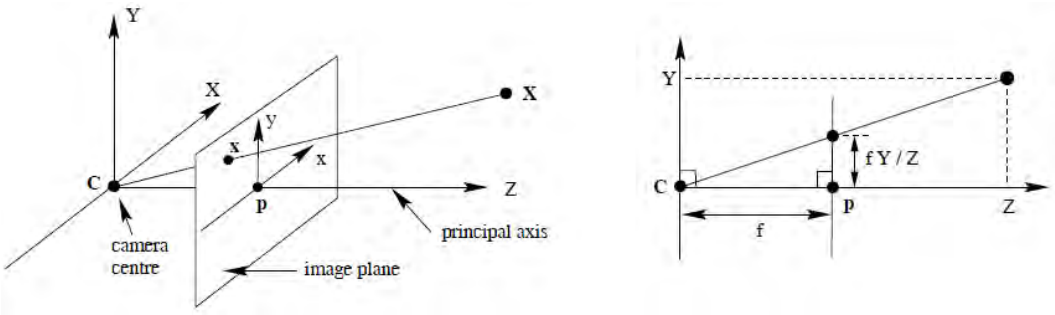


Figure 3.4: Pinhole camera model [56].

represents the orientation and translation (respectively) of the camera coordinate frame with respect to a world frame coordinate. The **R** in detail is

$$\mathbf{R} = \begin{bmatrix} f\alpha & \gamma & c_x \\ 0 & f\beta & c_y \\ 0 & 0 & f \end{bmatrix} \tag{3.23}$$

where $(c_x, c_y)^T$ is the center of the principal point, f is the focal length, α and β are the pixel physical dimensions, and γ is the angle between the axes.

Epipolar geometry

The epipolar geometry is the intrinsic projective geometry of two views. It depends only on the internal parameters of the cameras and their relative position. This kind of geometry is useful to search for a corresponding points either as a single moving camera or two fixed cameras. The system configuration that is going to use in this thesis is for two fixed cameras, also named, stereo camera. When the left and right cameras of the stereo rig simultaneously captures an image of a **X** point, this point is mapped as **x** in the left camera and $\mathbf{x}'$ in the right camera. These points creates a plane π, called epipolar plane. According the pinhole model of camera, such a plane must

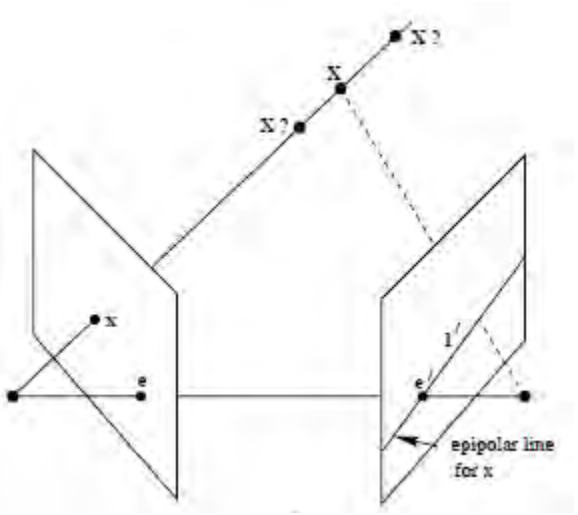


Figure 3.5: Epipolar geometry [56].

include the left and right centers of the camera, as shown in Figure 3.5, where also can be observed that any point in the line that joints the point $\mathbf{X}$ to the center of the left camera, projects to exactly the same point $\mathbf{x}$ in the left image plane, however in the right image the point $\mathbf{x}'$ do not. This means that if we do not know the location of the point $\mathbf{X}$, it can correspond to any point in the line $\mathbf{l}'$ in the right camera plane. The line $\mathbf{l}'$ is the right epipolar line, and it also is the intersection of the epipolar plane with the right image plane. The vice-versa occurs when referring to the other side of the stereo rig. As it was mentioned, there exists a mapping $\mathbf{x} \mapsto \mathbf{l}'$ that is expressed with the equation

$$\mathbf{l}' = \mathbf{F}\mathbf{x} \tag{3.24}$$

where if a point $\mathbf{x}$ is choosen from $\mathbf{l}'$, the following equation is satisfied

$$\mathbf{x}'^{T}\mathbf{F}\mathbf{x} = 0 \tag{3.25}$$

Figure 3.6: Epipolar lines [56].

where $\mathbf{F}$ is the fundamental matrix, and it contains the intrinsic parameters of both cameras. The matrix $\mathbf{F}$ is

$$\mathbf{F} = [\mathbf{e}']_{\times}\mathbf{P}'\mathbf{P}^{+} \tag{3.26}$$

where $\mathbf{P}^{+}$ is the pseudo inverse of $\mathbf{P}$ and $[\mathbf{e}']_{\times}$ is the right epipole vector expressed as a skew-symmetric matrix. The epipole is the intersection of the baseline with the image plane, so there are two epipoles in a two-view arrangement, like the stereo camera. As can be seen, the matrix $\mathbf{F}$ is a function of epipole, the right and left cameras, however, in this thesis is not necessary going in depth anymore (for details see [56]).

Image rectification

According to the Figure 3.5 the epipolar plane π intersects with both image views, creating the epipolar lines $\mathbf{l}$ and $\mathbf{l}'$. If both cameras faces to the same scene at the same time, there are points in $\mathbf{l}$ that correspond to the points in $\mathbf{l}'$. Usually the corresponding epipolar lines are slanted, as can be seen in Figure 3.6. This characteristic makes the operation of finding in the images correspondence points a slow task. To speed up this process both images can be rectified, which consists in applying a projective transformation that makes all the epipolar lines of both sides coincident in the same rows (or scanlines) of the images, so the search of a correspondence point in the first

image reduces to look for it only of the same row (v coordinate) of the other image. There are many algorithms to compute the homography matrices $\mathbf{H}_1$ and $\mathbf{H}_2$, necessary to transform the left and right images, respectively.

Most stereo rigs have both cameras facing front, its principal lines parallel and with the world origin at the left camera, so for this case, the camera matrices can be simplified as

$$\mathbf{P} = \mathbf{K}[\mathbf{I}|\mathbf{0}] \quad \mathbf{P}' = \mathbf{K}'[\mathbf{I}|\mathbf{t}]$$

where $\mathbf{t}$ is the translation of the right camera in the axis x.

3.2.2 Stereo correspondence

The correspondence problem consists in finding correspondence points in two or more images of the same object but taken from different positions. The aim of a stereo correspondence algorithm is to give a uni-valued function $d(u,v)$ that best represent the shapes of the surfaces captured by two displaced images. The special case when the images are taken from two cameras at the same time is called stereo correspondence. This problem is simpler when using a stereo camera with frontal parallel cameras, known epipolar geometry and rectified image pair. However an algorithm of finding stereo correspondence is not a trivial task, because the algorithm must measure the evidence that correspondence points in the two images are projections of the same scene point.

After both images are rectified, the comparison between right and left images is performed only in the same row of the images. There are three stages [57] of the stereo block corresponding algorithm:

- Employ a prefilter to normalize the input images, this is reduce the brightness differences between them and improve textures.

- Look for correspondences in the row (or horizontal) of both images using a SAD window.

- Apply a postfilter to remove bad-matched correspondences.

In the prefiltering stage, the normalization of images consists in running a window of a predefined size (like 5x5, 7x7,...,21x21 as maximum) over the image. The center of the window is a pixel represented by $I_c = min(max(I_c - \hat{I}, -I_{cap}), I_{cap})$, where $\hat{I}$ is the average value of the window and I_{cap} is a limit value.

After prefiltering, it is followed by the correspondence computation made by sliding the SAD window in the right image for each pixel in the left image. There is a predefined value of maximum disparity, and it must be defined according to the image resolution and if the cameras are frontal parallel or not. There may be missing features that are found in the left image but cannot be found in the right image, and vice-versa. this can be caused by occlusions and noise.

After correspondence algorithm, follows a postfiltering procedure which works removing false matches. Also can occur problems near the edges of the objects because in the window of one side is seen the foreground and in the other side is seen the background, resulting in a local region of large and small disparities called *speckle*. To avoid the edge matches, can be set a speckle detector with a speckle window. The speckle window works defining a pixel as a basis of making a component defined by a variable range. This variable includes a neighboring pixel only if it is within the range of the current pixel. When the connected component is calculated, the component is considered as speckle if it is smaller than the speckle window.

Some key terminology of the stereo correspondence is listed in [58] and complemented with [22]:

- Intensity **I** is the numerical value that represents the light intensity in an image location (pixel).

- Disparity is defined as simply $u - u'$ for a correspondence between $\mathbf{p} = (u, v)$ and $\mathbf{q} = (u', v')$ where $v = v'$ is assumed when the images are rectified. This term was used for first time for human vision, which describes the difference in location between two object correspondences seen by the left and right eyes. Frequently disparity is treated as the inverse of depth.

- Disparity space (u, v, d) is the fundamental concept of stereo correspondence methods.

- Disparity map for I_{ref} is an image made by the function $d(u, v)$, where $(u, v) \in I_{ref}$.

- Occlusions are those points of a scene that does not have a real correspondence in the both images. Common occlusion is a point that is visible from any of the two images but in the other image is not because there is another object closer to the cameras that is blocking the field of view.

- Calibration, is the information known about the cameras, which provides high-quality estimates of the epipolar geometry and better estimation of disparity

- Block-matching is the process of forming a small block around each pixel $(u, v) \in I_{ref}$ and find the disparity d that minimizes some cost function of a corresponding block around $(x - d, y) \in I_{match}$.

- Dense correspondence is the computation of each pixel correspondence in the images.

In [22] the authors made a taxonomy of stereo correspondence algorithms in general that can be described by the following steps:

- Matching cost is the computation at each pixel for all disparities under consideration. Every stereo correspondence algorithms uses a matching cost to measure

the similarity of image locations. The simplest examples of matching costs assume constant intensities at matching images locations, like absolute differences and squared differences.

- Cost aggregation is done by summing matching cost over the pixels surrounding a central pixel, commonly performed in a squared area, called window. Some examples of window-based matching cost are the sum of absolute differences (SAD), sum of squared differences (SSD), normalized cross-correlation (NCC) and census transform.

