الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

UNIVERSITE BADJI MOKHTAR - ANNABA
BADJI MOKHTAR – ANNABA UNIVERSITY

جامعة باجي مختار – عنابـــــة

Faculté : TECHNOLOGIE

Département : ELECTRONIQUE

Domaine : SCIENCES ET TECHNIQUES

Filière : AUTOMATIQUE

Spécialité : Automatique et Informatique Industrielle

Mémoire

Présenté en vue de l'obtention du Diplôme de Master

Thème:

Tracking mobile object with Unmanned Aerial Vehicle

Présenté par : *MALAOUI Nour el Amel & GHALABI Asma Ismahane*

Encadrant : *BENMOUSSA Samir* *MCA* *UBMA*

Jury de Soutenance :

HARKAT Mohamed Faouzi	Pr.	UBMA	Président
BENMOUSSA Samir	MCA.	UBMA	Encadrant
DOGHMANE Noureddine	Pr.	UBMA	Examinateur

Année Universitaire : 2021/2022

الملخص :

الهدف من هذا المشروع هو بناء طائرة بدون طيار رباعية مع نظام تتبع كائن متحرك. تمت مناقشة بعض الاهتمامات العملية ، مثل اختيار المكونات المناسبة. يتم أيضًا تقديم المشكلات النظرية ، مثل تطوير عنصر تحكم مشتق متكامل متناسب (PID) لزيادة استقرار طيران الطائرة بدون طيار / كوادكوبتر وتمكين الطيران المستقل باستخدام تقنيات الكائن والكشف. لتشغيل كوادكوبتر بشكل مستقل ، يتم استخدام استراتيجية تحكم تعتمد على الرؤية ، والتي تتكون من استخدام خوارزميات تتبع الكائن من أجل استخدام هذه البيانات كمدخلات للنظام. الكلمات الدالة كشف الكائن ، التتبع ، المروحية الرباعية ، متحكم STM32 وحدة التحكم ، PID.

Abstract :

The goal of this project is to build a quadcopter drone with a mobile object tracking system. Some practical concerns, such as choosing appropriate components are discussed. The theoretical issues, such as developing a Proportional-Integral-Derivative (PID) control to increase drone/quadcopter flight stability and enable autonomous flying utilizing object and detection techniques are also presented. To operate a quadcopter autonomously, a control strategy based on vision is used, which consists of using an object tracking algorithms in order to use this data as input for the system.

Keywords :

Object detection, Tracking, Quad-copter, STM32 microcontroller, PID controller.

Résumé :

L'objectif de ce projet est de construire un drone quadricoptère avec un système de suivi d'objets mobiles. Certaines préoccupations pratiques, telles que le choix des composants appropriés, sont discutées, ainsi que des problèmes théoriques, tels que le développement d'un contrôle proportionnel-intégral-dérivé (PID) pour augmenter la stabilité de vol du drone/quadricoptère et permettre un vol autonome utilisant des techniques d'objet et de détection. Pour faire fonctionner un quadricoptère de manière autonome, une stratégie de contrôle basée sur la vision est utilisée, qui consiste à utiliser des algorithmes de suivi d'objets afin d'utiliser ces données comme entrée pour le système.

Mots-clés

Détection d'objets, Poursuite, Quad-copter, microcontrôleur STM32, contrôleur PID.

ACKNOWLEDGMENTS

*We wish to thank **ALLAH**, the almighty for providing us with the chance to fulfill a dream, for given us a lot of patience, courage and the will to carry out this modest work*

We would like to express our deepest gratitude and heartfelt thanks to our supervisor, Mr. BENMOUSSA Samir for his patience, and above all for his trust, his comments and advice, his availability and kindness. We also thank him for giving us the opportunity to work on exciting project, and for his help at all times for which we are grateful.

We would like to express our heartfelt appreciation to the members of the examination jury in particular for their interest in this project.

A big thank you to our families, especially our parents who trust on us and who have always supported us in what we were undertaking. We thank you and we will be eternally grateful to you.

Finally, to our friends, thank you for giving us that extra level of support and an outlet when we needed it the most.

Nour el Amel MALAOUI *Ismahane GHALABI*

Contents

0.1 Introduction 6
0.2 Computer Vision Algorithm for Object Detection 6
0.2.1 RGB to Gray Scale 6
0.2.2 Filtering 7
0.2.3 Histogram equalization 9
0.2.4 Haar cascade classifier 10
0.3 Kalman Filter for Object Tracking 10
0.4 hardware implementation 13
0.4.1 Raspberry PI 13
0.4.2 Operating System 14
0.4.3 Features of the Raspberry pi Camera 14
0.4.4 Micro SD memory card 14
0.5 Software 14
0.5.1 Open CV 14
0.6 Implementation 14
0.6.1 The Kalman Filter module 15
0.6.2 The main program: face detection and application of the Kalman filter 17
0.7 Results and interpretations 20
0.8 Conclusion 21

1 Background of the quadcopters functioning 22
1.1 Introduction 22
1.2 Flight mode 23
1.2.1 Vertical flight 23
1.2.2 Hovering 23
1.2.3 Translational lift 23
1.3 Quad-copter flight dynamics 23
1.3.1 Reference system 23
1.3.2 Movement of the quadcopter 24
1.3.3 Euler angles 26
1.3.4 Physical effects acting on the quadcopter 27
1.4 Flight control logic 29
1.4.1 Block diagram 29
1.4.2 Micro controller 29

1.4.3 Inertial measurement unit (IMU) 30
1.4.4 PID Control of the quadcopte 30
1.5 Conclusion 31

2 Conception and realization of a quadricopter drone 32
2.1 Introduction 32
2.2 Quadcopter system architecture 32
2.3 Choice of material 33
2.3.1 The Frame 33
2.3.2 Propulsion 34
2.3.3 Power supply 35
2.3.4 Flight controller 36
2.4 CAD of quadracopter 37
2.5 Simscape simulation 39
2.6 Flight Controller Programming 41
2.6.1 Flight controller flowchart 41
2.6.2 Programming the MPU-9250 Inertial Unit 41
2.6.3 Programming a PID regulation 42
2.7 Deploying C-code to STM32 43
2.8 The communication between STM32 and Raspberry 44
2.8.1 Circuit 44
2.8.2 Program logic 45
2.9 Conclusion 45

List of Figures

1 Flowchart of the object detection and tracking. 7
2 RGB color decomposition 8
3 filters 8
4 image to which a low-pass filter has been applied (results on the right) 9
5 image to which a high-pass filter has been applied (results on the right) 9
6 histogram equalization 10
7 Flowchart of the proposed motion tracking approach using KF. 11
8 Raspberry PI 13
9 Result of raspberry pi camera processing using kalman filter 20

1.1 Rotational direction of quadcopter rotors 22
1.2 Reference system 24
1.3 Quadcopter reference frame. 24
1.4 Motors speeds for take-off and landing 25
1.5 Motors speeds for roll movement 25
1.6 Motors speeds for pitch movement 26
1.7 Motors speeds for yaw movement 26
1.8 Euler angles 27
1.9 Block diagram 29
1.10 PID logic diagram. 30
1.11 The adopted cascade loop for flight control. 31

2.1 Quadcopter system architecture 33
2.2 Drone's frame 33
2.3 Brushless motor 34
2.4 Electronic speed controller 40A 35
2.5 Propellers 10x47 35
2.6 STM32H7xx 36
2.7 IMU9250 36
2.8 Inner-Outer arm 37
2.9 Pieces 37
2.10 Quadcopter feet 38
2.11 Camera support 38
2.12 The quadcopter's CAD 38
2.13 The quadcopter's CAD 39

2.14 The quadcopter's CAD 39
2.15 The quadcopter's CAD 39
2.16 Simulink model of the quadcopter dynamic's. 40
2.17 Flight controller flowchart 41
2.18 The connection between MPU9250 and STM32 42
2.19 Euler angles from accelerations 42
2.20 Complementary filter 43
2.21 PID controller 43
2.22 Flight controller's Inputs/outputs. 44
2.23 Communication circuit 45
.

Introduction

Unmanned aerial Vehicles (Abrev. U.A.V) are unmanned aircraft capable of to carry out missions in a more or less autonomous and automatic way. Since some years, the development of drones continues to grow thanks to the remarkable progress in the field of embedded systems such as the miniaturization of sensors, actuators and the evolution of digital computers.

Among drones, the quadcopter stands out as one of the most popular systems. promising due to the diversity of applications for which it can be used. Among these applications we can cite: surveillance and observation missions, shots air traffic, pursuit, espionage, checking the condition of a building that is difficult to access or even the transport of goods.

In this project, a solution to the problem of an UAV capable of following a moving target (in this case a human face) is presented. The ability to track a face in a video stream opens up new possibilities for human computer interaction. Applications range from head gesture-based interfaces for physically handicapped people. Other possible use is like a video record someone in hazardous situations, like while practicing extreme sports.

The objective of this thesis is the study and the realization of an autonomous quadcopter capable of tracking a moving object. It is structured in 3 chapters:

The first chapter is dedicated to the vision system, that allows a detection and tracking of an object of interest, starting with a detailed presentation of the processing steps applied on an image before detection, then Kalman filter is used to track a moving object.

The second chapter is devoted to the modeling of the quad-copter, and to the proposed flight control logic based on PID controller.

In the last chapter, the quadcopter conception and realisation is presented. Starting with CAD model of the drone, the choice of materials, and the main board programming.

The thesis ends with conclusions and prospects.

0.1 Introduction

The human being is equipped with a complicated and very complex vision system which allows him to perceive and understand the surrounding environment. The human vision system is a complex biological device, composed of an eye, a brain and an optic nerve. With technological progress, all robot systems are equipped with an artificial vision that can reproduce certain functionalities of human vision through the analysis of acquired images. The aim of this chapter is to present the developed vision system that will allow the UAV to follow or track the object of interest, in this work, human faces. For this purpose, the Haar Cascade algorithm combined with Kalman filter are used and implemented on Raspberry PI card .

0.2 Computer Vision Algorithm for Object Detection

Object detection is a computer vision technique used to identify and locate objects in an image or a video. With this kind of identification and localization, object detection can be used to count objects in a scene and determine and track their precise locations, all while accurately labeling them.

The proposed approach for object detection and tracking is illustrated by Figure.1 and it is composed of the following steps:

1. Image acquisition and conversion: the camera streams an image in RGB format. This image is converted in Gray Scale.
2. Filtering is applied in order to reduce noise of an image .
3. Histogram equalisation is performed to improve the quality of an image .
4. Haar cascade classifier for object detection
5. Position prediction using the Kalman Filter

0.2.1 RGB to Gray Scale

The acquired images are described on Red Green Blue (RGB) color format work. Since, data processing on color image is heavy in terms of computer memory, an RGB to gray scale conversion if often necessary. Here, the image is represented by one color where its values range form 0 to 255 representing respectively the black and the white color. It exists different methods for conversion such as : the lightness method, the average method and the luminosity method.

