PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGH EDUCATION AND SCIENTIFIC

RESEARCH

UNIVERSITY YAHIA FARES OF MEDEA



Faculty of Technology

Department of Mechanical Engineering

MASTER THESIS

Branch: Electromechanical Engineering

Specialty: Mechatronics

Theme:

Artificial recognition drone

Presented by:

- **Mr.** Yaman Eyad Ayish Abu Mosameh

Proposed and Directed by:

- **Prof.** Abu Dura Salam

Academic Year 2023 / 2024

DEDICATION

I dedicate this humble work: **To ALLAH** with deepest gratitude for bestowing upon me the knowledge, perseverance, and strength to complete this endeavor.

My dear parents **(Eyad&Mayyada)** who sacrificed their lives for my success and Encourage them throughout my studies.

To my grandparents **(Zainab, Zizi, Ayish&Mohamad),**

to my dear brothers and dear sisters,

to my aunts, (**Ahlam, Dalal, Hanane, Iman, Jasmine**) uncles**, (Emad ,Ahmad Shatat,&kamal)** and cousins **(Omar Majdalo)** to all of my family Without forgetting all of my teachers who contributed to my work and participated in my education, especially to mention to all my **foreign colleagues** and **friends.**

To all the students in my class M1: **Mecatronics**. To all those I love, who love me, and to all who knew me. To everyone who helped me complete this project.

ACKNOWLEDGMENTS

I am incredibly fortunate to have a supportive network of family and friends. To my parents, [Parents**: Eyad&Mayyada**], thank you for your unwavering love and encouragement throughout my academic journey. Your belief in me has always been a source of strength.

I would like to express my deepest gratitude to my thesis supervisor, [**Prof.Abu Dura Salam**], for their invaluable guidance, encouragement, and expertise throughout this project.

To my friends, [**Muntaser Al-Shrafi, Fouad Yousri & Yacine Mami**] and specially to mention all my foreign colleagues and friends.

ملخص

هذه الوثيقة هي دليل شامل للطائرات بدون طيار (UAVs) ، والمعروفة أيضًا بالدرونز.يغطي جوانب مختلفة للطائرات بدون طيار ، بما في ذلك وظائفها وتكنولوجياتها وتطبيقاتها.الفصل الأول: الطائرات بدون طيار * يوفر فهمًا أساسيًا للطائرات بدون طيار. يستكشف تعريفها وخلفيتها التاريخية وتصنيفاتها والمكونات الأساسية للطائرة متعددة الدوارات والهياكل العامة ومزايا استخدام الطائرات متعددة الدوارات.الفصل الثاني: التعرف على الوجوه * يتعمق في تقنية التعرف على الوجوه المستخدمة في الطائرات بدون طيار. ويشرح مفهوم التعرف على الوجوه وأنظمة الاستحواذ وتقنيات استخراج الميزات والمطابقة ، واتخاذ القرار في سياق التعرف على الوجوه باستخدام الطائرات بدون طيار ، وتنفيذ أكواد Python للتعرف على الوجوه.الفصل الثالث: كشف النقاب عن NEO-M8N: وحدة GPS قوية * يركز على NEO-M8N ، وهي وحدةGPS تستخدم بشكل شائع في الطائرات بدون طيار. يستكشف الوظائف والميزات الأساسية لـ NEO-M8N ، مع تسليط الضوء على مزاياها مقارنة بالخيارات الأخرى. كما أنه يوضح بالتفصيل عملية تثبيت NEO-M8N باستخدام متحكم.Arduino الفصل الرابع: التحكم * PID يتعمق في التحكم في تحليق الطائرة بدون طيار باستخدام منظم PID (نسبية تكاملية مشتقة) على منصة .Arduino ويشرح مفاهيم الأنظمة المفتوحة والمغلقة ، ثم يفصل المكونات الثلاثة لمنظم .PID يحدد الفصل كيفية تنفيذ مُتحكم PID في رمز Arduino ، بما في ذلك الوظائف الأساسية والاعتبارات الخاصة بتشغيل طائرة بدون طيار بأمان.تختتم الوثيقة باستنتاج عام يلخص النقاط الرئيسية المستفادة من الفصول الفردية

كلمات مفتاحية : الطائرات بدون طيار(UAVs) ، طائرات التحكم عن بعد، مركبة جوية بدون طيار، مركبة جوية ذاتية التشغيل، التعرف على الوجوه باستخدام الطائرات بدون طيار، نظام تحديد المواقع العالمي(GPS) ، نظام الملاحة الفضائية العالمي(GNSS) ، NEO-M8N، نظام مفتوح الحلقة، نظام مغلق الحلقة، فعل نسبي(Kp) ، فعل تكاملي(Ki) ، فعل مشتق(Kd) ، منظمPID ، آردوينو(Arduino) ، وحدة تحكم السرعة الإلكترونية(ESC) ، معاملاتPID

Abstract:

This document is a comprehensive guide to Unmanned Aerial Vehicles (UAVs), also known as drones. It covers various aspects of drones, including their functionalities, technologies, and applications. *Chapter I: Drones* provides a foundational understanding of drones. It explores their definition, historical background, classifications, essential components for a quadcopter, general architectures, and the advantages of using quadrotors. *Chapter II: Facial Recognition* delves into facial recognition technology employed on drones. It explains the concept of facial recognition, acquisition systems, feature extraction and matching techniques, decision making in the context of drone-based facial recognition, and the implementation of Python codes for facial recognition. *Chapter III: Unveiling the NEO-M8N: A Powerful GPS Module* focuses on the NEO-M8N, a GPS module commonly used in drones. It explores the core functionalities and features of the NEO-M8N, highlighting its advantages over other options. It also details the installation process of the NEO-M8N using an Arduino microcontroller. *Chapter IV: PID Control* dives into controlling a drone's flight using a PID (Proportional-Integral-Derivative) regulator on an Arduino platform. It explains the concepts of open-loop and closed-loop systems, then details the three components of a PID regulator. The chapter outlines how to implement a PID controller in Arduino code, including essential functionalities and considerations for operating a drone safely. The document concludes with a general conclusion that summarizes the key takeaways from the individual chapters.

Keywords.

Unmanned Aerial Vehicles (UAVs), remotely controlled aircraft, pilotless aerial vehicle, autonomous aerial vehicle, Drone-based facial recognition, GPS (Global Positioning System), GNSS (Global Navigation Satellite System), NEO-M8N, Open-loop system, Closed-loop system, Proportional action (Kp), Integral action (Ki), Derivative action (Kd), PID regulator, Arduino, ESC (Electronic Speed Controller), PID coefficients.

Résumé :

Ce document est un guide complet sur les véhicules aériens sans pilote (UAV), également appelés drones. Il couvre divers aspects des drones, y compris leurs fonctionnalités, leurs technologies et leurs applications. *Chapitre I: Drones* fournit une compréhension fondamentale des drones. Il explore leur définition, leur contexte historique, leurs classifications, les composants essentiels d'un quadcoptère, les architectures générales et les avantages de l'utilisation de quadcoptères. *Chapitre II: Reconnaissance faciale* explore la technologie de reconnaissance faciale utilisée sur les drones. Il explique le concept de reconnaissance faciale, les systèmes d'acquisition, les techniques d'extraction et de correspondance de caractéristiques, la prise de décision dans le contexte de la reconnaissance faciale par drone et la mise en œuvre de codes Python pour la reconnaissance faciale. *Chapitre III: Dévoilement du NEO-M8N: Un module GPS puissant* se concentre sur le NEO-M8N, un module GPS couramment utilisé dans les drones. Il explore les fonctionnalités et les caractéristiques principales du NEO-M8N, en soulignant ses avantages par

Mots-clés : Véhicules aériens sans pilote (UAV), avion télécommandé, véhicule aérien sans pilote, véhicule aérien autonome, reconnaissance faciale par drone, GPS (Global Positioning System), GNSS (Global Navigation Satellite System), NEO-M8N, système ouvert, système fermé, action proportionnelle (Kp), action intégrale (Ki), action dérivée (Kd), régulateur PID, Arduino, ESC (contrôleur électronique de vitesse), coefficients PID.

Contents

DEDICATION i

ACKNOWLEDGMENTS ii

Abstract: iii

Contents iv

List of figure vi

List of tables vii

Abbreviations viii

Symbols and Units of Measurement ix

General Introduction 1

Chapter I: Drones History 2

I.1. Introduction: 2

I.2. Definition of a Drone 2

I.3. Historical Background: 2

I.4. Classification of Drones 3

I.4.1. According to wings: 3

I.4.2. According to Size: 3

I.4.3. According to Propulsion: 5

I.5 Essential Components for a Quadcopter: 8

I.5.1. Motors: 8

I.5.2. Electronic Speed Controllers (ESCs) [2]**:** 9

I.5.3 Battery: 10

I.5.4 NEO-M8N Recievers: 11

I.6. General architectures: 12

I.6.1 Drone Control Architectures: 12

I.6.2 Advantages of Quadrotor: 13

I.7. Conclusion: 15

Chapter II: 16

facial recognition 16

II.1.1 Introduction: 16

II.1.2. Main challenges of facial recognition: 16

II.1.3. The structure of a facial recognition system: 17

II.1.4. System hardware model: 21

II.2. Feature transformation based on keypoints: 22

II.2.1. Scale invariant feature transform (SIFT): 22
II.2.2. Approach : 23
II.2.3. Linear techniques 23
II.2.5. Recognition systems: 26
II.2.6. Rowley neural network: 26
II.2.7. Algorithme of rowley: 26
II.3. Hardware and software tools: 27
II.3.1. Material: 27
II.3.2. Modules: 27
II.4. Preconditions: 29
II.4.1. Installation requirements: 29
II.4.2. Configure faces to recognise: 30
II.4.3. Camera: 30
II.5. Create a facial recognition system in pythen: 31
II.5.1. Stabilization: 31
II.5.2. Create a folder and save photos of known people: 32
II.5.3. Building a line facial recognition system: 32
II.5.4. Python code to make a drone scan a specific area with facial recognitioin and direct the drone towards the recognised person: 40
II.5.5. Geographic coordinates: 42
II.6. Mathematical model and deriviation of geographic coordinate of the target: 43
II.6.1. Cartesian coordinates: 43
II.6.2. Gps technical overview: 43
II.6.3. Gps reciever technology: 43
II.6.4. World geodetic system (WGS-84): 44
II.6.5. Formula for transforming the geographic coordinate system: 44
II.6.6. Methodology: 45
II.6.7. Results and discussions: 46
II.6.8. Conclusion and suggestions: 48
II.7. Conclusion: 49
Chapter III: 55
Unveiling the NEO-M8N: A Powerful GPS Module 55
III.1. Introduction: 31
III.2. Core Functionality and Features 31

III.3. Advantages of the NEO-M8N 32
III.5. Installation of gps NEO-M8N using arduino microcontroller. 38
III.6. Conclusion: 39
Chapter IV: 40
PID Control 40
IV.1. Introduction: 40
IV.2. PID regulator: 41
IV.3. Proportional Action: 42
IV.4. Actions 43
IV.4.1. Integral Action 43
IV.4.2. Derivative Action 43
IV.4.3. PID action: 44
IV.5. Arduino installation: 44
IV.5.1. Implementation for Arduino: 44
IV.5.2. Calculation of instructions: 45
IV.5.4. PID regulator: 48
IV.6. Necessary functions: 51
IV.6.1. Start/stop: 51
IV.6.2. Compensate for battery drop 54
IV.6.3. Generate ESC signals: 56
IV.7. Wiring: 57
IV.7.1. Calibration of PID coefficients 57
IV.7.2. Test bench: 58
IV.7.3. Calibration procedure: 58
IV.8. First flight: 59
IV.8.1. Safety & legislation: 59
IV.9. Conclusion 60
General Conclusion: 61

List of figure

Figure.I. 1:Military surveillance drone 6
Figure.I. 2:Drone dropping supplies. 6
Figure.I. 3:Drone firing a missile 6
Figure.I. 4:Drone monitoring crops 7
Figure.I. 5:Drone delivering medical supplies 7
Figure.I. 6:Drone searching for survivors 8
Figure.I. 7:SunnySky V2216-11 brushless motor 9
Figure.I. 8:An electronic speed controller. 10
Figure.I. 9:4S LiPo (Lithium Polymer) battery 11

Figure. II. 1: Real-time Process Model 18
Figure. II. 2: Face Detection Using HOG 19
Figure. II. 3:steps to create gradients 19
Figure. II. 4 :Extracting a new facial pattern very similar to the general facial pattern. 20
Figure. II. 5:The 68 benchmarks detected by the dlib library. This image was created by Brandon Amos from CMU who works on OpenFace. /translate to english without any other thingr 21
Figure. II. 6:Figure. II.6. Quad-copter model used for the proposed system 22
Figure. II. 7:Rowley neural network 27
Figure. II. 8:PNG or JPG formats are accepted. 30
Figure. II. 9:detect faces like Zuckerberg's example 30
Figure. II. 10: Facial recognition system. 30

Figure.III. 1 Autonomous drone used for mapping areas 32
Figure.III. 2:Tructor used to harvest autonomously 33
Figure.III. 3:Tracking cargo trucks 33
Figure.III. 4:Microcontroller installation of Neo-M8N GPS 33
Figure.III. 5:Autonomous drone used scanning photography. 36
Figure.III. 6:Autonomous drone used for rescuing and searching. 36
Figure.III. 7:The way packages delivered by some autonomous drones. 37

Figure. IV. 1:Closed loop servo system 40
Figure. IV. 2:Block diagram of a PID regulator. 41
Figure. IV. 3:Example of a system responding to a given instruction 42
Figure. IV. 4: Response of an unstable system and oscillating system 43
Figure. IV. 5: Effect of varying the parameters of a PID regulator 44
Figure. IV. 6:• Roll: positive roll • Pitch: positive pitch • Yaw: positive yaw 49
Figure. IV. 7:Start/stop state machine 52
Figure. IV. 8:the receiver is powered by a BEC output from one of the ESCs 57
Figure. IV. 9:Benchmark example. 58

List of tables

Table.III. 1:summarizing the key technical specifications of the NEO-M8N 31

Table. IV. 1:Summary of proportional action 43
Table. IV. 2:Summary of derivative action 44
Table. IV. 3:Summary of the full action 44
Table. IV. 4:Roll axis table 49
Table. IV. 5:Pitch axis table: 50
Table. IV. 6:yaw axis table: 50

Abbreviations

AUV: Autonomous Underwater Vehicle

AR: Artificial Recognition

C**V:** Computer Vision

DOF: Degrees of Freedom

Kd: Derivative Gain

Ki: Integral Gain

Kp: Proportional Gain

MCU: Microcontroller Unit

MR: Machine Recognition

OR: Object Recognition

PD: Proportional-Derivative

PI: Proportional-Integral

PID: Proportional-Integral-Derivative

RMSE: Root Mean Square Error

RPAS: Remotely Piloted Aircraft System

SETPOINT: The desired reference value for the controlled variable

UAS: Unmanned Aircraft System

UAV: Unmanned Aerial Vehicle

Symbols and Units of Measurement

Length: meters (m)

Weight: kilograms (kg)

Time: seconds (s)

Velocity: meters per second (m/s)

Altitude: meters (m)

Battery Capacity: ampere-hours (Ah)

Power: watts (W)

Flight Time: minutes (min) or hours (hr)Accuracy: percentage (%)

Processing Time: seconds (s)

Resolution: pixels per inch (PPI)

Field of View: degrees (°)

Image Size: pixels (px)

Training Data Size: number of samples

Detection Rate: detections per second

Antenna Height: meters (m)

Latitude: degrees (°)

Longitude: degrees (°)

Altitude: meters (m)

Speed: meters per second (m/s)

Heading: degrees (°)

Accuracy: meters (m)

Time to First Fix: seconds (s)

Error: units corresponding to the controlled variable

Control Signal: units corresponding to the controlled variable

Time: seconds (s)

Gain: dimensionless

General Introduction

General Introduction

Introduction:

Unmanned aerial vehicles (UAVs), commonly known as drones, have revolutionized various industries with their versatility and capabilities. In this thesis, we delve into different aspects of drones, from their general overview to their integration with advanced technologies.

The first chapter provides a comprehensive understanding of drones. We explore the fundamental concepts, components, and capabilities of drones, shedding light on their definition and significance in today's world. Additionally, we discuss the wide range of applications and potential benefits that drones offer across various sectors.

Building upon the foundation of drones, Chapter 2 focuses on the integration of artificial recognition technologies. Specifically, we explore the combination of drones with artificial recognition, with a focus on object recognition. By leveraging techniques such as computer vision and machine learning, drones equipped with artificial recognition capabilities can enhance surveillance, monitoring, and search and rescue operations.

Accurate positioning is vital for drone operations, and the installation of a reliable Global Navigation Satellite System (GNSS) receiver is crucial in achieving precise location information. In Chapter 3, we delve into the installation process of the NEO-M8N receiver, a widely used GNSS receiver in drone systems. We cover topics such as antenna placement, configuration settings, and calibration procedures, ensuring optimal performance and accurate positioning of drones.

To achieve stable and precise control, the implementation of Proportional-Integral-Derivative (PID) control is essential in drone systems. Chapter 4 focuses on the theory, design, and implementation of PID control in drones. We explore the fundamental principles of PID control, tuning methods, and performance analysis. By understanding PID control and its application in drones, we can optimize control systems and improve the stability and performance of drones.

By delving into these chapters, we aim to provide a comprehensive understanding of drones, their integration with artificial recognition technologies, the installation of GNSS receivers, and the implementation of PID control. This knowledge will contribute to the advancement and utilization of drones in various industries, opening up new possibilities for their applications.

Chapter I: Drones History

I.1. Introduction:

Unmanned Aerial Vehicles (UAVs), commonly known as drones, are unmanned aircraft that utilize aerodynamic forces to achieve vertical flight. They can be remotely piloted, autonomous, or semi-autonomous. UAVs are capable of carrying various payloads, enabling them to perform specific tasks for varying durations depending on their capabilities. Initially developed for military applications such as surveillance, reconnaissance, target designation, and combat, drones have found diverse uses in civilian domains as well [1].

They are employed in natural phenomena observation (e.g., avalanches, volcanoes), pesticide spraying on agricultural fields, environmental monitoring (e.g., pollution measurement), surveillance of road networks, infrastructure maintenance, among others. This chapter provides an overview of drones, including their definition, historical background, and classification based on several criteria.

I.2. Definition of a Drone

There exist several definitions in literature; We present the following two:

Definition: A drone is a remotely controlled aircraft without a pilot, designed for missions such as surveillance, intelligence gathering, combat, or transport.

The term "drone" appeared in the French language in 1954, derived from the English meaning "male bee", indicating a small unmanned reconnaissance aircraft, remotely controlled or programmed. A broader definition states that a drone (UAV - Unmanned Aerial Vehicle) is a pilotless aerial vehicle equipped with payloads for missions such as combat with missiles or bombs. It can be piloted from ground stations with or without satellite relay and can operate autonomously. In the future, drones are expected to be assisted by control systems installed on aircraft (flying control stations).

