МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ АЕРОНАВІГАЦІЇ, ЕЛЕКТРОНІКИ ТА ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА АЕРОКОСМІЧНИХ СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
______________Юрій МЕЛЬНИК
«_____» ___________2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
«БАКАЛАВР»

Тема: «Система стеження об'єктів для квадрокоптера»

Виконавець: студент групи СУ-404 ___________ Валерій ГЛАДЧЕНКО

Керівник: к.т.н., доцент ___________________________Антоніна КЛІПА

Нормоконтролер: _____________________ Микола ДИВНИЧ

Київ 2024

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY
FACULTY OF AIR NAVIGATION, ELECTRONICS AND TELECOMMUNICATIONS
AEROSPACE CONTROL SYSTEMS DEPARTMENT

APPROVED FOR DEFENCE
Head of the Department
________________ Yurii MELNYK
“____” _____________ 2024

QUALIFICATION PAPER

(EXPLANATORY NOTE)

FOR THE ACADEMIC DEGREE OF BACHELOR

Title: “Object Tracking System for Quadcopter”

Submitted by: student of group CS-404 ___________Valerii HLADCHENKO

Supervisor: PhD, associate professor ___________________Antonina KLIPA

Standards inspector: _________________________ Mykola DYVNYCH

Kyiv 2024

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра аерокосмічних систем управління

Спеціальність: 151 «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____________ Юрій МЕЛЬНИК

«_____»___________2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Гладченка Валерія Григоровича

1. Тема кваліфікаційної роботи «Система стеження об’єктів для квадрокоптера» затверджена наказом ректора від «01» квітня 2024 р. № 511/ст.

2. Термін виконання роботи: з 13.05.2024 по 16.06.2024.

3. Вихідні дані роботи: програмований квадрокоптер, математична модель квадрокоптера.

4. Зміст пояснювальної записки: Теорія квадрокоптерів; Концепції комп’ютерного зору та відстеження об'єктів; Розробка системи стеження об’єктів та проведення лабораторних тестів.

5. Перелік обов’язкового ілюстративного матеріалу: Блок-схема основних етапів комп’ютерного зору, блок-схема алгоритму стеження об’єктів, таблиця коефіцієнтів ПД-регулятора, ілюстрації тестів розробленої системи стеження об’єктів для квадрокоптера.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Огляд літературних джерел та ознайомлення з математичною моделлю квадрокоптера	13.05.2024-17.05.2024	
2	Ознайомлення з теорією квадрокоптерів	18.05.2024-20.05.2024	
3	Дослідження концепцій відстеження об'єктів	21.05.2024-23.05.2024	
4	Розробка системи стеження об’єктів	23.05.2024-28.05.2024	
5	Оформлення пояснювальної записки	28.05.2024-01.06.2024	
6	Підготовка доповіді та презентації	01.06.2024-10.06.2024	

7. Дата видачі завдання: «13» травня 2024 р.

Керівник кваліфікаційної роботи __________________ Антоніна КЛІПА
(підпис керівника)

Завдання прийняв до виконання _____________________ Валерій ГЛАДЧЕНКО
(підпис випускника)

NATIONAL AVIATION UNIVERSITY

Faculty of Air Navigation, Electronics and Telecommunications

Aerospace Control Systems Department

Specialty: 151 “Automation and Computer-integrated Technologies”

APPROVED BY

Head of the Department

_____________ Yurii MELNYK

“______”______________2024

Qualification Paper Assignment for Graduate Student

Hladchenko Valerii Hrihorovich

1. The qualification paper title “Object Tracking System for Quadcopter” was approved by the Rector’s order of “ 01 ” April 2024 № 511/ст.

2. The paper to be completed between: 13.05.2024 and 16.06.2024

3. Initial data for the paper: programmable quadcopter, mathematical model of a quadcopter.

4. The content of the explanatory note: Theory of quadcopters; Concepts of computer vision and object tracking; Development of an object tracking system and laboratory tests.

5. The list of mandatory illustrations: Block diagram of the main stages of computer vision, block diagram of the object tracking algorithm, table of PD controller coefficients, illustrations of tests of the developed object tracking system for a quadcopter.

6. Timetable

№	Assignment	Dates of completion	Completion mark
1	Literature review and description of the mathematical model of a quadcopter	13.05.2024-17.05.2024	
2	Introduction to the Theory of Quadcopters	18.05.2024-20.05.2024	
3	Researching Object Tracking Concepts	21.05.2024-23.05.2024	
4	Object Tracking System Development	23.05.2024-28.05.2024	
5	Preparation of an explanatory note	28.05.2024-01.06.2024	
6	Preparation of the report and presentation	01.06.2024-10.06.2024	

7. Assignment issue date: “13” May 2024

Qualification paper supervisor ____________________ Antonina KLIPA
(the supervisor’s signature)

Issued task accepted ____________________ Valerii HLADCHENKO
(the graduate student’s signature)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи "Система стеження об’єктів для квадрокоптера" містить 84 сторінки, 56 ілюстрацій, 37 джерел.

Актуальність теми полягає в необхідності оснащення квадрокоптерів системами слідування за об'єктами в умовах зростання популярності використання квадрокоптерів в агропромисловості та охоронних підприємствах.

Об'єктом дослідження є система слідування за об'єктами на базі квадрокоптера.

Предметом дослідження є методи та алгоритми комп’ютерного зору та машинного навчання, а також система керування, що використовуються для реалізації функції слідування за об’єктами на квадрокоптері.

Метою роботи є дослідження та розробка алгоритмів системи слідування за об’єктами для квадрокоптера, з метою забезпечення точного автоматичного слідування за рухомими об’єктами.

Методи дослідження включають розробку алгоритму виявлення і відслідковування об’єктів, налаштування ПД-регуляторів для точного керування рухом квадрокоптера, експерементальні дослідження в лабораторних умовах для оцінки ефективності розробленої системи.

Ключові слова: КОМП’ЮТЕРНИЙ ЗІР, МАШИННЕ НАВЧАННЯ, СИСТЕМА АВТОМАТИЧНОГО КЕРУВАННЯ, ПД-РЕГУЛЯТОР.

ABSTRACT

The explanatory note to the qualification paper "Object tracking system for a quadcopter" contains 84 pages, 56 illustrations, and 37 sources.

The relevance of the topic lies in the need to equip quadcopters with object tracking systems in the context of the growing popularity of quadcopters in the agricultural and security industries.

The object of research is a quadcopter-based object tracking system.

The subject of the research is the methods and algorithms of computer vision and machine learning, as well as the control system used to implement the object tracking function on a quadcopter.

The aim of the work is to research and develop algorithms for an object tracking system for a quadcopter to ensure accurate automatic tracking of moving objects.

The research methods include developing an algorithm for detecting and tracking objects, tuning the PD controllers for precise control of the quadcopter's motion, and experimental studies in the laboratory to evaluate the effectiveness of the developed system.