As can be seen in the figure 4, the lightness method is the worst since it completely fails to capture the lightness of the input image. Among the other two, luminosity one is the best since the average method produces a darker gray-scale image.

-After that ,to improve the contrast, the image is smoothed to reduce noise that corrupt information and histogram equalization is applied also to improve the quality of an image:

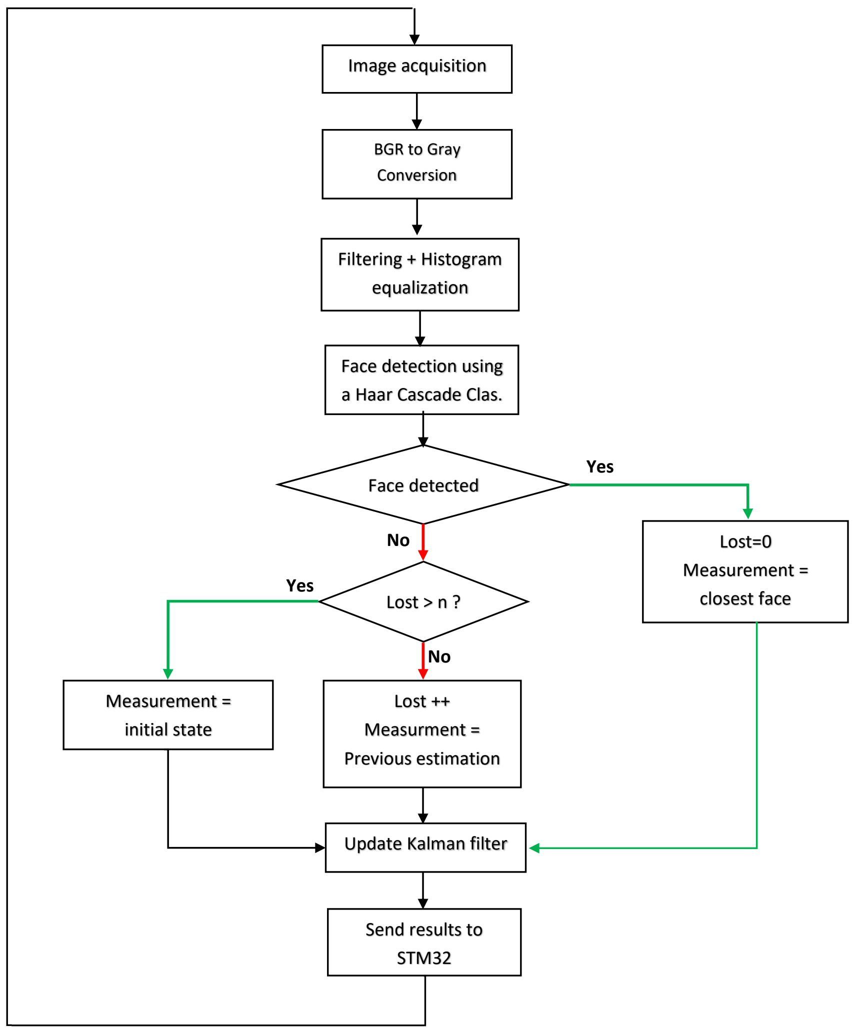


Image acquisition
BGR to Gray Conversion
Filtering + Histogram equalization
Face detection using a Haar Cascade Clas.
Face detected
Yes
No
Lost=0 Measurement = closest face
Lost > n ?
Yes
No
Measurement = initial state
Lost ++ Measurment = Previous estimation
Update Kalman filter
Send results to STM32

Figure 1: Flowchart of the object detection and tracking.

0.2.2 Filtering

It consists in modifying the value of the pixels of all or part of an image digital, generally with the aim of reducing noise in an image, detecting edges and preserving the shape of regions, also Accentuating variations in intensity of the image: enhancing the differences between pixels belonging to adjacent regions (contrast enhancement).

The following types of filtering are generally intended:

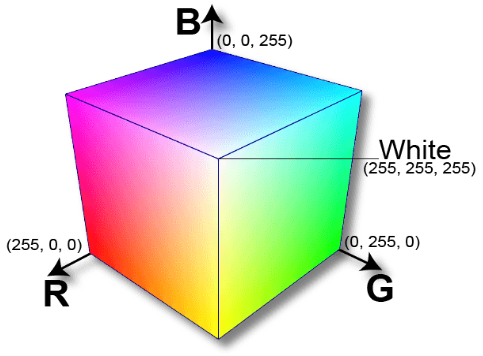


Figure 2: RGB color decomposition

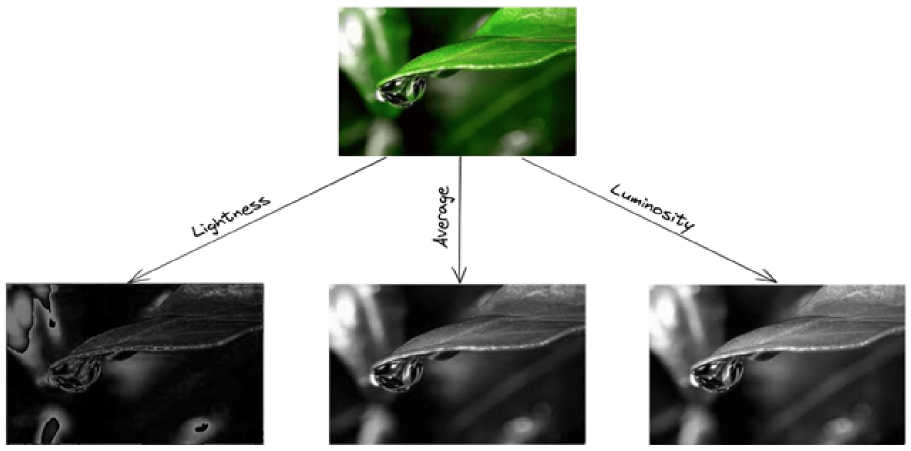


Figure 3: filters

Low-pass filter

amplifies more frequencies below a set frequency, typically used to reduce noise in an image

High-pass filter

unlike low-pass, amplifies the frequencies above a determined frequency and allow in particular to accentuate the details and the contrast, and therefore the difference between the neighboring pixels.

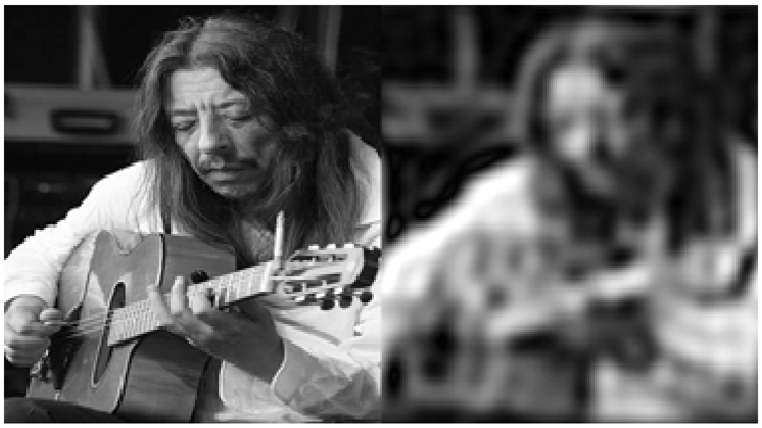


Figure 4: image to which a low-pass filter has been applied (results on the right)

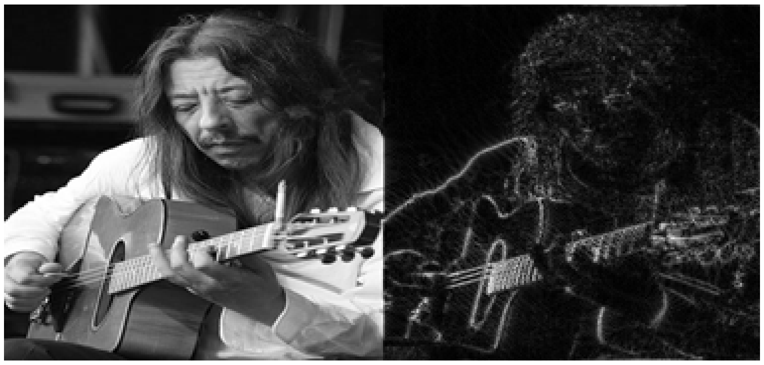


Figure 5: image to which a high-pass filter has been applied (results on the right)

0.2.3 Histogram equalization

Histogram modification is a technique among the most common image preprocessing techniques, which aims to improve the image by applying a punctual intensity transformation: to each pixel an intensity is associated by the increasing transformation of way to keep the contrasts between the regions, there are two types for the modification of the histogram:

dynamic expansion

consists in making the best use of the gray level scale available on the image acquisition system.

histogram equalization

This method is used in this project and consists of adjusting the contrast of a digital image that uses the histogram. This transformation consists in making the histogram of the image as flat as possible the gray level.

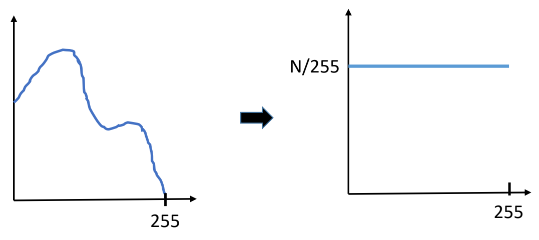


Figure 6: histogram equalization

0.2.4 Haar cascade classifier

An effective object detection approach ,basically a machine learning based algorithm where a cascade function is trained from a lot of images both:positive images for those that contains the images that will be identified by the classifier ,in this case images faces ,and negative which do not contain the object(without faces). This method was proposed in 2001 in a research paper [5] entitled ”Rapid Object Detection using a boosted cascade of simple feature”, the main advantages of it is that the Haar cascade are fast and work well in real time, also so simple to implement and less computing power required .

HOG CLASSIFIER

It is another method for detection known as Histogram of Oriented Gradients, HOG, This technique counts occurrences of gradient orientation in the localized portion of an image. This method is quite similar to Edge Orientation Histograms and Scale invariant a Feature Transformation (SIFT). The HOG descriptor focuses on the structure or the shape of an object. It is better than any edge descriptor as it uses magnitude as well as angle of the gradient to compute the features. For the regions of the image it generates histograms using the magnitude and orientations of the gradient.

0.3 Kalman Filter for Object Tracking

Object tracking is the problem of estimating the positions and other relevant information of moving objects in image sequences as illustrated by Figure.7. In Kalman filter, a tracking system of Eq.1 is considered where x_k is the state vector represents the dynamic behaviour of the object, the subscript k indicate the discrete time. The objective is to estimate x_k from the measurement z_k.