I.3. Historical Background:

The conceptualization of drones dates back to the end of World War I. While the United States was developing the Hewitt-Sperry automatic airplane project, in France, George Clemenceau, president of the Senate's army commission, initiated a project for "unmanned aircraft". Captain Max Boucher developed an automatic piloting system that enabled an aircraft called the Voisin BN3 to fly over a hundred kilometers.

At that time, all major powers sought a means to carry out aerial missions without risking the lives of their pilots. Thus, the roots of drones are deeply military before gradually expanding into the civilian domain.

I.4. Classification of Drones

There is no single way to classify drones; They can be categorized according to various criteria: altitude, mission, size, control system, and more. The aerodynamic functioning provides another possibility for classification. Thus, drones can be structured into three main families [1]:

I.4.1. According to wings:

1. Fixed-wing drones: These drones resemble traditional airplanes, featuring fixed wings that generate lift as the aircraft moves forward. They are typically used for long-range missions and can achieve high speeds and altitudes.

2. Rotary-wing drones: Also known as rotorcraft or multicopters, these drones utilize rotating blades to generate lift and propulsion. They are highly maneuverable and can hover in place, making them suitable for tasks requiring precise movement and close-quarters operations.

3. Hybrid drones: Combining features of both fixed-wing and rotary-wing aircraft, hybrid drones offer versatility in terms of flight capabilities.

They can take off and land vertically like rotary-wing drones while also benefiting from the efficiency and endurance of fixed-wing aircraft for longer missions.

Each classification offers distinct advantages and is chosen based on the specific requirements of the mission at hand. Additionally, drones within each category may vary in size, payload capacity, and level of autonomy, further diversifying their applications across various industries and sectors.

I.4.2. According to Size:

There are several categories of aerial drones, ranging from nano-drones weighing just a few grams to heavy drones capable of conducting missions lasting over 24 hours and covering thousands of kilometers from their base.

- **Nano-drones**: These are the smallest category of drones, often weighing only a few grams. They are compact, lightweight, and highly maneuverable, making them suitable for indoor or close-range operations such as surveillance or reconnaissance in tight spaces.

- **Micro-drones:** Slightly larger than nano-drones, micro-drones typically weigh a few hundred grams to a few kilograms. They offer improved stability and payload capacity compared to nano-drones, making them suitable for a wider range of tasks including photography, mapping, and environmental monitoring.
- **Small drones:** Small drones encompass a broad range of sizes, typically weighing between a few kilograms to around 25 kilograms. They are versatile and can be equipped with various payloads for tasks such as aerial photography, crop monitoring, search and rescue, and infrastructure inspection.
- **Medium drones:** Medium-sized drones are capable of carrying larger payloads and covering longer distances than their smaller counterparts. They are commonly used for applications such as surveillance, mapping, delivery, and agricultural spraying.
- **Large drones:** Heavy or large drones are designed for long-endurance missions and can carry payloads weighing several hundred kilograms. They are employed in tasks requiring extended flight times, such as border surveillance, maritime patrol, cargo transport, and military reconnaissance.

Each size category offers specific advantages and limitations, with selection depending on factors such as mission requirements, operational environment, and budget constraints.

- **Tactical Drones:**

Tactical drones are used for reconnaissance or battlefield surveillance missions, flying at altitudes ranging from 200 meters to 5 kilometers, with a range of 30 to 500 kilometers. Examples include the Sperwer (SAGEM) and the Aerostar (Aeronautics Defense Systems).

These drones are specifically designed for close-range operations, providing real-time situational awareness and intelligence gathering for military forces. Their ability to fly at low to medium altitudes allows them to operate effectively in complex terrain and urban environments while minimizing the risk of detection by enemy air defenses.

Tactical drones play a critical role in modern warfare, providing commanders with vital information for decision-making and enhancing the effectiveness of military operations. They are equipped with various sensors, including electro-optical, infrared, and radar systems, enabling them to perform a wide range of reconnaissance and surveillance tasks in diverse operational scenarios.

Other types of drones include Mini Air Vehicles (MAV), characterized by flight times of a few hours, dimensions on the order of meters, and altitudes of 300 metres. Additionally, there are micro drones MAV, distinguished by their compact size.

These mini and micro drones serve various purposes, ranging from reconnaissance and surveillance to research and development in fields such as robotics and aerospace engineering. Their small size and maneuverability make them ideal for navigating tight spaces and conducting missions in urban environments or other confined areas where larger drones may struggle to operate effectively. They are also utilized in applications requiring rapid deployment and agile maneuvering, such as search and rescue operations or environmental monitoring in challenging terrain.

I.4.3. According to Propulsion:

Drones can be categorized into three main families based on propulsion [3]:

• Fixed-wing drones are similar to airplanes and are efficient for long-range missions due to their speed and range. They come in two types: heavier-than-air (like airplanes) and lighter-than-air (dirigibles).

• Flapping-wing drones mimic birds and insects with their flapping wings for flight. They are good for hovering and navigating narrow spaces.

• Rotary-wing drones use rotors to fly. They are slower than fixed-wing drones but can take off and land vertically. This category is further divided into multirotor drones (most common, with multiple rotors for stability) and single-rotor drones (like helicopters, with more lift capacity and endurance). The text also mentions Bi-rotor drones which have some advantages of both single and multirotor configurations.

• Single-rotor drones offer good lift capacity and endurance for long flights and heavy payloads.

• Bi-rotor drones provide stability, maneuverability, and payload capacity through two rotors.

• Multi-rotor drones are versatile, easy to operate, and popular for various applications due to their ability to hover.

Military Applications

• Surveillance and Intelligence Gathering: Drones equipped with high-tech cameras and sensors can gather real-time information on enemy movements and terrain, helping with strategic decision making.



Figure.I. 1:Military surveillance drone

• Combat Support: Drones can provide target designation, communication relay, support for special operations, and jamming enemy signals.



Figure.I. 2:Drone dropping supplies.

• Direct Combat (Future): Armed drones are being explored for aerial combat and potentially engaging enemy aircraft. Civilian Applications



Figure.I. 3:Drone firing a missile

• Surveillance and Observation: Drones are used for scientific studies, weather monitoring, search and rescue, traffic monitoring, and infrastructure inspection.



Figure.I. 4:Drone monitoring crops

• Aerial Missions: Drones are used for delivering goods, mapping and surveying, filming stunning aerial shots, and delivering aid in disaster zones.



Figure.I. 5:Drone delivering medical supplies

• Specific Missions: Drones can be communication relays in remote areas, detect hazardous materials in spills or leaks, and aid in search and rescue efforts. Overall, drones are a versatile tool

with a wide range of applications that improve efficiency, safety, and data collection in many sectors.



Figure.I. 6:Drone searching for survivors .

I.5 Essential Components for a Quadcopter:

I.5.1. Motors:

The four motors used are SunnySky V2216-11 brushless motors, characterized by the following specifications [1]:

• Dimensions: 48 x 28mm

• Weight: 39g

• Power: 210W

• KV: 900 RPM/V

• Voltage: 10V

• Current: 0.4A

These brushless motors are specifically designed for quadcopter applications, offering a balance of power, efficiency, and reliability. With their compact size and lightweight design, they provide the necessary thrust to lift and maneuver the quadcopter efficiently. The high KV rating indicates their ability to spin at high speeds, ensuring responsive performance during flight maneuvers. Additionally, their low current draw helps optimize battery life, allowing for longer

flight durations. Overall, the SunnySky V2216-11 motors are integral components in the propulsion system of the quadcopter, contributing to its stability, agility, and overall flight performance.



Figure.I. 7:SunnySky V2216-11 brushless motor

I.5.2. Electronic Speed Controllers (ESCs) [2]**:**

The ESCs, or electronic speed controllers, are essential components responsible for regulating the speed of the motors. They must be compatible with both the motors and the battery used in the quadcopter. Each motor requires its own ESC. Here are the key characteristics of the ESC [2] :

• Maximum Speed: 210,000 RPM for 2 poles, 70,000 RPM for 6 poles, 35,000 RPM for 12 poles.

• Output Current: 40A continuous.

• Built-in Battery Eliminator Circuit (BEC): Provides a stable 3A output at 5V for powering onboard electronics.

• Dimensions : 55mm x 28mm x 12mm.

• Extremely low output resistance, ensuring efficient power delivery and minimal heat generation.

• Multiple Protections: Equipped with safeguards such as low voltage cutoff, temperature protection, and loss of signal detection to prevent damage to the ESC and motor.

• Three Startup Modes: Normal, Soft, and Super-Soft startup options for smooth and controlled motor acceleration.

These ESCs play a critical role in translating the electronic signals from the flight controller into precise control of motor speed, allowing the quadcopter to maintain stability, maneuverability, and responsiveness during flight. Their compact size, high efficiency,

and advanced features make them indispensable components in the propulsion system of the quadcopter.



Figure.I. 8:An electronic speed controller.

I.5.3 Battery:

The battery used in the quadcopter is a, with the following specifications [2]& [3]:

• Capacity: 6300mAh

• Voltage: 14.8V

• Continuous Discharge Current: 30C

• Energy: 93.2Wh

• Dimensions: 155 x 48 x 24mm

• Weight: 375g

• Main Connector: EC5 female

• Balancing Connector: JST-XH

This LiPo battery serves as the power source for the quadcopter, supplying the necessary electrical energy to the motors, flight controller, and other onboard systems. With its high capacity and voltage rating, it provides sufficient power for extended flight durations and stable performance. The continuous discharge rating of 30C indicates the maximum safe discharge rate, ensuring reliable power delivery during high-demand maneuvers. The compact dimensions and lightweight design contribute to the overall agility and maneuverability of the quadcopter. Additionally, the use of specific connector types such as EC5 female for the main connection and JST-XH for balancing facilitates easy and secure integration into the quadcopter's electrical

system. Overall, the choice of battery plays a crucial role in determining the flight endurance and performance capabilities of the quadcopter.



Figure.I. 9:4S LiPo (Lithium Polymer) battery

I.5.4 NEO-M8N Recievers:

- General Characteristics:

- Manufacturer: u-blox
- Model: NEO-M8N

- Physical Specifications:

- Dimensions: Approx. 12mm x 16mm (compact design for easy integration)
- Weight: Lightweight (weight varies depending on the specific variant, typically around 5 grams)

- Performance Characteristics:

- Number of Channels: N/A (The NEO-M8N doesn't operate on channels like a traditional receiver. It focuses on acquiring satellite data for positioning.)
- Navigation: Concurrent multi-GNSS (GPS, GLONASS, BeiDou, Galileo) for reliable signal reception in various environments.
- Positioning Accuracy: 2.0 meters CEP (Circular Error Probable)
- Update Rates: Up to 18 Hz (single GNSS), 10 Hz (concurrent GNSS) for fast data updates.
- Sensitivity: -167 dBm exceptional sensitivity for receiving weak GPS signals.
- Cold Start Time: 26 seconds for rapid acquisition of a signal upon startup.

- Communication Interfaces:

- UART (Universal Asynchronous Receiver Transmitter): Enables communication with microcontrollers and flight controllers.
- I2C (Inter-Integrated Circuit): Another communication protocol for integration with various electronic components.

Additional Features:

- Support for SBAS augmentation systems (WAAS, EGNOS, MSAS) for enhanced positioning accuracy.
- Low power consumption compared to some high-performance GNSS modules.
- Easy integration with popular development boards like Arduino and Raspberry Pi.

This radio controller offers precise and responsive control over a wide range of aircraft types, making it suitable for various flying applications. The inclusion of advanced features such as model memory, programmable mixings, and data transfer functionality enhances convenience and flexibility for users. With its ergonomic design and intuitive interface, the Neo-M8N provides an efficient and enjoyable control experience for pilots of all skill levels [11].

I.6. General architectures:

I.6.1 Drone Control Architectures:

Drones possess specific characteristics that make the design of control algorithms challenging. Particularly, rotary-wing drones exhibit underactuated systems, are sensitive to aerodynamic disturbances, and feature highly nonlinear dynamics. Moreover, they entail significant coupling between various system state variables and control inputs. To mitigate these undesirable effects, the development of control strategies for autonomous aerial vehicles has been the subject of numerous publications over recent decades. The most commonly employed control engineers in aerial robotics applications include[4] [6]:

• PID Control

• Linear Approaches

• Quadratic Linear Control

• Predictive Control

• Nonlinear Approaches

• Hierarchical Control

• Backstepping Control

• Sliding Mode Control

• Fault-Tolerant Control

• Pole Placement Control

• Gain Scheduling Control

These architectures offer diverse methodologies for addressing the complex dynamics and control challenges encountered in drone systems. Researchers and engineers explore these approaches to enhance stability, performance, and robustness in the control of autonomous aerial vehicles. Each method has its strengths and weaknesses, and the choice of architecture depends on factors such as the specific application requirements, system characteristics, and performance objectives[5]. Overall, the evolution of control architectures

Continues to advance the capabilities and reliability of drone technology for various applications.

I.6.2 Advantages of Quadrotor:

The quadrotor offers significant advantages compared to other configurations:

• Compact Size and Maneuverability: Its small size and agility enable navigation through both confined spaces and open environments while avoiding obstacles.

• Mechanical Simplicity: The mechanical simplicity of the quadrotor design facilitates maintenance tasks.

• No Clutch Required: Unlike some other rotorcraft, no clutch is necessary between the motor and the rotor, and there are no requirements regarding rotor pitch angles.

• Popular Multi-Rotor Design: The quadrotor design is the most popular among multi-rotor configurations due to its simplicity and versatility.

• Vertical Takeoff and Landing (VTOL): Quadrotors are capable of VTOL operations, making them suitable for various applications.

• Control via Motor Speed Variation: Control of the quadrotor is achieved by varying the rotation speed of its four motors.

• Enhanced Lifting Capability: The presence of four rotors provides increased lifting capacity compared to single-rotor configurations.

• Lower Dynamics than Helicopters: Quadrotors have lower dynamics compared to helicopters, requiring less rapid reaction times.

• Reduced Kinetic Energy: The use of four small rotors instead of a single large rotor significantly reduces stored kinetic energy and minimizes damage in the event of accidents.

These advantages make quadrotors highly suitable for a wide range of applications, including aerial photography, surveillance, inspection, and various research and development endeavors. Their versatility, simplicity, and maneuverability contribute to their widespread use in both commercial and research settings.

I.7. Conclusion:

In conclusion, drones have evolved from their military roots into versatile tools with a broad range of applications across various civilian sectors. Their ability to fly autonomously or be remotely controlled makes them suitable for tasks that can be dangerous, dull, or difficult for humans. The text has covered the definition, historical background, classification systems, essential components, and general control architectures of drones, specifically focusing on quadcopters. With continuous advancements in technology, drones are expected to play an even more significant role in the future, shaping various industries and improving our lives in numerous ways.

Chapter II:

facial recognition

II.1.1 Introduction:

Ensuring the security of citizens is one of the most important and complex tasks that governments around the world must accomplish. Street crimes and thefts pose the greatest threats to citizens and their property. In order to ensure security, there is an urgent need for a system that can identify a criminal in a busy area. This chapter allows us to study a facial recognition system using a local dual-pattern facial recognition system installed on drone technology. Facial recognition capability is an essential feature that a drone must find or recognize in a crowd. With the advent of drone technology in the proposed system, we can also use it as a reconnaissance drone through which a larger area can be covered compared to the fixed system. Once the system identifies the wanted person, it tags them and sends the image along with the location coordinates to the relevant authorities using the installed global positioning system (GPS). The proposed system is capable of identifying a person with an accuracy of approximately 89.1%.

Military organizations prefer facial recognition technologies over fingerprints or hand scanning. Since the introduction of artificial intelligence (AI), the facial recognition system has become a valuable tool for an application such as this. It is becoming more and more popular among researchers around the world in many applications such as medicine, engineering, security, etc.

II.1.2. Main challenges of facial recognition:

Face recognition presents a major obstacle similar to a complex visual task for the human brain. Creating a system that can mimic human capabilities of effortless object detection and recognition is a daunting task. The difficulty increases when dealing with very similar images, because the limited variation between individuals makes the task difficult. On the other hand, the wide range of facial characteristics of individuals creates an even greater challenge. Various factors contribute to this complexity.

Facial recognition technology helps focus attention from the center of the world and into different areas [10]. Proposed a facial recognition algorithm using a sparse representation classifier to detect facial features and identify individuals. Educational institutions have also adopted this technology for targeted student assessments. Karambar et al. Introduction of an automated attendance tracking system using facial recognition. The Local Binay Pattern Face Recognizer

(LBPH) is a pre-trained classifier capable of identifying faces given a dataset of interest as a reference [11-12]. Real-time applications mainly use facial recognition algorithms used by security companies and military organizations around the world [13]. Applications such as remote monitoring, especially over long distances, often require hardware platforms such as the Raspberry Pi [14-15].

This chapter delves into the complex process of facial recognition facilitated by the DIP model, deeply rooted in the principles of deep learning, revealing the process in four distinct stages. Initially, face detection is performed after real-time video capture, using histogram of oriented gradients (HOG) as the basic methodology. Next comes the facial alignment phase, which consists of extracting 68 facial landmarks. Moving to the preprocessing stage, a deep neural network is leveraged to extract 128 facial measurements for coding purposes. Finally, in the final phase, the input face is compared to the faces in the dataset, using Euclidean distance calculation to determine the closest match.