Keywords: COMPUTER VISION, MACHINE LEARNING, AUTOMATIC CONTROL SYSTEM, PD-CONTROLLER.

CONTENTS

LIST OF ABBREVIATIONS

Ryze – name of the Chinese startup;

DJI – name of the Chinese manufacturer of multicopters;

Tello – name of the quadcopter;

UAV – Unmanned Aerial Vehicle;

IMU – Inertial Measurement Unit;

GPS – Global Positioning System;

Parrot – name of the French manufacturer of multicopters;

Parrot AR.Drone – name of the quadcopter;

MEMS – micro-electromechanical system;

FAA – Federal Aviation Administration;

LiDAR – Light Detection and Ranging;

VPS – Vision Positioning System;

VPU – Vision Processing Unit;

UDP – User Datagram Protocol

PID – Proportional-Integral-Derivative controller;

CNN – Convolutional Neural Network;

FCN – Fully Convolutional Network;

AABB – Axis-aligned bounding box;

LQR – Linear Quadratic regulator;

SMC – Sliding Mode Control;

MPC – Model Predictive Control.

INTRODUCTION

In the 21st century quadcopters are gaining popularity every year and become an indispensable tool in various spheres of activity, from aerial photography, to search and rescue victims of various natural or man-made disasters. The system of recognizing and following objects is the key in such tasks. It also ensures the efficiency and autonomy of quadcopters.

The relevance of the qualification work "Object tracking system for quadcopter" is due to the wide demand for systems of this type. Such systems can be used in agriculture to accompany the movement of livestock, in film production - to shoot dynamic scenes in open spaces, where the rental of a specialized helicopter is very costly, and in closed spaces, where a compact drone with a good camera is suitable. It can also be used in security and rescue operations, where timely detection of a person plays a crucial role.

The object of study in this paper is the Tello programmable quadcopter from Ryze in collaboration with DJI. The object tracking system will be implemented on it.

The aim of the qualification work is to develop an object tracking system for a quadcopter that will be able to recognize, classify and track various objects in real time, providing fast and accurate object tracking. In this work, it was decided to choose a human face as an example of a tracked object. In the process of work will be considered various methods and approaches in the field of computer vision, machine learning, as well as automatic control systems for the realization of the task.

In order to achieve the set goal, the following tasks will be realized in the paper:

1. Analyzing the technical characteristics of the object of study;
2. Study of popular methods in the field of computer vision and automatic control theory;
3. Design and implementation of object detection and quadcopter control system;
4. Laboratory testing and analysis of results.

SECTION 1
THEORY OF QUADCOPTERS

1.1. Problem statement

According to the importance and practical value of quadcopters, the main purpose of the qualification work is to develop a system of automatic control of the aircraft using computer vision technologies for the fulfillment of various tasks in a fully autonomous mode. To accomplish this objective it is necessary:

- To study how this type of aircraft appeared, the principle of its operation, for what tasks is suitable and what advantages and disadvantages has in comparison with other types of aircraft.
- To investigate the methods and approaches to the realization of automatic control systems. It is also necessary to understand the principles of computer vision algorithms for the realization of object tracking system.
- To develop a system of automatic control of quadcopter using special software and various mathematical methods. The result of the development of such a system should be a quadcopter capable of moving in 3 planes without human intervention.
- Conduct appropriate flight tests and analyze the developed control system.

1.2. The evolution of quadrotor-type drones

The advancement of microprocessor technology and computer science has led to a significant breakthrough in aviation, enabling the creation of a novel category of unmanned aerial vehicles (UAV) – quadcopters [1]. The quadcopter represent a burgeoning rotorcraft design for UAV applications.

This aircraft comprises four rotors in total, with two sets of counter-rotating, fixed-pitch blades positioned at the four corners of the vehicle. The compact size and reduced inertia of quadcopters enable the implementation of a notably simplified flight control system, significantly enhancing the practicality of small quadcopters in this context.

The initial reference to quadcopters dates back to the early 20th century. Louis Charles Breguet pioneered the creation of the first quadrotor in 1907 (see Fig.1.2.1) [2].

In the 1920s, Etienne Oehmichen experimented with rotorcraft designs, including the Oehmichen No. 2 (see Fig.1.2.2). This design featured four two-blade rotors and eight propellers, all driven by a single engine. The rotor blade angles could be adjusted, with five propellers providing lateral stabilization, one for steering at the nose, and the remaining pair for forward propulsion.[3-4]

Charles Richet and Dr. George de Botezat's work in 1956 introduced propellers for controlling quadrotors in roll, pitch, and yaw angles. Further enhancements in weight, battery life, and density, alongside breakthroughs in processors and cost-effective sensors, have significantly propelled quadrotor research forward [5].

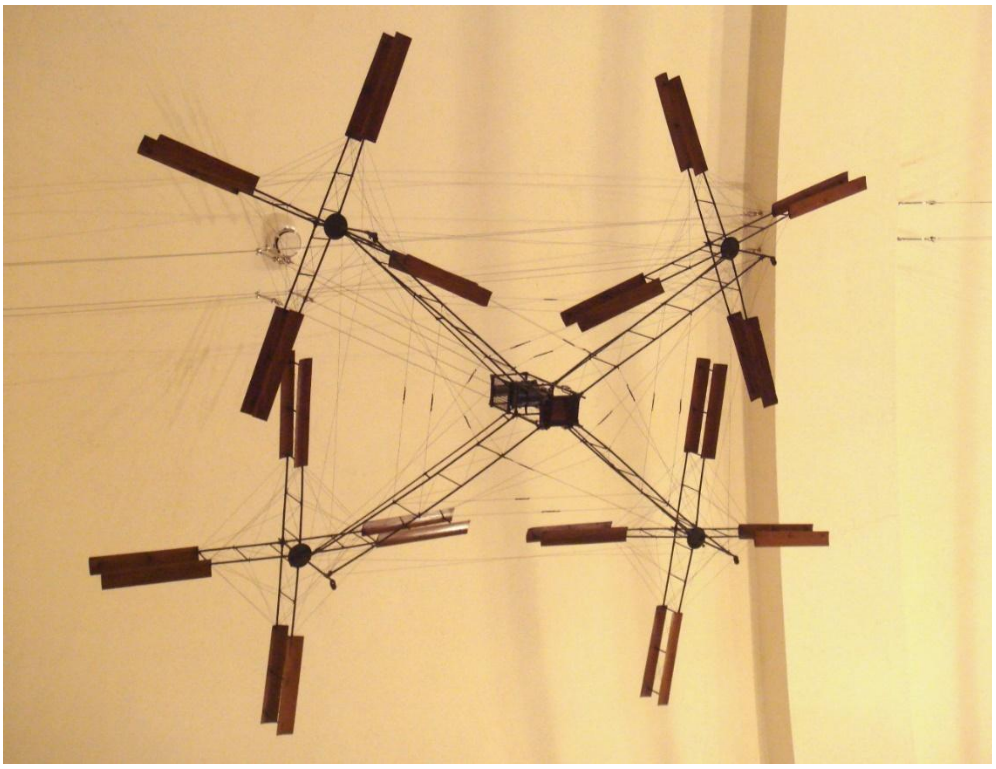


Fig.1.2.1. Model of the Bréguet-Richet Gyroplane No.1

Не удается отобразить рисунок. Возможно, рисунок поврежден или недостаточно памяти для его открытия. Перезагрузите компьютер, а затем снова откройте файл. Если вместо рисунка все еще отображается красный крестик, попробуйте удалить рисунок и вставить его заново.