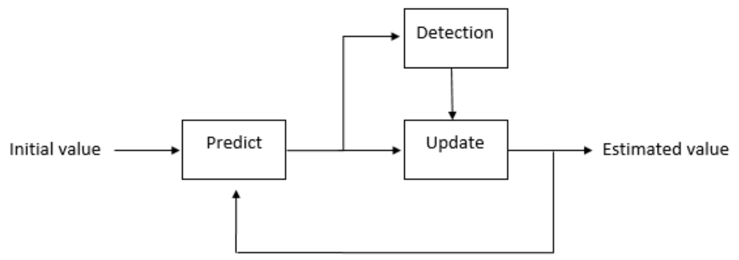


Figure 7: Flowchart of the proposed motion tracking approach using KF.

Following is the mathematical description of the Kalman filter:

1. Process equation:
$$x_k = Ax_{k-1} + w_{k-1} \tag{1}$$
Where A represents the transition matrix and x_k the state at time $k-1$ to k . w_{k-1} is the Gaussian process noise

2. Measurement equation:
$$z_k = Hx_k + v_k \tag{2}$$
Where H is the measurement matrix and z_k is the measurement observed at time $k-1$ to k respectively . v_k is the Gaussian measurement noise .

3. Time update equations:
Equation 1 and 2 describes a linear model at time k.
As x_k is not measured directly, therefore the information provided by measured z_k is used to update the unknown states x_k. A priory estimate of state $x_{\hat{k-1}}$ and covariance error $\hat{x}_k$ estimate is obtained for the next time step K.
$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{3}$$
$$P_k^- = AP_{k-1}A^T + Q \tag{4}$$

4. Measurement update equations
These equations are associated with the feedback of the system.The objective is to estimate a posterior estimating $\hat{x_k}$ which is a linear combinations of the a priory estimate and the new measurement z_k. these equations are given below :

$$\hat{K}_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{5}$$

$$\hat{x}_{k-1} = \hat{x}_k^- + K_k (z_k - H\hat{x}_k^-) \tag{6}$$

$$P_k = (I - K_k H) P_k^- \tag{7}$$

K_k is the Kalman gain which is computed by above the measurement update equations.

After that a posterior state estimate $\hat{x}_k$ and a posterior error estimate P_k is computed by the measurement z_k. The time and measurement equations are calculated recursively with previous a posterior estimates to predict new a prior estimate. This recursive behaviour of estimating the states is one of the highlights of the Kalman filter.

For the studied system, the state vector is defined as x = $\begin{bmatrix} px & py & D \end{bmatrix}^T$,where x, y and d represents the position and the diameter of the center of circle enclosing the object of interest in this case human face and its diameter d ,while z = $\begin{bmatrix} \dot{p}_x & \dot{p}_y & \dot{D} \end{bmatrix}^T$ its time derivative From its corresponding equations, the resulting transition matrix is:

$$A = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

where dt is the time estimating .
For simplification purposes, the targets are moving in a uniform rectilinear motion, therefore the process noise can be omitted. Then, Two different observation models are considered. The first assumes that the sensors can measure the position as well as velocity, in which case, the observation matrix is defined as the identity matrix: H_2. The other case used here which assumes that the sensors can only measure positions. In this case, the observation matrix is defined as: H_1

$$H_1 = [6,6] \tag{9}$$

$$H_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

A state estimation equation $\hat{x}_k$ that calculates the posterior status estimation must be calculated in order to get the Kalman filter equations. It requires the calculation formula of $\hat{x}_k$ is the linear combination of a priory estimation and weighted difference between true measurements and measured forecast value.

model 3B+	
soc : Broadcom BCM2837B0	CPU : 4coeurs ARM cortex A53 1,4GHZ
GPU : Broadcom video core IV	Memory : 1GO
USB2 : 4, USB3 : 0	Ethernet : 10/100/300
video : HDMI Full HD	WIFI : 802.11ac
Bluetooth : 4.2	Consummation : 800mA

Table 1: Technical specifications of the Raspberry Pi

0.4 hardware implementation

0.4.1 Raspberry PI

The Raspberry Pi (Figure .8) is a computer that runs under the operating system Linux. It has two USB ports to which you can connect a keyboard and a mouse, and an HDMI (High-Definition Multimedia Interface) video output to which you can connect a TV or monitor. When the Raspberry Pi starts, we get the Linux desktop It is really a complete computer, with an office suite, video playback features, games, etc. But he does not run on Microsoft Windows; instead we have the open source competitor Windows, Linux (a Debian distribution), and a windowing environment is called LXDE. The technical specifications are given by Table.1

Figure 8: Raspberry PI

0.4.2 Operating System

The raspberry pi card requires an operating system for its management and which provides access and manages the hardware side and the inputs/outputs. For this, this system is based on Linux which is called Raspbian. Raspbian is a free operating system based on the GNU/Linux distribution, and optimized for the smallest computer in the world. Raspbian does not just provide a basic operating system, it also comes with more of 35,000 precompiled packages delivered in an optimized format.

0.4.3 Features of the Raspberry pi Camera

The camera that has been used in this project is 5MP Omni Vision OV5647 camera module which is a fully compatible with both the Model A and Model B Raspberry Pi. The image resolution is 2592 x 1944 and supports 1080p @ 30fps , 720p @ 60fps and 640x480p 60/90 recording videos.

0.4.4 Micro SD memory card

A memory card is used since it will contain the operating system and the programs that we are going to create. It is preferable that it is with a high Capacity (at least 16 GByte) and of good quality since it is in this memory card is going to be used as a disk hard where the operating system and created programs will be stored and the speed data transmission between the micro SD and the processor must be the fastest possible.

0.5 Software

0.5.1 Open CV

Open CV (Open Source Computer Vision) is a library of programming functions mainly aimed at real time computer vision , originally developed by Intel,it is a library of computer software and machine learning software, it was designed for various purposes such as computer vision, algorithm, operations mathematics, video capture, image processing... etc, it has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac BONE. This library contains more than 2500 optimized algorithms,which have excellent accuracy and computing efficiency,it can be used to detect and recognize faces, identify objects, camera movement tracking, moving object tracking, extraction 3d models of objects and many other things.

0.6 Implementation

The code is divided into two modules, one dedicated to face detection, and the other to the prediction thanks to the Kalman filter

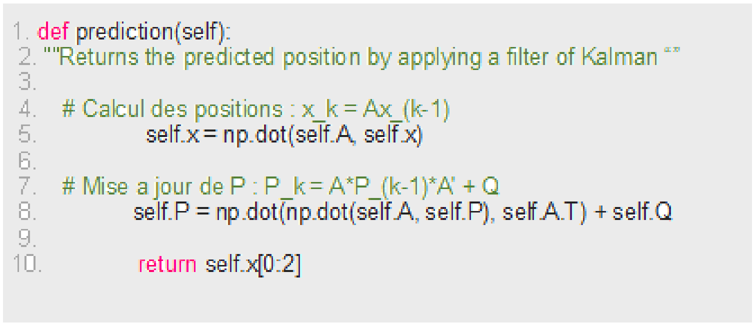
0.6.1 The Kalman Filter module

This module is the lung of the project, it is in this one that all the operations related to the Kalman filter will be define .

For this module, the NumPy package is needed, in particular at the end to facilitate matrix calculations. First, start by initializing all the parameters of the filter, namely: the matrix of the initial state, the matrices of transition, observation, covariance and transition and observational noise matrices.

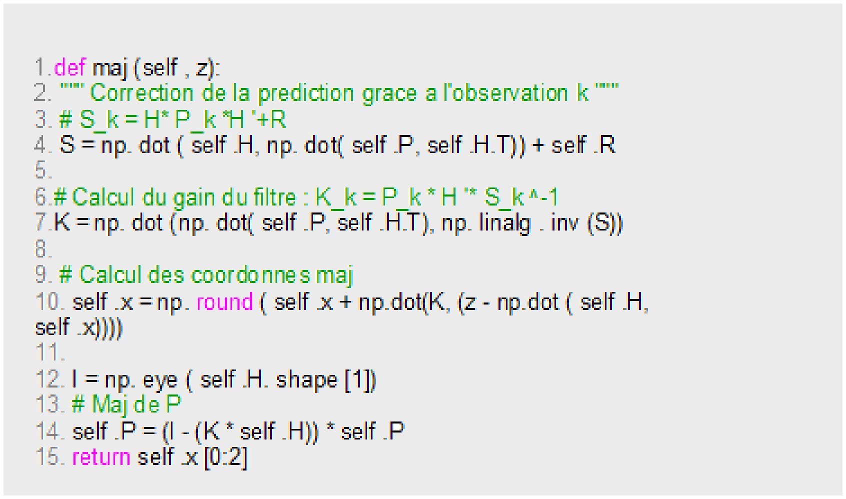
```
import numpy as np


class KalmanFilter(object):
def __init__(self, dt, u_x, u_y, std_acc, x_std_meas, y_std_meas):

      #:param dt: sampling time (time for 1 cycle)
      #:param u_x: acceleration in x-direction
      #:param u_y: acceleration in y-direction
      #:param std_acc: process noise magnitude
      #:param x_std_meas: standard deviation of the measurement in x-direction
       #:param y_std_meas: standard deviation of the measurement in y-direction


    # Define sampling timeself.dt
        self.dt = dt

    # Define the control input variables
         self.u = np.matrix([[u_x],[u_y]])

    # Intial State
          self.x = np.matrix([[0], [0], [0], [0], [0], [0]])

    # Define the State Transition Matrix A
          self.A = np.matrix([[1, 0, 0, self.dt, 0, 0],
                              [0, 1, 0, 0, self.dt, 0],
                              [0, 0, 1, 0, 0, self.dt],
                              [0, 0, 0, 1, 0, 0],
                              [0, 0, 0, 0, 1, 0],
                              [0, 0, 0, 0, 0, 1]])

    # Define the Control Input Matrix B
                self.B = np.matrix([[a2, 0, 0],
                              [0, a2, 0],
                              [0, 0, a2],
                              [self.dt, 0, 0],
                              [0, self.dt, 0],
                              [0, 0, self.dt]])

    # Define Measurement Mapping Matrix
               self.H = np.matrix([[1, 0, 0, 0, 0, 0],
                              [0, 1, 0, 0, 0, 0]]),


   #Initial Process Noise Covariance
               self.Q = np.matrix([[(self.dt**4)/4, 0, 0, (self.dt**3)/2, 0, 0],
                               [0, (self.dt**4)/4, 0, 0, (self.dt**3)/2, 0],
                               [0, 0, (self.dt**4)/4, 0, 0, (self.dt**3)/2],
                               [(self.dt**4)/4, 0, 0, self.dt**2, 0, 0],
                               [0, (self.dt**4)/4, 0, 0, self.dt**2, 0],
                               [0, 0, (self.dt**4)/4, 0, 0, self.dt**2]]) * std_acc**2

  #Initial Measurement Noise Covariance
                self.R = np.matrix([[x_std_meas**2, 0],
                              [0, y_std_meas**2]]),


  #Initial Covariance Matrix
                self.P = np.eye(self.A.shape[1])
```

then define a first function: the prediction which, like its name the indicators will predict the positions. For this,the formulas current views is applied, i.e.: $x_k = Ax_{k-1}$ for position prediction, and $P_k = AP_{k-1}A^T + Q$ for updating the covariance matrix



```
def prediction(self):
""Returns the predicted position by applying a filter of Kalman ""

  # Calcul des positions : x_k = Ax_(k-1)
          self.x = np.dot(self.A, self.x)

  # Mise a jour de P : P_k = A*P_(k-1)*A' + Q
         self.P = np.dot(np.dot(self.A, self.P), self.A.T) + self.Q

          return self.x[0:2]
```

The second function (maj) updates the filter. The calculations are based once again on the calculations seen in progress: $S_k = HP_kH^T + R$; the filter gain : $K_k = P_k.HTS_{K-1}$ at the end the update of P:
$P_k = P_{k-1} - K_kS_kK_k^T \Leftrightarrow P_k = (I - K_kH)P_{k-1}$.



```
def maj (self , z):
""" Correction de la prediction grace a l'observation k """
# S_k = H* P_k *H '+R
S = np. dot ( self .H, np. dot( self .P, self .H.T)) + self .R

# Calcul du gain du filtre : K_k = P_k * H '* S_k ^-1
K = np. dot (np. dot( self .P, self .H.T), np. linalg . inv (S))

# Calcul des coordonnes maj
self .x = np. round ( self .x + np.dot(K, (z - np.dot ( self .H,
self .x))))

I = np. eye ( self .H. shape [1])
# Maj de P
self .P = (I - (K * self .H)) * self .P
return self .x [0:2]
```

The functions of this module will be used in the main program described below.