II.1.3. The structure of a facial recognition system:

The architecture of the facial recognition system includes the general generation of additional files. Initially, HOG (Histogram of Oriente Gradients) method is used to detect images. Once the face is detected, the corresponding image is saved and summarized with additional preparation to mitigate issues such as harsh lighting and facial curvature and tilt. Then, a facial feature estimation algorithm is used to identify 68 landmarks in the image registration. Through a simple conversion process, the image undergoes rotation, cropping and adjusts the image to ensure perfect alignment of the image and the center of the image registration. [34]

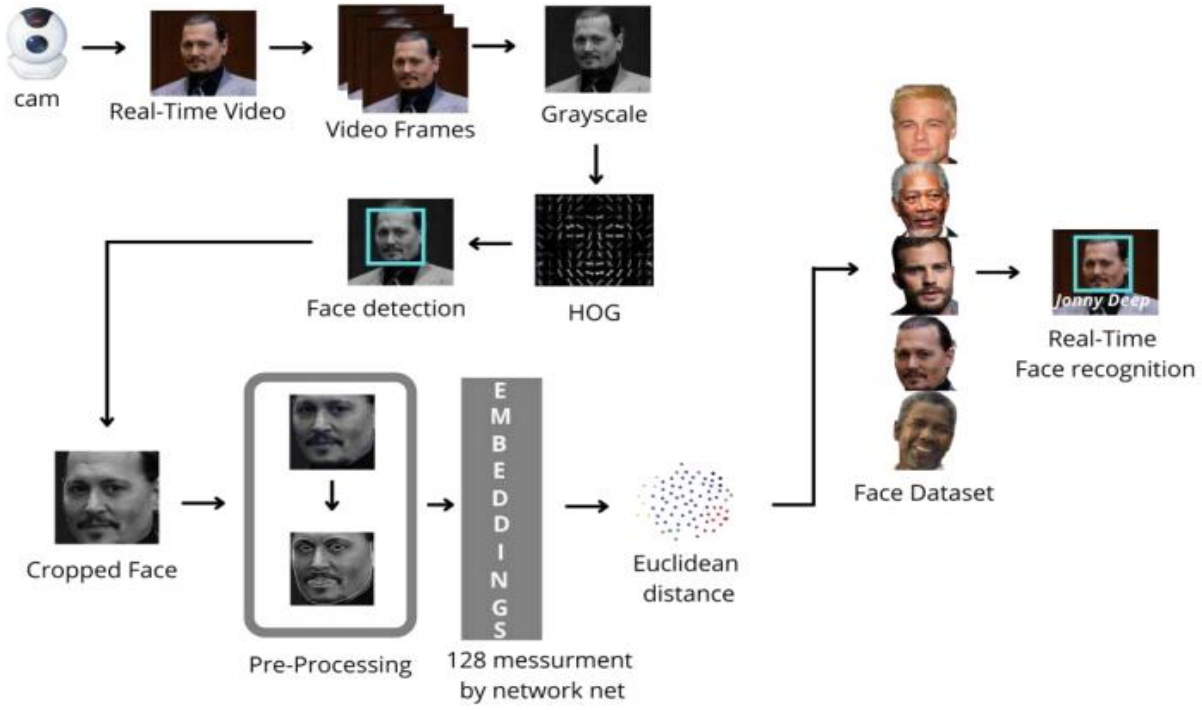


Figure. II. 1 Real-time Process Model

After feeding the preprocessed image into a trained neural network, it extracts the features and produces a set of 128 embeds, representing the facial measurements. These features are obtained by passing the preprocessed image through a containing neural network which generates a feature vector 128 embeddings. The last step is to classify the images. This is done by comparing the 128 guarantees with those in the database. This comparison includes calculating the Euclidean distance and choosing the smallest distance that corresponds to the recognized face. The complete structure and its detailed framework are displayed. [33]

Face detection by HOG - (Histogram of oriented gradients):

Face separation is the first and most important task after capturing real-time face images to remove unwanted and redundant information, such as background, from images. We will use a

method called Histogram of Oriented Gradients (HOG) to detect face images. HOG is basically a feature descriptor used for image processing and computer vision techniques.

After capturing an image of the face in live video frames, convert that image to grayscale, then HOG focuses on the structure of the object to extract the edge width information as well as the orientation of the edges. Then the face part in the input image is extracted. Finally, the image of the face is cropped [35], as shown in (Figure II.2).

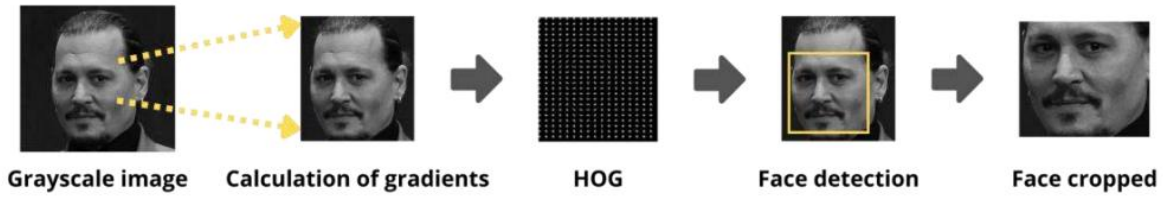


Figure. II. 2 Face Detection Using HOG

To find the faces in an image, we just need to make the image gray because we don't need color data to find the faces, then each pixel in the image is analyzed along with the pixels surrounding it, and knowing how dark the current is pixel is. Relative to the surrounding pixel directly to draw an arrow in the darkest direction. After generalizing it to the whole image, each pixel will be replaced by an arrow. These arrows are called gradients, showing the flow from light to dark, this is shown in (Figure II.3)

.

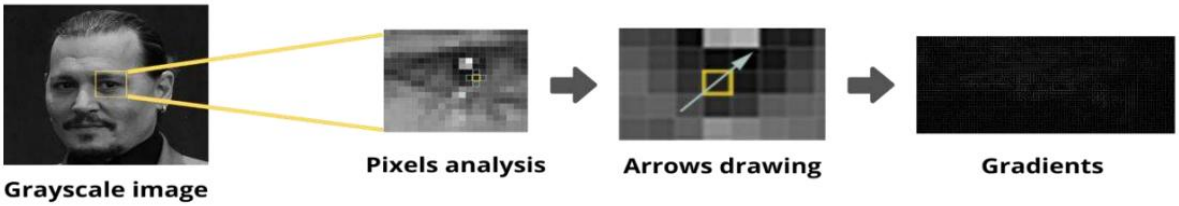


Figure. II. 3:steps to create gradients

This calculation is repeated for the entire grayscale image and we will obtain a gradient image. The next step is to calculate the strongest gradients in 16 x 16 pixel windows and replace the

gradients in that window with the strongest gradient. This will result in a very simple representation that captures the basic structure of the face in a simple way.

To locate the face in the input image captured in real-time video, we located only the part of the image that remarkably resembles a known HOG pattern and cropped that part of the input image, by consequently, the image of the face is cropped, As shown in (Figure II.4).

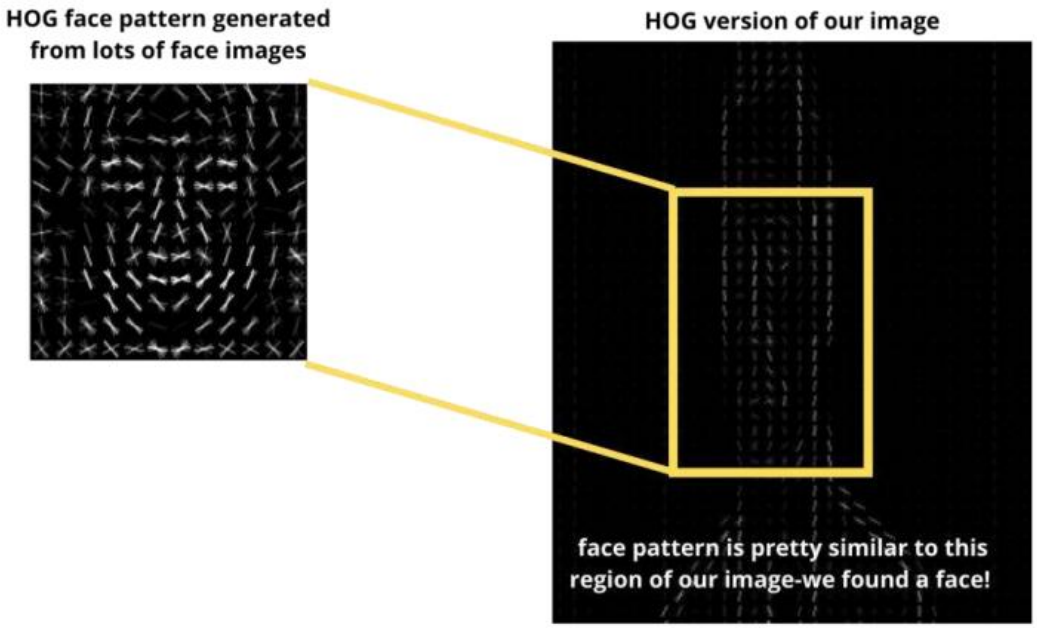


Figure. II. 4 :Extracting a new facial pattern very similar to the general facial pattern.

The process continues by posing and projecting faces, in order to deal with faces that are in different directions or angles.

There are many ways to do this, but we will use the approach pioneered by Vahid Kazemi and Josephine Sullivan in 2014, using a facial landmark estimation algorithm, a machine learning algorithm capable of detecting 68 specific human facial landmarks on an image of the face. [16], as shown in (Figure 2.5), which includes two steps:

➢ Locate the face in the image.

➢ Detect key facial structures on the QUEEN face.

The basic idea is to extract the 68 specific points (called Landmarks) located on each face, these landmarks locate the eyes, nose, chin, lips and eyebrows etc., then we will train a learning algorithm automatic to be able to find these 68 specific points on any face.

II.1.4. System hardware model:

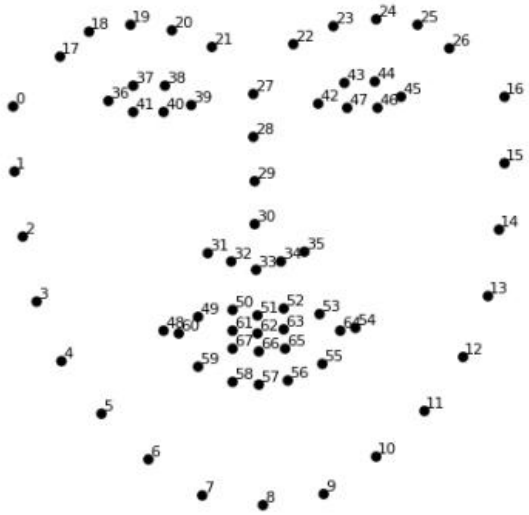


Figure. II. 5:The 68 benchmarks detected by the dlib library. This image was created by Brandon Amos from CMU who works on OpenFace.
/translate to english without any other thingr

The Figure III.6 shows the functional diagram of the system. After once the system is configured, the drone will navigate to the location provided to it in the form of coordinates. Either the RF transceiver sends these coordinates to the drone, or we transmit them to them first. We can also control the drone manually if necessary. If the drone identifies the target person, it will notify the competent authority by sending the number or photo of the identified person. Then they can control the drone to manually track the offender. The camera mounted on the drone acts as a sensory object through which it can detect or track the person. We use a Raspberry Pi module to interact with the camera. It is necessary to use a camera with a good aspect ratio and resolution to achieve better precision and results. The camera captures an image and applies Harar Cascade to detect faces in the crowd using the installed Pi module. All detected faces are cropped and the LBPH face recognition system is run on each of the detected faces and compared to the trained dataset. If the system identifies the person it is looking at, it will tag them and capture the frame. It also sends the captured frame to the concerned party for appropriate action. If the system does not detect anyone in the frame, the drone will move to the next waypoint. We can also perform

this process manually. The whole process is repeated again at the new location where the drone has just arrived. The installed camera starts capturing new information while trying to discover the person they are looking for. We offer Haar Cascade for face detection process. It can also detect facial organs such as eyes and mouth. Using a drone is the most efficient and cost-effective way this study offers. If the camera is stationary, then it will only be able to identify or identify the desired person nearby. The reason for choosing the Raspberry Pi module as an on-board computer is its lightweight and ability to perform complex operations such as image processing and machine learning. Since the proposed model uses a face recognition machine learning (LBPH) algorithm, it is also an AI-based application

II.2. Feature transformation based on keypoints:



Figure. II. 6:Figure. II.6. Quad-copter model used for the proposed system

II.2.1. Scale invariant feature transform (SIFT):

SIFT (Scale-Invariant Feature Transform) is an algorithm used to detect and describe local features of an image. This algorithm is widely used to link two images by their local descriptors, which contain information allowing a correspondence between them. The main idea of the SIFT writer is to transform the image into a representation composed of points of interest. The algorithm is carried out in four steps:

1. Detection of maximum and minimum points on the spatial scale.

2. Location of characteristic points.

3. Allocation of orientation.

4. descriptor of the characteristic point.

II.2.2. Approach:

They rely on well-known statistical analysis techniques. It is not necessary to mark certain characteristic points of the face (such as the center of the eyes, the nostrils, the center of the mouth, etc.) except to standardize the mages. In these methods, images of faces (which can be seen as arrays of pixel values) are processed.
Overall and are generally transformed into vectors, easier to manipulate. The main advantage of global methods is that they are relatively quick to implement and the basic calculations are of medium complexity. On the other hand, they are very sensitive to variations in lighting, pose and facial expression. This is easily understood since the slightest variation in the conditions of the ambient environment leads to inevitable changes in the values of the pixels which are directly processed. These methods mainly use face subspace analysis. The use of subspace modeling techniques has significantly advanced facial recognition technology.

II.2.3. Linear techniques

The most popular linear techniques used for facial recognition systems are the Eigenfaces (Principal Component Analysais - PCA) technique, the Fisher faces (Liner Discriminative Analysais - LDA) technique and the Independent Component Analysais (ICA).

II.2.4. Linear discriminant analysis:

Linear discriminant analysis or normal discriminant analysis or discriminant function analysis is a commonly used dimensionality reduction technique for supervised classification problems. It is used to model differences between groups, that is, to separate two or more classes. LDA mainly focuses on projecting features in higher dimensional space to lower dimensions. This can be achieved in three steps.

1. First, calculate the separation between classes (the distance between the mean of different classes). This is called interclass variance.

$$S_b = \sum_{i=1} N_i(\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})$$

2. Second, calculate the distance between the mean and the sample of each class. It is also called interclass variance.

$$S_w = \sum_{i=1}(N_i - 1)S = \sum_{i=1}^{g}\sum_{j=1}^{i}(x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)$$

3. Finally, the construction of the lower dimensional space (maximizes the variance between classes and minimizes the variance within the class). P is considered the lower dimensional spatial projection, also called Fisher's criterion.

$$P_{LDA} = arg\,\max_P \frac{|P^T S_b P|}{|P^T S_w P|}$$

The system will work by having a drone broadcast the video feed from its camera to a computer. This computer will run facial detection and recognition algorithms to perform a real-time analysis of each frame of the video stream in search of a specific singular individual. If this individual is found, the tracking part of the system will activate and the computer will send movement

instructions to the drone, designed to keep the specified individual in frame. All of these small systems working together will result in an autonomous drone capable of finding and tracking a specified individual.

The use of drones has become more common in recent years as they have become more widely available. These days, people are using drone technology in ways that were previously unimaginable; Like delivering packages, planting trees, and even quickly deploying medical equipment to those in need. A drone is made up of 11 basic components including: chassis, motors, propellers, electric motor controller (ESC), power distribution board (PDB), flight controller, battery, receiver, the camera, the video transmitter (VTX) and the sensors . And sensors. Drones also have four degrees of freedom. These four degrees are composed of three translations and one rotation. All three translations allow drones to move up and down, left and right, and back and forth. The unique rotation allows the drones to rotate clockwise and counterclockwise. When working with the drone, parameters such as height, distance and depression angle are kept in mind.

Ryze Tech Tello drone is a small, lightweight drone weighing only 80 grams but offering a flight time of 13 minutes, a flight distance of 100 meters and a 5 megapixel camera capable of transmitting HD videos in 720P. The djitellopy python library was also used for this project, which makes it easy to send commands and receive information from the drone directly to a computer. Thanks to the connect function, a connection could be established between the drone and a computer.

The Stream on function then transmits the video captured from the drone directly to a computer allowing the image to be examined for facial recognition purposes. Even if a live video is broadcast, it is broken down frame by frame for review by the facial recognition system. The system is also capable of displaying the feed on the computer and providing additional information if people are identified.

djitellopy library also contained many different functions designed to control the drone's movements. The user can specify a person as a target in the system and when a specified target is found, the drone will follow it. As images are reviewed and people are specified, a box is drawn around each face, displaying the person the system has identified them as.

If one of these identified people is the target, the system uses the size of the box drawn around them to issue movement commands. If the box area decreases or is already small, the drone will move closer until it reaches a specified range. This works the same for targets if the area is too big or the target is getting closer. Similar motion controls are used to rotate the drone if the target moves from side to side, ensuring it can be tracked.

However, optimally, all tracking should be done with the front side of the target visible. Side profile images lead to reduced accuracy and images from the back of the head lead to no accuracy at all.

II.2.5. Recognition systems:

II.2.6. Rowley neural network:

Rowley's algorithm, based on a neural network, aims to detect grayscale upright faces in images by using multiple neural networks to analyze different parts of the image. Each network is trained to recognize the presence or absence of a face. The method relies on the fact that faces can be characterized by pixel intensities in images, but training a neural network for this task is difficult because it is complicated to characterize images that are not faces. Unlike facial recognition, which distinguishes between different faces. [17]

Face detection needs to differentiate images with faces from those without faces, which makes it difficult to obtain a representative sample of faceless images for training. Rowley, Baluja, and Kanada proposed a solution to this problem using a selective sampling technique for non-facial images.

II.2.7. Algorithme of rowley:

First step:

The first neural network-based component of our system is a filter that receives as input a 20x20 pixel region of the image, and generates an output ranging from 1 to -1, signifying the presence or

absence of a face, respectively. To detect faces anywhere in the input, the filter is applied at each location in the image. To detect faces larger than the window size, the input image is repeatedly scaled down (by downsampling), and the filter is applied to each size. This filter must have a certain position and scale invariance. The amount of invariance determines the number of scales and positions at which it should be applied.

Second step:

Merging overlapping detection and adjudication. The examples in

Figure II.7 show that the raw output of a single network will contain a number of false detections. In this section, we present two strategies to improve detector reliability: fusion of overlapping

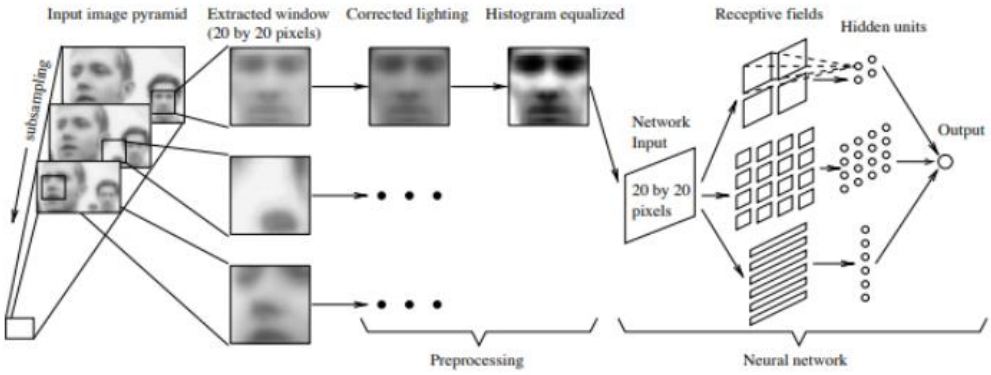


Figure. II. 7:Rowley neural network

detections from a single network and arbitration between multiple networks [18]

II.3. Hardware and software tools:

II.3.1. Material:

Processor: Intel Core i5 5300U CPU 2.30GHz processor -2 core(s) -4 logical processor(s).

RAM: 4GB DDR3.

GPU: Intel HD Graphics 5500.

Operating system: Linux: “Kali Linux”.

language: Python.

II.3.2. Modules:

Open CV (open source computer vision):

Open CV (Open Source Computer Vision Library) is a library of programming functions primarily intended for real-time computer vision. Initially developed by Intel, it was later supported by Willow Garage and then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the Apache 2 open source license. As of 2011, Open CV offers GPU acceleration for real-time operations.

It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. The library contains over 2,500 enhanced algorithms, which incorporate an extensive set of excellent machine learning algorithms. These algorithms can be used to distinguish and perceive faces, recognize objects, group human activities in recordings, track camera developments, track moving items, extract 3D models of objects.

The library is widely used in organizations, research groups and administrative bodies.