- Disparity computation/optimization is the selection of the disparity which corresponds to the lowest matching cost, i.e. winner-takes-all (WTA).

- Refinement of disparities consists on removing peaks, checking the consistency, interpolating gaps, or increasing the accuracy by sub-pixel interpolation (upsampling).

Stereo algorithms classification

Local algorithms, also called window-based algorithms, computes the disparity at a given point depending only on intensity values within a finite window. These methods makes emphasis on the matching cost computation and on the cost aggregation. To compute the final disparities simply choose at each pixel the disparity associated with the minimum cost value. Then, a WTA local optimization is performed at each pixel. However, the inconvenience of this algorithm is the uniqueness of matches is only enforced for the reference image (usually the left one) while points in the other image might get matched to multiple points.

Global algorithms make explicit smoothness assumptions and then solve an optimization problem. Typically does not perform an aggregation step, but rather seek a disparity assignment that minimizes a global cost function that combines data for

matching cost and smoothness terms. Many global methods are formulated in an energy-minimization framework. The objective is to find a disparity function d that minimizes a global energy

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d)$$

where $E_{data}(d)$ measures how well the disparity function d agrees with the input image pair. Its formulation is

$$E_{data}(d) = \sum_{(u,v)} C(u,v,d(u,v))$$

where C is the matching cost Disparity Space Image or disparity map. The smoothness term $E_{smooth}(d)$ encodes the smoothness assumptions of the algorithm, and is often restricted to only measuring the differences between neighboring disparities of pixel

$$E_{smooth}(d) = \sum_{(u,v)} \rho(d(u,v) - d(u+1,v)) + \rho(d(u,v) - d(u,v+1))$$

where ρ is a monotonically increasing function of disparity difference. Some methods use more terms for penalizing occlusions, alternatively treating visibility, enforcing a left/right or symmetric consistency between images or weight the smoothness term according to segmentation information.

Exists a Semi-Global Matching (SGM or SGBM) method [24] which is an approach between local and global matching, it can use a pixelwise matching of Mutual Information or Birchfield-Tomasi.

The pixel-wise matching cost calculation of this method needs two input images that are assumed to have a known epipolar geometry (see Subsection 3.2.1). The matching cost is calculated using a reference image pixel $\mathbf{p}$ from its intensity I_{rp} and the suspected correspondence I_{mp} with $\mathbf{q} = e_{rm}(\mathbf{p}, d)$ of the match image. The function $\mathbf{q} = e_{rm}$ symbolizes the epipolar line in the match image for the base image pixel $\mathbf{p}$

with the line parameter d.

SGM aims to minimize a global 2D energy function $E(D)$ by solving a large number of 1D minimization problems, the energy used is

$$E(D) = \sum_{p}(C(\mathbf{p}, d_p) + \sum_{q \in N_p} P_1 T[|d_p - d_q| = 1] + \sum_{q \in N_p} P_2 T[|d_p - d_q| > 1]) \qquad (3.27)$$

where $C(\mathbf{p}, d_p)$, as defined before, is a pixel-wise matching cost, $\mathbf{p}$ is the vector of the pixel (u, v) in the reference image I_{ref} and d_p is its correspondent disparity. The function $T[]$ returns 1 if its argument is true and 0 otherwise, so it penalizes small differences of neighboring pixels N_p of $\mathbf{p}$ with the cost P_1. In similar manner, the third term of Equation (3.27) penalizes larger disparity steps (discontinuities) with a higher penalty P_2.

It is important the size and shape of the area that is considered for matching, since the robustness of matching increases with large areas. However, the implicit assumption of constant disparity inside the area is violated at discontinuities, causing blurred objects borders and fine structures. So the constant disparities of the neighborhood of $\mathbf{p}$ is discarded, so only the intensities I_{rp} and I_{mq} itself can be used for matching cost computation.

Birchfield-Tomasi measure

Among the matching cost methods used in the stereo community, one of the most widespread algorithms is the Birchfield-Tomasi (BT) which measures dissimilarity of pixels which is insensitive to image sampling since it uses the linearly interpolated intensity functions surrounding the pixels. As a note, must be mentioned that this explanation and the Figure 3.7 is based on the original paper of S. Birchfield and C. Tomasi [59], where the imaged point is represented by the vector $\mathbf{p} = (x, y)$, which come into conflict with the convention of $\mathbf{p} = (u, v)$, so after the explanation the convention

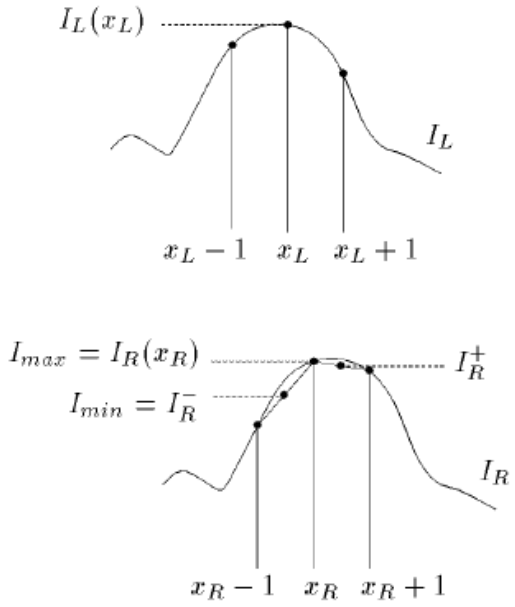


Figure 3.7: Computation of dissimilarity [59].

of this thesis is taken again.

The cost $C_{BT}(\mathbf{p},d)$ is calculated using the absolute minimum difference of intensities in $\mathbf{p}$ and $\mathbf{q} = e_{rm}(\mathbf{p},d)$. Let I_L and I_R be the intensity functions and it is sampled at discrete points by the image sensor. The goal is to calculate the dissimilarity between a pixel at position x_L in the left scanline and the pixel in the position x_R. Then $\hat{I}_R$ is defined as the linearly interpolated intensity function between the sample points of the right scaline. This function is defined in terms of I_R^- and I_R^+ (see Figure 3.7)

$$\begin{aligned} I_R^- &= \hat{I}_R(x_R - \tfrac{1}{2}) = \tfrac{1}{2}(I_R(x_R) + I_R(x-1)) \\ I_R^+ &= \hat{I}_R(x_R + \tfrac{1}{2}) = \tfrac{1}{2}(I_R(x_R) + I_R(x+1)) \end{aligned} \tag{3.28}$$

We then define the same for $\hat{I}_L$. Now the dissimilarity between a pixel at position x_L and the pixel x_R, is defined as

$$\bar{d}(x_L,x_R,I_L,I_R) = \max(0, I_L(x_L) - I_R^{max}(x_R), I_R^{min}(x_R) - I_L(x_L)) \tag{3.29}$$

where $I_R^{min}(x_R)$ and $I_R^{max}(x_R)$ are the minimum and maximum, respectively, of the intensities $I_R(x_R)$ and the values of its interpolated values $I_R^-(x_R)$ and $I_R^+(x_R)$

$$\begin{aligned} I_R^{min}(x_L) &= \min(I_R^-, I_R(x_R), I_R^+) \\ I_R^{max}(x,y) &= \max(I_R^-, I_R(x_R), I_R^+) \end{aligned} \tag{3.30}$$

and are similarly defined for $I_L^{min}(x_L)$ and $I_L^{max}(x_L)$ in the right side.

The dissimilarity D between the pixels is defined symmetrically as the minimum of the two quantities

$$D(x_L, x_R) = \min(\bar{d}(x_L, x_R, I_L, I_R), \bar{d}(x_R, x_L, I_R, I_L)) \tag{3.31}$$

rewriting the function is

$$\begin{aligned} D(x_L, x_R) = \min(&\max(0, I_L(x_L) - I_R^{max}(x_R), I_R^{min}(x_R) - I_L(x_L)), \\ &\max(0, I_R(x_R) - I_L^{max}(x_L), I_L^{min}(x_L) - I_R(x_R))) \end{aligned} \tag{3.32}$$

Until now, we have omitted the y and disparity d in order to simplify the definition of dissimilarity, but we need to express the cost as a function of the dissimilarity. First of all, we need to change the variables $x_L = x$ and $x_R = x - d$ in the dissimilarity function. Then need to check every pixel in a window with center pixel (x_0, y_0) and with a constant disparity and sum all the minimum values. The matching cost is $C_{BT}(x_0, y_0, d) = \sum_{(x,y)\in W} D(x, y, d)$, i.e.

$$\begin{aligned} C_{BT}(x_0, y_0, d) = \sum_{(x,y)\in W} \min(&\max(0, I_L(x,y) - I_R^{max}(x-d, y), I_R^{min}(x-d, y) - I_L(x,y)), \\ &\max(0, I_R(x-d, y) - I_L^{max}(x,y), I_L^{min}(x,y) - I_R(x-d, y))) \end{aligned}$$

Taking in count the convention of the image point $\mathbf{p} = (u, v)$

$$C_{BT}(u_0, v_0, d) = \sum_{(u,v)\in W} \min(\max(0, I_L(u,v) - I_R^{max}(u-d,v), I_R^{min}(u-d,y) - I_L(u,v)), \\ \max(0, I_R(u-d,v) - I_L^{max}(u,v), I_L^{min}(u,v) - I_R(u-d,v))) \tag{3.33}$$

3.3 Kalman Filter

The Kalman Filter was published in [60] by Rudolf Emil Kalman in 1960, and from then on, it has been extensively applied in autonomous navigation. The KF is a set of equations that computes the estimated values of the state of a process. It is able to estimate past, present and future states, even if the modeled system is not precisely known. This computation is performed by iterations in the time, with a specified period.