0.6.2 The main program: face detection and application of the Kalman filter

In this part every function in the program will be explained ,The database used is an XML file gathering raw data allowing to detect faces; this file is named ”Haar cascade frontal face default.xml”.

```
#we create the waterfall
face_cascade=cv2.CascadeClassifier(‘haarcascade_frontalface_default.xml’)
```

Once the face Cascade has been created, the video needs to be played in order to get treated. For this, the Video Capture method were used:

```
# ’0’to get a video feed from the camera
video = cv2 . VideoCapture (0')
```

also initialize the Kalman filter as well as two variables: height and width which correspond to the height and width of the face.

```
largeur = 0
hauteur = 0
# Filter initialization
KF = FiltreKalman(1/30, 1, 0.1, 0.1)
```

then enter an infinite loop to process the video to completion, the steps described below are inside the loop. lets start by initializing the list that will host the positions of the faces and play the video frame by frame with the read() method and then convert the video to gray scale for easier processing.

```
 centers=[]
#Frame-by-frame video playback
ret, img = cap.read()
#convert Video frame to Greyscale
gray  = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Once the image has been converted to gray scale, detecting faces must be fulfilled using the **detectMultiscale** function. This function takes as input the gray scale image. It has two parameters:

1. -minNeighbors, which allows not to detect 10 times the same face in indicating the minimum possible distance between two faces.
2. -minsize, which makes it possible not to consider very small elements like faces by specifying minimum face size.

```
 # Detection des visages dans l'image en nv de gris
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5)
```

after that, green squares are around each face in order to identify them, and the ID of the person detected is printed over the rectangle. the ID is recognized from the available data in trainer.yml file .

```
for (x, y, w, h) in faces:
    roi_gray = gray[y:y+h, x:x+w#Convert Face to greyscale
    id_, conf = recognizer.predict(roi_gray) #recognize the Face
    if conf>=80:
        font = cv2.FONT_HERSHEY_SIMPLEX #Font style for the name
        name = labels[id_] #Get the name from the List using ID number
        cv2.putText(img, name, (x,y), font, 1, (0,0,255), 2)
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2
```

due to the prediction method ,the predicted coordinates must be calculated so a blue rectangle is draw.

```
#prédiction thanks to kalman filter
(x_pred, y_pred) = KF.prediction()
# Draw a rectangle at the predicted position
cv2.rectangle(img, (int(x_pred), int(y_pred)), (int(x_pred + largeur), int(y_pred + hauteur)), (255, 0, 0), 2)
# Adding a legend
cv2.putText(img, "Position predite", (int(x_pred) + 15, int(y_pred)), 0, 0.5, (255, 0, 0), 2)
```

All to do now is update the filter and possibly correct the prediction (if and only if a face is detected) using the function maj.

```
# Fase de maj du filtre Kalman
if centers !=[ ]:
(x_estim, y_estim) = KF.maj(centers[0])
# Dessine un rectangle à la position estimée grâce à Kalman
  cv2.rectangle(img, (int(x_estim), int(y_estim)), (int(x_estim + largeur), int(y_estim + hauteur)), (0, 0, 255), 2)
.# Adding a legend
    cv2.putText(img, "Position estimee", (int(x_estim + 15), int(y_estim + 10)), 0, 0.5, (0, 0, 255), 2)
```

The following code is optional but very practical since it allows you to get out of the loop in finished without having to interrupt the program manually. wait Key waits for a keyboard event (goes to True when a key is pressed).

```
# To exit the loop by pressing ‘q’
 if cv2 . waitKey (1) & 0xFF == ord('q'):
 break
```

of course, don’t forget to display the images:

```
#show result
Cv2.imshow ('preview',img)
```

And finally, as soon as the loop finished ,stop playing the video and destroy the window displaying the results