Installation command: $ pip Install open cv-python

Dlib:

Dlib is an Open Source C++ toolkit. It contains various algorithms and machine learning tools to create complex software. Dlib was solving real-world problems. It is useful in industry and academia, especially in robotics, embedded devices, mobile phones, and large high-performance computing environments [19].

Our Dlib network architecture for face recognition is inspired by ResNet-34 from the Deep Residual Learning for Image Recognition paper by He et al., but with fewer layers and the number of filters halved.

Installation command: $ pip Install dlib

Installation command: $ pip Install cmake

➢ cmake because dlib was developed in a C-based programming language.

Facial Recognition Library:

Face recognition module is used to recognize and manipulate faces from Python or from the command line of the world's simplest face recognition module.

Built using dlib's cutting-edge, deep learning-based facial recognition. This provides a simple facial recognition command line tool that allows you to perform facial recognition on a folder of images from the command line [20].

Installation command: $ pip Install face-recognition

The system works by having a drone broadcast a video feed from its camera to a computer. This computer will run facial detection and recognition algorithms to perform a real-time analysis of each frame of the video feed in search of a specific individual. If this person is found,

The tracking part of the system will activate and the computer will send movement instructions to the drone, designed to keep the specific person in frame.

All of these small systems working together will result in an autonomous drone capable of finding and tracking a specified individual.

djitellopy library also contains many different functions designed to control the movement of drones, and these functions were used to implement the tracked part. The user can select someone as a target in the system, and when a specific target is found, the drone will follow it. As images are scanned and people are identified, a box is drawn around each face, displaying the people recognized by the system. If one of these identified people is the target, the system uses the size of the frame drawn around them to issue movement commands. If the box space decreases or is already small, the drone will approach until reaching a specified range.

II.4. Preconditions:

II.4.1. Installation requirements:

Make sure you have the following libraries installed in the Python environment:

1. Open CV

2. dlib

3. Jumpy

4. Mutilated

5. Pillow

II.4.2. Configure faces to recognise:

Known_faces guide is updated with images of the individuals we seek to identify, taking care to frame the face correctly. Please note that only PNG or JPG formats are accepted.

```
/known_faces/Zuckerberg.png
/known_faces/YourPicture.jpg
```

Figure. II. 8:PNG or JPG formats are accepted.

Note that the recognition name displayed is taken from the file name (without extension) it corresponds to in the known_faces folder.

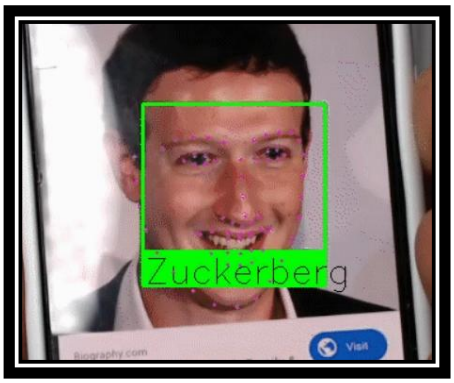


Figure. II. 9:detect faces like Zuckerberg's example

II.4.3. Camera:

You need a camera connected to your PC since the program will broadcast the camera image on your screen and recognize the displayed face if the face is part of the folder



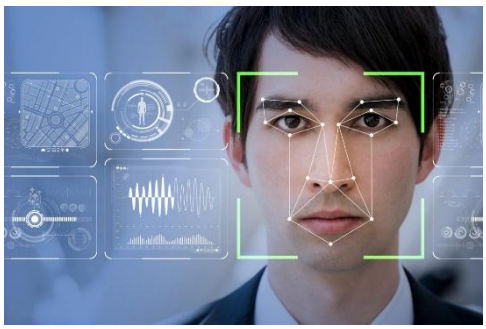


Figure. II. 10: Facial recognition system.

II.5. Create a facial recognition system in pythen:

II.5.1. Stabilization:

To run this program, the first thing we must do is download and install

VisualStudio. You might be wondering what Visual Studio has to do with all this. Believe me, I did it

Also. But because of the functionality it provides through (dlib).

Second, we install the latest version of Python.

Now we install the required packages via command (pip) in CMD command prompt

by typing the following command:



cmake:

CMake is a build system and project configuration tool.

dlib ==19.18.0:

(Dlib) is a C++ library for machine learning and computer vision. Please

Make sure you install the specified version.

face-recognition:

is a Python module for facial recognition? This is the module that does the magic for us. Feel free to check out the documentation here.

OpenCV-Python:

(OpenCV) is a computer vision library and (OpenCV-Python) refers to its Python bindings

Used for image and video processing.

II.5.2. Create a folder and save photos of known people:

We can create a folder and save photos of known people. People that we want to know through live streaming. The images in this folder are those on which our software to train for . Then it will be able to recognize anyone from that folder in the live stream. Photos must be recorded with the names of the people.



II.5.3. Building a line facial recognition system:

And now let's get to the exciting part. Coding! We will start by importing the libraries:

```
import tkinter as tk, numpy as np, cv2, os, face_recognition
from datetime import datetime

import json
from urllib.request import urlopen
```

Afterwards:

```
# Initialize empty lists to store images and people's names.
known_faces = []
face_labels = []

# Get a list of all images in the TrainingImages directory.
image_files = os.listdir("persons")

# Loop through the images in the directory.
for image_name in image_files:
    # Read each image and add it to the known_faces list.
    current_image = cv2.imread(f'persons/{image_name}')
    known_faces.append(current_image)

    # Extract the person's name by removing the file extension and add it to the face_labels list.
    face_labels.append(os.path.splitext(image_name)[0])
```

What we're doing here is going through all the images in our folder (people) and adding them to a list for further processing (encoding and comparison). We also extract the names of known people by removing the extension. So from Miloud.jpg we remove Miloud and display it as the person's name if it is recognized.

Next, we create a function to get the encodings of the images

```
# Function to get face encodings from a list of images.
def get_face_encodings(images):
    encoding_list = []
    for image in images:
        # Convert the image to RGB format. RGB is Red Green Blue.
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        # Get the face encoding for the first face found in the image.
        face_encoding = face_recognition.face_encodings(image)[0]
        encoding_list.append(face_encoding)
    return encoding_list

```

Next, we create a function that does the documentation:

```
# Define a function to document the recognized face.
def document_recognised_face(name, filename='records.csv'):
    # Get the current date in the YYYY-MM-DD format et gps.
    capture_date = datetime.now().strftime("%Y-%m-%d")
    url='http://ipinfo.io/json'
    response=urlopen(url)

    # Check if the specified CSV file exists.
    if not os.path.isfile(filename):
        # If the file doesn't exist, create it and write the header.
        with open(filename, 'w') as f:
            f.write('Name,Date,Time,gps');# Create the file and write the header.


    # Open the CSV file for reading and writing ('r+')
    with open(filename, 'r+') as file:
        # Read all lines from the file into a list.
        lines = file.readlines()

        # Extract the names from existing lines in the CSV.
        existing_names = [line.split(",")[0] for line in lines]

        # Check if the provided name is not already in the existing names.
        if name not in existing_names:
            # Get the current time in the HH:MM:SS format.
            now = datetime.now()
            current_time = now.strftime("%H:%M:%S")
            gps = json.load(response)

            # Write the new entry to the CSV file including name, capture date, and time.
            file.write(f'\n{name},{capture_date},{current_time},{gps}')
```

The above function documents the person's name, date/time and location of identification and saves the information to a Records.csv file.

Then we get the known image encodings and create a function to start the program. This is necessary because this is a GUI-based program, and we need to pass this functionality as a button action:

```
# Get face encodings for known images.
known_face_encodings = get_face_encodings(known_faces)


# Function to start the Facial recognition program.
def start_recognition_program ( ) :
    # Open a webcam for capturing video. If you are using your computer's webcam,
change 1 to 0.
    # If using an external webcam, leave it as 1.
video_capture = cv2.VideoCapture( 0 )
```

```
    while True :
        # Read a frame from the webcam.
frame = video_capture.read ( 0 )

        # Check if the frame is not None (indicating a successful frame capture).
        if frame is not None :
frame = frame[ 1 ] # The frame is usually the second element of the tuple
returned by video_capture.read().

            # Resize the image to a smaller size.
resized_frame = cv2.resize(frame, ( 0 , 0 ), None , 0.25 , 0.25 )
resized_frame = cv2.cvtColor(resized_frame, cv2.COLOR_BGR2RGB)

            # Detect faces in the current frame.
face_locations = face_recognition.face_locations (resized_frame )

            # Get face encodings for the faces detected in the current frame.
current_face_encodings = face_ recognition.face _encodings(resized_frame,
face_locations)

            # Loop through the detected faces in the current frame.
            for face_encoding, location in zip ( current_face_encodings,
face_locations):
                # Compare the current face encoding with the known encodings.
matches = face_ recognition.compare _faces(known_face_encodings, face_encoding)
face_distances = face_ recognition.face _distance(known_face_encodings,
face_encoding)

              # Find the index of the best match. That is, the best resemblance.
best_match_index = np.argmin (face_distances)

                if matches[best_match_index]:
                    # If a match is found, get the name of the recognized person.
recognized_name = face_labels[best_match_index ].upper ()

                    # Extract face location coordinates.
top, right, bottom, left = location
top, right, bottom, left = top * 4 , right * 4 , bottom * 4 , left * 4

     # Draw a rectangle around the recognized face.
cv2.rectangle(frame, (left, top), (right, bottom), ( 0 , 255 , 0 ), 2 )
# Draw a filled rectangle and display the name above the face.
cv2.rectangle(frame, (left, bottom - 35 ), (right, bottom), ( 0 , 255 , 0 ),
cv2.FILLED)
cv2.putText(frame, recognized_name, (left + 6 , bottom - 6 ),
cv2.FONT_HERSHEY_COMPLEX, 1 ,
```

```
( 255 , 255 , 255 ), 2 )
document_recognized_face(recognized_name)
   if not matches[best_match_index]:

     # Extract face location coordinates.
top, right, bottom, left = location
top, right, bottom, left = top * 4 , right * 4 , bottom * 4 , left * 4

         # Draw a rectangle around the recognized face.
cv2.rectangle(frame, (left, top), (right, bottom), ( 0 , 0 , 255 ), 2 )

     # Draw a filled rectangle and display the name above the face.
cv2.rectangle(frame, (left, bottom - 35 ), (right, bottom), ( 0 , 0 , 255 ),
cv2.FILLED)
cv2.putText(frame, "unknown" , (left + 6 , bottom - 6 ),
cv2.FONT_HERSHEY_COMPLEX, 1 ,
( 255 , 255 , 255 ), 2 )

 # Display the image with recognized faces.
cv2.imshow( "Webcam" , frame)

   # Check for key press
key = cv2.waitKey( 1 ) & 0x FF

 # Check if the 'q' key is pressed to exit the program.
if key == ord ( 'q' ):
   break

    # Release the video capture and close all OpenCV windows.
video_capture.release ( )
cv2.destroyAllWindows()


# Create the main application window.
root = tk.Tk ()
root.title ( "Face Recognition Program" )

# Create a label
label = tk.Label (root, text = "Click the button to start the facial recognition
program" )
label.pack ( pady = 10 )

# Create a button to start the program
start_button = tk.Button (root, text = "Start Recognition" , command =
start_recognition_program)
start_button.pack ( pady = 10 )
```

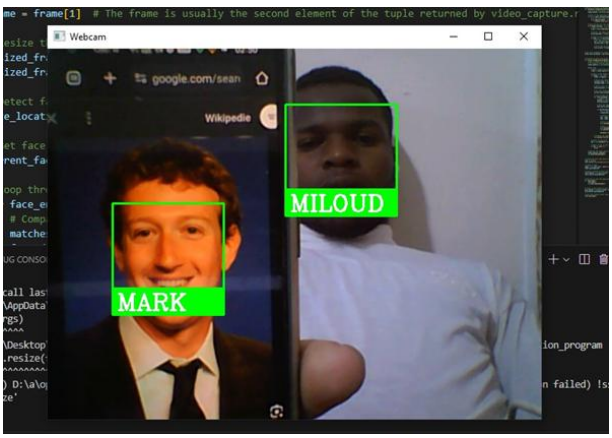
```
# Function to exit the application. This is for leaving the entire program. To
leave the webcam stream, hit q.
def quit_app ( ) :
    root.quit ()
cv2.destroyAllWindows()

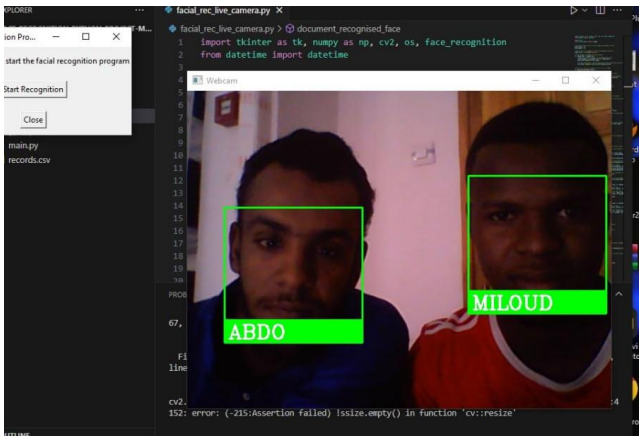

# Create a quit button to exit the application.
exit_button = tk.Button (root, text = "Close" , command = quit_app)
exit_button.pack ( pady = 10 )

# Start the Tkinter event loop.
root.mainloop ()
```

And there you go! We just built a facial recognition system that documents activity!

Here is a demo: The csv file is created automatically:



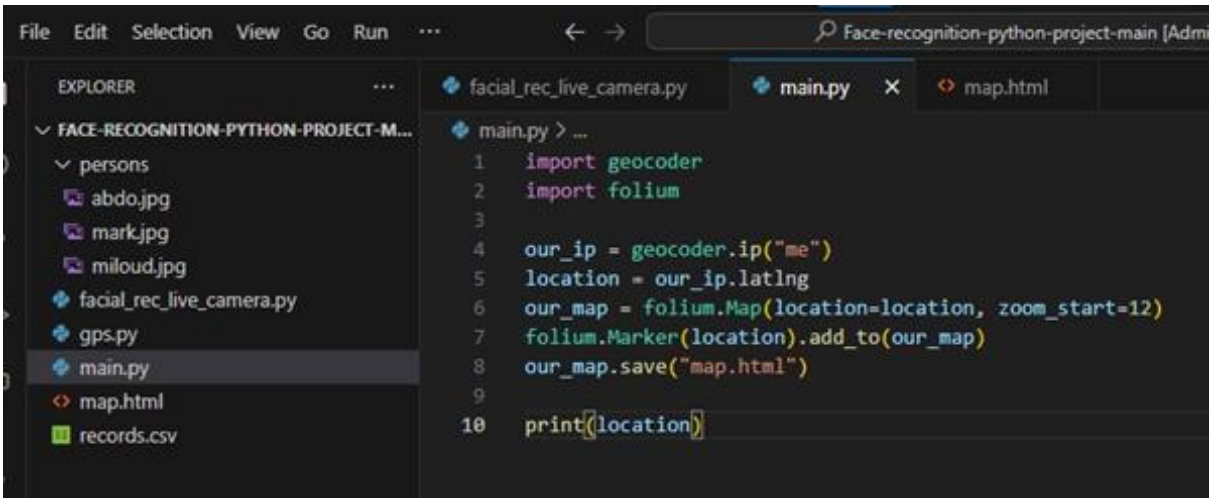


Using facial recognition and IP address extraction technology, we can develop an integrated system combining these two technologies. This system includes additional integrated software

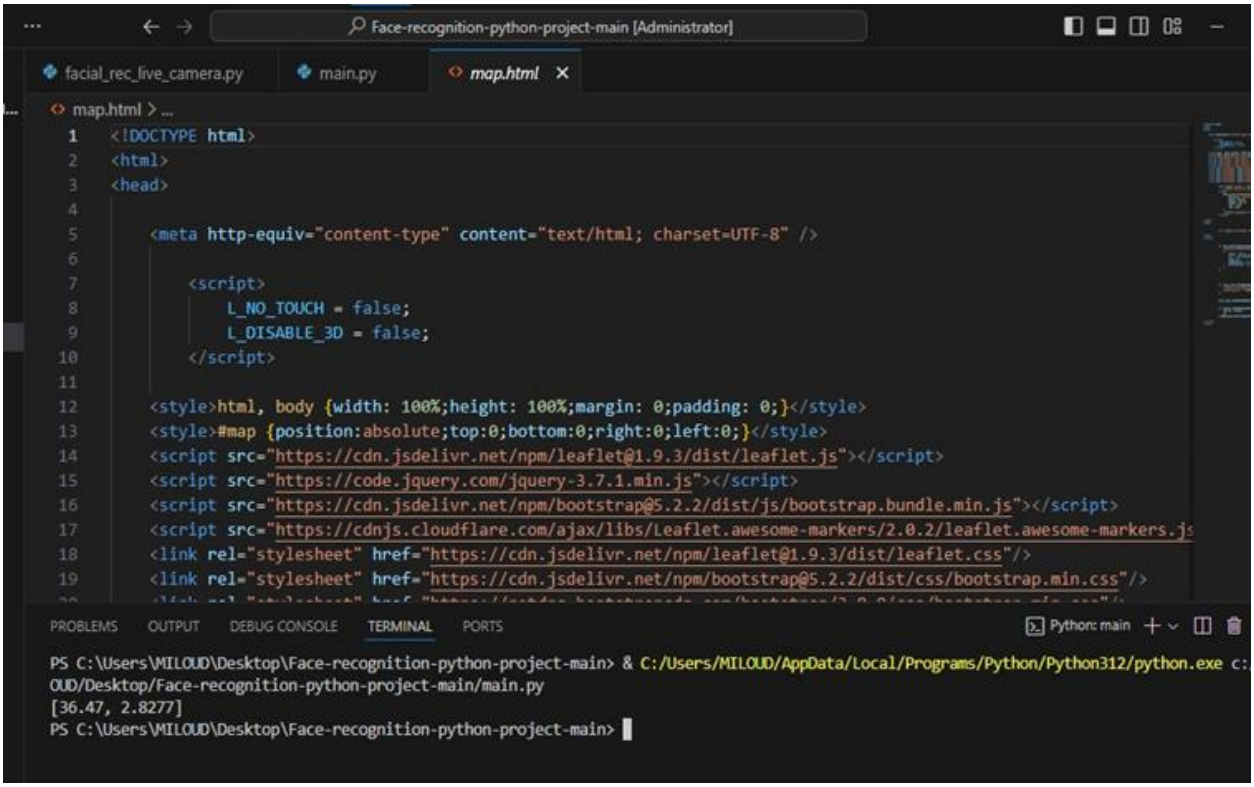


With facial recognition software, which extracts the actual geographic location of detected IP addresses and displays them on an HTML map.