Fig.1.2.2. Oehmichen No. 2 two-blade quadrotor

Initially, helicopter development overshadowed quadcopters, yet recent technological advancements have redirected attention and resources towards quadcopter research. Unlike helicopters, which rely on tail rotors to counterbalance torque from a single main rotor, quadcopters offer superior control, reduced power consumption, and lower production costs [6].

In the late 50's the US army had a request to create a new type of aircraft for reconnaissance missions. It had to be light, maneuverable, easy to operate and not expensive to produce. In 1958 the Aerophysics Development Corporation, a subsidiary of Curtiss-Wright, presented its prototype of "Flying Jeep" called VZ-7 to the Army [7]. This vehicle was operated by a single pilot and had good performance, but was not accepted for armament and put into mass production, because it did not satisfy the army standards.

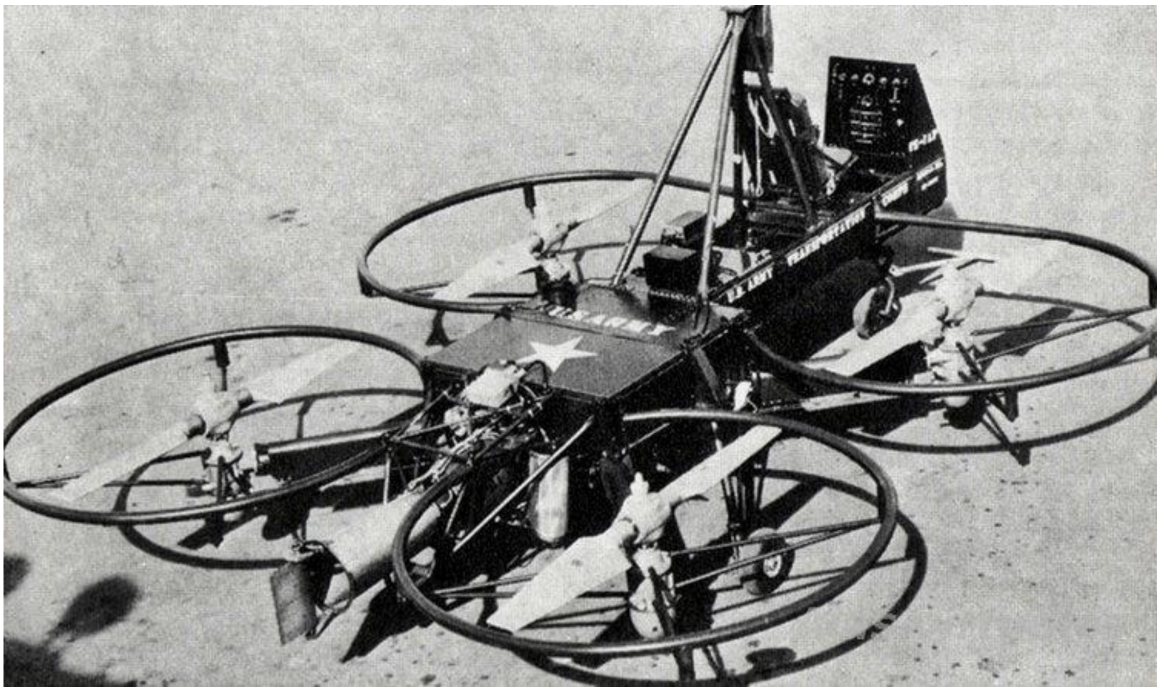


Fig.1.2.3. Prototype of VZ-7 “Flying Jeep”

During the initial decades of the 21st century, the quadcopter configuration has gained prominence as a preferred choice for miniature UAVs or drones. The demand for aircraft endowed with enhanced maneuvering capabilities and hovering proficiency has stimulated a surge in quadcopter-related research endeavors. The quadcopter design, characterized by four rotors, offers a balance between simplicity of construction and exceptional reliability and maneuverability. Ongoing research endeavors aim to augment the capabilities of quadcopters through advancements in multi-vehicle communication, environmental exploration, and maneuvering techniques. The convergence of these evolving attributes holds the potential for enabling quadcopters to execute sophisticated autonomous missions that presently remain beyond the scope of other aerial platforms [3].

Between the years 2005 and 2010, notable advancements in electronics facilitated the development of cost-effective lightweight flight controllers, inertial measurement units (IMUs) utilizing accelerometers, global positioning systems (GPS), and cameras. Consequently, the quadcopter configuration gained popularity for small UAVs. Featuring miniature size and maneuverability, these quadcopters are capable of flying both indoors and outdoors.

One prominent exemplar among the early small-scale quadrocopters is the Parrot AR.Drone, developed by the eponymous French company Parrot (Fig.1.1.4). The AR Drone has been engineered using lightweight and durable materials. Its primary framework consists of carbon-fiber tubes and fiber-reinforced plastic components, while the hulls are fabricated using injected expanded polypropylene[8].