```
#when we exit the loop we close everything
Video . release ()
cv2.DestroyAllWindows()
```

0.7 Results and interpretations

The results are quite conclusive, the face tracking works very well and the prediction is quite reliable despite some bugs. Indeed, if we completely changes direction at the time of the prediction, the latter will be wrong. Also, the model chosen is that of constant acceleration; however, this is not always the case during the movement of people, and it is therefore not may not be the most suitable model for predicting human movements, which like certain mechanical or physical movements, are not completely predictable. Nevertheless, the program seems to work in the majority of cases, and the screenshots below show this.It is clearly seen on these captures that the trajectory prediction is coherent.

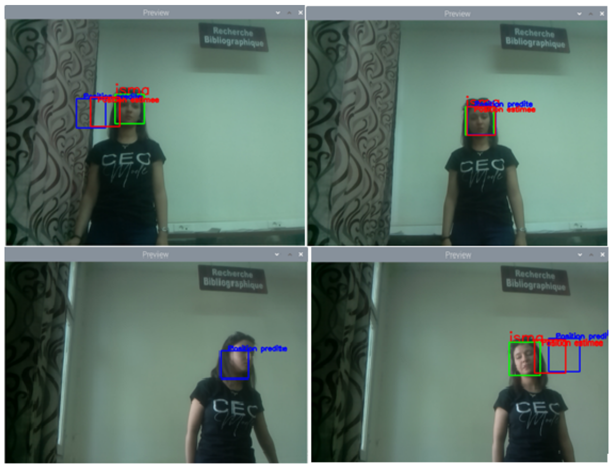


Figure 9: Result of raspberry pi camera processing using kalman filter

0.8 Conclusion

In this chapter, we are interested in two of the major problems of motion analysis in a sequence of images; the detection and tracking of moving objects, more precisely the tracking of a human face. In the next chapter, the quadcopter modeling and the flight control logic will be presented.

Chapter 1

Background of the quadcopters functioning

1.1 Introduction

The Drone is a rotor craft that can perform missions without a human being on board. This first basic characteristic justifies the Uninhabited (or Unmanned) Aerial Vehicle (UAV).

In present study, the world of drones will be presented and more precisely that of the quadrotor, which is, in other words, an aerial mobile robot with four rotors defined in space by 6 degrees of freedom (DOF).

The basic structure that keeps all of the components together is the quadcopter body, which is made up of two perpendicular bars (cross-shaped) connected by a central core. This structure is typically symmetric. The crucial component for lifting the entire frame is a motor located at the end of each bar. Each motor contains a rotor inside, which is the moving part that keeps the rotating wings that provide lifting. Each rotor lift force may produce a torque that cannot be eliminated, preventing the drone from taking off. Therefore,the rotors must rotate in different directions. Motor 1 and 3 will rotate in the same direction, while motor 2 and motor 4 will rotate in the other direction. Thus, each motor will rotate in the opposite direction as the motor next to it rotates, as shown by Figure 1.1.

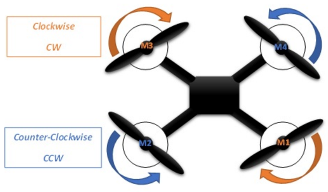


Figure 1.1: Rotational direction of quadcopter rotors

The functioning of a quadcopter is quite particular; by skillfully varying the power of the motors, it is possible to make it go up/down on the z-axis, to tilt it to the left/right on the x-axis, or forward/backward on the y-axis, or to make it pivot on itself. These movements provide the six degrees of freedom that need to be controlled using only four motors, so the system is under-actuated, that is, the number of inputs is lower than the number of outputs.
The rotation around the z-axis is called YAW. A variation in the speed of rotation on the pair of propellers one and three or two and four according to the desired angle causes a modification in the yaw angle.
Rotation around the y-axis is called PITCH, and around the x-axis it is called ROLL. These movements can be ensured by acting on the speed of one of the four motors, and consequently on the force of the rotors. A variation of the roll angle is obtained due to a speed difference between the right and left motors. A variation in the pitch angle is obtained due to a speed difference between front and back motors.

1.2 Flight mode

1.2.1 Vertical flight

The net weight and the aerodynamic resultant are two forces acting in opposite directions, allowing the drone to go up or down depending on the aerodynamic effect, which can be higher or less than the drone's weight.

1.2.2 Hovering

If the lift force produced by all rotors is equal to the weight of the quadcopter, then it is a hover mode. In this case, the same average speed is applied to all rotors.

1.2.3 Translational lift

The movement of the drone on a horizontal plane is known as translational flight. Tilting, pitching, and rolling motions are used to assure it.

1.3 Quad-copter flight dynamics

1.3.1 Reference system

To describe the flight dynamics of the quadcopter, a set of basic references and notation must be defined. Figure 1.2 shows the standard coordinate system for dealing with quadcopter orientation. In addition, the inertial frame reference and the body frame reference are two separate reference points in the coordinate system to consider. The first is a fixed coordinate system with an external environment, while the second is a fixed coordinate system with reference to the quadcopter body.

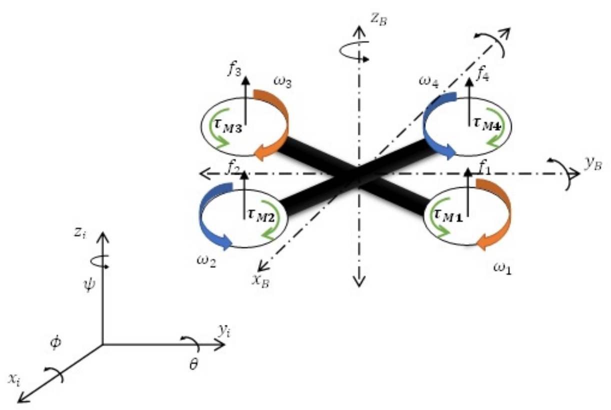


Figure 1.2: Reference system

1.3.2 Movement of the quadcopter

Different types of movements can be achieved by different combinations of force produced by each rotor. Based on these possible movements, the drone can perform three flight modes:

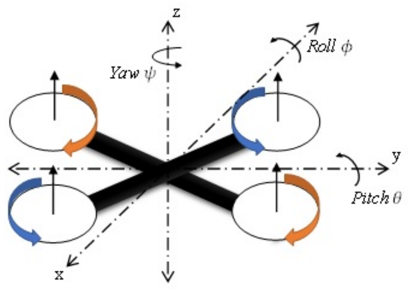


Figure 1.3: Quadcopter reference frame.

Vertical flight

For take off, the speed of all the rotors must be equally high, which results in a greater net force (which will now be greater than the absolute value of the total mass). and vice versa for landing.

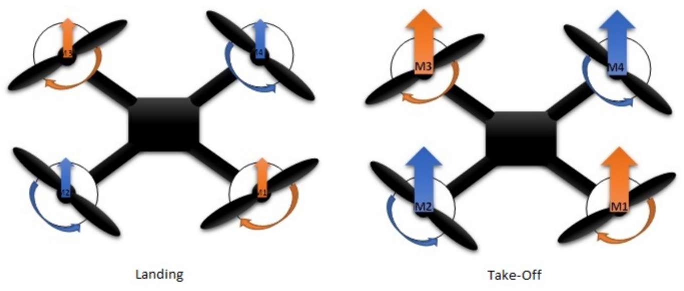


Figure 1.4: Motors speeds for take-off and landing

Roll movement

Since we cannot produce a pure lateral force,lateral movement must be carried out by a rotation around the axis that is orthogonal to the axis over which to translate and to the vertical axis simultaneously.
In other words, a rotation around the x axis is coupled with a translational movement along the y axis, which is the rolling motion.

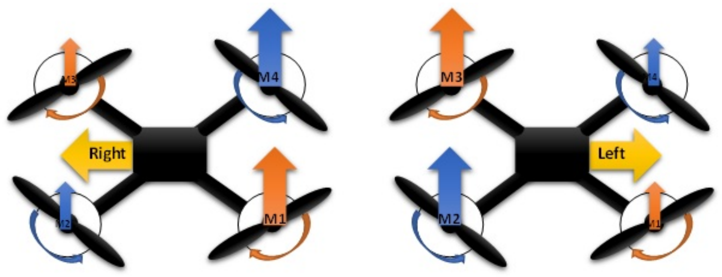


Figure 1.5: Motors speeds for roll movement

Pitch movement

This is a rotation around the y axis. Decrease the speed of the forward motors and increase the speed of the rear motors ensuring forward movement and vice versa for reverse. This movement is coupled to a translational movement along the axis x.

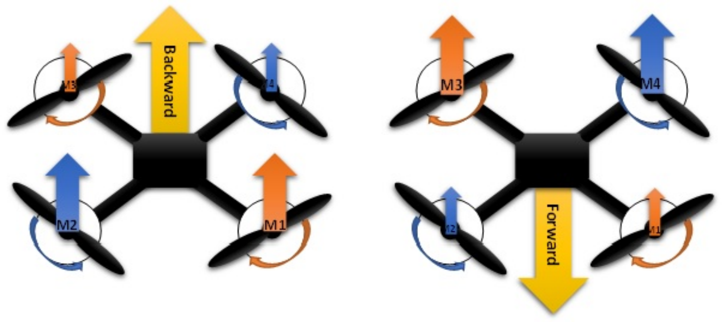


Figure 1.6: Motors speeds for pitch movement

Yaw Movement

The yaw movement is used to spin the quadcopter around its own axis z. It is made by raising the speed of two motors on the same axis and vice versa.

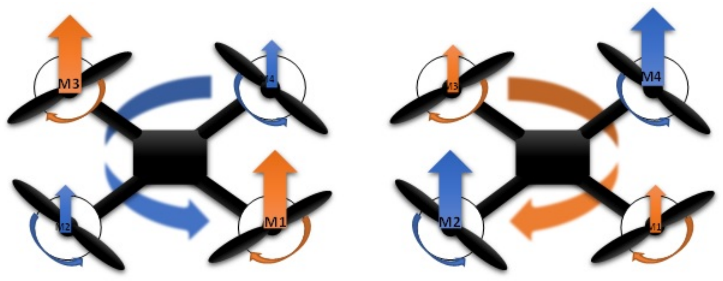


Figure 1.7: Motors speeds for yaw movement

1.3.3 Euler angles

The coordination between landmarks may be defined using an orthogonal rotation matrix.The parameterization of the rotation matrix by Euler angles is commonly used in robotic applications.
Because the conservative forces exist (the weight is always in the direction Z of the Earth's fixed reference $i(O_i,x_i,y_i,z_i)$), a 3D rotation matrix is required to train them in the mobile frame B (O_B,x_B,y_B,z_B).
As a result, the forces will be formulated using the angles shown in figure 1.8:

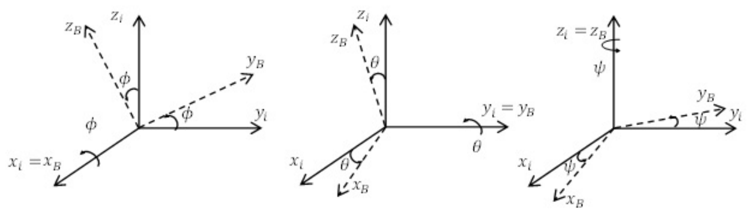


Figure 1.8: Euler angles

Rotation around the X-axis:

$$R(X,\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{1.1}$$

Rotation around the Y-axis:

$$(Y,\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{1.2}$$

Rotation around the Z-axis:

$$R(X,\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

$R(\phi,\theta,\psi) = R(Z,\psi) * R(Y,\theta) * R(X,\phi) =$

$$\begin{bmatrix} \cos\psi * \cos\theta & \sin\phi * \sin\theta * \cos\psi - \sin\psi * \cos\phi & \cos\phi * \sin\theta * \cos\psi + \sin\psi * \sin\phi \\ \sin\psi * \cos\theta & \sin\phi * \sin\theta * \sin\psi + \cos\psi * \cos\theta & \cos\phi * \sin\theta * \sin\psi - \sin\phi * \cos\psi \\ -\sin\theta & \sin\phi * \cos\theta & \cos\phi * \cos\theta \end{bmatrix} \tag{1.4}$$

1.3.4 Physical effects acting on the quadcopter

Assumption:

Mass-gravity is the primary force acting on the quadcopter. The propellers create thrust vectors to counterbalance this force. The aerodynamic effects are not included because these forces are supposed to be negligible. Wind and drag are two examples of aerodynamic effects that have been adjusted to zero to simplify system modeling. Figure 1.2 also depicts the angular velocities as well as torques, and forces generated by the four rotors.

Forces

The main force acting on the quad-copter is the weightlessness defined by:

$$P = m \times g \tag{1.5}$$

with **P**: Weight ; **m** : massand **g** :Gravity.

Propellers generate a force that is perpendicular to their rotational plane. As a result, this force is proportional to the angular velocity of the propeller.

$$f_i = b \times \omega_i^2 \tag{1.6}$$