Since I'm creating a simulated program on the computer, I'm going to extract my site:

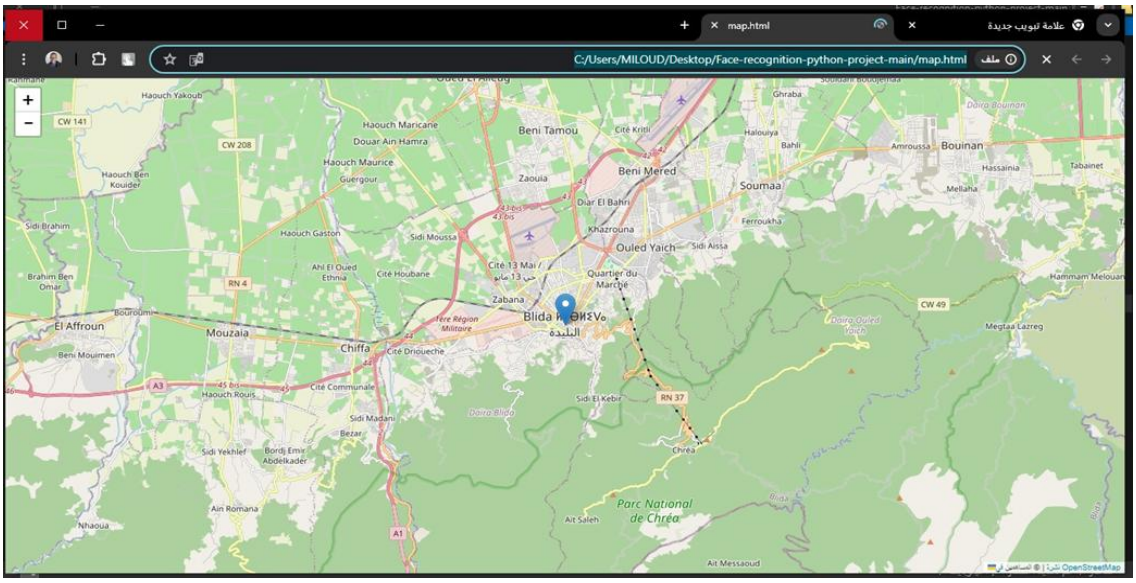


We save the location to the HML:



When we open the file we can see the exact location:

This capability allows for the location of a person with high accuracy, making it a valuable tool for law enforcement teams. For example, this system could be programmed to operate on a police drone, sending the exact location of a wanted person to the nearest police patrol to ensure their arrest.



II.5.4. Python code to make a drone scan a specific area with facial recognitioin and direct the drone towards the recognised person:

Note: This code may not work. This code applies to advanced software and

Technologies that tell you facial recognition and drone piloting. We need a drone to extract this code Requirements:

• API compliant drone

• A library of facial recognition software (like OpenCV or Dlib)

• A library of software for flying drones (like DroneKit or pymavlink)

• A camera attached to the drone.

Not:

Import libraries:

```
new.py
1   import cv2
2   import time
3   from dronekit import connect, VehicleMode, LocationGlobalRelative
4
```

```
8
9   video_capture = cv2.VideoCapture(1)
10
```

Establish a connection with the drone:

Configuring the drone camera:

Replace “1” with the drone camera ID

```
5
6   vehicle = connect('COM1', baudrate=57600)
7
```

Facial recognition function:

```
def detect_face(frame):

    faces = face_detector.detectMultiScale(frame)


    if len(faces) > 0:
        for (x, y, w, h) in faces:
            # حدد موقع الوجه
            face_center = (int(x + w / 2), int(y + h / 2))
            return face_center
    else:
        return None

```

Function to move the drone:

```
def move_drone(face_center):

    if face_center is not None:

        x_offset = face_center[0] - image_width / 2
        y_offset = image_height - face_center[1]

        speed = min(abs(x_offset) / 100, abs(y_offset) / 100)

        if x_offset > 0:
            vehicle.set_attitude(target_roll=0, target_pitch=0, target_yaw=-speed)
        elif x_offset < 0:
            vehicle.set_attitude(target_roll=0, target_pitch=0, target_yaw=speed)
        if y_offset > 0:
            vehicle.set_velocity(vx=0, vy=0, vz=speed)
        elif y_offset < 0:
            vehicle.set_velocity(vx=0, vy=0, vz=-speed)
    else:

        vehicle.set_attitude(target_roll=0, target_pitch=0, target_yaw=0)
        vehicle.set_velocity(vx=0, vy=0, vz=0)

```

Main program:

Defines the search area

Determines flight altitude

The drone has started

```
search_area = [
    (latitude1, longitude1),
    (latitude2, longitude2),
    (latitude3, longitude3),
    (latitude4, longitude4),
]


flight_altitude = 10


vehicle.mode = VehicleMode("GUIDED")
vehicle.armed = True
vehicle.takeoff(flight_altitude)

while True:

    ret, frame = video_capture.read()

    gray
```

II.5.5. Geographic coordinates:

Geographic coordinates for locations are always given as latitude and longitude. The latitude and longitude are associated with a specific earth figure, which may be a sphere for most atlas maps and imprecise work, but for rigorous purposes is a mathematical figure that much more nearly approximates the form of the actual earth. The earth's form is supposed to be an ellipsoid of revolution (a figure defined by an ellipse rotated about its minor axis), sometimes known as a spheroid (although not at the epicenter). Several ellipsoids are used to map various nations, each characterized by its semi-major axis (6378.137 km) and semi-minor axis.

Each of these ellipsoids generally corresponds to one or more countries. An ellipsoid appropriate for one region of the Earth is not always appropriate for other regions [22]. Another property is needed to uniquely specify geographic positions. This is the position and orientation of the ellipsoid relative to the earth. The term used to describe this adjustment of an ellipsoid to the earth is geodetic datum. Numerous geodetic data exist throughout the world, each generally associated with the national survey of a particular country or continent. More recently, several new geodetic

data have been successfully derived, from accumulating satellite and other data, to provide the global best fit [21].

All points and precisely features studied and mapped on the Earth's surface will be uniquely defined in position by indicating their coordinates (grid or geographic), the projection if it is a grid, and the geodetic data, including its terrestrial ellipsoid. Therefore, to be unambiguous and accurate in defining the position, it is essential that in addition to citing latitude, longitude and height, it is also necessary to specify the geodetic data, notably ellipsoid and the level at which the value of the height coordinate is linked [22].

II.6. Mathematical model and deriviation of geographic coordinate of the target:

II.6.1. Cartesian coordinates:

The earth being assumed to be rigid, the frame of reference must be fixed at its center. The Z axis is taken parallel to the average axis of rotation. The These axes are parallel to those classically adopted for reference geodesic spheroids. The origin can be taken at the center of any convenient spheroid reference on which one of the astrotriangulation stations can be located. [21].

II.6.2. Gps technical overview:

The technical and operational characteristics of GPS are organized into three segments:

1. The space segment.

2. The operational control segment.

3. The user equipment segment.

GPS signals, which are broadcast by each satellite and carry data to both user equipment and ground control facilities, connect the segments together into a single system. [25].

II.6.3. Gps reciever technology:

GPS technology is based on accurate time and location information. To calculate the user's position on Earth, a GPS receiver receives signals from three or more satellites at the same time [23].

GPS receivers cannot operate in buildings that block high-frequency satellite signals. Before turning on the GPS receiver, the user must first choose a suitable location to stand, preferably on a flat surface with a clear horizon [26].

Then it acquires the satellite signals and displays the signal strength to indicate a successful connection. When signals from three satellites are received, the system can perform a preliminary calculation of the geographic position and provide information [24].

II.6.4. World geodetic system (WGS-84):

The standard physical model of the Earth used for GPS applications is the World Geodetic System-1984 (WGS- 84) [27].

Part of WGS-84 is a detailed model of Earth's gravitational irregularities.

Such information is necessary to obtain accurate satellite ephemeris information; However, a GPS receiver's estimation of latitude, longitude, and height is affected. To this end, WGS-84 provides an ellipsoidal model of the Earth's shape,

In this model, the cross sections of the Earth parallel to the equator plane are circular. The equatorial cross-section of the Earth has a radius of (6378.137 km), which is the mean equatorial radius of the Earth. In the WGS-84 Earth model, the cross sections of the Earth normal to the equatorial plane are ellipsoidal.

$$e = \sqrt{\frac{\left(a^2 - b^2\right)}{a^2}} \quad \text{---------- (1)}$$

In an ellipsoidal section containing the Z axis, the major axis coincides with the equatorial diameter of the Earth. The semi-major axis (a) therefore has the same value as the average equatorial radius given above. The minor axis of the ellipsoidal section. corresponds to the polar diameter of the earth; the semi-minor axis (b) in WGS-84 is considered to be (6356.752 km.) Thus, the eccentricity of the earth's ellipsoid (e) is: [28]

II.6.5. Formula for transforming the geographic coordinate system:

Latitude (φ) is the angle measured between the equatorial plane and the normal line of the ellipsoid and longitude (λ) is the angle measured from the Greenwich meridian [29].

Latitude (φ) and longitude (λ), in terms of a geographic coordinate system, can be expressed in terms of a geocentric (Earth-centered) Cartesian coordinate system (X, Y, Z) with the axis Z

corresponding to the positive polar axis. Towards the north, the x axis passing through the intersection of the Greenwich meridian and the equator.

$$X = (N + h).\cos\varphi.\cos\lambda \text{.....(2)}$$
$$Y = (N + h).\cos\varphi.\sin\lambda \text{.....(3)}$$
$$Z = \left(\frac{b^2}{a^2}.N + h\right).\sin\varphi \text{.....(4)}$$

This requires that the equivalent geographic coordinate system be based on the Greenwich Meridian. Using the formulas below, the geographic coordinate system based on a non-Greenwich meridian must first be transformed into its Greenwich equivalent. If the semi-major axis of the Earth's spheroid is (a) equal to (6,378.137 km) and the semi-minor axis is (b) equal to (6,356.852 km), the equations of the Cartesian coordinate system (X, Y, Z)) are [30].

Where N is the first vertical radius of curve at latitude φ and is equal to:

φ and λ are respectively the latitude and longitude (linked to the original meridian) of the point, h is the height above the ellipsoid [31].

II.6.6. Methodology:

The GPS receiver was installed on the WGS-84 and the geographic coordinates (latitude,

$$N = \frac{a}{\sqrt{(1 - e^2.\sin^2\varphi)}} \text{ ---------- (5)}$$

longitude and height) of the site were determined also the units of the coordinates (deg., meter) were determined. The following steps are then carried out:

1. After selecting the appropriate observer position based on the unknown location.

Four satellites were considered to determine the geographic coordinates of the observer position.

2. An approximate value of the distances between the observer's position and the unknown site's position (i.e. distance (R)) was measured using the laser range finder.

3. Using equation (2, 3 and 4), the geographical coordinates (φi, λi, hi) of the observer were transformed into Cartesian coordinates (Xi, Yi, Zi), where (i=1, 2, 3, 4…).

4. Derivation of mathematical equations to calculate the Cartesian coordinates of the unknown position in three dimensions (X a k. Y unk ., Z unk .).

5. These Cartesian coordinates of the unknown position transform into coordinated geographic coordinates using equations (6 and 7):[37].

$$\varphi_{unk.} = \tan^{-1}\left(\frac{Z_{unk.} + N_i.e^2.\sin\varphi_1}{\sqrt{X^2_{unk.} + Y^2_{unk.}}}\right) \text{ --------- (6)}$$

$$\lambda_{unk.} = \tan^{-1}\left(\frac{Y_{unk.}}{X_{unk.}}\right) \text{ ---------- (7)}$$

II.6.7. Results and discussions:

The importance of this research is to solve the problem of determining the unknown position coordinates of a point or site at long distances. To calculate these equations to find the unknown position, four observer measurement positions away from the unknown position using the GPS receiver were required. Then, the distances between the observer's positions and the unknown position are measured using the laser range finder (LRF).

Through this derivation, the final form of the three equations needed to determine the Cartesian coordinates of the unknown position was found. The derivation of athematic equations to measure the Cartesian coordinates of the unknown position in three dimensions (X unk, Y unk .,Z unk .) located at a distance from the observer's position. Using four observer positions (Xi, Yi, Zi) and the range (Ri) between the observer and the unknown positions, the resulting equations were the foundations: The X axis of Cartesian coordinates:

Or:

$$X_{unk.} = \frac{D_3 D_4 - D_1 D_6}{D_2 D_6 - D_3 D_5}, \text{ ....(8)}$$

$$D_1 = \frac{C_1}{(Z_2 - Z_1)} - \frac{C_2}{(Z_3 - Z_2)}$$

$$C_1 = \frac{A_1 - A_2 + B_2 - B_1}{2},$$

$$A_1 = R_1^2, A_2 = R_2^2,$$

$$B_1 = X_1^2 + Y_1^2 + Z_1^2, B_2 = X_2^2 + Y_2^2 + Z_2^2, \text{ and}$$

$$C_2 = \frac{A_2 - A_3 + B_3 - B_2}{2}, A_3 = R_3^2, B_3 = X_3^2 + Y_3^2 + Z_3^2,$$

$$D_2 = \frac{X_1 - X_2}{Z_2 - Z_1} - \frac{X_2 - X_3}{Z_3 - Z_2},$$

$$D_3 = \frac{Y_1 - Y_2}{Z_2 - Z_1} - \frac{Y_2 - Y_3}{Z_3 - Z_2},$$

$$D_4 = \frac{C_2}{Z_3 - Z_2} - \frac{C_3}{Z_4 - Z_3}, C_3 = \frac{A_3 - A_4 + B_4 - B_3}{2},$$

$$A_4 = R_4^2,$$

$$B_4 = X_4^2 + Y_4^2 + Z_4^2,$$

$$D_5 = \frac{X_2 - X_3}{Z_3 - Z_2} - \frac{X_3 - X_4}{Z_4 - Z_3},$$

$$D_6 = \frac{Y_2 - Y_3}{Z_3 - Z_2} - \frac{Y_3 - Y_4}{Z_4 - Z_3},$$

The Y axis of Cartesian coordinates:

$$Y_{unk.} = -\frac{1}{D_6}[D_4 + D_5 \, X_{unk.}] \text{.....(9)}$$

The Z axis of Cartesian coordinates:

$$Z_{unk.} = \frac{1}{(Z_2 - Z_1)}\left[C_1 + (X_1 - X_2)\left(\frac{D_3D_4 - D_1D_6}{D_2D_6 - D_3D_5}\right) + \frac{D_2}{D_3}(Y_1 - Y_2)\left(\frac{D_1D_6 - D_3D_4}{D_2D_6 - D_3D_5}\right) - \frac{D_1}{D_3}(Y_1 - Y_2)\right] \text{.....(10)}$$

After that, calculating the unknown position in three dimensions (X unk , Y unk ., Z unk) by the deviated equations could be carried out. Then this search could perform another transformation from Cartesian coordinates of unknown position (Xunk ., Yunk ., Zunk.) to geographic coordinates (φunk ., λunk .).

II.6.8. Conclusion and suggestions:

From the result obtained, the following conclusions and suggestions can be derived:

1. 1. From the final evolved equations (8, 9 and 10) which are used to measure the unknown position located at a distance from the observer's position must be found four observers to hold the GPS receiver.

2. LRF must be found to give the value of the distance between the observer and the unknown position.

3. These mathematical equations are successful in practical applications.

Practical example

Suppose we have the following geographic point

Latitude: ∅:52.52^0

Longitude:λ:13.42^0

Height: h:34 m

Not

Convert latitude and longitude to radians

∅rad = π/180*52.52=0.9163 rad

λrad =π/180*13.52=0.2339 rad

Calculate horizontal radius NOT :ℯ^2= (b/a)^2-1

a = 6356752.3142; b = 6378137 Substituting into equation 5

we find NOT≈6390278.661

By substituting a number (2) (3) (4) in the equation we find the following Cartesian coordinates:

X=3771793.97m.

Y = 884593.67m.

Z=5038470.79m.

By comparing the results obtained with the results obtained from the drone and following the same steps, we find almost zero convergence, which indicates that the two geographic coordinates are identical.

II.7. Conclusion:

Facial recognition technology has evolved significantly with the advent of deep learning and traditional methods such as Local Binary Pattern (LBP) and Histogram of Oriented Gradients (HOG). Deep learning techniques, particularly convolutional neural networks (CNN), have revolutionized facial recognition by automatically learning features from raw data, enabling high accuracy even with variations in lighting, pose and image. 'expression. LBP and HOG, on the other hand, are traditional feature extraction methods that encode facial texture and shape information, respectively. While deep learning excels in end-to-end feature learning and overall accuracy, LBP and HOG are computationally efficient and robust to some variations. Combining these approaches can improve facial recognition systems by leveraging the strengths of each method to improve performance in real-world scenarios.

This is the approved program that has been studied and approved in all security, surveillance and people tracking systems. You simply take a photo of the targeted people and submit it to the database, and the system will identify the faces you want to search for or identify, whether through a live broadcast (video) or a stored photo, and with the help of a tracking camera on a drone or any type of surveillance camera.