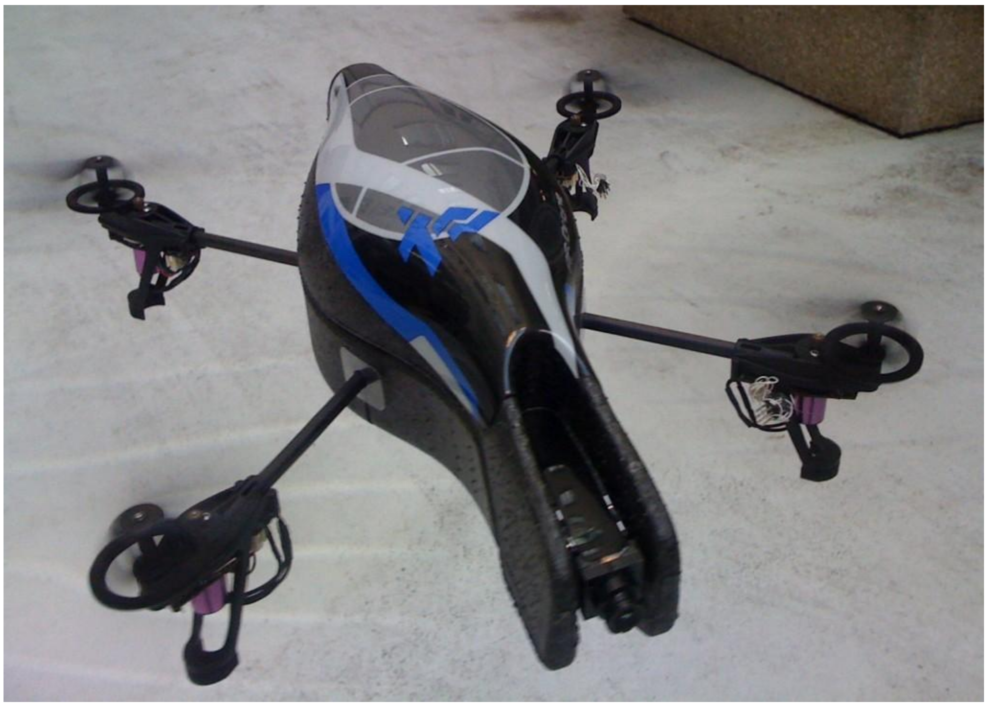


Fig.1.2.4. Parot AR drone

This and all subsequent quadcopters of this type are equipped with the following plants and sensors:

- Engines and propellers that create the lifting force for flight.
- IMU based on MEMS gyroscopes and accelerometers used to control quadcopter in flight
- Frontal camera which used for visual navigation in space
- Vertical camera with ultrasound sensor for navigation and vertical stabilization

1.3. Applications of quadcopters

Due to ability to hover in the same place, which quadcopters inherited from helicopters, small size, electric motors that create enough thrust to lift not only the frame itself, but also additional equipment, as well as a relatively small price per flight hour has made quadcopters very popular in various commercial fields and their popularity is only growing every day.

In 2015, the Federal Aviation Administration (FAA) [9] conducted a study in which it examined the use of quadcopters in commercial sectors and identified the top industries that actively take advantage of the benefits of this type of aircraft. The results are shown in the diagram below (Fig.1.3.1).

The following areas of application are mainly distinguished [10].

- **Agriculture:** The use of quadcopters offers manufacturers the opportunity to optimize production by increasing efficiency and reducing physical work. Quadcopters can save a lot of time in planting seeds, agricultural research, predicting crop yields, fertilizing crops and monitoring livestock.
- **Photography and cinema:** Quadcopters with a professional camera allow you to shoot various video clips, commercials and movies from a bird's eye view.
- **Surveying and mapping:** Quadcopters which are equipped with cameras and LiDAR sensors are also widely used for terrain mapping and geodetic surveying for land development and infrastructure planning.
- **Real estate:** Aerial photography will help buyers to better understand the features of the layout, design of real estate, as well as help to assess the boundaries and neighborhoods of the purchased property.
- **Logistics and delivery:** In the logistics industry quadcopters are used to deliver food, parcels, or other commodities. They are preferred for transportation of emergency or frequently sent small packages, as well as for delivery to hard-to-reach areas.

- **Construction:** In the construction industry, aerial drones are used when architects need accurate information about the terrain, existing buildings nearby and other terrain features. As well as providing real-time imagery, the quadcopters help monitor the construction process, helping project managers to clearly assess potential problems and ensure that the project is completed on schedule.

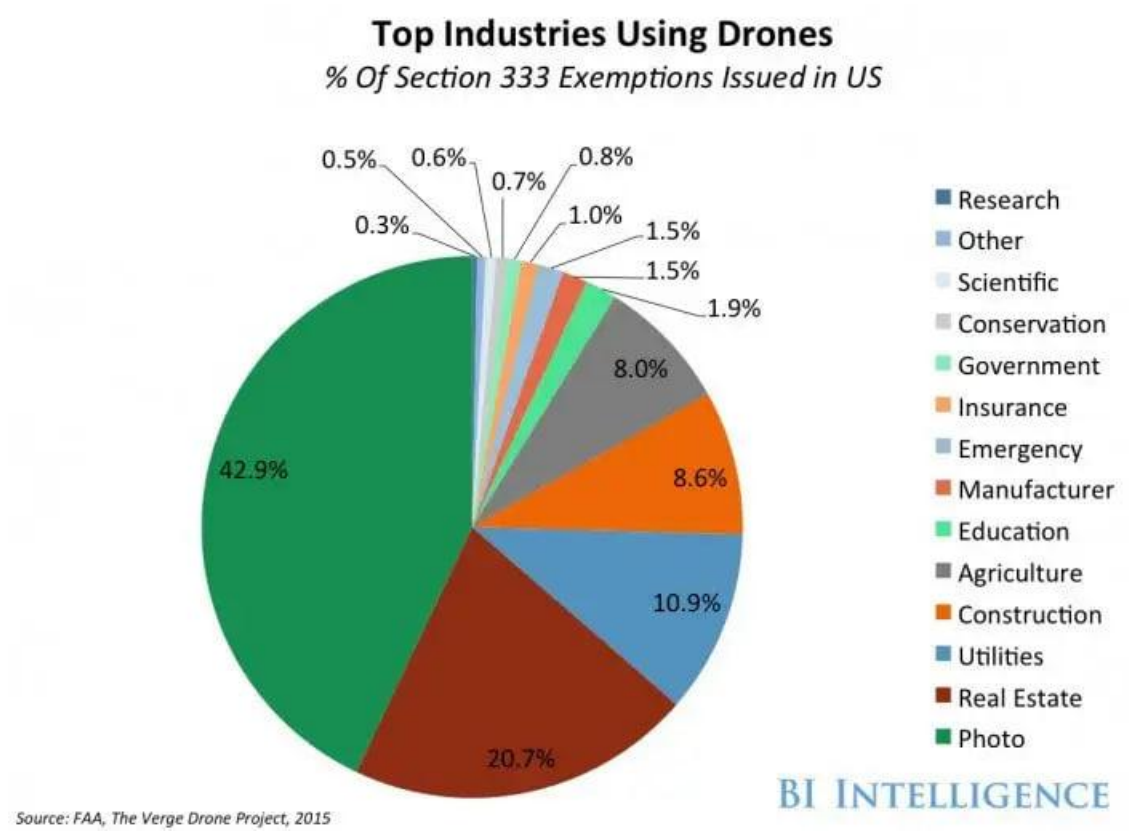


Fig.1.3.1. FAA research shows that 42.9% of drones were used in photo industry in 2015

1.4. Quadcopter work principle

A quadcopter, characterized by its four motors, plays a pivotal role in altering the pitch, roll, and yaw angles through the thrust exerted by each motor. Understanding this relationship is crucial, considering the quadcopter's underactuated nature, where control is exerted over six degrees of freedom (roll, pitch, yaw, and translational motion along the x, y, and z axes) using only four inputs. Notably, the translational movements along the x and y axes are interrelated, dependent on the craft's attitude concerning the remaining degrees of freedom. Strategies for addressing and controlling the underactuated aspects of quadcopters are essential when devising control methodologies. Furthermore, quadcopters typically feature four motors, with two rotating clockwise and the other two counter-clockwise [5]. Precise translation and rotation along the three axes are achieved through meticulous manipulation of the rotational velocities of the individual rotors. The variation in lift among these rotors dictates the orientation of the aircraft concerning the roll and pitch angles, facilitating translational motion [11]. Diagram on Fig.1.4.1 show the basic concept of movement of quadcopter along pitch, roll and yaw axes.