According to the simplified lecture note [61] the following description of the simple Kalman Filter assumes that the system state equation has the following general linear form:

$$\mathbf{x}_t = \mathbf{A}_t \mathbf{x}_{t|t-1} + \mathbf{B}_t \mathbf{u}_t + \mathbf{w}_t \tag{3.34}$$

where $\mathbf{x}_t$ is the state vector which contains the interesting terms of the system (e.g., position, velocity, acceleration) at time t; $\mathbf{u}_t$ is the vector containing the control inputs (steering angle, throttle setting, braking force); $\mathbf{A}_t$ is the state transition matrix which applies the effect of each system state parameter at time $t-1$ on the system state at time t (e.g., the position and velocity at time $t-1$ both affect the position at time t); $\mathbf{B}_t$ is the control input matrix which applies the effect of each control input parameter in the vector; $\mathbf{u}_t$ on the state vector (e.g.,applies the effect of the throttle setting on the system velocity and position); $\mathbf{w}_t$ is the vector containing the process noise terms for each parameter in the state vector. The process noise is assumed to be drawn from a zero mean multivariate normal distribution with covariance given by the covariance

matrix $\mathbf{Q}_t$.

$$p(\mathbf{w}_t) \sim N(0, \mathbf{Q}_t)$$

Also the KF assumes a system measurement model that has the form

$$\mathbf{y}_t = \mathbf{C}_t \mathbf{x}_t + \mathbf{v}_t \tag{3.35}$$

where $\mathbf{y}_t$ is the vector of measurements; $\mathbf{C}_t$ is the transformation matrix that maps the state vector parameters into the measurement domain; $\mathbf{v}_t$ is the vector containing the measurement noise terms for each observation in the measurement vector. Like the process noise, the measurement noise is assumed to be zero mean Gaussian white noise with covariance $\mathbf{R}_t$.

$$p(\mathbf{v}_t) \sim N(0, \mathbf{R}_t)$$

The Kalman filter algorithm consists of two stages: prediction and measurement update. For prediction stage, the equations are

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{A}_t \hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t \mathbf{u}_t \tag{3.36}$$

$$\mathbf{P}_{t|t-1} = \mathbf{A}_t \mathbf{P}_{t-1|t-1} {\mathbf{A}_t}^T + \mathbf{Q}_t \tag{3.37}$$

The measurement update stage consists of the equations

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t (\mathbf{y}_t - \mathbf{C}_t \hat{\mathbf{x}}_{t|t-1}) \tag{3.38}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{K}_t \mathbf{C}_t \mathbf{P}_{t|t-1} \tag{3.39}$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} {\mathbf{C}_t}^T (\mathbf{C}_t \mathbf{P}_{t|t-1} {\mathbf{C}_\mathbf{t}}^T + \mathbf{R}_t)^{-1} \tag{3.40}$$

The Equation (3.36) can be called State Extrapolation equation, Prediction equation or more commonly Transition equation. This extrapolates the current state of the system

to the next state, makes a prediction.

The Equation (3.37) is obtained in a similar way to the state extrapolation, the estimate uncertainty extrapolation is done with the dynamic model equations. The estimate uncertainty extrapolation equation is called Covariance Extrapolation Equation.

The Equation (3.38) is called the State Update equation where $\mathbf{K}_t$ is called the Kalman Gain, and the subscript t indicates that this gain can change with every iteration. The term $(\mathbf{y}_t - \mathbf{C}_t\hat{\mathbf{x}}_{t|t-1})$ can be called "innovation" because it contains new information. Since this formula is based on measurements, the first iteration can be performed with an "initial guess" that can be just a reference value or approximated value.

The Equation (3.39) updates the estimate uncertainty of the current state. It is called the Covariance Update Equation.

The Equation (3.40) is called the Kalman Gain equation, and it tells how much it needs to change the estimate by a given measurement. It follows a restriction

$$0 < \mathbf{K}_t < 1.$$

This gain has an effect on Equations (3.38) and (3.39) of weighting between measurement and estimated values, i.e., if Equation (3.38) is reformulated as

$$\hat{\mathbf{x}}_{t|t} = (1 - \mathbf{K}_t\mathbf{H}_t)\hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\mathbf{z}_t$$

is evident that $\mathbf{K}_n$ is the weight given to the measurement, and the $(1 - \mathbf{K}_t\mathbf{C}_t)$ is the weight given to the estimate. In the case of Equation (3.39) when it is reformulated it is obtained

$$\hat{\mathbf{P}}_{t|t} = (1 - \mathbf{K}_t\mathbf{C}_t)\hat{\mathbf{P}}_{t|t-1}$$

it is observed that as $\mathbf{K}_t$ increases the measurement uncertainty decreases. A complete

derivation can be studied in details in [61], [62] and [63].

3.4 Robotic Operating System

It is a meta-operating system for general purpose computers including Windows, Linux, Mac, Android, iOS, among others. The ROS is a framework that consists of libraries, tools and conventions that aim to simplify the development of robotic systems. ROS is focused on maximizing code reusing in the robotics research and development, taking the collaboration in robotics to a global level [64].

Next is briefly defined the used ROS terms for this work

- Node: Refers to the smallest unit of processor running in ROS. It is like a executable program with a single purpose assigned (as recommended) to each node. Examples of purposes are a sensor drive, sensor data conversion, obstacle recognition, motor drive, encoder input, etc.

- Package: is the basic unit of ROS, and contains either a configuration file to launch other packages or nodes.

- Message: It is the communication via between nodes. The type of data can be of type integer, floating point and boolean.

- Topic: it is literally like a topic in a conversation. It is created by a publisher node and can publish messages through them. Subscriber nodes can request to connect to the publisher node to exchange messages as a topic.

CHAPTER 4

Methodology

In this section is presented the system overview, the information acquisition processes for 1D-LiDAR and stereo camera rig. Also, the implemented KF is presented as a sensor fusion method. Lastly, the obstacle methodology is going to be described.

4.1 System overview

The system is conformed by sensors, microcomputers, software and a quadrotor UAV. In this part we begin by describing the sensors, particularly the stereo camera rig and the 1D-LiDAR. How the provided information by these sensors is processed and interpreted is presented next. Later, we present results about implementation.

First, consider the pair of images provided by stereo camera. For each frame two measurements are captured and then processed according to the system overview shown in Figure 4.1. The KF have the aforementioned measurements as inputs which corre-

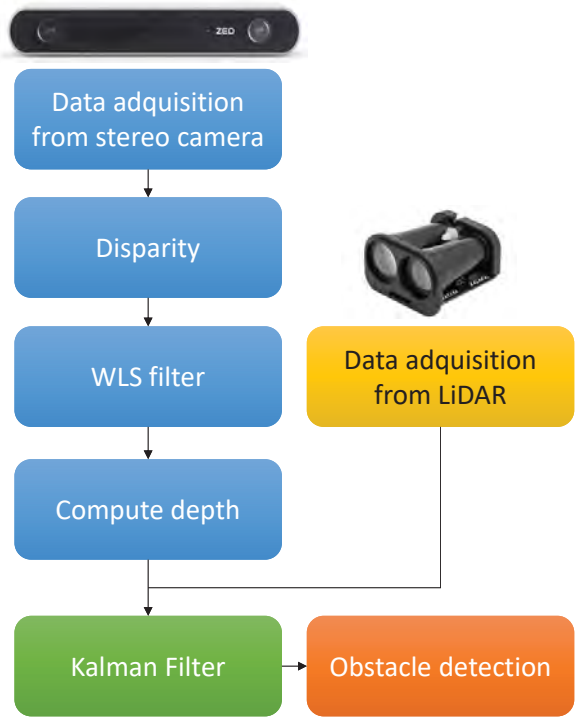


Figure 4.1: System overview. Blue blocks: Camera processes. Yellow block: 1D-LiDAR processes. At bottom, both information are fused in a Kalman Filter.

spond to the disparity map and the distance acquired by the 1D-LiDAR. After that, inside the depth map, it is computed a rectangle with the height and width corresponding to the dimensions of our UAV at a determined distance in front the UAV. The dimensions of our UAV is about of 120 cm of width and 40 cm of height. The center of the image captured corresponds to the same physical point that the 1D-LiDAR measures as distance. Then, the rectangle is an area that must be free of any obstacles to avoid any possible collision. If we are able to measure the distance with accuracy, the size of quadrotor can be projected ahead and determinate if the UAV can access freely across of the rectangle.

4.2 Platform setup

The UAV is a quadrotor Tarot XS690 quadrotor frame, as can be seen in the Figure 4.2. As stereo camera rig we have mounted a ZED camera developed by Stereolabs, which is attached in the upper part of the quadrotor. This stereo rig has a specified range up to 40 meters and a number of third party support, among them, ROS and OpenCV, where we have coded the obstacle detection algorithm. Particularly we have used the ZED camera node provided by ROS, which gives the rectified right and left images that are required to compute the disparity at a frame rate of 29.9 fps. The disparity and autonomous operation is performed on-board with a Jetson TX2 development kit, a embedded computing board developed by Nvidia that stands out for being designed for machine learning applications. A laser distance sensor SW20/C [65] is fixed near to the left and right camera. Both cameras and sensor are positioned facing forward, in x direction of the quadrotor.

It is worth mentioning that ZED camera driver needs the Jetson computer to have installed JetPack 3.3 [66] which includes CUDA toolkit. In order to remove bad-matched pixels that causes noise in the disparity map, is implemented a postfilter called WLS filter. This filter needs to have OpenCV version 3.4.2 built together with the Extended Image Processing module (ximgproc), that is included in OpenCV Contrib repositories. The builder used for this purpose is the CMake. Nevertheless, WLS filter decreases significantly the frame rate of disparity map computing, so the algorithm is written in C++ language, since it has a faster performance than Python in real-time image processing.

4.3 Nvidia Jetson TX2

The data given by the sensors must be processed, this requires a fast processing in order to reduce the latency and increase the runtime of obstacle detection, the algorithm

Figure 4.2: Tarot XS690 quadrotor.

was then considered to be performed by the NVIDIA Jetson TX2 which is one of the best high processing embedded computers designed specially for artificial intelligence in autonomous machines, whose specifications could be seen in [67]. Moreover the Jetson TX2 is mounted over a Development Kit (JDK) [68] that exposes the hardware capabilities and interfaces of the Jetson TX2 Module. In Figure 4.3 we can observe the computer and sensors conected via USB.

Jetson TX2 comes with preinstalled Ubuntu 16.04, with optional-flashed NVIDIA JetPack, a Software Development Kit that includes libraries for deep learning, computer vision, GPU computing and multimedia processing.