in which the lift constant is b. The angular velocity of rotor i,indicated by ωi

The combined forces of the rotors provide thrust T.

$$T = \sum_{i=1}^{4} f_i = b \sum_{i=1}^{4} \omega_i^2, \quad T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{1.7}$$

All of these forces are acting in the z direction. The consequence is that the quadcopter cannot traverse the x, y plane without rotating.

Torques

Torque is equal to the cross product of distance and force, according to physics. The torques $\tau\phi$, $\tau\theta$, and $\tau\psi$ in the direction of the respective body frame angles make up torque τ_x.

$$\tau_x = l \times T = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} l\,(f_1 + f_4 - f_2 - f_3) \\ l\,(f_1 + f_2 - f_3 - f_4) \\ l\,(f_1 - f_2 + f_3 - f_4) \end{bmatrix} \tag{1.8}$$

where l is the distance between the center of gravity of the quad-copter and the center of the rotor.

These equations are adapted for a symmetrical X configuration quadcopter.

Gyroscopic effect

The gyroscopic effect occurs when an object rotates around an axis. This refers to an object's capacity to retain its rotational axis, or, more accurately, its angular momentum.
The name ”gyroscope” is made up of two Greek words: gyro, which means rotation, and scope, which means to observe. The gyroscopic effect is the mode of operation of a gyroscope, a motion measurement instrument used in aviation.
In the world of aviation, there are two gyroscopic moments: the propeller gyroscopic moment and the gyroscopic moment caused by quadcopter motions.

1.4 Flight control logic

The most significant component of a drone is the flight controller, which allows the drone to fly. It features a micro controller that receives and analyzes IMU data, receives orders from the communication system, and controls the ESCs with them. Several control algorithms may be used to develop control laws that can be used to pilot and stabilize a quad-copter. However, when the drone travels away from its equilibrium points, these approaches suffer a performance loss. Furthermore, disruptions could cause these vehicles to become unstable. Adaptive approaches can be utilized to achieve a stable system in this case. usually, a PID corrector used to manage the altitude movement, as well as the angles of pitch, roll, and yaw.

1.4.1 Block diagram

This diagram has two subsystems: The motor unit is made up of four brushless direct current (BLDC) motors and the Electronic speed controllers (ESCs) that accompany them. The control unit contains STM32 and sensors that supports the flight controller.

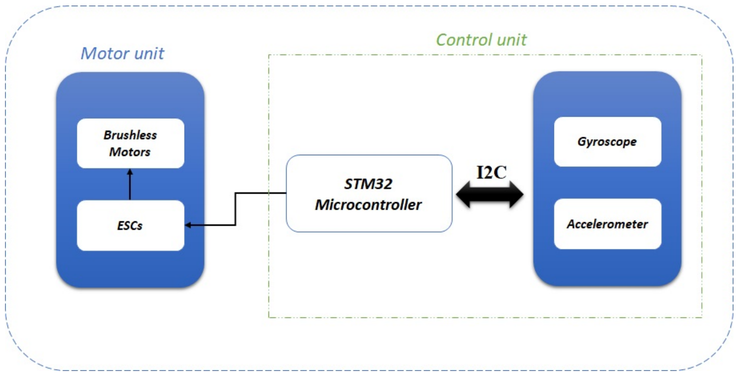


Figure 1.9: Block diagram

1.4.2 Micro controller

A micro controller is a device that is integrated into a system and controls a single function. It accomplishes this by employing its core CPU to evaluate data received from its I/O peripherals. The micro controller's temporary data is saved in its data memory, where the processor retrieves it and decodes and applies the incoming data using instructions stored in its program memory. It then communicates with its I/O peripherals and takes the required action.

1.4.3 Inertial measurement unit (IMU)

The gyroscope and accelerometer sensors are integrated in the IMU, which refers for inertial measurement unit. The accelerometer measures acceleration, whereas the gyroscope measures angular velocity. Both of these sensors give data in three axes, allowing to determine the drone's rotation speed and orientation. This data is provided into a PID controller, which ensures that the drone is flying as desired by correcting any inconsistencies between the commands received and the actual movement determined by the sensors.

1.4.4 PID Control of the quadcopte

Proportional-Integral-Derivative controllers are frequently employed due to its simplicity and great efficiency in many applications. A well-tuned PID can address the majority of industry problems. The proportional controller (P), the integral controller (I), and the derivative controller (D) are the three separate controllers that make up the PID controller. Each of these three components has a simple and straightforward concept behind it. A proportional(P) controller produces a control signal proportionate to the current error. Put another way, as the error increases, the P block will output greater control signals in an attempt to reduce the error.The integral (I) term produces a signal proportional to the total of previous error values. When a single P controller fails to obtain zero error in steady state, this block comes in handy. When this happens, a constant error remains in the controller's input,and a simple I block may integrate that constant to provide an increasing control signal, leading to the error being reduced to zero. Finally,the derivative (D) term generates a control signal proportionate to the error signal's rate of change. As a result, it has a forewarning impact. To avoid responding to noisy input, this term usually has a built-in low-pass filter.This block has no effect on steady state.In many circumstances, a basic P controller is all that is required to maintain a stable system. Although it is occasionally preferable to employ a complete PID structure or even a more complicated controller [1].

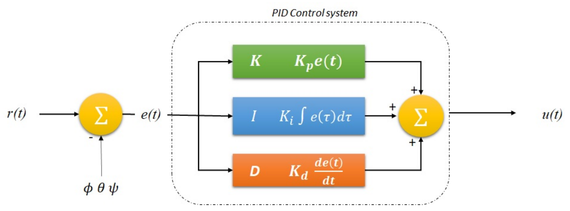


Figure 1.10: PID logic diagram.

Adopted control logic

The technique of stabilizing and controlling the quadcopter's flight is depicted in figure 1.11. The system has two control loops : an inner loop and an outer one, that feed into each other continually. The position reference (provided by the vision module), and angle accelerations measures, and the thrust reference are the system's inputs. The model can track the drone's position relative to the desired position due to the outer loop procedure.

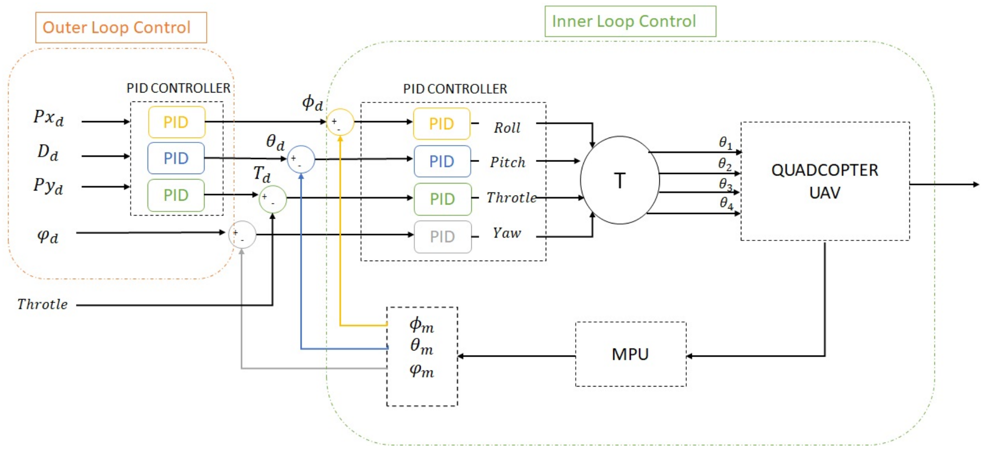


Figure 1.11: The adopted cascade loop for flight control.

The system states are sent as input into the internal control loop after the quadcopter has reached the coordinate points.

Four PID controllers for roll, pitch, yaw, and thrust control this part of the block diagram. To establish motor commands, these command inputs are transmitted to the quadcopter's motors.The factory controls cause flying adjustments in all six degrees of freedom. These control loops keep adjusting till the drone reaches the desired destination. This flight control system guarantees that the quadcopter's position is always within the desired tolerances.

Inner loop : The PID command of the inner loop makes the comparison between the measurements from the gyro and the desired angles (from outer loop), in order to ensure a stable movement in the three directions of the drone.

outer loop : The objective behind the outer loop is to ensure that the drone is in the desired position. for this the PID command compares the desired position (defined by the programmer) and the measured position (provided by the vision module).

The programmed controller is developed in Matlab/Simulink. Full details of the program are explained in Chapter 3.

1.5 Conclusion

At first, we presented the aerodynamics of the flight of a quad-copter by citing the physical efforts acting on it. as well as a brief description of the principle of operation of the flight controller is given. To finalize the chapter we opted for a cascade PID control loops in order to obtain a better stabilization of the quadcopter.

Chapter 2

Conception and realization of a quadricopter drone

2.1 Introduction

The overall architecture of our quadricopter system, as well as the many electronic components that make it up, were discussed in this chapter.

We next presented the quadcopter's CAD model, which we created on the Solidworks platform, in order to convert it to matlab simulink using the simscape multibody function. After implementing the PID command of our system as a block under the simulink platform, we converted it into C code in order to implement it on the stm32 board. is this a benefit of utilizing a stm32 board that is compatible with the matlab environment, which is really interesting.

2.2 Quadcopter system architecture

Our quad-copter is made with four brushless DC electric motors. The control board is based on an STM32H743VI board. The data processing of the different sensors is processed by a program developed in C language for programming the ARM Cortex-M7 processor integrated on the STM32 under the Stm32Cube IDE development environment. A 9 degree-of-freedom (DOF) sensor block is connected to the STM32 board. The sensor block contains a three-axis magnetometer, a three-axis accelerometer, and a three-axis gyroscope for roll, pitch, and yaw motion control. The commands for the four motors are delivered in the form of a PWM (Pulse Width Modulation) signal, which is sent to the motors by the ESCs, which also decode the data from the receiver. Figure 3.1 shows a block schematic of the architecture specific to the quadcopter created in this thesis.