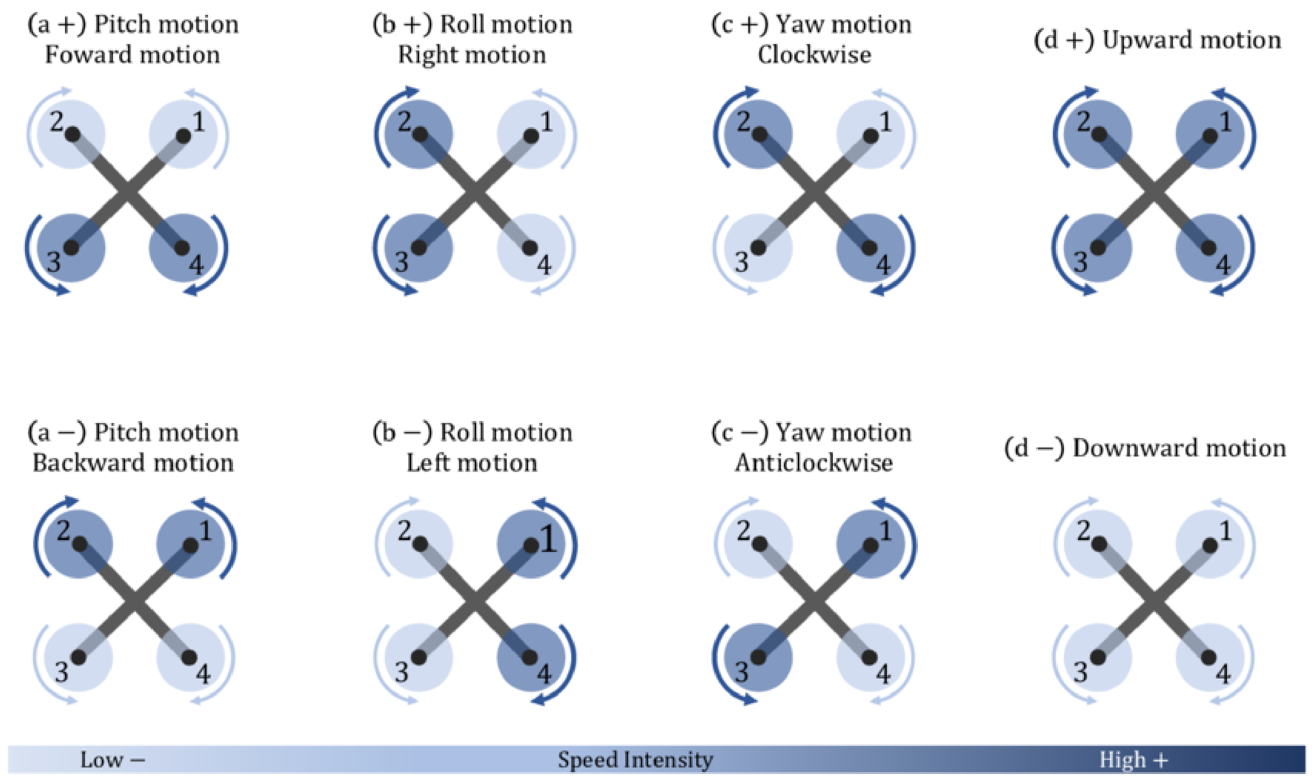


Fig.1.4.1. Motion concept scheme of quadcopter

The quadrocopter, like any other aircraft, is controlled by the above-mentioned mechanisms of rotation in space, such as roll, pitch and yaw, which in essence is the rotation of the body around the longitudinal axis, lateral axis and vertical axis respectively. Visual description of aircraft axis is presented in Fig.1.4.2.

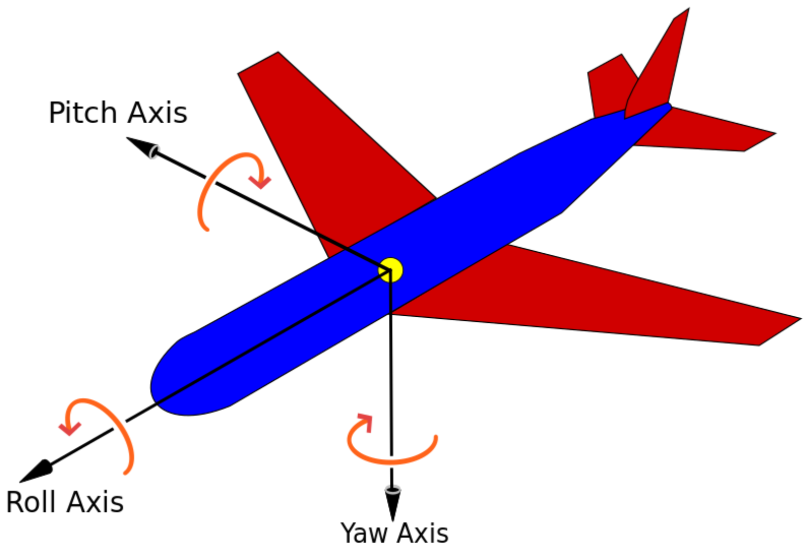


Fig.1.4.2. Axis of aircraft

These rotations manage the altitude of the quadcopter. To monitor altitude, a two-coordinate system is used: the body frame, attached to the quadcopter's center of gravity, and the earth frame, fixed to the ground. The difference in angles between these frames defines how the quadcopter behaves in space. The attitude system is determined by rotating the body frame around the earth frame's z-axis (yaw angle), followed by rotation around the y-axis (pitch angle), and then around the x-axis (roll angle) [5].

1.5. Object of study overview

The subject of the study was chosen the Tello quadcopter from the Chinese quadcopter manufacturer DJI in cooperation with Ryze Tech [13].

DJI Tello is a lightweight and small-sized entry-level quadcopter designed for both entertainment and educational purposes. This quadcopter is suitable for beginner drone operators without experience in flying such aircrafts, as it has intuitive control via smartphone application.

The specifications of the quadcopter are presented in Fig.1.5.1.