4.4 OpenCV

OpenCV is an open source computer vision and machine learning software library, and can be used in a variety of programming languages, including Python and C++,

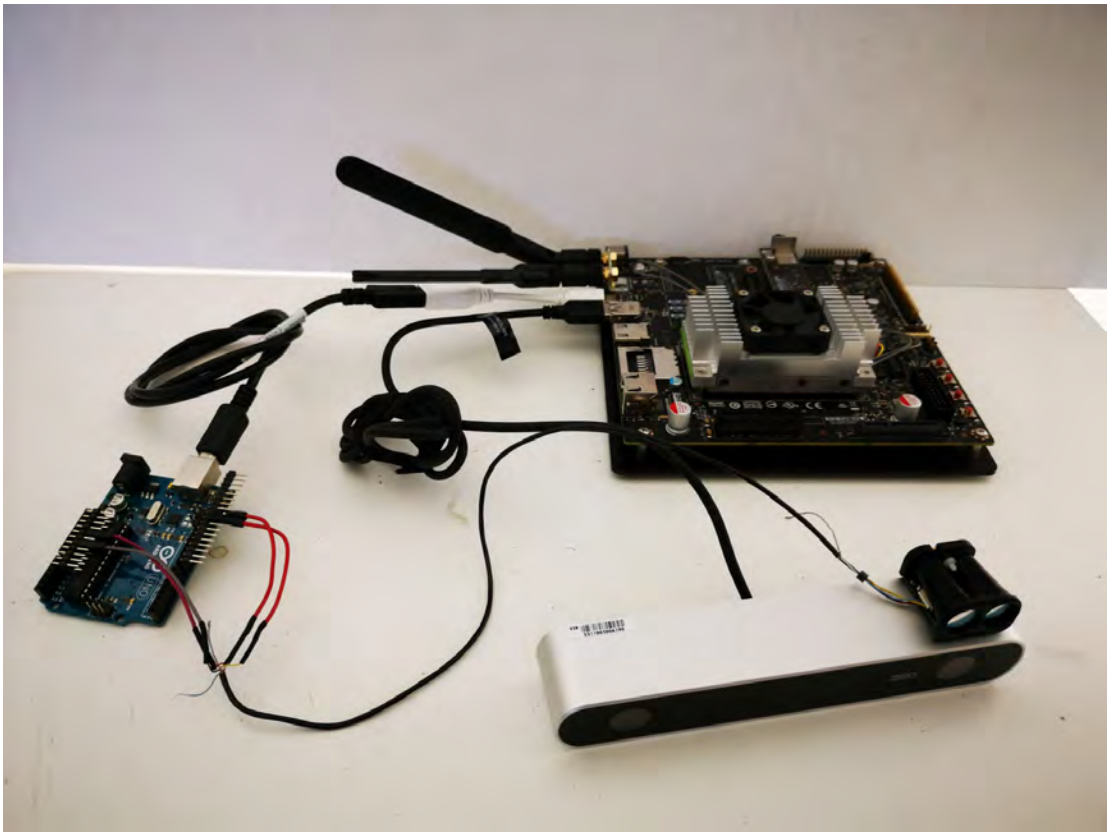


Figure 4.3: Components: ZED camera, 1D-LiDAR sensor, Jetson TX2 and Arduino UNO.

supporting Windows or Linux operating systems. Python was contemplated as a first option for stereo matching because of its extended use in artificial intelligence, however C++ was used instead Python because of faster runtime. The faster runtime of the stereo vision, the faster can detect and avoid obstacles, on the other hand, for using GPU processing Python requires additional libraries and compilers, C++ has direct access to GPU and more support from OpenCV and NVIDIA.

4.5 Stereo camera

The ZED camera is a stereo rig that includes a SDK that makes more efficient the development of applications with its stereo camera, taking in count the third-party support by ROS and OpenCV, in addition, SDK supports Windows and Linux operating systems.

It is worth mentioning that the JDK needs JetPack 3.3 [66] which includes CUDA toolkit that is necessary to install the ZED drivers and its SDK.

Once ROS Kinetic is set as a framework, ZED SDK can be installed and run as a node called *zed_ros_wrapper*. This node publishes to several topics, among them, the interesting ones for this work are image type and camera info nodes.

4.5.1 Stereo Correspondence

Once the right and left images are accessed, the following process is stereo correspondence. In Section 3.2, it is defined some of the necessary stereo matching parameters that needs to be specified for processing the stereo images, in this Section is going to be presented some extra parameters. OpenCV presents two stereo algorithms, the first one is Block Matching (BM) presented as StereoBM using a simple block matching similar to the proposed by K. Konolige, [69] with SAD cost function. The disadvantage of this algorithm is the low detection of features in low-textured scenes, like indoors. The other algorithm is Semi-Global Block Matching, and is presented as an object called StereoSGBM [70] which is based on the original paper of H. Hirschmuller [24] but using a Birchfield-Tomasi cost function [59] which is mentioned in the paper as a simpler option to the Mutual Information cost function. SGBM algorithm was chosen as the stereo correspondence method despite this algorithm is slower than BM algorithm, but it is more accurate.

The StereoSGBM parameters list is the following:

- *minDisparity*: the smallest disparity allowed, the value is stablished as 0, because

it corresponds to a far object detections.

- *numDisparities*: the maximum disparity allowed, and it should be greater than min Disparity. It limits the nearer object detections.

- *blockSize*: the matched block size. It must be an odd value because the block size should have a central pixel. In the original paper, the author suggests to use 7 as maximum value.

- *P1*: corresponds to the P_1 parameter, and controls the disparity smoothness. Its suggested value is according to the following: $8*image\_channels*SADWindowSize*SADWindowSize = 8*3*3*3 = 216$

- *P2*: controls the disparity smoothness. The larger the values are, the smoother the disparity is. The algorithm requires P2 $>$ P1 . It is suggested set the value in: $32*image\_channels*SADWindowSize*SADWindowSize = 32*3*3*3 = 864$

- *disp12MaxDiff*: the maximum allowed difference (in integer pixel units) between the left-right and right-left disparity check. If the two checks are different by more than disp12MaxDiff, is considered as unknown. It is set to 1 to make sure that it is the correct matching.

- *preFilterCap*: this value is the parameter I_{cap} that limits the output of the pre-filtering stage. Its default value is 30.

- *uniquenessRatio*: Margin in percentage by which the best (minimum) computed cost function value should ”win” the second best value to consider the found match correct. Normally, a value within the 5-15 range is good enough. For this work is set to 15.

- *speckleWindowSize*: Maximum size of smooth disparity regions to consider their

noise speckles and invalidate. If it is set to 0 disables speckle filtering. Otherwise, set it somewhere in the 50-200 range. In this code its value is 0.

- *speckleRange*: Maximum disparity variation within each connected component. If speckle filtering is used, set the parameter to a positive value, it will be implicitly multiplied by 16. Normally, 1 or 2 is good enough. It is set to 2.

- *mode*: sets the mode of disparity directions, which can be 3, 5 and 8 directions. The 8-directions mode requires great amount of memory, so the set value is for 3 directions, which makes the algorithm faster.

4.6 1D-LiDAR

The sensor SF20/C [65], is a low cost sensor LRF manufactured by Lightware, it is specially designed for drones and robots and can be connected to different controllers using I^2C or serial protocol. This sensor has a measurement reach from 0.2m up to 100m in sunlit conditions. In scanning mode can made up to 388 measurements with a resolution of 0.01m and an accuracy of 10cm

It can be connected to the JDK using I^2C interface pins or using an Arduino controller as interface to USB protocol. The first choice needs the computer a special setup and takes longer. The second choice is the easiest and fastest manner to connect the sensor despite the direct connection to JDK is better to obtain the maximum scanning rate, however by time constraints it is left as a future work. The device can be observed above the stereo camera at the Figure 4.3, specifically on the left camera.

The data acquisition from laser distance sensor to the Jetson TX2 is performed with a code that reads data from I^2C bus of an Arduino Uno microcontroller and then sent to Jetson TX2 via USB. On the other hand hand, in ROS environment the acquisition is performed with Lidar Lite ROS package [71], although we have modified the code to our requirements.

4.7 Kalman Filter

Since the quadrotor is planned to stop to measure the distance of the object in front, the measurement model of the system it is going to be assumed that the object in front is static while the algorithm is measuring distance along the z axis.

$$\begin{aligned} x_{t+1} &= x_t + \Delta t \dot{x}_t \\ \dot{x}_{t+1} &= \dot{x}_t \end{aligned} \tag{4.1}$$

Therefore, the state vector is comprised of the position and velocity of the obstacle in relation to the quadrotor, this is

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} \tag{4.2}$$

This vector can be expressed in matrix form as

$$\begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ \dot{x}_{t-1} \end{bmatrix} + \mathbf{w} \tag{4.3}$$

In discrete time the state space model and measurement model are given as follows

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t|t-1} + \mathbf{B}\mathbf{u}_t + \mathbf{w} \tag{4.4}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v} \tag{4.5}$$

where the matrix $\mathbf{A}$ relates the state at the past time step $t-1$ to the state at the current step t state transition matrix is

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{4.6}$$

in which Δt is the sampling time, which is the time between each frame computed. The matrix $\mathbf{B}$ is the control input matrix. Since we have two sensors that measure the same variable, i.e. distance, the matrix $\mathbf{C}$ can be expressed as

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}. \tag{4.7}$$

The outputs are given by the sensor measurements, i.e. the 1D-LiDAR and the disparity map; the $(\mathbf{w}, \mathbf{v})$ are the process and measurement noise, respectively. It is assumed that both variables are independent (of each other), with zero mean and normal probability distributions

$$\begin{aligned} p(w) &\sim N(0, Q) \\ p(v) &\sim N(0, R) \end{aligned} \tag{4.8}$$

Due to the fact that the space and measurement model have been expressed as linear can be solved by a simple discrete Kalman filter. The value of the state $\mathbf{x}_t$ cannot be observed directly, is in this case when Kalman filter can provide an estimate of the real state $\hat{\mathbf{x}}_t$ by means of probability density functions.