ANNEX:

Face recognition code:

```
import tkinter ace tk , numpy ace np , cv2 , os , face_recognition
from datetime import datetime

import json
from urllib . request import urlopen


# Initialize empty lists to store images and people's names.
known_faces = []
face_labels = []

# Get a list of all images in the TrainingImages directory.
image_files = bone . listdir ( "persons" )

# Loop through the images in the directory.
for image_name in image_files :
    # Read each image and add it to the known_faces list.
    current_image = cv2 . imread ( f'persons / { image_name } ' )
    known_ faces . append ( current_image )

    # Extract the person's name by removing the file extension and add it to the
face_labels list.
    face_ labels . append ( os.path.spliext ( image_name ) [ 0 ] )


# Function to get face encodings from a list of images.
def get_face_encodings ( images ):
    encoding_list = []
    for picture in pictures :
        # Convert the image to RGB format. RGB is Red Green Blue.
        picture = cv2 . cvtColor ( image , cv2.COLOR_BGR2RGB )
        # Get the face encoding for the first face found in the image.
        face_encoding = face_recognition . face _encodings ( image )[ 0 ]
        encoding_ list . append ( face_encoding )
    return encoding_list


# Define a function to document the recognized face.
def document_recognized_face ( name , filename = ' records.csv' ):
    # Get the current date in the YYYY-MM-DD format and GPS.
    capture_date = datetime . now ( ). strftime ( "%Y-%m- %d " )
    url = 'http://ipinfo.io/json'
    response = urlopen ( url )
```

```
    # Check if the specified CSV file exists.
    if not bone . path . isfile ( filename ):
        # If the file doesn't exist, create it and write the header.
        with open ( filename , 'w' ) as f :
            f . write ( 'Name,Date,Time,gps' ); # Create the file and write the
header.


    # Open the CSV file for reading and writing ('r+')
    with open ( filename , 'r+' ) as file :
        # Read all lines from the file into a list.
        lines = file . readlines ()

        # Extract the names from existing lines in the CSV.
        existing_names = [ line . split ( "," )[ 0 ] for line in lines ]

        # Check if the provided name is not already in the existing names.
        if name not in existing_names :
            # Get the current time in the HH: MM:SS format.
            now = datetime . now ( )
            current_time = now . strftime ( "%H:%M:%S" )
            gps = json . load ( response )

            # Write the new entry to the CSV file including name, capture date,
and time.
            file . write ( f ' \n { name } , { capture_date } , { current_time }
, { gps } ' )


# Get face encodings for known images.
known_face_encodings = get_face_encodings ( known_faces )


# Function to start the Facial recognition program.
def start_recognition_program ( ) :
    # Open a webcam for capturing video. If you are using your computer's webcam,
change 1 to 0.
    # If using an external webcam, leave it as 1.
    video_capture = cv2 . VideoCapture ( 0 )

    while True :
        # Read a frame from the webcam.
        frame = video_capture . read ( 0 )

        # Check if the frame is not None (indicating a successful frame capture).
        if frame is not None :
```

```
            frame = frame [ 1 ] # The frame is usually the second element of the
tuple returned by video_capture.read().

            # Resize the image to a smaller size.
            resized_frame = cv2 . resize ( frame , ( 0,0 ) , None , 0.25,0.25 )
            resized_frame = cv2 . cvtColor ( resized_frame , cv2 . COLOR_BGR2RGB
)

            # Detect faces in the current frame.
            face_locations = face_recognition . face _locations ( resized_frame )

            # Get face encodings for the faces detected in the current frame.
            current_face_encodings = face_recognition . face _encodings (
resized_frame , face_locations )

            # Loop through the detected faces in the current frame.
            for face_encoding , location in zip ( current_face_encodings ,
face_locations ):
                # Compare the current face encoding with the known encodings.
                matches = face_recognition . compare _faces (
known_face_encodings , face_encoding )
                face_distances = face_recognition . face_distance (
known_face_encodings , face_encoding )

                # Find the index of the best match. That is, the best
resemblance.
                best_match_index = np . argmin ( face_distances )

                if matches [ best_match_index ]:
                    # If a match is found, get the name of the recognized person.
                    recognized_name = face_labels [ best_match_index ].upper ()

                    # Extract face location coordinates.
                    top , right , bottom , left = rental
                    top , right , bottom , left = top * 4 , right * 4 , bottom *
4 , left * 4

                    # Draw a rectangle around the recognized face.
                    cv2 . rectangle ( frame , ( left , top ), ( right , bottom ),
( 0 , 255 , 0 ), 2 )

                    # Draw a filled rectangle and display the name above the
face.
                    cv2 . rectangle ( frame , ( left , bottom - 35 ), ( right ,
bottom ), ( 0 , 255 , 0 ), cv2 . FILLED )
                    cv2 . putText ( frame , recognized_name , ( left + 6 , bottom
- 6 ), cv2 . FONT_HERSHEY_COMPLEX , 1 ,
```

```
( 255 , 255 , 255 ), 2 )
                    document_recognized_face ( recognized_name )
                if not matches [ best_match_index ]:

                    # Extract face location coordinates.
                    top , right , bottom , left = rental
                    top , right , bottom , left = top * 4 , right * 4 , bottom *
4 , left * 4

                    # Draw a rectangle around the recognized face.
                    cv2 . rectangle ( frame , ( left , top ), ( right , bottom ),
( 0 , 0 , 255 ), 2 )

                    # Draw a filled rectangle and display the name above the
face.
                    cv2 . rectangle ( frame , ( left , bottom - 35 ), ( right ,
bottom ), ( 0 , 0 , 255 ), cv2 . FILLED )
                    cv2 . putText ( frame , "unknown" , ( left + 6 , bottom - 6
), cv2 . FONT_HERSHEY_COMPLEX , 1 ,
( 255 , 255 , 255 ), 2 )

            # Display the image with recognized faces.
            cv2 . imshow ( "Webcam" , frame )

        # Check for key press
        key = cv2 . waitKey ( 1 ) & 0x FF

        # Check if the 'q' key is pressed to exit the program.
        if key == ord ( 'q' ):
            break

    # Release the video capture and close all OpenCV windows.
    video_ capture . release ()
    cv2 . destroyAllWindows ()


# Create the main application window.
root = tk . Tk ()
root . title ( “Face Recognition Program” )

# Create a label
label = tk . Label ( root , text = "Click the button to start the facial
recognition program" )
label . pack ( pady = 10 )

# Create a button to start the program
```

```
start_button = tk . Button ( root , text = "Start Recognition" , command =
start_recognition_program )
start_button . pack ( pady = 10 )


# Function to exit the application. This is for leaving the entire program. To
leave the webcam stream, hit q.
def quit_app ( ) :
    root . exit ()
    cv2 . destroyAllWindows ()


# Create a quit button to exit the application.
exit_button = tk . Button ( root , text = "Close" , command = quit_app )
exit_button . pack ( pady = 10 )

# Start the Tkinter event loop.
root . mainloop ()
```

GPS code :

```
import json
from urllib . request import urlopen

url = 'http://ipinfo.io/json'
response = urlopen ( url )
gps = json . load ( response )

print ( gps )
```

location code :

```
import geocoder
import folium

our_ip = geocoder . ip ( '105.235.234.564' )
rental = our_ ip .latlng
our_map = folium . Map ( location = location , zoom_start = 12 )
folium . Marker ( location ). add_to ( our_map )
our_ map . save ( "map.html" )

print ( location )
```

Chapter III:

Unveiling the NEO-M8N: A Powerful GPS Module

III.1. Introduction:

The Global Positioning System (GPS) has revolutionized our ability to navigate the world with precision. From smartphones to commercial airlines, GPS technology plays a vital role in various applications. This chapter delves into a popular GPS module, the NEO-M8N, manufactured by u-blox. We'll explore its functionalities, advantages, and considerations [36-37].

III.2. Core Functionality and Features

The NEO-M8N stands out as a concurrent multi-GNSS module. This means it can track signals from multiple satellite constellations simultaneously, including GPS (American), GLONASS (Russian), BeiDou (Chinese), and Galileo (European). This capability offers several advantages:

• **Enhanced Accuracy:** By utilizing signals from various constellations, the NEO-M8N can achieve improved positioning accuracy, especially in challenging environments with limited satellite visibility.

• Faster Time to First Fix (TTFF): TTFF refers to the time it takes a GPS receiver to acquire a signal and provide a valid location fix. The NEO-M8N boasts rapid cold start times (initial signal acquisition), ensuring your project is operational quickly.

Here's a table summarizing the key technical specifications of the NEO-M8N:

Table.III. 1:summarizing the key technical specifications of the NEO-M8N

Feature	Specification
Navigation Sensitivity	-167 dBm
Cold Start Time	26 seconds
Update Rates	18 Hz (single GNSS), 10 Hz (concurrent GNSS)
Accuracy	2.0 meters CEP (Circular Error Probable)
Supported Interfaces	UART (Universal Asynchronous Receiver Transmitter), I2C (Inter-Integrated Circuit)
Physical Dimensions	Approx. 12mm x 16mm

III.3. Advantages of the NEO-M8N

The NEO-M8N offers a compelling set of advantages that make it a popular choice for various GPS-based projects:

• **Superior Accuracy and Reliability:** The combination of concurrent GNSS and high sensitivity translates to more accurate and reliable positioning data. This is crucial for applications like:

• **Drone Navigation:** Precise location information is essential for safe and autonomous drone flight.



Figure.III. 1 Autonomous drone used for mapping areas

• **Faster Time to First Fix:** Rapid cold start times minimize delays in acquiring a signal, ensuring projects are operational quickly. This is beneficial for applications that require immediate location data upon startup.

• **Versatility Across Applications:** The NEO-M8N's ability to leverage multiple GNSS constellations and its compact form factor make it adaptable to a wide range of applications. Here are a few examples:

• **Precision Agriculture:** GPS data from the NEO-M8N can be used for tasks like automated tractor guidance and optimizing crop yields.



Figure.III. 2:Tructor used to harvest autonomously .

• **Asset Tracking:** The NEO-M8N can be integrated into devices to track the location of vehicles, equipment, or personnel in real-time.



Figure.III. 3:Tracking cargo trucks .

• **Ease of Integration:** The NEO-M8N supports common communication protocols like UART and I2C, simplifying integration with microcontrollers and development boards like Arduino or Raspberry Pi. This makes it accessible for hobbyists and developers with varying levels of experience.

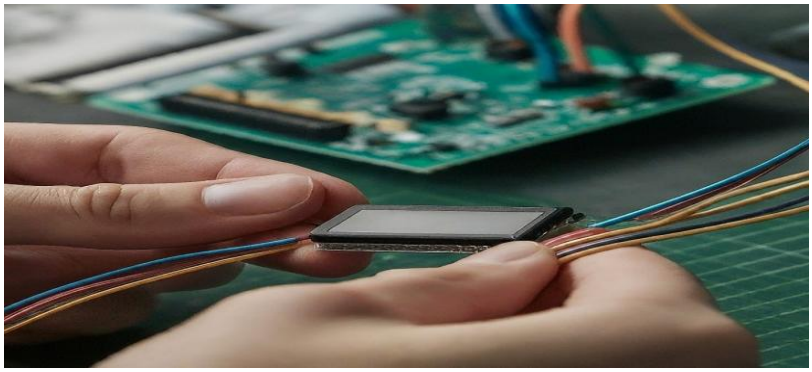


Figure.III. 4:Microcontroller installation of Neo-M8N GPS .

• **Cost-Effective Solution:** Compared to higher-end GNSS modules, the NEO-M8N offers a balance of performance and affordability. This makes it a suitable choice for budget-conscious projects and hobbyist applications.

Considerations and Potential Drawbacks

While the NEO-M8N is a powerful tool, it's not without limitations:

- **Power Consumption:** The NEO-M8N does draw some power, which might be a concern for battery-powered applications where low-power operation is critical. Depending on your project's requirements, you might need to consider power saving strategies like reducing update rates or utilizing sleep mode.

Learning Curve:

The learning curve refers to the relationship between the effort required (time, resources) and the proficiency gained in a particular skill or task. In the context of the NEO-M8N, the learning curve represents the time and effort needed to become familiar with its setup, configuration, and utilization within your drone project.

Here's a breakdown of the learning curve for the NEO-M8N:

- **Steeper Curve:** For beginners with limited experience in electronics or programming, the initial setup and configuration of the NEO-M8N might involve a steeper learning curve. Understanding communication protocols, interpreting data output, and integrating it with your drone's flight controller might require some research and practice.
- **Shorter Curve:** For individuals with experience in electronics, programming, or familiarity with similar GPS modules, the learning curve for the NEO-M8N can be shorter. The readily available resources, tutorials, and user communities can further simplify the process.

III.4. The Power of Precision: Why GPS Matters for Drones

Imagine a drone tasked with capturing stunning aerial photos of a sprawling landscape. Without accurate GPS data, the drone might struggle to maintain a consistent flight path or return to its designated landing zone. This is where the NEO-M8N comes into play.

The NEO-M8N boasts impressive features that translate to enhanced drone performance:

• **Concurrent Multi-GNSS:** This capability allows the NEO-M8N to track signals from multiple satellite constellations (GPS, GLONASS, BeiDou, Galileo) simultaneously. This is crucial for maintaining a strong signal even in environments with limited satellite visibility, such as urban canyons or dense forests.

• **High Sensitivity:** The NEO-M8N's exceptional sensitivity (-167 dBm) enables it to receive weak GPS signals, ensuring reliable positioning even under challenging conditions.

• **Fast Time to First Fix (TTFF):** Rapid cold start times minimize delays in acquiring a signal, allowing drones to become operational quickly after takeoff.

These features combine to provide drones with precise location data, enabling them to:

• **Maintain Stable Flight Paths:** Accurate GPS information allows the drone's flight controller to maintain a consistent course, even in windy conditions.

• **Perform Autonomous Operations:** Drones equipped with the NEO-M8N can be programmed to follow pre-defined waypoints, autonomously capturing aerial footage or delivering packages to designated locations.

• **Enhance Safety:** Reliable GPS data is essential for features like geofencing, which restricts the drone's flight area, preventing accidental collisions or trespassing.

Drone Applications That Benefit from the NEO-M8N

The NEO-M8N's capabilities cater to a wide range of drone applications, including:

• **Aerial Photography and Videography:** Drones equipped with the NEO-M8N can capture breathtaking aerial shots with precise positioning, ensuring consistent framing and smooth flight paths.



Figure.III. 5:Autonomous drone used scanning photography.

• **Search and Rescue Operations:** The ability to navigate autonomously and maintain a strong signal in challenging environments makes the NEO-M8N ideal for search and rescue drones, allowing them to locate missing persons quickly and efficiently.



Figure.III. 6:Autonomous drone used for rescuing and searching.

• **Precision Agriculture:** Drones with the NEO-M8N can be used for tasks like crop mapping, field scouting, and targeted pesticide application. Precise data location ensures accurate data collection and efficient use of resources.

• **Delivery Drones:** Reliable GPS navigation is crucial for the success of autonomous delivery drones. The NEO-M8N helps ensure accurate deliveries, even in complex urban environments.



Figure.III. 7:The way packages delivered by some autonomous drones.

The NEO-M8N offers benefits beyond core GPS functionality for drones:

• **Compact Design:** The small size of the NEO-M8N allows for easy integration into various drone designs without adding significant weight or bulk.

• **Ease of Integration:** Communication protocols like UART and I2C simplify integration with flight controllers commonly used in drones.

• **Cost-Effective Solution:**

Compared to high-end GNSS modules, the NEO-M8N provides a balance of performance and affordability, making it a suitable choice for hobbyist drone builders and commercial drone applications.

• **Considerations and Potential Challenges**

While the NEO-M8N offers significant advantages for drones, some considerations need to be addressed:

• **Power Consumption:**

While not excessive, the NEO-M8N does draw power. For extended drone flights, managing power consumption might be necessary. Strategies like reducing update rates or utilizing sleep mode can be implemented.

• **Flight Regulations:**

Drone regulations vary depending on the region. Pilots should always be aware of and comply with local regulations regarding drone operation, particularly for autonomous flights.

III.5. Installation of gps NEO-M8N using arduino microcontroller.

Code:

```
#include <SoftwareSerial.h>

// Choose two Arduino pins to use for software serial

int RXPin = 2;

int TXPin = 3;

//Default baud of NEO-6M is 9600

int GPSBaud=9600;

// Create a software serial port called "gpsSerial"

SoftwareSerial gpsSerial (RXPin, TXPin);

void setup () {

// Start the Arduino hardware serial port at 9600 baud

Serial.begin(9600);

// Start the software, serial port at the GPS's default baud

gpsSerial.begin(GPSBaud);

} void loop() {

// Displays information when new sentence is available.

while (gpsSerial.available() > 0)

Serial.write(gpsSerial.read()); }
```

III.6. Conclusion:

Beyond its current applications, the NEO-M8N's potential soars in tandem with the burgeoning drone industry. As developers explore uncharted territories in autonomous flight, navigate vast distances, and integrate drones with complex onboard systems, the NEO-M8N's reliable performance and user-friendly design solidify its position as a cornerstone for future advancements. This versatile GPS module, by offering a rock-solid foundation for precise positioning, is poised to become a trusted partner on the exhilarating journey towards a future brimming with groundbreaking drone innovation.

Chapter IV:

PID Control

IV.1. Introduction:

A. open loop system:

An open loop system does not take into account the response of this system, that is to say it does not have a feedback loop. Concretely, this amounts, for example, to drive a car with your eyes closed, hoping that the vehicle will follow the desired trajectory. An open loop system is all rubbish, what good can it do? Most of the time, open loop control is used when the system response is difficult to measure or diagnose. That is to say, when it is very complicated to get feedback on the result obtained. Or simply because the system does not require a feedback loop. For example, automatic watering does not need to have any feedback, it just waters. Finally, the open-loop study also makes it possible to draw the Nyquist diagram of a closed-loop system in order to study its stability. This last point is a fairly advanced idea of automaticity that we will not deal with here. In the case of our drone, such a system would be ineffective and would definitely send it into the background [38].

B. Closed loop system:

A system is said to be closed loop when the output of the process is taken into account to calculate the input. As part of our drone, the system will have sensors which will provide information on its state (tilt, angular speed, acceleration). These will be taken into account to correct the trajectory of the drone. Schematically, we represent a closed loop system as follows [38]:

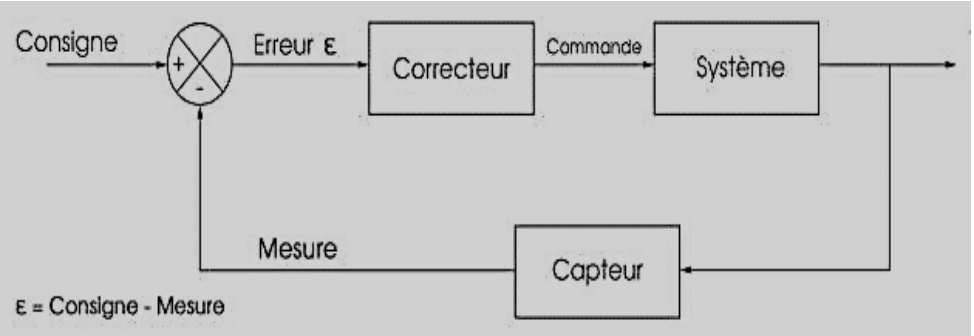


Figure. IV. 1:Closed loop servo system

The set point arrives at the system input. This is the objective that we give to our system (a speed, a direction, a temperature, etc.). From this instruction we subtract a measurement carried out on the system: we then obtain the error (notedε).This error is then passed as input to the corrector (or regulator) which then calculates the output value of the system. For us, the output value will be the rotational speed of each motor. Remember, it is by adjusting the speed of each motor that we can orient our drone in space and make it follow a trajectory.

IV.2. PID regulator:

The regulator (or corrector) will apply a correction to the system based on the measured error. From a mathematical point of view, a PID regulator can be expressed as follows in the time domain:

$U(t) = Kp\ e(t) + Ki \int_0^t e(t')\,dt' + Kd.\ De(t)/dt$

Now we'll break it all down to understand what it's about. But first, what does PID mean? PID is an acronym for Proportional, Integral, Derivative. These are the three ways in which the regulator acts:

• Proportional action: the error is multiplied by a gain Kp.

• Integral action: the error is integrated and multiplied by a gain.

• Derived action: the error is derived and multiplied by a gain Kd. If we take the diagram of the closed loop system and integrate the PID regulator, this is what it looks like [38-39]:

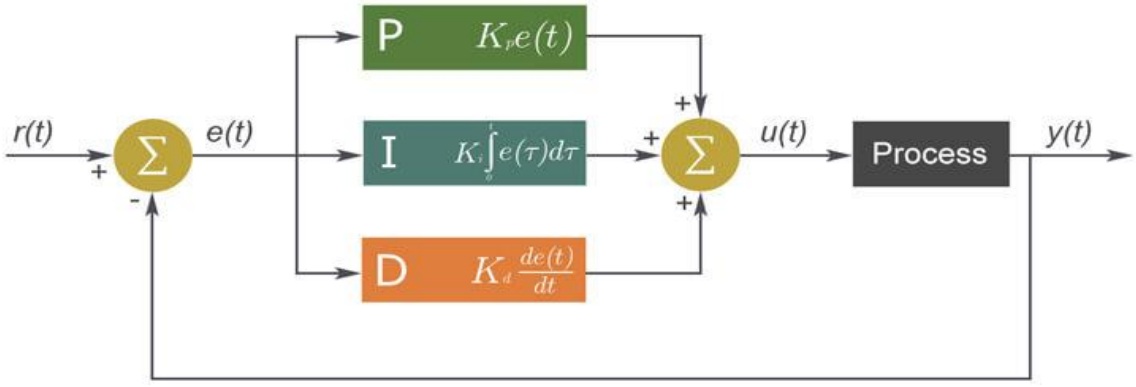


Figure. IV. 2:Block diagram of a PID regulator.