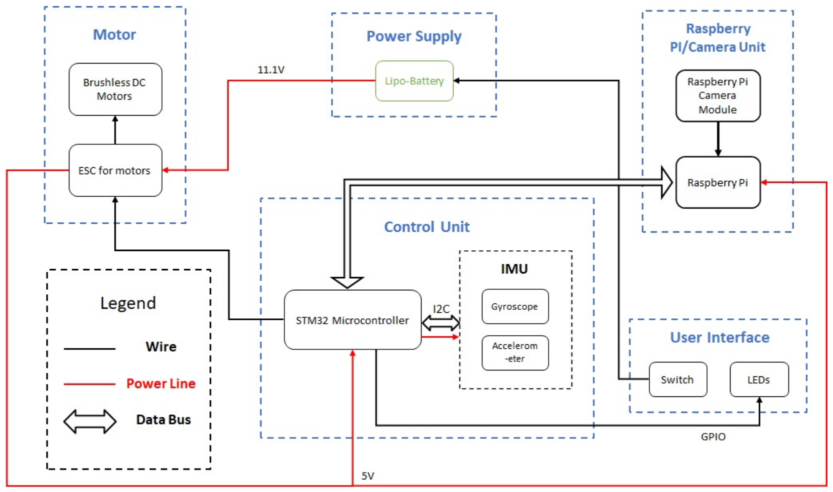


Figure 2.1: Quadcopter system architecture

2.3 Choice of material

2.3.1 The Frame

The weight of the chassis, which will be determined by the materials selected, as well as its shock resistance, are important considerations. A drone's chassis can be composed of a variety of materials. The choice of material is vital, depending on the intended use: a hardwood chassis will be more expensive, while a carbon fiber chassis will be lighter and more robust. Many things are determined by the chassis; in fact, if you choose a huge chassis, you will have to adjust everything to it. As a result, it is the first element to be selected.
We chose the JMT 380mm built in carbon fiber, because it is known to be strong and light, it is even equipped with a power distribution system to feed the ESCs.

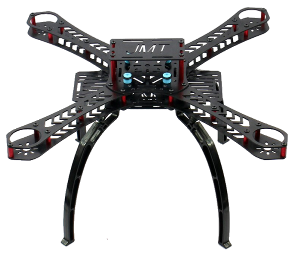


Figure 2.2: Drone's frame

2.3.2 Propulsion

The Motors

Brushes are the most common flaw in DC motors, as they cause friction and interference, as well as reducing the motor's life. Brushless Motors Direct Current (BLDC) are used in our quad-copter to eliminate all of these issues. A BLDC motor is an electric machine that belongs to the synchronous machine category, with a rotor made up of one or more permanent magnets and a rotor position sensor as standard. Brushless motors are substantially more efficient and resistant than "brushed" motors. They are, however, heavier. A motor's designation is determined by two factors: the size of its stator and its kV (speed of rotation for 1 Volt, with no load) which makes it possible to define the maximum speed of rotation of the motor according to the voltage of the battery.
Our choice is based on the mass of the quad-copter and the type of flight. We chose a brushless motor with 920 KV.

Figure 2.3: Brushless motor

Speed controllers

The ESC is an electronic speed controller that connects a motor's control signal to its power circuit. It is used to regulate the motor's speed and maximum current consumption. It is connected to an accumulator, which will give the motor with the necessary power. Pulse width modulation (PWM) is used to regulate the ESC: it is a pulse over a period of time t, and the higher it is during this period, the more power the ESC will give to the motor. An ESC will be selected on the basis of the consumption (A amps) and voltage (V volts) of the motor it will be controlling. It is advised that the controller be oversized by roughly 50 percent in comparison to the intensity (A) that it will have to sustain in order to ensure optimal and, above all, long-term functioning.

Propellers

The bigger the propeller, the more lift it can generate (more stable flight). Propellers frequently have two dimensions: diameter and pitch. The diameter affects the traction, whereas the pitch affects the speed of the model.

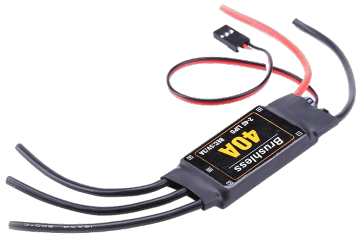


Figure 2.4: Electronic speed controller 40A

- **Pitch**:
  - Small pitch provides more traction at low speeds but restricts maximum speed.
  - Large pitch: low-speed pull is smaller, while top-speed is considerable.
- **Rotational speed and propellers:**
  - High KV (rapid spinning) combined with a short propeller equals great aerobatics and speed (racers).
  - Low KV (slower rotation) with a huge propeller make for a more steady flying and autonomy (UAV).

As a result, we chose propellers with an 10-inch diameter and a 4.7-inch pitch.

Figure 2.5: Propellers 10x47

2.3.3 Power supply

Battery

In the model world, LiPo batteries are the most common. A LiPo has the benefit of being able to be charged 100 times without losing capacity. Each cell in a LiPo has a nominal voltage of 3.7 volts and a maximum charging voltage of 4.2 volts. A LiPo of 5000 mAH has been selected to give to quadcopter more autonomy.

2.3.4 Flight controller

Microcontroller

The STM32 microcontroller series from STMicroelectronics was chosen because it has the computational capability to perform all the tasks required. STM32h743VI is used which is based on the high performance Arm® Cortex®-M7 32 bit RISC core operating at up to 480 MHz, which is more than enough to run the PID loop at 9KHz, which is the maximum rate the IMU can output data. It has a FPU (floating point unit) which is required to make fast calculations on the IMU data. It has DMA (direct memory access), which is necessary for faster communication with the ESCs, as they can directly read the speed control without the STM32 CPU in between.

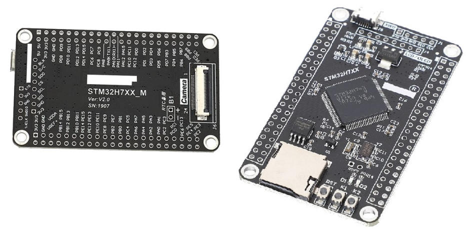


Figure 2.6: STM32H7xx

Sensors

MPU-9250 is a multi-chip module (MCM) consisting of two dies integrated into a single QFN package. One die houses the 3-Axis gyroscope and the 3-Axis accelerometer. The other die houses the AK8963 3-Axis magnetometer from Asahi Kasei Microdevices Corporation. Hence, the MPU-9250 is designed to interface with multiple non-inertial digital sensors, such as pressure sensors, on its auxiliary I2C port [3].

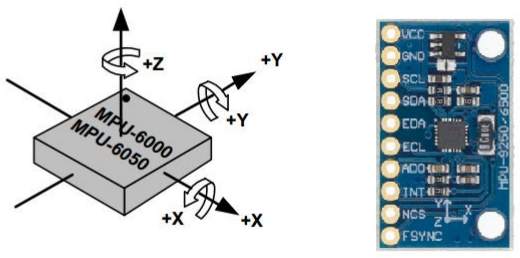


Figure 2.7: IMU9250

Acceleration and accelerometers:
The accelerometers contain an internal MEMS (Micro Electro Mechanical Systems) technology that, like a spring mass system, allows acceleration to be measured, taking into consideration that even if there is no movement, the accelerometer will always sense gravity's acceleration.

Angular velocity and gyroscope:
The rate of change of angular displacement per unit time, or the rate at which a body spins about its axis, is known as angular velocity. The Coriolis effect is used by gyroscopes to detect angular velocity using a MEMS (Micro Electro Mechanical Systems) device. The angular velocity can be measured with a gyroscope, and the angular displacement may be calculated by integrating the angular velocity with respect to time (angular position).

2.4 CAD of quadracopter

This type of realization represents the different points of view of parts of the mechanical design software SolidWorks, which is a computer-aided design (CAD) software that uses parametric design. Generates 3 types of files related to three basic concepts: the part, the assembly, and the drawing. These files are related; any modification at any level is reflected in all the files involved.

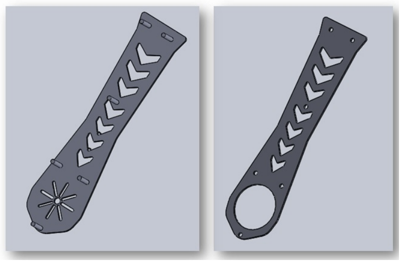


Figure 2.8: Inner-Outer arm

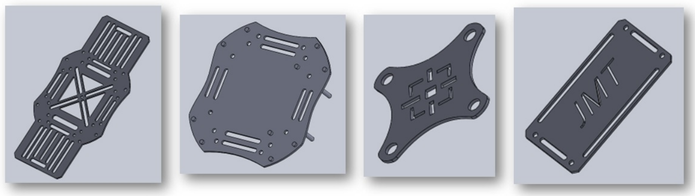


Figure 2.9: Pieces

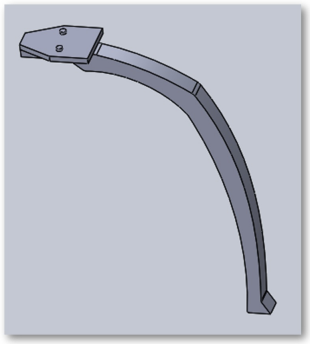


Figure 2.10: Quadcopter feet

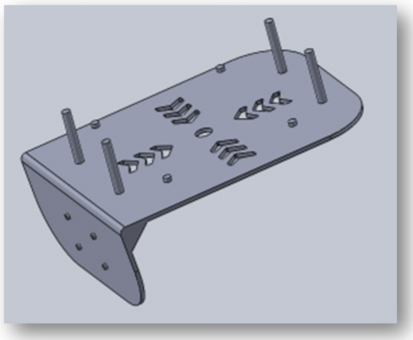


Figure 2.11: Camera support

Once the different parts of the drone are made, the parts must be assembled. The figures 2.12, 2.13, 2.14, and 2.15 show the CAD model from different angles.

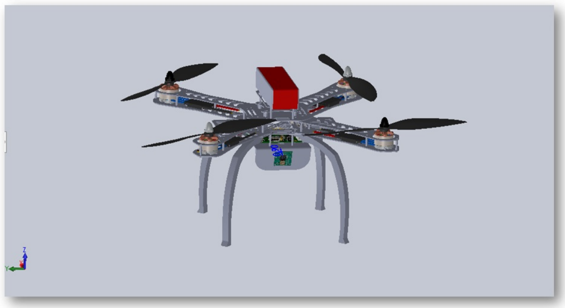


Figure 2.12: The quadcopter's CAD

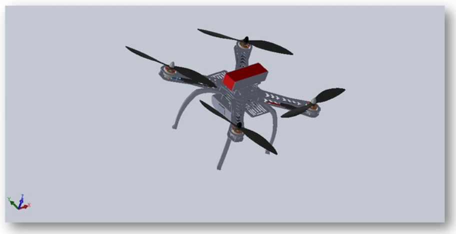


Figure 2.13: The quadcopter's CAD

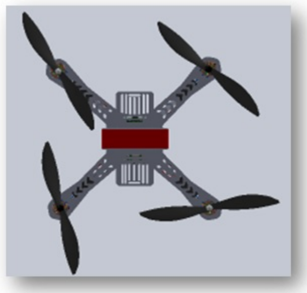


Figure 2.14: The quadcopter's CAD

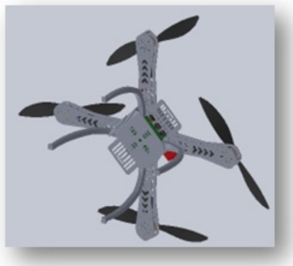


Figure 2.15: The quadcopter's CAD

2.5 Simscape simulation

Simscape™ enables you to rapidly create models of physical systems within the Simulink® environment. With Simscape you build physical component models based on physical connections that directly integrate with block diagrams and other modeling paradigms. You model systems such as electric motors, bridge rectifiers, hydraulic actuators, and refrigeration systems by assembling fundamental components into a schematic. Simscape add-on products provide more complex components and analysis capabilities [4].

The Solidworks CAD model is exported to Matlab using a Simscape Multibody, which links the two platforms, and then routed to the Simulink tool.

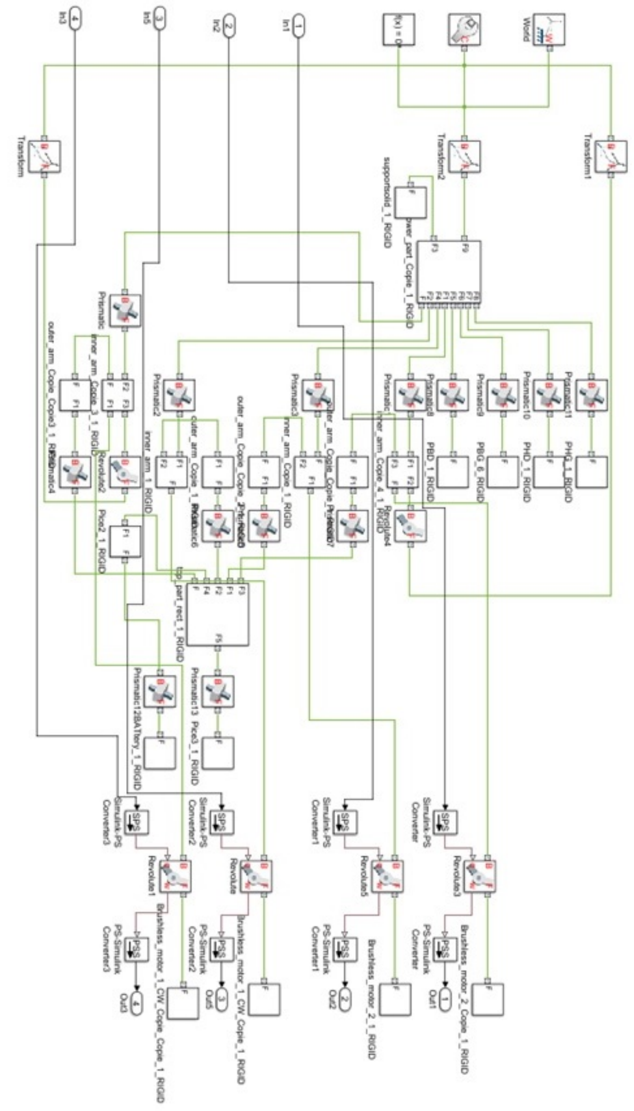


Figure 2.16: Simulink model of the quadcopter dynamic’s.

2.6 Flight Controller Programming

2.6.1 Flight controller flowchart

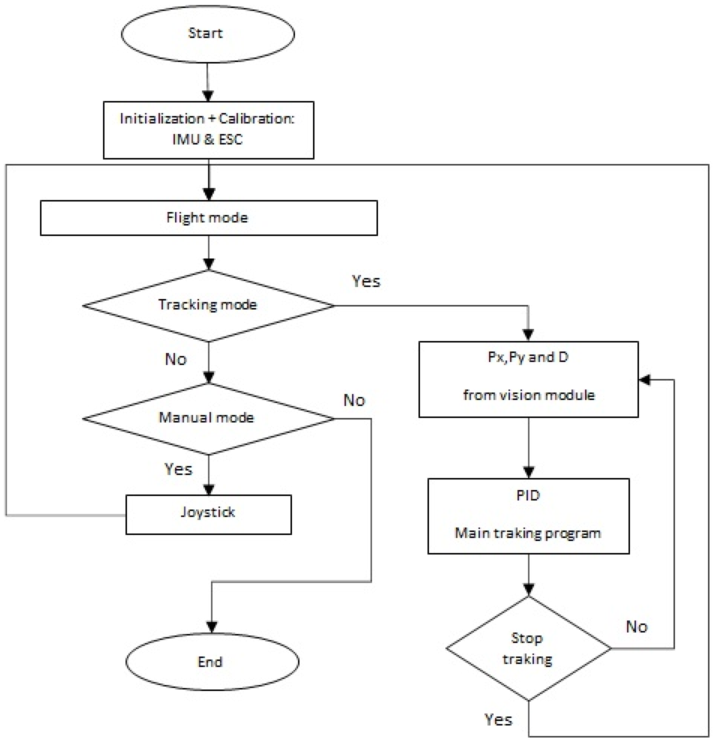


Figure 2.17: Flight controller flowchart

2.6.2 Programming the MPU-9250 Inertial Unit

The connection of MPU9250 with the STM32 is presented by Figure .2.18

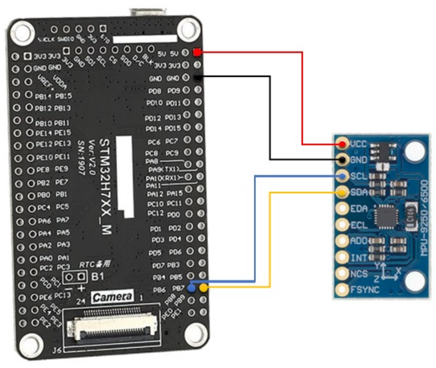
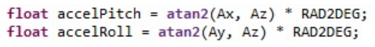


Figure 2.18: The connection between MPU9250 and STM32

Calculation of Euler angles from accelerations

If we consider that the sensor is subjected to only one force: gravity. As gravity is always vertical, the values obtained in the accelerometer component correspond to gravity, and the angles of the resultant are the tilt of the sensor plane. As a result, the apparent position of the acceleration vector may be used to determine sensor rotation [2].