The covariance matrix $\mathbf{R}_t = E[vv^T]$ of measurement vector y can be expressed as

$$\mathbf{R}_t = \begin{bmatrix} \sigma_z & 0 \\ 0 & \sigma_l \end{bmatrix} \tag{4.9}$$

where σ_z and σ_l are the variances of the depth map estimation and 1D-LiDAR sensor, respectively. This measurement noise matrix is diagonal, since we suppose the acquisition data from sensors is independent between both [72]. The covariance matrix $\mathbf{R}_t$ of the process is represented as 2×2 matrix, which seems to Equation (4.9), but the process covariance values are considered smaller than the measurement covariance matrix because as we stated, we assumed the measurement process when the quadrotor is

static and modeled with the Second Law of Newton. To simplify this we assumed the process variances equal to zero.

4.8 Depth map estimation

There are several algorithms under research for depth map estimation based on stereo cameras, some of them are faster than others, but there is a tendency that the faster the algorithm the less accuracy of the depth map. It is important to mention that in the quadrotor navigation, the algorithm efficiency is crucial. There are two important parameters for disparity map estimation: the quality of the depth map and the computation time. According to the Middlebury Stereo Evaluation [21], Semi-Global Block Matching [73] (SGBM) algorithm has an absolute average error of 14.3 pixels and 0.68 sec/megapixels which can be implemented with OpenCV [24] easily. The absolute average error is the absolute difference of disparity between the computed disparity map and the ground truth map, whereas the time of computation is the time it takes a one million of pixels disparity map to be computed. The SGBM algorithm is based on smoothness constrains applied on pixel-wise matching which removes outliers.

The approach used to achieve the disparity map from camera is performed through OpenCV libraries, which is based on Hirschmüller algorithm with some modifications added by OpenCV developers. Once the disparity is obtained we used a Weighted Least Squares filter [74] (WLS) to remove holes due to half-occlusions, in other words WLS filter fills the image to get a uniform segmentation. Depth is computed from the following equation

$$\mathrm{D} = \frac{fB}{d} \tag{4.10}$$

where D is the depth, f is the focal distance in mm, B is the baseline between both cameras in pixels, d is the disparity got previously in pixels. In (4.10) the depth depends on focal distance and camera baseline, which are intrinsic parameters of camera and

they are unique values for each camera rig. Then, depth depends on disparity and on the camera calibration matrix. According to epipolar geometry [56] the disparity is the difference in pixels between the projection of a 3D point in the right camera and the same 3D point projected on the left cameras, as the images were overlapped. If there is an erroneous disparity or calibration, the depth will be erroneous. For that reason, parameters are modified to achieve a better matching between two images obtained from the camera. The application of the WLS filter help to remove the NaN (Not a number) and Inf (infinity) values. WLS filter was implemented as a fast global smoother but running on CPU; the implementation on GPU is left for future work.

4.9 Obstacle detection

Let $\mathscr{W} \subset \mathbb{R}^3$ be the set of all points in a world coordinate system that are mapped inside the field of view (FOV) of the stereo camera. The FOV can include objects, some of them can be considered as obstacles if they are in the path space $\mathscr{P} \subset \mathscr{W}$. The obstacles are represented by the set $\mathscr{O}$. Then, if the following equality holds

$$\mathscr{O} \cap \mathscr{P} \neq \emptyset \tag{4.11}$$

means that the path is free to navigate. Let consider the filtered depth map explained in the previous section. There is a maximum reach distance of the depth map, named D_{max}, please refer to Figure 4.4 a) (page 56). Also, there is a *window* in which the UAV can pass trough without any obstruction shown at Figure 4.4 b) . Such window is at a distance D_{min} in the z-coordinate from the stereo camera, please see Figure 4.4 a).

On the other hand, in epipolar geometry [56] exists a function that maps 3D points to 2D points, called 3D projection. Every 3D point and set in the field of view has a projection to the 2D plane in the left camera sensor (by convention) of the stereo rig,

then

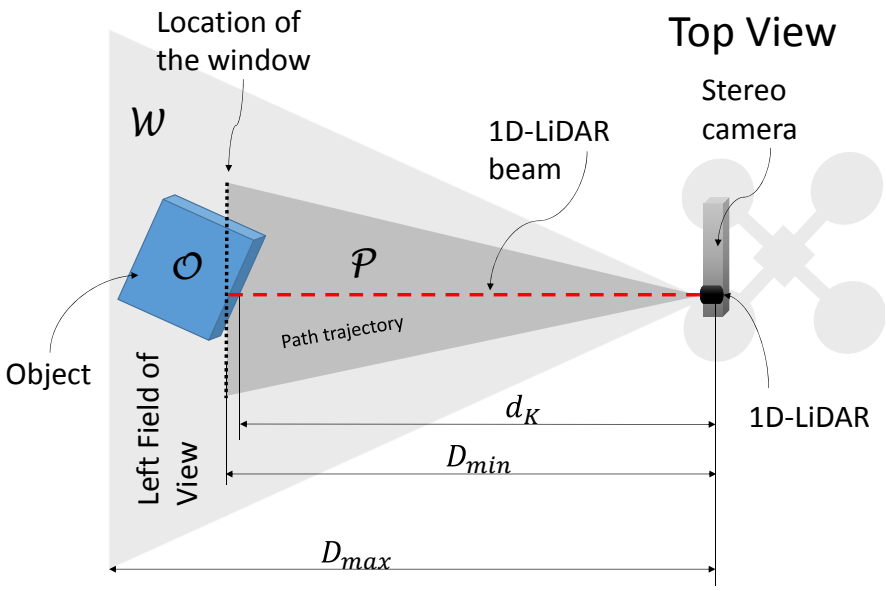
$$\mathbf{W}_i \mapsto \mathbf{w}_i$$

$$\mathbf{P}_i \mapsto \mathbf{p}_i$$

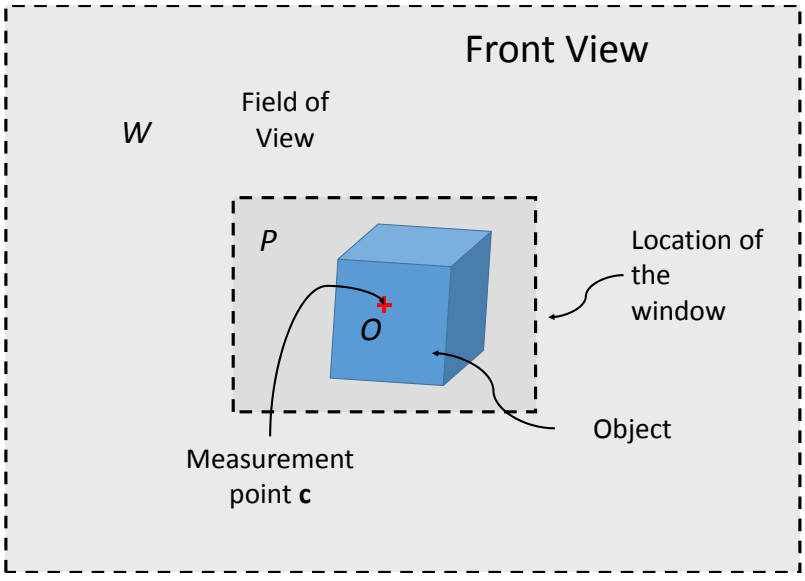
$$\mathbf{O}_i \mapsto \mathbf{o}_i$$

where $\mathbf{W}_i = (x_i, y_i, z_i) \in \mathscr{W}$ represents a point in the FOV; $\mathbf{P}_i \in \mathscr{P}$ is a point inside the path of the UAV; $\mathbf{O}_i \in \mathscr{O}$ is a point of a physical object; $\mathbf{w}_i \in W \in \mathbb{R}^2$ is a point of the FOV projected in the camera sensor; $\mathbf{p}_i \in P \in \mathbb{R}^2$ is a projected point of a path in the camera sensor, in other words, it is the window projection in the camera; and $\mathbf{o}_i \in O \in \mathbb{R}^2$ is a projected point of an obstacle, see Figure 4.4b . All the above for $i = \{$ 0,1,2,3,...$\}$.

The depth function of Equation (4.10) is used by means of the disparity between a point $\mathbf{p}_i$ seen from the right and the left camera sensors, in this way $D(\mathbf{p}_i)$ is the depth value in the position (u_i, v_i) of the depth map. Every value of (u_i, v_i) in the depth map represents the z_i coordinate value of the point $\mathbf{P}_i$ that is projected to the image seen with respect to the left camera. Since the approach only considers displacements in the z coordinate (forward direction) with respect to the UAV, it is proposed to use the window projection as a rectangular region centered inside the depth map. Such a region represents a free area in which the UAV can navigate. Then, $\mathbf{p}_i \in P \in \mathbb{R}^2$ represents the set of all points contained inside such a rectangular region, i.e. $P \subset W$. Inside the window region, we highlight the window centroid given by $\mathbf{c} \in P$ which is ideally the same point mapped to $\mathbf{P}_i$ that the 1D-LiDAR is pointing at. At $\mathbf{c}$, we get two measurements: d_L which is the distance measured by the 1D-LiDAR, and $D_c = D(\mathbf{c})$ which is the distance obtained from the depth map. When both measurements are fused together we obtain d_K, which is compared with d_{min}; if $d_K < d_{min}$ then the point $\mathbf{c} \in \mathscr{O}$, i.e. it is part of an obstacle.



(a) Top view of the system.



(b) Front view, what the camera sees.

Figure 4.4: Diagram representing the main elements of our approach. Part a) is the top view of the obstacle detection; and b) is the view seeing by the camera mounted on the drone.

CHAPTER 5

Experiments

In this section the experimental platform and the obtained results is presented with the approach seen in previous sections. The ROS general explanation is first explained because all the hardware and software implementation is made through ROS. The next subsections specifically talks about each ROS node.

5.1 ROS

The complete package created is shown in the Figure 5.1 The KF can be implemented in Robotic Operating System (ROS) with the contribution of D. Ratasich et al [75], which can be used for generic sensor fusion purposes. The sensor fusion node can be configured according to the requirements of the system, in this manner, it is just necessary to define the state space matrices including the process noise matrices and the initial state.