IV.3. Proportional Action:

Proportional action is the most intuitive of the three. This involves applying a coefficient Kp to the error.

Kp. e(t)

This is naturally what we do when we drive: when the car starts to deviate from its path, we jerk the steering wheel in the opposite direction. The error here is the difference between the desired trajectory and that estimated by our eyes and the correction made is proportional to the error. The more we increase the gain, the more quickly the system reacts, but it then gains in uncertainty and loses in precision [39].

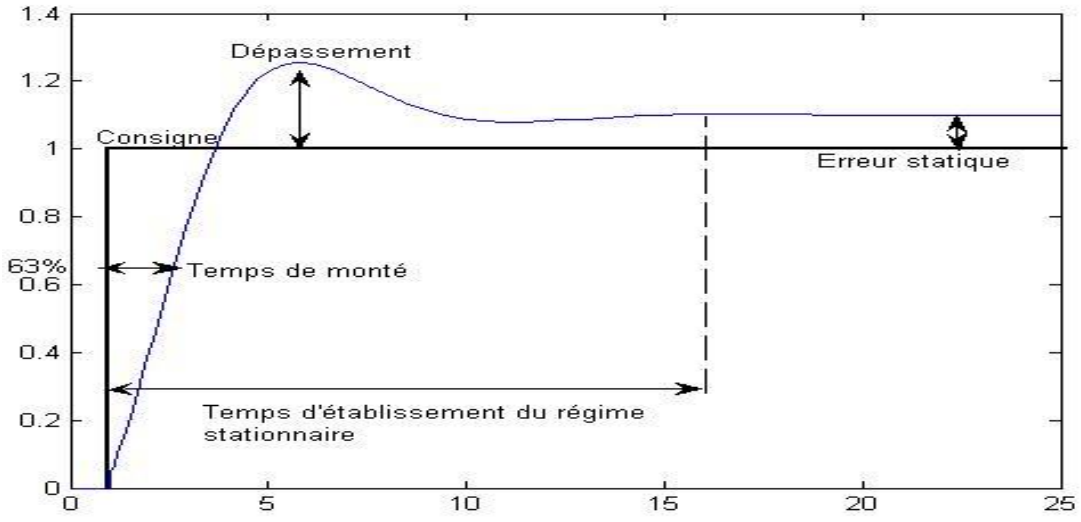


Figure. IV. 3:Example of a system responding to a given instruction

The more we increase the gain, the closer the system gets to its point of anxiety. When the latter is reached, we then obtain an oscillator. It is on this very principle that electronic oscillators work. When we exceed this point, the system is then unstable.

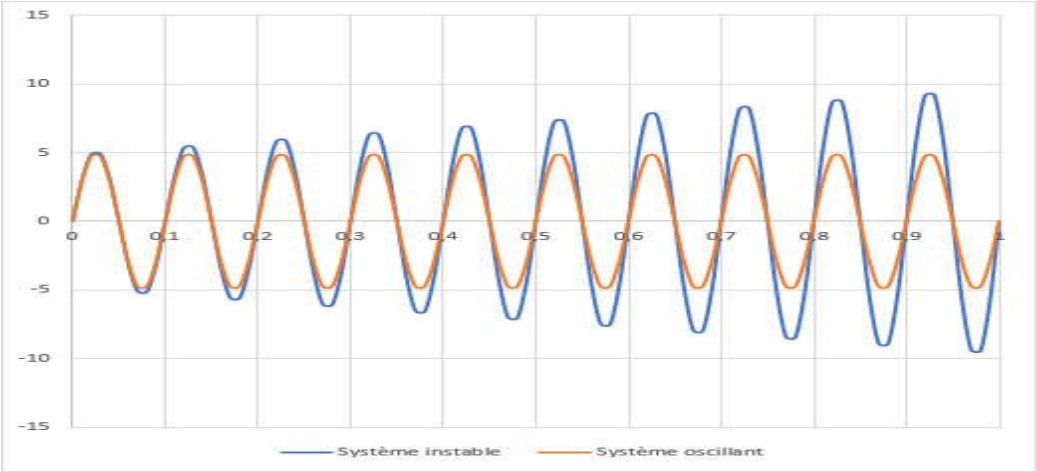


Figure. IV. 4: Response of an unstable system and oscillating system

And we quickly understand why: if I jerk the wheel hard to correct the camera of my car, the correction made will be even worse than the initial error. The next time it will be even worse and so on.

Table. IV. 1:Summary of proportional action

Summary of proportional action		
Precision	**Stability**	**Speed**
↘	↘	↗

IV.4. Actions

IV.4.1. Integral Action

The integral action consists of summing the measured errors over time and applying a coefficient Ki to all:

$$Ki \int_{t-0} e(t') dt'$$

This component makes it possible to reduce the static error and therefore improve the precision of the system. On the other hand, it increases stability and reduces speed [40].

IV.4.2. Derivative Action

The derived action consists of calculating the difference between the current error and the error at t-1 and applying a coefficient Kd to the result:

$K_i \int_{t-0} e(t') dt'$

This component increases the stability of the system but at the expense of speed.

Table. IV. 2:Summary of derivative action

Summary of derivative action		
Precision	**Stability**	**Speed**
-	↗	↘

Table. IV. 3:Summary of the full action

Summary of the full action		
Precision	**Stability**	**Speed**
↗	↘	↘

IV.4.3. PID action:

Ultimately, a PID regulator is a clever mix of these three actions.

In the following animation we can see the effects of each action on the response of a system to a step type instruction [40]:

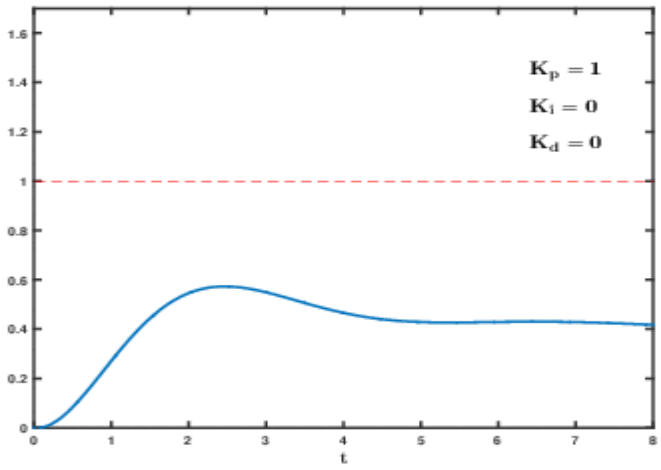


Figure. IV. 5: Effect of varying the parameters of a PID regulator

IV.5. Arduino installation:

IV.5.1. Implementation for Arduino:

Our algorithm will have a lot of things to do. Here are the different steps:

1. Reading sensors

2. Calculation of angles & angular velocities

3. Radio receiver playback

4. Calculation of instructions

5. Error calculation

6. Calculation of outputs: PID regulator

7.Battery drop compensation

8. Generation of output signals

IV.5.2. Calculation of instructions:

To translate the signals received into a dimension usable by the regulator we will use the following function:

```
/**
 * Calcul la consigne d'un axe en degré/sec.
 *
 * @param float angle         Angle mesuré en degré sur l'axe (X, Y, Z)
 * @param int   channel_pulse Durée en µs de l'impulsion reçue pour l'axe (comprise entre 1000µs et 2000µs)
 * @return float
 */
float calculateSetPoint (float angle, int channel_pulse) {
   float level_adjust = angle * 15;
   float set_point    = 0;
   // On s'accorde une marge 16µs pour de meilleurs résultats
   if (channel_pulse > 1508) {
      set_point = channel_pulse - 1508;
   } else if (channel_pulse < 1492) {
```

```
      set_point = channel_pulse - 1492;
    }
    set_point -= level_adjust;
    set_point /= 3;
    return set_point;
}
```

Arduino

This calculation makes it possible to obtain a setpoint expressed in °/sec. We apply this calculation for each of the 3 axes.It comes from where the value 1515?

This coefficient is used to limit the maximum value of the drone's inclination. With the value1515, we obtain a maximum inclination of ±32.8°.

Demonstration

The above function can be expressed by the following formula (for pulse > 1508):

set_point t= {puLse − 1508 − hasnotgLe × 15}/3

From this formula we extract the angle:

hasnotgLe = {−3 × set_point + puLse −1508}/15

By definition, when the drone has reached the setpoint, set_point is worth 0°/s. In this case, the expression for the angle becomes:

hasnotgLe = {puLse – 1508}/15

The maximum pulse value is2000µs, which then gives a maximum angle value of 32.8°.If you want to change the maximum angle, you are free to change the coefficient 15 by the value of your convenience.

IV.5.3. Calculation of errors:

Now that the setpoints are calculated, we can calculate the different errors that the PID regulator needs:

```
/**
 * Calcul des erreurs
 */
void calculateErrors() {
```

```
    // Calcul des erreurs courantes
    errors[YAW]   = angular_motions[YAW]   - pid_set_points[YAW];
    errors[PITCH] = angular_motions[PITCH] - pid_set_points[PITCH];
    errors[ROLL]  = angular_motions[ROLL]  - pid_set_points[ROLL];

    // Calcul des sommes d'erreurs : composante intégrale
    error_sum[YAW]   += errors[YAW];
    error_sum[PITCH] += errors[PITCH];
    error_sum[ROLL]  += errors[ROLL];

    // on s'assure que ∫e.Ki ne dépasse pas 400
    error_sum[YAW]   = minMax(error_sum[YAW],   -400/Ki[YAW],   400/Ki[YAW]);
    error_sum[PITCH] = minMax(error_sum[PITCH], -400/Ki[PITCH], 400/Ki[PITCH]);
    error_sum[ROLL]  = minMax(error_sum[ROLL],  -400/Ki[ROLL],  400/Ki[ROLL]);

    // Calcul du delta erreur : composante dérivée
    delta_err[YAW]   = errors[YAW]   - previous_error[YAW];
    delta_err[PITCH] = errors[PITCH] - previous_error[PITCH];
    delta_err[ROLL]  = errors[ROLL]  - previous_error[ROLL];

    // L'erreur courante devient l'erreur précédente pour le prochain tour de boucle
    previous_error[YAW]   = errors[YAW];
    previous_error[PITCH] = errors[PITCH];
    previous_error[ROLL]  = errors[ROLL];
}
```

Arduino

IV.5.4. PID regulator:

The goal of our PID regulator will be to calculate the duration of the pulse to be provided for each ESC to act directly on the rotation speed of the motors.Remember, the pulse duration is between 1000µs and 2000µs.We know the face of a PID regulator: e. Kp + ∫e. Ki + Δe. Kd, and we now have all the elements to carry out this calculation on each of the axes[40]:

```
// PID = e.Kp + ∫e.Ki + Δe.Kd
yaw_pid = (errors[YAW] * Kp[YAW]) + (error_sum[YAW] * Ki[YAW]) + (delta_err[YAW] * Kd[YAW]);
pitch_pid = (errors[PITCH] * Kp[PITCH]) + (error_sum[PITCH] * Ki[PITCH]) + (delta_err[PITCH] * Kd[PITCH]);
roll_pid = (errors[ROLL] * Kp[ROLL]) + (error_sum[ROLL] * Ki[ROLL]) + (delta_err[ROLL] * Kd[ROLL]);
```

Arduino

Ok great, but that doesn't give us the pulse duration of each ESC!

And that's where it gets complicated. Let's look at the diagram of our drone and the orientation conventions:

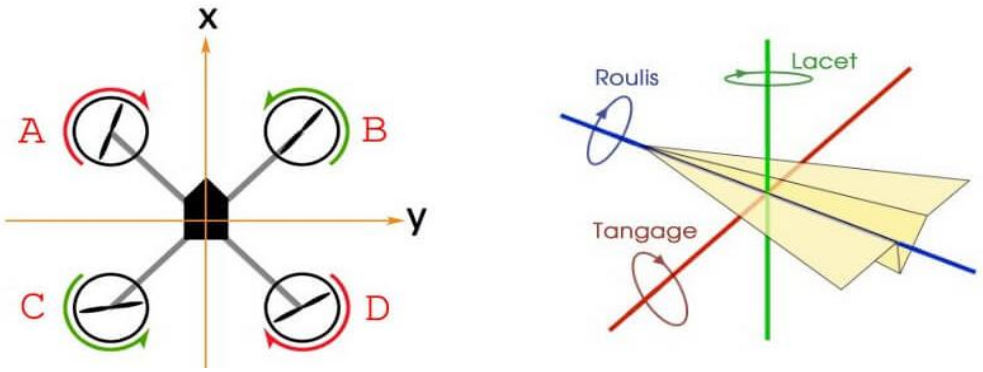


Figure. IV. 6:• Roll: positive roll • Pitch: positive pitch • Yaw: positive yaw

The letters A, B, C, and D refer to the four arms of the quadcopter. The Z axis is orthogonal to the X and Y axes.

Axis orientation

Let's imagine that our drone turns on its roll axis at +10°/s (it therefore leans to the right) and that the setpoint is 0°/s. The error compared to the instruction is $\epsilon = +10 - 0 = +10$ With a positive error and positive coefficients Kp, Ki & Kd, we deduce that roLL_pid >0

To right the drone, motors A & C must slow down and motors B & D must speed up.

For motor A for example, it is therefore necessary to extract roLL_pid to make it slow down.

Table. IV. 4:Roll axis table

Roll (measure)	**Order**	**Error = measurement – instruction**	**roll_pid**	**Correction to apply**	**Operation to apply**
+10°/s	**0°/s**	**+10°/s**	**> 0**	**A & C ↘** **B & D ↗**	**A= roll_pid** **B += roll_pid** **C -= roll_pid** **D += roll_pid**

We apply the same logic for the pitch axis:

Table. IV. 5:Pitch axis table:

Pitch (measure)	Order	Error = measurement – instruction	pitch_pid	Correction to apply	Operation to apply
+10°/s	0°/s	+10°/s	> 0	A & B ↘ C & D ↗	A-= pitch_pid B -= pitch_pid C += pitch_pid D += pitch_pid

And for the yaw axis table:

Table. IV. 6:yaw axis table:

Yaw (measure)	Order	Error = measurement – instruction	Yaw_pid	Correction to apply	Operation to apply
+10°/s	0°/s	+10°/s	> 0	A & B ↘ C & D ↗	A+ = Yaw_pid B - = Yaw_pid C - = Yaw_pid D + = Yaw_pid

Basically, we initialize the rotation speed of each motor with the value of the *throttle*. In the end, by putting all of this together we obtain:

```
pulse_length_esc1 = throttle - roll_pid - pitch_pid + yaw_pid;
pulse_length_esc2 = throttle + roll_pid - pitch_pid - yaw_pid;
pulse_length_esc3 = throttle - roll_pid + pitch_pid - yaw_pid;
pulse_length_esc4 = throttle + roll_pid + pitch_pid + yaw_pid;
```

Arduino

And to be sure not to end up with completely excessive values, we will simply make sure to keep them in the meantime. [1100,2000] µs.

```
pulse_length_esc1 = minMax(pulse_length_esc1, 1100, 2000);
pulse_length_esc2 = minMax(pulse_length_esc2, 1100, 2000);
pulse_length_esc3 = minMax(pulse_length_esc3, 1100, 2000);
pulse_length_esc4 = minMax(pulse_length_esc4, 1100, 2000);
```

Arduino

I set 1100µs as the minimum value to be sure that the motors idle once the drone starts.

IV.6. Necessary functions:

IV.6.1. Start/stop:

For security reasons, we will implement a drone start/stop function. By default, the drone is in the STOPPED state. As long as it is in the STOPPED state, you can move the radio control sticks, nothing should happen.Once the drone is in the STARTED state, the motors idle by default and the PID regulator does the rest.To avoid a chaotic takeoff, we will re-initialize the PID controller variables each time the STARTED state is entered.

The commands will be as follows:

• Start: move the left stick to the lower left corner and bring it back to the center.

• Stop: move the left stick to the lower right corner.

We can represent this algorithm by a simple state machine:

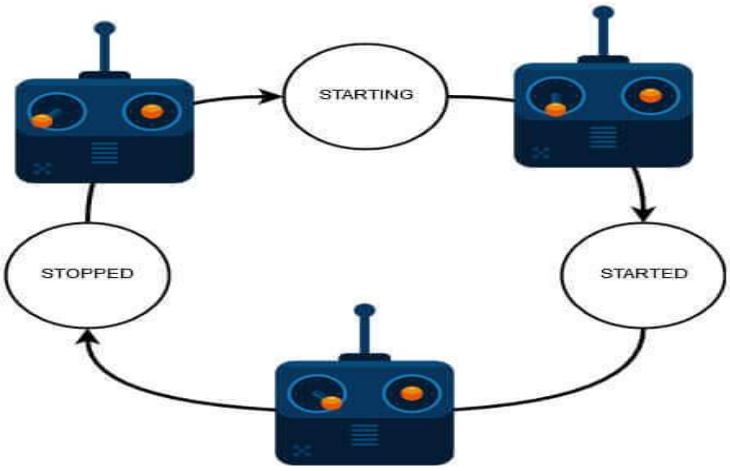


Figure. IV. 7:Start/stop state machine

To do this, we will implement a function `is Started ()` which will calculate the status of the drone and return `true` whether it is STARTED:

```
// On declare quelques constantes pour la lisibilité
#define STOPPED  0
#define STARTING 1
#define STARTED  2
// On initialise l'état à STOPPED
int status = STOPPED;


/**
 * @return bool
 */
bool isStarted()
{
    // Quand le stick de gauche est positionné dans le coin inférieur gauche
```

```
  if (status == STOPPED && pulse_length[mode_mapping[YAW]] <= 1012 && pulse_length[mode_mapping[THROTTLE]] <= 1012) {
    status = STARTING;
  }
  // Quand le stick de gauche revient au centre
  if (status == STARTING && pulse_length[mode_mapping[YAW]] == 1500 && pulse_length[mode_mapping[THROTTLE]] <= 1012) {
    status = STARTED;

    // Ré-initialise les variables du PID pour éviter un démarrage chaotique
    resetPidController();

    resetGyroAngles();
  }

  // Quand le stick de gauche est positionné dans le coin inférieur droit
  if (status == STARTED && pulse_length[mode_mapping[YAW]] >= 1988 && pulse_length[mode_mapping[THROTTLE]] <= 1012) {
    status = STOPPED;
    // On s'assure de stopper touts les moteurs
    stopAll();
  }
  return status == STARTED;
}
```

Arduino

Then just call it in `loop ()`:

```
void loop()
{
   // ...
   if (isStarted()) {
      pidController();
   }
   // ...
}
```

Arduino

IV.6.2. Compensate for battery drop

After a while of use, the battery voltage will decrease, which directly results in a reduction in the drone's lift. To compensate for this, the rotation speed of the motors must be increased.