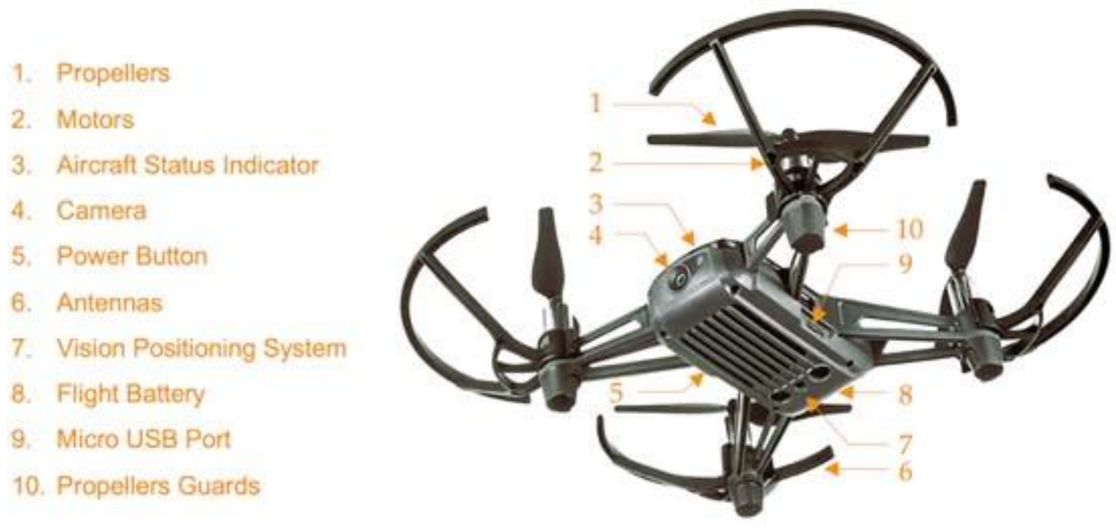


Fig.1.5.1. Basic components of Tello quadcopter

Key features of DJI Tello:

- **Lightweight and Portable**: Weighing only 80 grams
- **5-megapixel high definition camera** with electronic image stabilization
- **Vision Positioning System (VPS)** onboard which allows stable flight in areas without GPS signals. VPS consist of downward-facing camera and infrared sensor. Working together they capture images of the surface and measure distance to it
- **4 antennas** allow connection to any device that supports Wi-Fi UDP protocol
- **Intel Movidius Myriad 2 VPU (Vision Processing Unit)** for onboard image processing and computer vision tasks
- **Programable with Python:** This feature allows the user to expand the quadcopter's capabilities by adding artificial intelligence and developing a control system for automated flight
- **Features propeller protection**, making the quadcopter safe for indoor use

Disadvantages of DJI Tello:

- **Limited operational range:** The maximum flight range is limited to 100 meters. It also has a altitude limitation of a maximum 30 meters
- **Limited operational time:** Due to the small battery, the quadcopter can stay in the air for about 10 minutes
- **Limited payload capacity:** The quadcopter's low weight and low-powered motors make it impossible to suspend a heavy load. The quadcopter's payload is limited to 35 grams of additional weight
- **Limited wind resistance:** The quadcopter is very sensitive to wind. Even a weak airflow can disrupt the quadcopter's stabilization, which in effect limits the quadcopter's area of application to enclosed indoor areas or completely windless weather
- **Lack of GPS module:** This feature makes it impossible to navigate the quadcopter at night, as the vision positioning system works well only in the presence of good illumination

1.6. Problem statement and mathematical model of the object

In order to develop a quality control system for quadcopter, it is necessary to calculate its mathematical dynamic model. This model helps to understand the dynamics of quadcopter flight and the forces and torques applied. Also, it allows to determine the inputs, outputs, state variables, and disturbances in the system.

Based on the model calculated by R, Benotsmane and J, Vásárhelyi, the rotation matrix, state vector, inputs outputs of the system and state space matrices are taken from the paper. For more details see the article [12].

The dynamic structure of the Tello quadcopter within its body and inertial frame, where respective angular velocities, torques, and forces created by the four rotors are presented, is illustrated in the Fig.1.6.1.

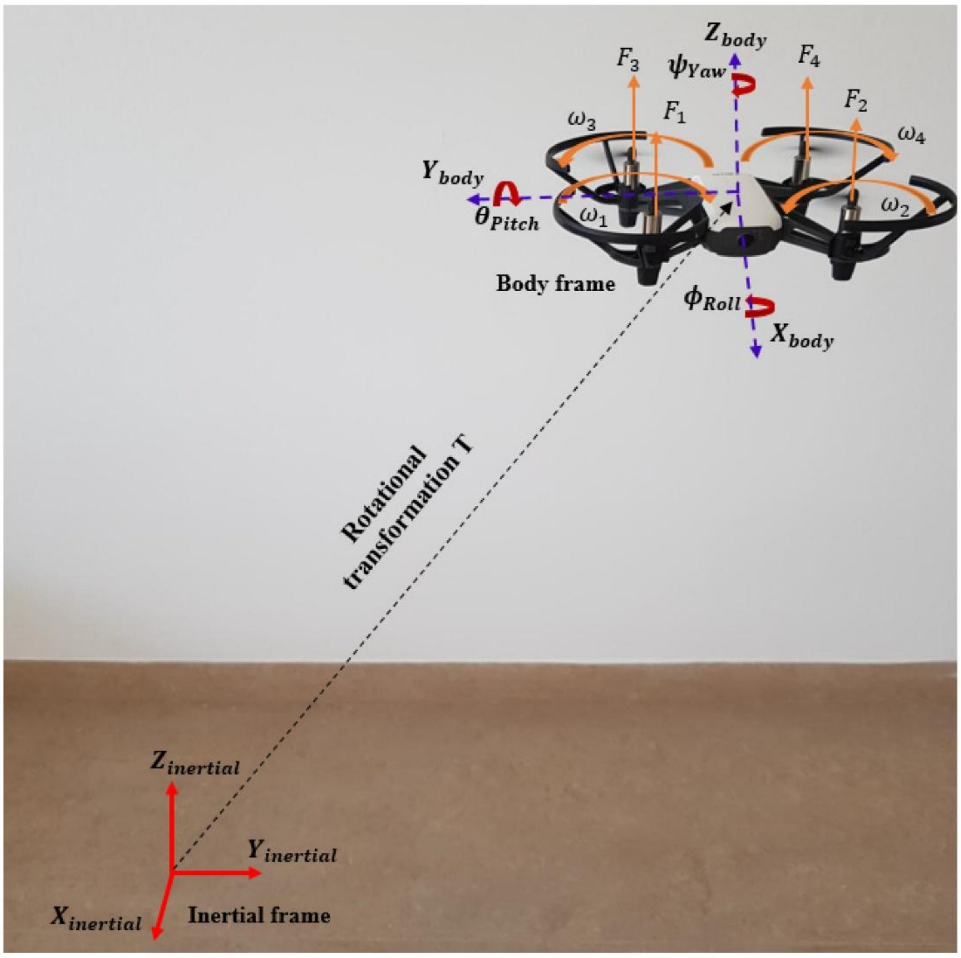


Fig.1.6.1. Connection between quadcopter's Body frame and Inertial frame

The rotation matrix R describing rotation from the body reference frame to the inertial reference frame is presented in equation below.