```
float accelPitch = atan2(Ax, Az) * RAD2DEG;
float accelRoll = atan2(Ay, Az) * RAD2DEG;
```

Figure 2.19: Euler angles from accelerations

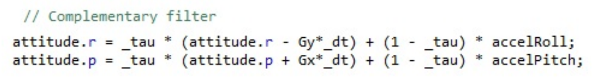
Implementation of a Complementary filter

The complementary filter's concept is to blend slow moving data from an accelerometer with rapid moving signals from a gyroscope.
In static situations, the accelerometer is a reliable measure of orientation.
In dynamic settings, a gyroscope is an excellent indication of tilt. The goal is to combine the accelerometer and gyroscope signals after passing them through a low-pass filter and a high-pass filter, to get the final rate.
The crucial element here is that the low-pass and high-pass filters' frequency responses sum up to 1 at all frequencies [2].

2.6.3 Programming a PID regulation

PID implementation

The inner loup gains are chosen as follows:



```
// Complementary filter
attitude.r = _tau * (attitude.r - Gy*_dt) + (1 - _tau) * accelRoll;
attitude.p = _tau * (attitude.p + Gx*_dt) + (1 - _tau) * accelPitch;
```

Figure 2.20: Complementary filter

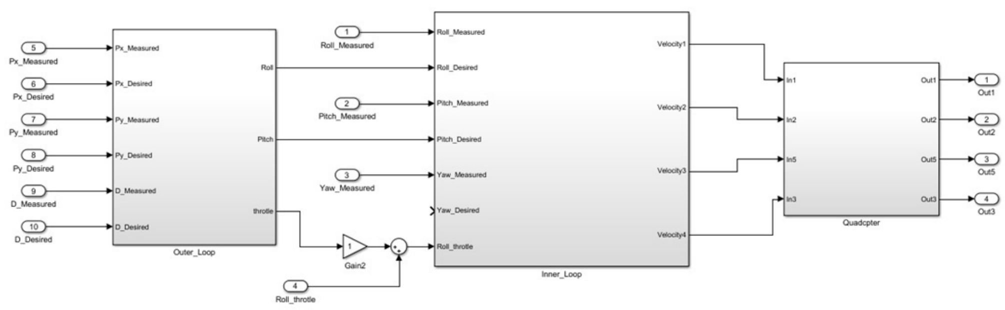


Figure 2.21: PID controller

	θ_1	θ_2	θ_3	θ_4
ROLL	+	-	-	+
PITCH	+	+	-	-
YAW	+	-	+	-

The coefficients of a PID

Testing may be used to adjust the K_p, K_i, and K_d coefficients of a PID. First of all, a basic proportional regulator must be set up (the coefficients K_i and K_d are therefore equal to zero). It is required to alter the K_p coefficient through testing in order to enhance the system's reaction time. After we've set this coefficient, we may go on to the Ki coefficient. This one will allow to correct the system's ultimate error. Finally, we may go on to the last coefficient K_d, which allows to improve the system's stability.

2.7 Deploying C-code to STM32

The flight controller has been developed under Simulink/Matlab environment. Using Matlab Embedded coder and Simulink Coder, an optimised C-code can be generated. This c-code is then deployed to the STM32 board. For this, the c-code of developed flight controller of the Figure.2.21 is generated following these steps :

- Specify the flight controller inputs and outputs, as illustrated by Figure .2.22.

- Select quick start on Embedded Coder,
- Select C-code as the output of the generated code
- Select the appropriate Target Hardware processor type (Arm 7 for STM32H743)
- Select the generated code objective : execution efficiency vs RAM efficiency.

Once all the parameters are chosen correctly, a set of headers and c codes are generated. These files are then integrated on STM32cube IDE under source codes.

One important parameter to be fixed carefully is the sampling time. In this project, a sampling time of 0.01 sec is chosen. This choice is a comprise to get an execution efficiency within the stm32 board, and to handle different information from MPU9250 and the Raspberry PI.

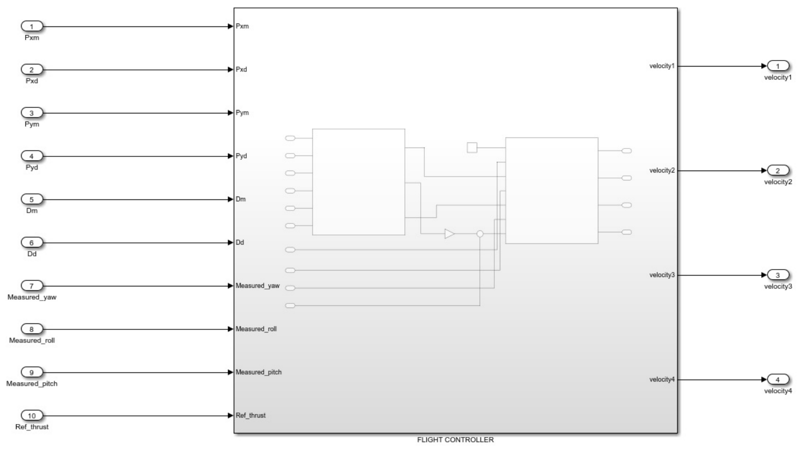


Figure 2.22: Flight controller's Inputs/outputs.

2.8 The communication between STM32 and Raspberry

In order to connect STM32 with the Raspberry Pi to allow communication between them, we have chosen a communication under I2C protocol.

The I2C protocol involves using two lines to send and receive data; a serial clock pin (SCL) that the Raspberry board (master) pulses at a regular interval, and a serial data pin (SDA) over which data is sent between the two boards.

2.8.1 Circuit

We connect respectively, the pin 3 (SDA) and the pin 5 (SCL) of the Rasspberry pi to the pin PB7 and PB6 of the STM32.

we connect also the Ground (GND) pin of these two boards with a wire.

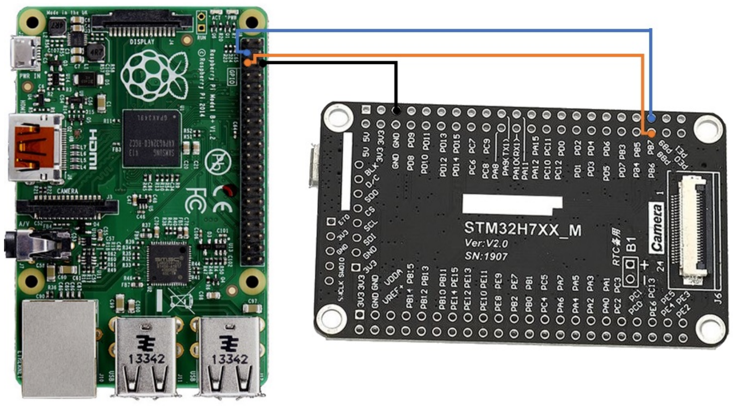


Figure 2.23: Communication circuit

2.8.2 Program logic

the program that we made it in both cards allow only the raspberry pi sends the data to the stm32 via the address 0x08 that we chose in the configuration of the I2C bus in the stm32cubeMX

2.9 Conclusion

In this chapter, we have presented the experimental part of the project. First of, the CAD model of the quadcopter has been developped under solidworks environnment and exported to Simscape/Matlab to test and to validate the flight control algorithm. Then, a c-code has been generated and implemented on STM32 Arm cortex card to control the quadcopter. An i2c communication between the two boards was made in order to process the gyroscope measurements and the vision module data in stm32 to track the object of interest.

Once the software programs implementation was completed, we moved on to assembling all the components which were successfully completed, and so we concluded our project.

Conclusion and prospects

In the present thesis, we presented a tracking moving object algorithm by unmanned aerial vehicles. This project is composed of two parts.

In the first part, the tracking algorithm of moving object has been presented. The developed algorithm is based mainly on OpenCV packages since a Rassberry Pi board has been used. In case of losing of the target object, and to ensure that the quadcopter continues following the object, a Kalman estimator is considered assuming that the target movement has a constant acceleration.

In the second part of the project, a cascade of PID controller is presented. This controller has been implemented on STM32 Arm Cortex board.

The presented algorithm is completely embedded which makes the quadcopter autonomous. There is still a lot of improvement in sense that the moving target do not follows a constant acceleration. Also, the identification of the object of interest is not always accurate; it will be interesting to test another recognition algorithm instead of Haar cascade algorithm.

This project was a challenge in the beginning. At the end, we can estimate that the initial objectives have been achieved and that through this project, we learned a lot of complementary information to our master's training in automation and acquired skills such as programming (python for example) and managing projects.

Bibliography

[1] [FELIPE RIBAS SILVA DE AZEVEDO. Complete system for quadcopter control. In ., 2014.

[2] O. BENHAMOU et B. BOUDEBZA. Réalisation d'un drone quad-copter. 2019.

[3] InvenSense. Mpu-9250 product specification revision 1.1. In *InvenSense*, 2020.

[4] Mathworks. Model and simulate multidomain physical systems. In *Get Started with Simscape*, 2020.

[5] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.