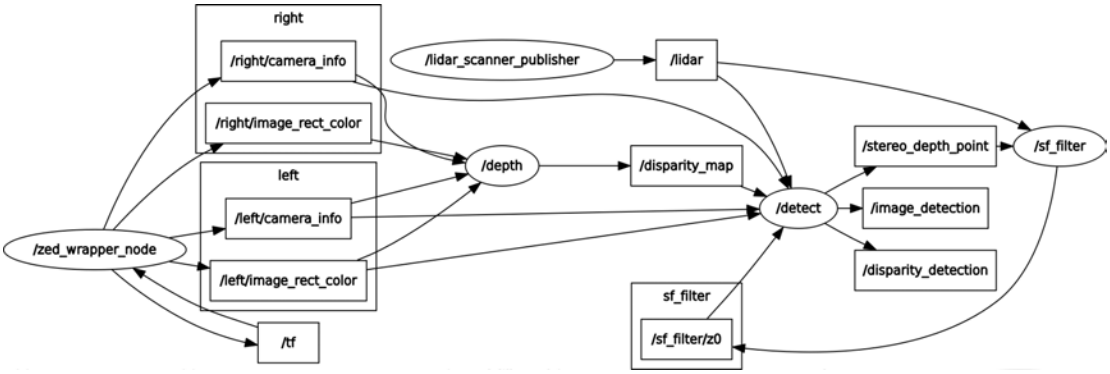


Figure 5.1: ROS nodes

The parameters of the KF were set in the node */sf_filter* accordingly to the model of our system explained in Section 4.7, using the Equations (4.6), (4.7) and (4.9), i.e. covariance matrices, inputs, outputs, matrix dimensions and states. Furthermore, the parameters of SGBM algorithm, implemented in node */depth* using OpenCV, were set as explained in the Subsection 4.5.1.

The ROS package used to get the stereo rectified images is the official one named *Zed Ros Wrapper*. In Figure 5.1 can be seen that the node */depth* subscribes to the topic */left/image_rect_color* and */right/image_rect_color* published by */zed_wrapper_node* (created by the stereo rig manufacturer), the */depth* node was developed for this work and its purpose is to perform the block matching algorithm in the stereo images. The images are rectified to correct errors in the lens structure and published in a ROS topic. The */depth* node also subscribes to the topics */left/camera_info* and */right/camera_info* in order to obtain the calibration matrix of the stereo camera and get the epipolar values of the stereo rig. The camera calibration parameters corresponding to each resolution are stored in the package *Zed Ros Wrapper*. Thus we can compute the 3D position pointed out by the center of the image, applying the method of 3D reconstruction theory [56]. Meanwhile, the node */lidar_scanner_publisher* from the package */lidar_scanner_driver* publishes the topic */lidar* with the distance measure of the approximated center of the

image. This node is in charge of receive the data from the USB port which is transmitted by the 1D-LiDAR.

There is a node called */sf_filter* which is in charge of mixing the sensor measurements coming from the */lidar* and */stereo_depth_point*. As mentioned before, this node uses Kalman Filter algorithm to fuse the data that is published through the topic */sf_filter/z0* whose subscriber is the node */detect* which is in charge of work with the window of depth seen in section 4.8 and with the fused measurement of the center of the window of depth as described in section 4.9.

5.2 Variance measurement

The covariance matrices were computed taking samples with the camera in a period of time whose center was pointing at the same point that laser sensor, and using the statistics covariance formula [76].

$$\sigma^2 = \frac{\sum(x_i - \bar{x})^2}{n-1} \tag{5.1}$$

where n is the number of samples; x_i is the measurement; and $\bar{x}$ is the expected value. We obtain both measurement noise. In this calculation, we fixed the stereo camera and the 1D-LiDAR. For stereo depth measurement 150 samples was taken, pointing at a fixed object at a distance of 3.75 meters and taking measurements at 8.62 frames per second. The frame rate was taken with the help of the ROS command *rostopic hz /stereo_depth_point*.

The samples were taken from the ROS node */detect* which is in charge of read the measurements of the topics */lidar* and */disparity_map* and then display their values in terminal console of Ubuntu. That values was used to calculate the variances using the formula 5.1 in a spreadsheet. The Figure 5.2 shows the graph of the samples taken to obtain the variance of the stereo depth in the center of the window, when it is pointing

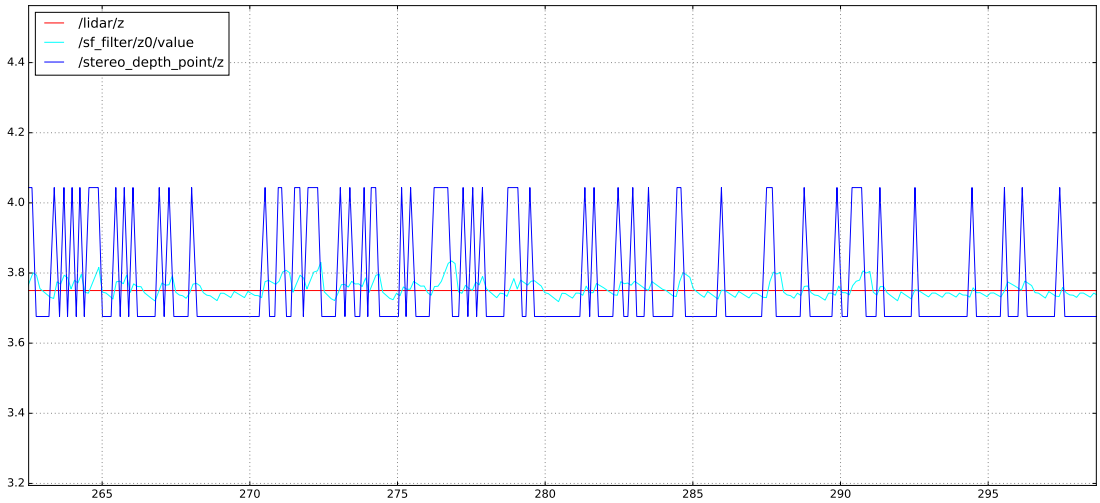


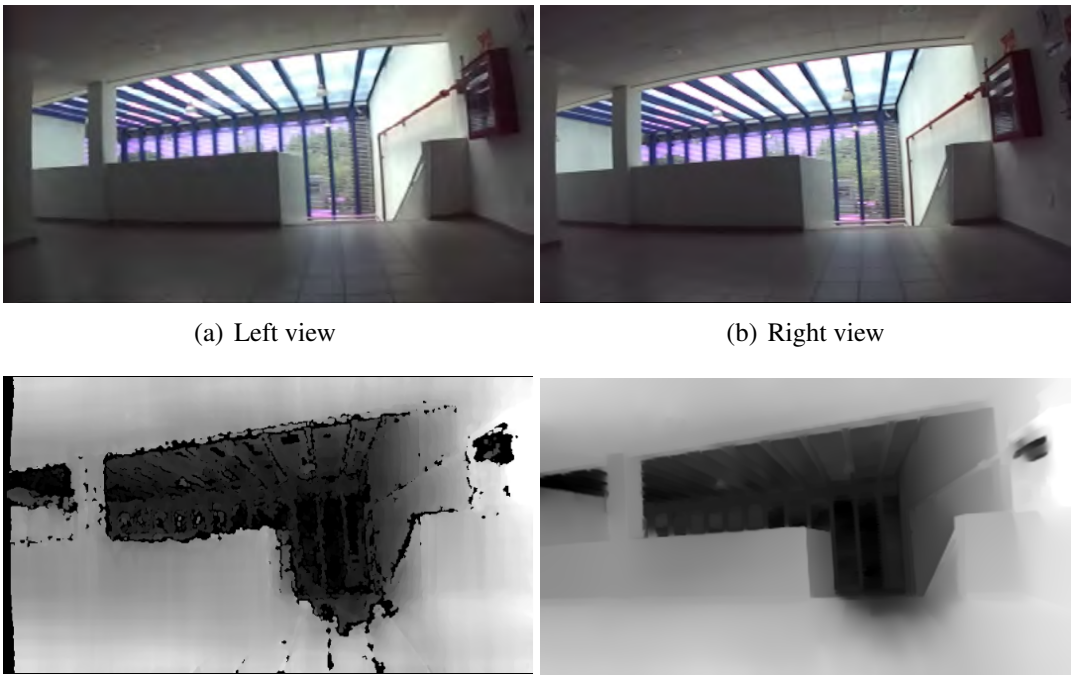
Figure 5.2: Graph of the static measurement of variance.

at a low-texture area which causes noise. However when the object has enough texture, the stereo depth value does not change in that magnitude.

According to the experiments it has a minimum detection distance of 8cm, and its accuracy changes depending on the surface the laser beam is pointing at. When it is an opaque surface, its accuracy is much less than 10cm, but if the surface is more or less transparent, the distance accuracy is affected some times more than 10cm; for instance, when measuring the distance to a computer monitor, of measuring through a glass window, etc. The 1D-LiDAR position is near the left camera since the disparity and distance measuring must match with the same object to the extent as possible.

5.3 Depth map ameliorate

The results obtained with disparity algorithm was conducted with OpenCV using three processes to an image of 672x376 pixels: rectification, StereoSGBM function, WLS filter. With the help of the ROS tool RViz which is a visualizer of sensor data, as presented in Figure 5.3 (a) and (b) where is presented the difference between the left



(a) Left view (b) Right view

(c) Disparity map before WLS filter (d) Disparity map after WLS filter

Figure 5.3: Raw image and rectified image (a) and (b). Disparity with and without WLS filter, using RViz to visualize the images (c) and (d).

camera view and the same scene from the right camera view. The next stage is to compute the dense stereo matching, which produces a disparity map, shown in the Figure 5.3 (c) and (d) where is illustrated the same images after being filtered with WLS for disparity maps. It can be observed that the non-filtered disparity map has black edges which are the occlusions, and the black point widespread in the image are bad-matched pixels, therefore, it generate bad measures of depth when they appear in the center of the map, which usually are caused by NaN, negative or infinity values. Nevertheless, the images with a complete disparity without black spots show smoother images with a better disparity, this means that the computation of depth map has less noise.