The following state-space vector is used to describe the physical system of the quadcopter:

Where:

– physical location within the Earth fixed system;

– angles that are related to roll, pitch and yaw respectively;

– quad velocities along X,Y,Z axes;

– angular velocities.

In the state vector, only the first 6 states are measured. They form the output vector y:

Also, the model has 4 inputs, which form the following input vector U:

;

where: F_t is a translatory force of moving body and M_x, M_y, M_z - inertial moments around X-, Y- and Z-axes respectively.

A state-space model obtained after linearization of the nonlinear model is also attached.

;

where A is a system matrix of ( ), B is the input matrix of ( ), C is the output matrix of ( ), and D is the feed forward matrix of ( ). State space matrices are presented below.

The main parameters specified for the dynamic model are summarized in Table 1.6.1 [12].

J_{xx} denotes the inertia tensor of a symmetric solid body around the X-axis, J_{yy} is the inertia tensor of this body around the Y-axis, J_{zz} is the inertia tensor around the Z-axis, and g is the coefficient of free-fall acceleration on the Earth's surface.

;

;

;

.

Table 1.6.1

Main parameters of dynamic model

Parameters	Value
Mass	
	9.81

1.7. Conclusion

In the first section we set the goals of our work and the tasks to be solved to achieve them. We have understood the design and operating principle of quadcopters, which will help us in further designing an effective automatic control system in the next sections of the paper.

Also, we have defined a mathematical model of this apparatus, which is a critical element in the cycle of synthesis of the necessary regulators and control system as a whole.

SECTION 2
OBJECT TRACKING CONCEPTS

To realize the task of object detection and tracking on quadcopter, it should be divided into two subtasks: The task of object detection using computer vision technologies and the task of object tracking, which is realized by adding a Proportional-Integral-Derivative controller (PID) to the quadcopter's control loop. Let us first consider the concept of object detection using artificial intelligence technologies.

2.1. Theory of Computer Vision

Computer vision is a technology based on machine learning and neural network algorithms that allow computers to analyze, process and manipulate information from digital images, video or any other source of visual information. The basis of computer vision is the ability to process the resulting image at the pixel level, for example, classifying the object by the level of brightness of pixels [14].

The principle of computer vision can be divided into several stages, namely: image acquisition, pre-processing, feature extraction, segmentation, object detection, object recognition, classification and post processing. This algorithm is presented in the form of a block diagram in Fig 2.1.1. Let us consider each step of this algorithm in detail.

1) Image Acquisition. At this stage, the light from the light source enters the camera lens where it passes through a set of moving lenses, focuses and hits the photosensitive sensor. The matrix is a device that perceives the image projected onto it. The matrix itself consists of millions of photosensitive elements called pixels. The more pixels on the matrix, the clearer the image can be obtained.

Since semiconductor photodetectors are approximately equally sensitive to all colors of the visible spectrum, to perceive a color image, each photodetector is covered with a light filter of one of the primary colors: red, green, blue (RGB color model).

This matrix of color filters is called Bayer filter [15]. After the passage of light photons through the photosensitive elements, they are converted into an electrical signal, and then the signal enters the analog-to-digital converter at the output of which we have a digital picture that is stored in internal or external storage. At the end of all these manipulations comes the stage of image pre-processing.

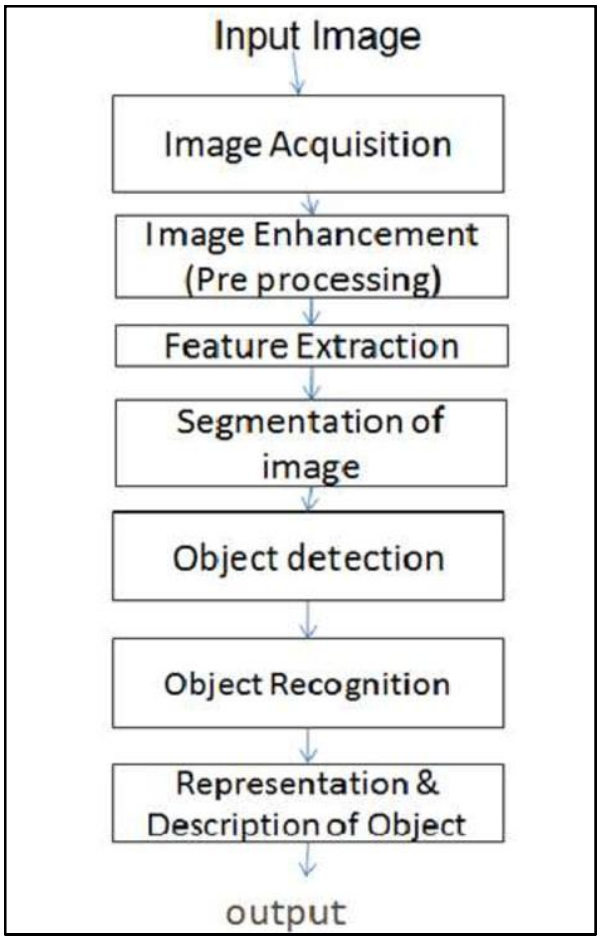


Fig 2.1.1. Main steps of computer vision algorithm

2) Image Pre-processing [16]: Before feeding the collected image set to the input of the computer vision model, it must be pre-processed to ensure that the algorithm works with the highest possible accuracy. This step involves several techniques that help to enrich the image data and extract useful information while reducing the computational complexity of downstream processing. The main techniques that are used most often include (Fig.2.1.2):

- **Image Resizing:** Image resizing allows you to reduce the size of the dataset and fit all photos to the same size. This is necessary to reduce the load on the hardware on which the computer vision model will be run.
- **Noise Reduction:** Noise removal is required for more accurate further image processing (e.g. edge detection). One of the popular methods is median filtering. The main idea of the median filter run through the image from pixel to pixel and change the pixel values to the median value of two neighboring pixels. This way, if there are a lot of broken pixels in the image, they are smoothed out, but the image may lose a little bit of sharpness in the process.
- **Contrast Enhancement:** Contrast enhancement should be used on low contrast images to accurately identify image features. For contrast enhancement such techniques as Contrast Stretching, Histogram Equalization, Adaptive Histogram Equalization, Contrast-Limited Adaptive Histogram Equalization are used.
- **Normalization** Image normalization is a process that changes pixel intensity values between 0 and 1. Normalization is necessary to process images with poor contrast, for example, due to glare.
- **Grayscale conversion:** Color images are often converted to grayscale images, which simplifies the data and reduces computational requirements. This is especially useful when color does not play a significant role in the analysis.