We will therefore simply read the supply voltage with an analog input from the Arduino and apply a compensation factor to the motors.

Voltage divider bridge

We simply create a voltage divider bridge connected to the analog input A0 of the Arduino and add this piece of code:

```
// Variable globale
int battery_voltage = 0;


void loop() {
   // ...
   compensateBatteryDrop();
   // ...
```

```
}


/**
 * Compense la baisse de tension de la batterie.
 */
void compensateBatteryDrop() {
  if (isBatteryConnected()) {
    int coeff = ((1240 - battery_voltage) / (float) 3500);


    pulse_length_esc1 += pulse_length_esc1 * coeff;
    pulse_length_esc2 += pulse_length_esc2 * coeff;
    pulse_length_esc3 += pulse_length_esc3 * coeff;
    pulse_length_esc4 += pulse_length_esc4 * coeff;
  }
}
/**
 * Retourne TRUE si la batterie est bien connectée, FALSE sinon.
 */
bool isBatteryConnected() {
  // On applique un simple filtre passe-bas pour filtrer le signal (Fc ≈ 10Hz et gain de ~2.5dB dans la bande passante)
  battery_voltage = battery_voltage * 0.92 + (analogRead(0) + 65) * 0.09853;
  return battery_voltage < 1240 && battery_voltage > 800;
}
```

Arduino

IV.6.3. Generate ESC signals:

We arrive at the last step: generating the ESC control signals.

Why don't we just use the Servo library as for ESC calibration?

Because the Servo library can only generate signals with a frequency of 50Hz. In the case of our drone, we need a frequency of 250Hz.And as we have already set up a timer at 250Hz for the IMU part, we will be able to reuse it for signal generation. we will manipulate the input/output ports for performance reasons [40].

The algorithm is quite simple: we switch all outputs to the HIGH state at the same time. We wait until the duration corresponding to each pulse has elapsed to return the output concerned to the LOW state.

Here is what it looks like in Arduino language:

```
void applyMotorSpeed() {
  // Fs = 250Hz : on envoie les impulsions toutes les 4000µs
  while ((now = micros()) - loop_timer < 4000);
  // Update loop timer
  loop_timer = now;
  // On active les broches #4 #5 #6 #7 (état HIGH)
  PORTD |= B11110000 ;
  // On boucle tant que toutes les sorties ne sont pas retournées à l'état LOW
  while (PORTD >= 16) {
    now = micros();
    difference = now - loop_timer;
    if (difference >= pulse_length_esc1) PORTD &= B11101111; // Passe la broche #4 à LOW
    if (difference >= pulse_length_esc2) PORTD &= B11011111; // Passe la broche #5 à LOW
    if (difference >= pulse_length_esc3) PORTD &= B10111111; // Passe la broche #6 à LOW
    if (difference >= pulse_length_esc4) PORTD &= B01111111; // Passe la broche #7 à LOW
  }
```

```
}
```

IV.7. Wiring:

And finally, here's how to wire everyone:

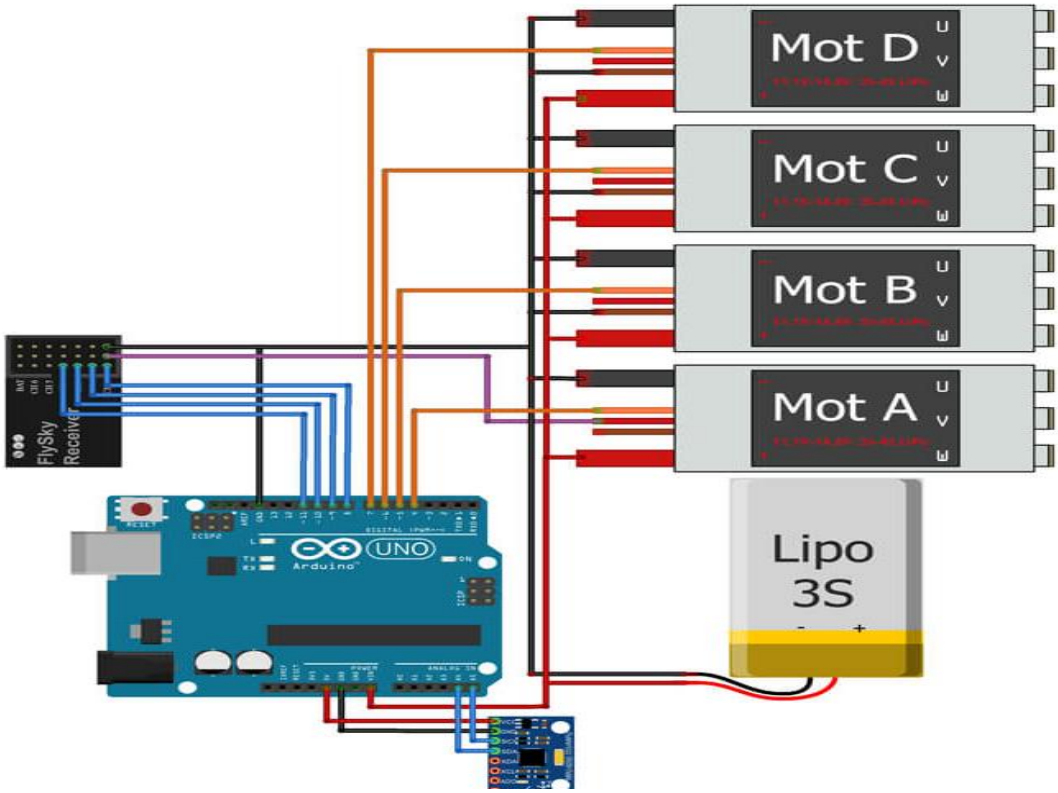


Figure. IV. 8:the receiver is powered by a BEC output from one of the ESCs

IV.7.1. Calibration of PID coefficients

If your drone has the same dimensions as mine, the default coefficients should do the trick and you can move on directly.

Otherwise, you will have to determine the values of the coefficients Kp, Ki & Kd for the three axes yaw, pitch & roll. As the motors are organized in a square, the coefficients of the pitch & roll axes are the same. Which makes 6 values in total to define.

These coefficients can be calculated or determined by experiment. To calculate them, you need to know the transfer function of the system to be controlled, information that we do not have here. We will therefore see the method through experience [40].

IV.7.2. Test bench:

For this calibration step, I highly recommend that you make a test bench to attach the drone to, like this:

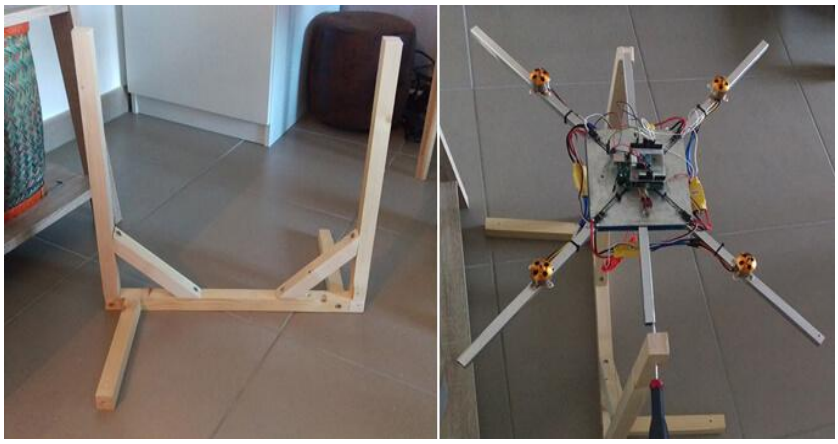


Figure. IV. 9:Benchmark example.

This will prevent you from shooting your drone by sending it into the kitchen table...

IV.7.3. Calibration procedure:

1. Start by initializing all coefficients to 0 and the Z axis coefficients like this:

• Kp=3

• Ki=0.02

• Kd=0

Z axis does not need derivative correction

2. Next, increase the value of Kd (roll & pitch). Turn on the throttle until the drone just begins to rise.

Increase Kd until the drone starts to oscillate. Then its value decreases until the drone becomes stable again. In my case I get the value of Kd=20. Decrease this value by 25% which gives the final value of Kd=15

3. Then increase Kp in steps of 0.2 until the drone starts to oscillate. Reduce the found value by 25% and you get the final value of Kp. In my case I find Kp=1.3.

4. Increase the Ki gain in steps of 0.01. The drone is supposed to become more and more stable. Continue increasing Ki until the drone begins to oscillate slowly. Then decrease Ki by 25% which gives you its final value. In my case I find Ki=0.05.

5. Finally increase Kp again until the drone oscillates quickly. Then slightly decrease this value and you will have a stable drone!

You can continue adjusting the coefficients until you find the config that suits you best.

IV.8. First flight:

IV.8.1. Safety & legislation:

As cool as the aircraft model is, it is subject to regulations. In Algeria there are many rules that it is better to know if you do not want to risk a fine or confiscation of equipment:

Find out about the legislation in your country before embarking on a flight session.

• Protective glasses: A drone that eats is fragments of propellers that fly in all directions at very high speed.

I took one in the forehead (it didn't go far from the eye), it was bloody, so we protect ourselves!

• Good weather: I know you're in a hurry but there's no point going out if it's windy or rainy. Wait until the weather is nice and the wind drops, otherwise you will find your drone in a tree or worse...

• Open terrain: find an open area, far from homes and roads. If you lose control of your drone this will prevent you from a serious accident. Especially since the first few times, it is difficult to master the bouzin.

If necessary, ask permission to squat a field from a farmer during your session

• Accompanists: it's always nicer to share a moment like this with several people. And if something goes wrong, you won't be alone.

And then it's the opportunity to immortalize this moment by asking a friend to film it!

• Battery: make sure you have recharged your LiPo and radio control batteries. It would be a shame to realize on site that the batteries are flat

I'm not going to make an exhaustive list but you get the idea: safety & fun!

IV.9. Conclusion

Throughout this project we will have explored the range of possibilities of our good old Arduino Uno. It’s because he’s got it in his stomach after all ! This is a subject that is as fascinating as it is complex, so don't be discouraged if it doesn't work the first time. I'll tell you a secret: it never works the first time.Persevere, do your research, talk to enthusiasts and everything will be fine. You will have felt it, the drone flies but still quickly tends to slide in the wind and hovering is far from being as simple as you think!

We could then imagine adding new sensors to improve the flight experience:

• altimeter to maintain a stable altitude (hovering)

• compass to maintain a course

• rangefinder for takeoff/landing phases

• PHOTOVOLTIC CELL

This may be the subject of future articles but nothing is certain.

General Conclusion:

In conclusion, this compilation explores various technological advancements that are shaping the future. Unmanned aerial vehicles (UAVs), also known as drones, have become increasingly sophisticated, with applications ranging from military use to environmental monitoring. Facial recognition technology, coupled with drones, has the potential to introduce new functionalities in security and surveillance. The NEO-M8N GPS module represents a reliable and efficient tool for precise positioning, crucial for drone operation. Finally, the understanding of open-loop and closed-loop systems is fundamental for designing effective control systems, essential for ensuring stable and accurate drone flight.

Références:

[1]. D. Varghese, S. Joy, and A. Biju, "Design and Fabrication of a Multi-spice dryer," vol. 09, no. 08, 2022.

[2]. P. Rajendran, N. M. Mazlan, and H. Smith, "Single Cell Li-Ion Polymer Battery Charge and Discharge Characterizations for Application on Solar-Powered Unmanned Aerial Vehicle," *Key Eng. Mater.*, vol. 728, pp. 428–433, Jan. 2017, doi: 10.4028/www.scientific.net/KEM.728.428.

[3]. O. Tatale, N. Anekar, S. Phatak, and S. Sarkale, "Quadcopter: Design, Construction and Testing".

[4]. M. A. Wahid, A. N. Asmi, N. Othman, M. I. Ardani, M. Z. M. Zain, and S. Mansor, "The effect of different motor constants to an commercial propeller," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 884, no. 1, p. 012096, Jul. 2020, doi: 10.1088/1757-899X/884/1/012096.

[5]. K. N. Mogensen, "Motor-control considerations for electronic speed control in drones," 2016.

[6]. F. Ghallabi, A. Kawamura, Y. Pyo, T. Tsuji, and R. Kurazume, "Control architecture for service drone in informationally structured environment," in *2015 IEEE/SICE International Symposium on System Integration (SII)*, Nagoya, Japan: IEEE, Dec. 2015, pp. 611–618. doi: 10.1109/SII.2015.7405049.

[7]. J. Hernandez *et al.*, "Network Controlled Data Acquisition Drone," in *2008 GSW Proceedings*, unknown: ASEE Conferences, Dec. 2021, p. 38573. doi: 10.18260/1-2-620-38573.

[8]. A. Phinyomark, S. Hirunviriya, C. Limsakul, and P. Phukpattaranont, "Evaluation of EMG Feature Extraction for Hand Movement Recognition Based on Euclidean Distance and Standard Deviation".

[9]. V. Rodríguez-Fernández, H. D. Menéndez, and D. Camacho, "A study on performance metrics and clustering methods for analyzing behavior in UAV operations," *J. Intell. Fuzzy Syst.*, vol. 32, no. 2, pp. 1307–1319, Jan. 2017, doi: 10.3233/JIFS-169129.

[10]. Cheng E-J, Chou K-P, Rajora S, et al. Deep sparse representation classifier for facial recognition and detection system. Pattern Recogn Lett 2019; 125: 71–77.

[11]. Deeba F, Memon F, Dharejo FA, et al. LBPH-based enhanced real-time face recognition. Int J Adv Comput Sci Appl 2019; 10(5): 274–280.

[12]. Wijaya EP, Sitompul OS and Mawengkang H. Combina- tion of the local binary pattern histogram method andSHA256 bit in the face recognition system. Int J Res

Rev2020; 7(2): 135–139.

[13]. Ahmed A, Hasan T, Abdullatif FA, et al. A digital signa-ture system based on real time face recognition. In: 2019 IEEE 9th International Conference on System Engineering and Technology (ICSET), Shah Alam, Malaysia, 7 Octo-ber 2019. New York: IEEE.

[14]. Ambre S, Masurekar M and Gaikwad S. Face recogni- tion using Raspberry PI. In: Gujan VK, Zurada JM, Balasubramanian R, et al. (eds) Modern approaches in machine learning and cognitive science: A walkthrough. Cham: Springer, 2020, pp. 1–11.

[15]. Singhal M and Shinghal K. Recent advances in signature verification and facial recognition algorithms. TEST Engineering & Management 2020; 82: 11371–11377.

[16]. V. Kazemi, J. Sullivan, " One Millisecond Face Alignment with an Ensemble of Regression Trees. "

[17]. Nazil Perveen, Darshan Kumar, Ishan Bhardwaj « An Overview on Template Matching Methodologies and its Applications», ID: 54957226, 2013 .

[18]. P. Phillips, P. Grother, R. Michaels, D. Blackburn, E.Tabassi, and M. Bone. «Face recognition vendor test», 2002.

[19]. https://www.grandviewresearch.com

[20]. https://face-recognition.readthedocs.io

[21].. Bomford,G.(1977),Geodesy, 3rded., OxfordUniversity.

[22]. Htm,Epicentre User Guide,POSC Specifications(1997),Coordinate Systems-A Deeper Understanding,Geography Department, University Of Texas.

[23]. Kumar, B. H.;Reddy, K. C. andNamassivaya, N. (2008), Determination of GPS ReceiverPosition Using Multivariate Newton-Raphson Technique for Over Specified Cases, International Journal of Applied Engineering Research (IJAER), 13, 1457-1460.

[24]. Al-Delami, O.M. (2002), Accuracy Measurement of Geodesy and GPS, Tech.University.

[25]. Edu/Openbook(2000),The Global Positioning System; A Shared National Asset,The National Academy Of Sciences.

[26].. Grejner,B.;Dorota, A. and Wang, J. (1998),Gravity Modeling for high accuracy GPS/INS integration,navigation,45,209-220.

[27]. Defense Mapping Agency(1992),world geodetic 1984 (WGS-84),DMA TR 8350.2, 2nd.ed.,Fairfax, UA.

[28]. Kaplan, E.D. and lava, J.I. (1993), Fundamentals of Satellite Navigation, Hughes space and communication.

[29]. Staff Members of Surveying and Faculty of Engineering (1994),Geodetic surveying, Dar El-Rateb Alexandria- Beirut Arab University.

[30].. Hofman, Wellenhof, B. Lichtenegger, H. andEllins, J. (1993),GPS Theory and Practice,New York.

[31].. Epicentrevsage guide, (2000), Geographic Coordinates System Transformations,POSC Specifications Version 2.2,

[32]. Ash, T. ;Coatsee, J. ;Silva, R. and Brown, A. (1995),A Low Cost GPS/Inertial Mapping (GIM) System,Mobile Mapping Symposium, Columbus,OH.

[33]. OTMANI, Hamza, and Ramzi SAIDI. Reconnaissance faciale par Deep Learning. Diss. UNIVERSITE KASDI MERBAH OUARGLA.

[34]. SAKHRI, Ahmed Rami. "Optimisation de la détection pour la reconnaissance de visage."

[35]. BEN_TOUMI, Adel, and AbdElHak BEN_BOUZID. "Reconnaissance de visage par méthode hybride PCA2D ADL2D."

[36]. R. Kikutis, J. Stankūnas, D. Rudinskas, and T. Masiulionis, "Adaptation of Dubins Paths for UAV Ground Obstacle Avoidance When Using a Low Cost On-Board GNSS Sensor," *Sensors*, vol. 17, no. 10, p. 2223, Sep. 2017, doi: 10.3390/s17102223.

[37]. I. Sheridan, "Drones and global navigation satellite systems: current evidence from polar scientists," *R. Soc. Open Sci.*, vol. 7, no. 3, p. 191494, Mar. 2020, doi: 10.1098/rsos.191494.

[38]. V. Grabe, H. H. Bulthoff, and P. R. Giordano, "On-board velocity estimation and closed-loop control of a quadrotor UAV based on optical flow," in *2012 IEEE International Conference on Robotics and Automation*, St Paul, MN, USA: IEEE, May 2012, pp. 491–497. doi: 10.1109/ICRA.2012.6225328.

[39]. Nikunj Mehta, Dharmendra Chauhan, Sagar Patel, Siddharth Mistry, and CSPIT, Changa, Gujarat, India, "Design of HMI Based on PID Control of Temperature," *Int. J. Eng. Res.*, vol. V6, no. 05, p. IJERTV6IS050074, May 2017, doi: 10.17577/IJERTV6IS050074.

[40]. Tsung-Tai Huang, Hung-Yuan Chung, and Jin-Jye Lin, "A fuzzy PID controller being like parameter varying PID," in *FUZZ-IEEE'99. 1999 IEEE International Fuzzy Systems. Conference Proceedings (Cat. No.99CH36315)*, Seoul, South Korea: IEEE, 1999, pp. 269–276 vol.1. doi: 10.1109/FUZZY.1999.793247.