(a) No obstacle inside window in RGB. (b) No obstacle inside window in depth map.

(c) Obstacle inside window in RGB (d) Obstacle inside window in depth map

Figure 5.4: Examples of detection of obstacle in exterior, using RViz to visualize the images.

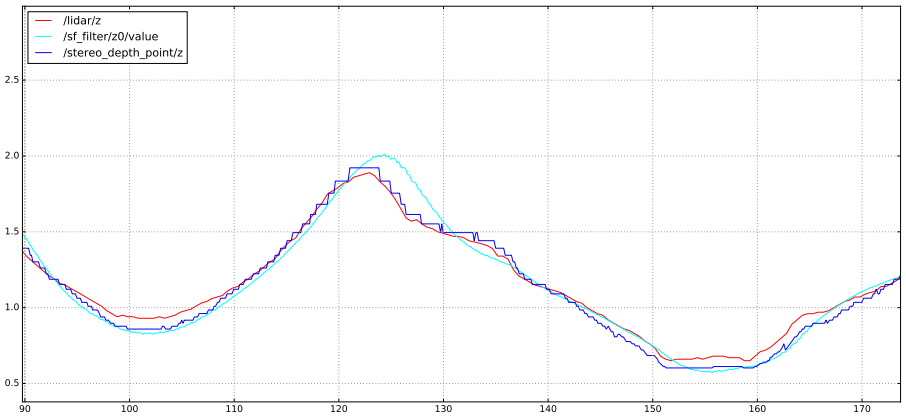
5.4 Obstacle detection

The stage of obstacle detection algorithm was written in the ROS node */detect*, and the topics that it publishes is presented using RViz, as shown in Figure 5.4, here is shown the results of obstacle detection in exterior when the detection window is 3 meters ahead from the quadrotor. In a) and b) the quadrotor is looking between two trees with no detection of obstacles. But if we put an obstacle inside the center and from 1.5 meters away, it is detected as an obstacle in that flying direction.

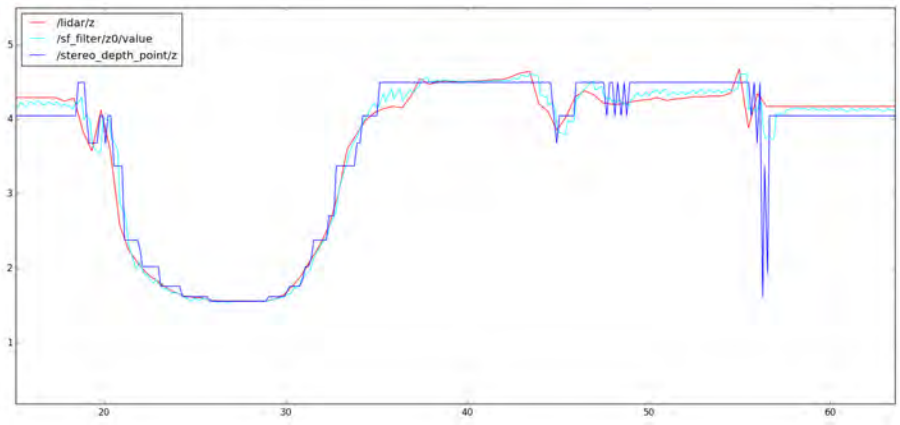
5.5 Fusion Sensor

Figure 5.5 shows the behavior of depth respect to disparity, 1D-LiDAR and Kalman Filter (KF) implementation; the distance is expressed in the x-axis in meters. The plot depicted at Figure 5.5 (the graphs was taken using the ROS tool *rqt_plot*), where (a) shows a graph where the depth stereo resolution changes according to the sensing distance from the object causing a stepped-shape line, where the more distance of detection the higher is the ”step” of the graph. However the Kalman estimation is still not configured correctly, due to the period should be bigger, causing a smoother shape that appears not following the lidar graph. In Figure 5.5(b) is showing the shape of the graph when the period is 0.116 seconds, how ever the shape of the estimated value is noisy due to a big values of the elements of the process covariance matrix, near to 1. The noise increases when the difference between both input signals diverge. In Figure 5.5(c) is observed a graph with less noise using a process noise with diagonal elements equals to 0.25, which is a value obtained with trial and error. There are many methods based on the work [77], but this process of estimation is left as a future work.

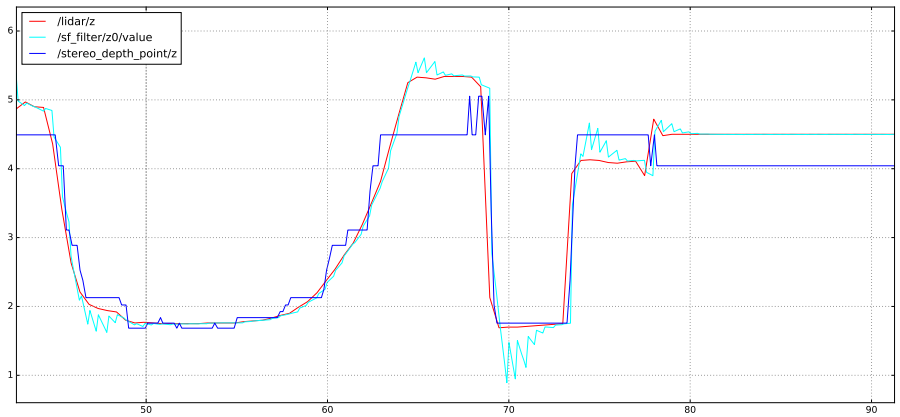
Depth obtained from ZED is noisier compared with the signal from 1D-LiDAR, so the covariance from ZED is bigger than covariance from 1D-LiDAR. Observe the effects of having a smaller 1D-LiDAR covariance than the disparity map coming from the stereo camera rig. At this point, the advantage of using disparity map instead of only 1D-LiDAR is that with the help of disparity map we can provide to the system information regarding visual perception. Also, we can extend the covered area of the potential obstacle; the 1D-LiDAR is used only to provide more precision to our estimation of free-obstacle area. The Figure 5.5 (b) and (c) shows a peak caused by the computation of depth from stereo camera does not largely affect the filter response.



(a)



(b)



(c)

Figure 5.5: Blue line represents the output of disparity, red line corresponds the output of 1D-LiDAR and cyan line shows the output from Kalman filter.

CHAPTER 6

Conclusions

This work presents a low-cost implementation to estimate a free navigation area in front of a quadrotor that is interpreted as a free-obstacle navigation area. For that aim, it is proposed a Kalman-filter-based algorithm that uses information from a stereo camera rig and a 1D-LiDAR. The results show that the estimation of free obstacle areas are considerably ameliorated with this approach since: a) using only a 1D-LiDAR the quadrotor cannot have any visual perception of their environment; and b) using only disparity maps conducts to a noise response. Then, mixing these sensors in an appropriate Kalman Filter together with WLS filter results in more trustworthy depth map for obstacle navigation inside a depth window, which is a predefined area depending on the size of the drone in which this can navigate freely.

The implementation of the WLS filter greatly removes noise of the depth measuring, as we can see from Figure 5.5, where it is shown that the depth map does not have any big peaks or NaN values. However, the distance becomes more imprecise when the

distance increases and shows small peaks that represents the increase of a pixel in the disparity map. This is caused by the small distance between the pixel where is an object in the left image and the right image. So, while more depth, minus disparity and more noise. In this case the distance sensor does not suffer that problem. The graphs showed that the estimated measurement approaches to the laser sensor due to its variance was smaller than the depth map. In this way, we can compute with more precision the depth inside the depth window where is safe for the quadrotor to pass through.

There are some areas of improvement and points that cannot be finished due to time restrictions, the most important of them is the experimental implementation of the proposed obstacle avoidance approach working together with a navigation algorithm in real environments. Regarding computing time, the implementation of our algorithm in GPU instead of only CPU would increase the obstacle detection rate.

Bibliography

[1] A. Zhahir, A. Razali, and M. R. M. Ajir, "Current development of UAV sense and avoid system," *IOP Conference Series: Materials Science and Engineering*, vol. 152, p. 012035, oct 2016. [Online]. Available: https://doi.org/10.1088%2F1757-899x%2F152%2F1%2F012035

[2] B. N. Chand, P. Mahalakshmi, and V. P. S. Naidu, "Sense and avoid technology in unmanned aerial vehicles: A review," in *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, Dec 2017, pp. 512–517.

[3] G. Fasano, D. Accado, A. Moccia, and D. Moroney, "Sense and avoid for unmanned aircraft systems," *IEEE Aerospace and Electronic Systems Magazine*, vol. 31, no. 11, pp. 82–110, November 2016.

[4] S. Hottman, K. Hansen, and M. Berry, "Literature review on detect, sense, and avoid technology for unmanned aircraft systems," 01 2009.

[5] A. D. Zeitlin, "Performance tradeoffs and the development of standards," 2012.

[6] T. B. Billingsley, M. J. Kochenderfer, and J. P. Chryssanthacopoulos, "Collision avoidance for general aviation," *IEEE Aerospace and Electronic Systems Magazine*, vol. 27, no. 7, pp. 4–12, July 2012.

[7] E. Valovage, "Enhanced ads-b research," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 22, pp. 35 – 38, 06 2007.

[8] "Distance measuring sensors." [Online]. Available: http://www.sharp-world.com/products/device/lineup/selection/opto/haca/diagram.html

[9] "Ultrasonic sensors." [Online]. Available: https://www.sick.com/ag/en/distance-sensors/ultrasonic-sensors/c/g185671

[10] "Lidar rangefinders." [Online]. Available: https://lightware.co.za/collections/lidar-rangefinders

[11] D. Falanga, S. Kim, and D. Scaramuzza, "How fast is too fast? the role of perception latency in high-speed sense and avoid," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1884–1891, April 2019.

[12] "Arducam." [Online]. Available: https://www.arducam.com/

[13] "Echoflight airborne radar." [Online]. Available: https://echodyne.com/products/echoflight/

[14] "2d lidar sensors tim5xx / outdoor." [Online]. Available: https://www.sick.com/mx/en/detection-and-ranging-solutions/2d-lidar-sensors/tim5xx/tim561-2050101/p/p369446

[15] "Single-chip 76-ghz to 81-ghz automotive radar sensor integrating mcu and hardware accelerator." [Online]. Available: http://www.ti.com/product/AWR1443

[16] "The dynamic vision sensor." [Online]. Available: https://inivation.com/dvs/

[17] "Blasterx senz3d." [Online]. Available: https://us.creative.com/p/web-cameras/blasterx-senz3d

[18] "The camera that senses space and motion." [Online]. Available: https://www.stereolabs.com/zed/

[19] "Intel realsense depth camera d435." [Online]. Available: https://www.intelrealsense.com/depth-camera-d435/

[20] "Middlebury stereo evaluation - version 2." [Online]. Available: http://vision.middlebury.edu/stereo/eval/

[21] "Middlebury stereo evaluation - version 3." [Online]. Available: http://vision.middlebury.edu/stereo/eval3/

[22] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," in *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, Dec 2001, pp. 131–140.

[23] L. Li, S. Zhang, X. Yu, and L. Zhang, "Pmsc: Patchmatch-based superpixel cut for accurate stereo matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 3, pp. 679–692, March 2018.

[24] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, Feb 2008.

[25] J. A. Jackson, J. D. Boskovic, and D. Diel, "Sensor fusion for sense and avoid for small uas without ads-b," *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 784–793, 2015.

[26] S. Ramasamy and R. Sabatini, "A unified approach to cooperative and non-cooperative sense-and-avoid," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2015, pp. 765–773.

[27] K. Rahul Sharma, D. Honc, and F. Duek, "Sensor fusion for prediction of orientation and position from obstacle using multiple ir sensors an approach based on kalman filter," in *2014 International Conference on Applied Electronics*, Sep. 2014, pp. 263–266.

[28] K. Park, S. Kim, and K. Sohn, "High-precision depth estimation with the 3d lidar and stereo fusion," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2156–2163.

[29] G. Fasano, D. Accardo, A. E. Tirri, A. Moccia, and E. D. Lellis, "Radar/electro-optical data fusion for non-cooperative uas sense and avoid," *Aerospace Science and Technology*, vol. 46, pp. 436 – 450, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1270963815002540

[30] J. Kim and Y. K. Kwag, "Multi-sensor fusion based target detection using eo/ sar," in *EUSAR 2014; 10th European Conference on Synthetic Aperture Radar*, June 2014, pp. 1–2.

[31] S. A. Rethnaraj Rambabu, Muhammad Rijaluddin Bahiki, "Multi-sensor fusion based uav collision avoidance system," *Jurnal Teknologi*, vol. 76, no. 8, p. 89, 2015. [Online]. Available: http://dx.doi.org/10.11113/jt.v76.5630

[32] H.-R. Song, W. sub Choi, and H. dong Kim, "Depth-aided robust localization approach for relative navigation using RGB-depth camera and lidar sensor," in *The 2014 International Conference on Control, Automation and Information Sciences (ICCAIS 2014)*, Dec 2014, pp. 105–110.

[33] H. R. Roopa, P. Parimala, and J. R. Raol, "Image sensor data fusion using factorized kalman filter," in *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, May 2016, pp. 1217–1220.

[34] G. Fasano, D. Accado, A. Moccia, and D. Moroney, "Sense and avoid for unmanned aircraft systems," *IEEE Aerospace and Electronic Systems Magazine*, vol. 31, no. 11, pp. 82–110, November 2016.

[35] J. J. Ruz, O. Arevalo, G. Pajares, and J. M. de la Cruz, "Decision making among alternative routes for uavs in dynamic environments," in *2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007)*, Sep. 2007, pp. 997–1004.

[36] B. Vanek, T. Pni, P. Bauer, and J. Bokor, "Vision only sense and avoid: A probabilistic approach," in *2014 American Control Conference*, June 2014, pp. 1204–1209.

[37] A. Strobel and M. Schwarzbach, "Cooperative sense and avoid: Implementation in simulation and real world for small unmanned aerial vehicles," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2014, pp. 1253–1258.

[38] A. Zarandy, T. Zsedrovits, B. Pencz, M. Nameth, and B. Vanek, "A novel algorithm for distant aircraft detection," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2015, pp. 774–783.

[39] A. Stanoev, N. Audinet, S. Tancock, and N. Dahnoun, "Real-time stereo vision for collision detection on autonomous uavs," in *2017 IEEE International Conference on Imaging Systems and Techniques (IST)*, Oct 2017, pp. 1–6.

[40] R. Labayrade, D. Aubert, and J. . Tarel, "Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2, June 2002, pp. 646–651 vol.2.

[41] B. T. Lopez and J. P. How, "Aggressive 3-d collision avoidance for high-speed navigation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5759–5765.

[42] M. S. Aman, M. A. Mahmud, H. Jiang, A. Abdelgawad, and K. Yelamarthi, "A sensor fusion methodology for obstacle avoidance robot," in *2016 IEEE International Conference on Electro Information Technology (EIT)*, May 2016, pp. 0458–0463.

[43] M. ki, J. cha, and H. Lyu, "Detect and avoid system based on multi sensor fusion for uav," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct 2018, pp. 1107–1109.

[44] Y. Zuqiang, F. Zhou, and L. Ping, "A bio-inspired collision-free 4d trajectory generation method for unmanned aerial vehicles based on tau theory," in *2015 34th Chinese Control Conference (CCC)*, July 2015, pp. 6961–6968.

[45] A. L. Smith, "Uas collision avoidance algorithm that minimizes the impact on route surveillance," 01 2009.

[46] N. E. Smith, R. Cobb, S. Pierce, and V. Raska, *Optimal Collision Avoidance Trajectories for Unmanned/Remotely Piloted Aircraft*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2013-4619

[47] E. J. Forsmo, E. I. Grøtli, T. I. Fossen, and T. A. Johansen, "Optimal search mission with unmanned aerial vehicles using mixed integer linear programming," in

2013 International conference on unmanned aircraft systems (ICUAS). IEEE, 2013, pp. 253–259.

[48] C. Lpez-Franco, J. Zepeda, N. Arana-Daniel, and L. Lpez-Franco, "Obstacle avoidance using pso," in *2012 9th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, Sep. 2012, pp. 1–6.

[49] F. Rehmatullah and J. Kelly, "Vision-based collision avoidance for personal aerial vehicles using dynamic potential fields," in *2015 12th Conference on Computer and Robot Vision*, June 2015, pp. 297–304.

[50] B. Du and S. Liu, "A common obstacle avoidance module based on fuzzy algorithm for unmanned aerial vehicle," 06 2016, pp. 245–248.

[51] C. Fu, M. A. Olivares-Mendez, R. Suarez-Fernandez, and P. Campoy, "Monocular visual-inertial slam-based collision avoidance strategy for fail-safe uav using fuzzy logic controllers," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 513–533, Jan 2014. [Online]. Available: https://doi.org/10.1007/s10846-013-9918-3

[52] T. W. Liao, "Uav collision avoidance using a* algorithm," 2012.

[53] R. Austin, *Introduction to Unmanned Aircraft Systems (UAS)*, 04 2010, pp. 1–15.

[54] R. Beard and T. McLain, "Small unmanned aircraft: Theory and practice," *Small Unmanned Aircraft: Theory and Practice*, 02 2012.

[55] "Connect escs and motors." [Online]. Available: http://ardupilot.org/copter/docs/connect-escs-and-motors.html

[56] A. Z. Richard Hartley, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[57] A. Kaehler and G. Bradski, *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*, 1st ed. O'Reilly Media, Inc., 2016.

[58] R. J. Radke, *Computer Vision for Visual Effects*. Cambridge University Press, 2012.

[59] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 4, pp. 401–406, April 1998.

[60] R. E. Kalman, "A new approach to linear filtering and prediction problems," *ASME Journal of Basic Engineering*, 1960.

[61] R. Faragher, "Understanding the basis of the kalman filter via a simple and intuitive derivation," *IEEE SIGNAL PROCESSING MAGAZINE*, 2012.

[62] G. Welch and G. Bishop, "An introduction to the kalman filter," Chapel Hill, NC, USA, Tech. Rep., 1995.

[63] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[64] L. J. D. L. Yoonseok Pyo, Hancheol Cho, *ROS Robot Programming (English)*. ROBOTIS, 12 2017.

[65] "Sf20/c (100 m)." [Online]. Available: https://lightware.co.za/products/sf20-c-100-m

[66] "Jetpack." [Online]. Available: https://developer.nvidia.com/embedded/jetpack

[67] "Technical specifications." [Online]. Available: https://developer.nvidia.com/embedded/develop/hardware

[68] "Harness ai at the edge with the jetson tx2 developer kit." [Online]. Available: hhttps://developer.nvidia.com/embedded/jetson-tx2-developer-kit

[69] K. Konolige, "Small vision systems: Hardware and implementation," in *Robotics Research*, Y. Shirai and S. Hirose, Eds. London: Springer London, 1998, pp. 203–212.

[70] "cv::stereosgbm class reference." [Online]. Available: https://docs.opencv.org/3.4.3/d2/d85/classcv_1_1StereoSGBM.html#a6b4a2763384ad5c6746189be22b56130a0f746667febe92e1189e924c40752660

[71] "Ros driver for personal project." [Online]. Available: https://github.com/r3n33/lidar-lite-ros-scanner-driver

[72] H. Pishro-Nik, *Introduction to Probability, Statistics, and Random Processes*. Kappa Research, LLC, 2014.

[73] "Stereo correspondance algorithms." [Online]. Available: https://docs.opencv.org/3.4.2/d1/d9f/classcv_1_1stereo_1_1StereoBinarySGBM.html

[74] "Disparitywlsfilter class reference." [Online]. Available: https://docs.opencv.org/3.3.1/d9/d51/classcv_1_1ximgproc_1_1DisparityWLSFilter.html#details

[75] D. Ratasich, B. Frmel, O. Hftberger, and R. Grosu, "Generic sensor fusion package for ros," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 286–291.

[76] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*. Brooks/Cole Publishing Co., 2012.

[77] J. P. Valappil and C. Georgakis, "Systematic estimation of state noise statistics for extended kalman filters," 2000.