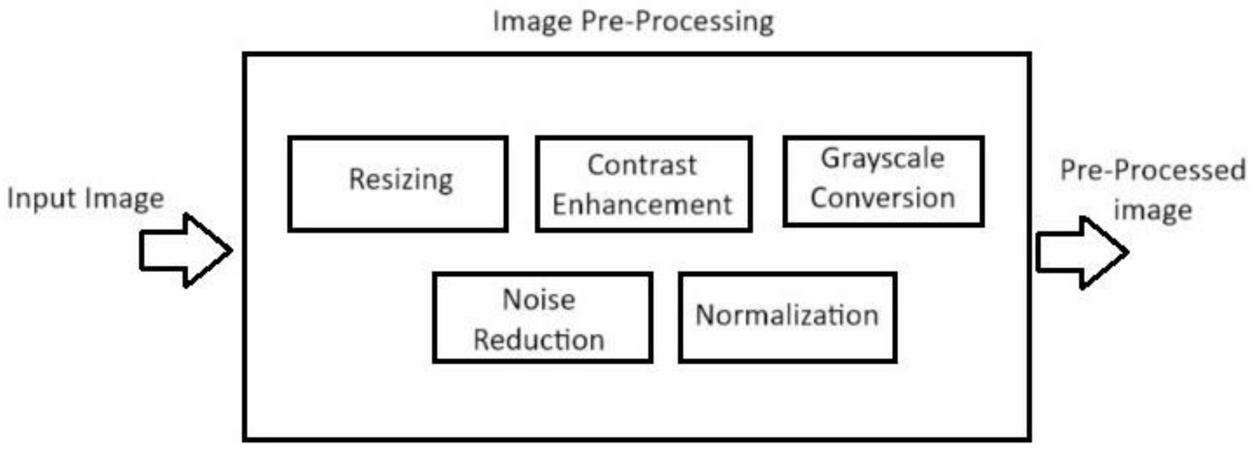


Fig.2.1.2 Image Pre-Processing techniques

3) Feature Extraction: Feature extraction is the third step in the computer vision algorithm, which involves identifying and capturing distinctive visual patterns or structures from images. These extracted features serve as the basis for subsequent analysis such as object detection, recognition or classification. Feature extraction include widespread techniques [17-18]:

- Edge Detection: Identifying abrupt intensity changes representing object boundaries
- Corner Detection: Detecting points with significant intensity gradients in multiple directions
- Blob Detection: Identifying regions with homogeneous intensity values.
- Texture Analysis: Characterizing spatial patterns using statistical or frequency-based methods
- Feature Descriptors: Capturing local appearance or shape information around key points
- Deep Learning Features: Learning hierarchical representations directly from raw image data using convolutional neural networks (CNN)

4) Segmentation [19]: Image segmentation, it is a key technique in computer vision, which is used to accomplish most applications. The concept of this technique is to divide the image into a set of multiple segments where each pixel in the image is mapped to an object. The major task is to clarify and customize the visual representation of an image, making more meaningful and easier to understand and analyze. The scope of this technology includes object detection, face recognition, traffic control systems, automated vehicle control systems and much more. There are several commonly used types of segmentation:

Threshold based segmentation [19-20] represents a fundamental and widely employed methodology in image processing. This approach entails establishing a threshold parameter to partition pixels into discrete regions according to their respective intensity or color attributes. Such a method proves efficacious particularly in scenarios where target objects exhibit discernible disparities in intensities or colors relative to the surrounding background.

Edge-based segmentation [19;21] techniques based on edges concentrate on identifying and leveraging the boundaries of objects within an image through edge detection (Fig.2.1.3). These edges denote substantial alterations in pixel intensities and can be discerned through various algorithms including Sobel, Canny, and Laplacian of Gaussian.

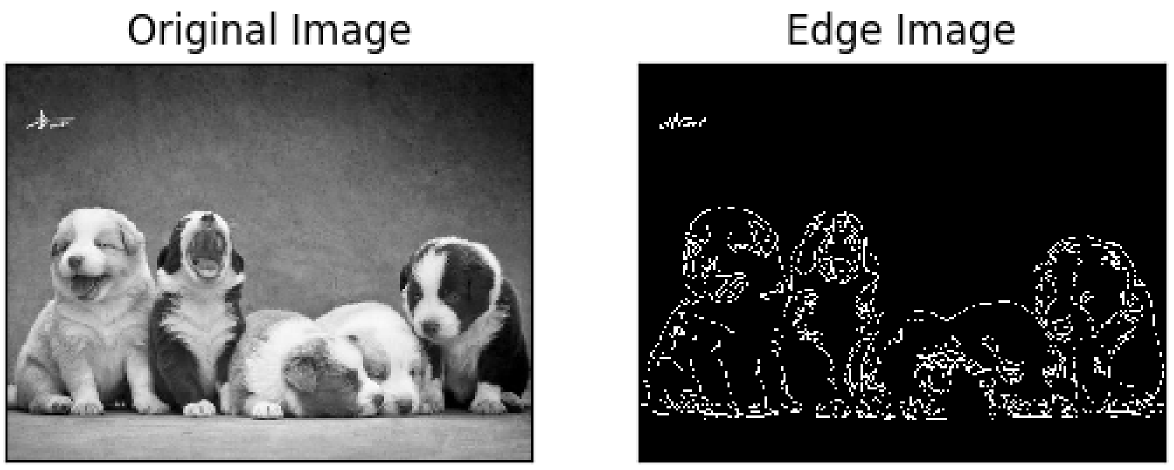


Fig.2.1.3 Illustration of edge-based segmentation

Following the detection of edges, region-based segmentation methodologies become applicable. Notably, Active Contour Models, commonly referred to as Snakes, are extensively employed for the precise demarcation of object boundaries by iteratively modifying a curve to conform to the identified edges. While edge-based segmentation facilitates precise object localization, it may exhibit sensitivity to noise and necessitates clearly defined edges.

Clustering-based segmentation [19] techniques based on clustering endeavor to cluster comparable pixels or regions together according to their inherent features. Prominent clustering algorithms utilized in image segmentation encompass K-means and Mean Shift. Color-based clustering strategies amalgamate pixels sharing akin color values, whereas texture-based approaches center on the aggregation of pixels exhibiting similar texture attributes. Clustering-based segmentation demonstrates resilience to fluctuations in lighting conditions and possesses the capacity to manage intricate scenes featuring multiple objects. Nevertheless, it necessitates meticulous feature curation, is susceptible to noise interference, and can potentially yield instances of over-segmentation or under-segmentation (Fig.2.1.4).

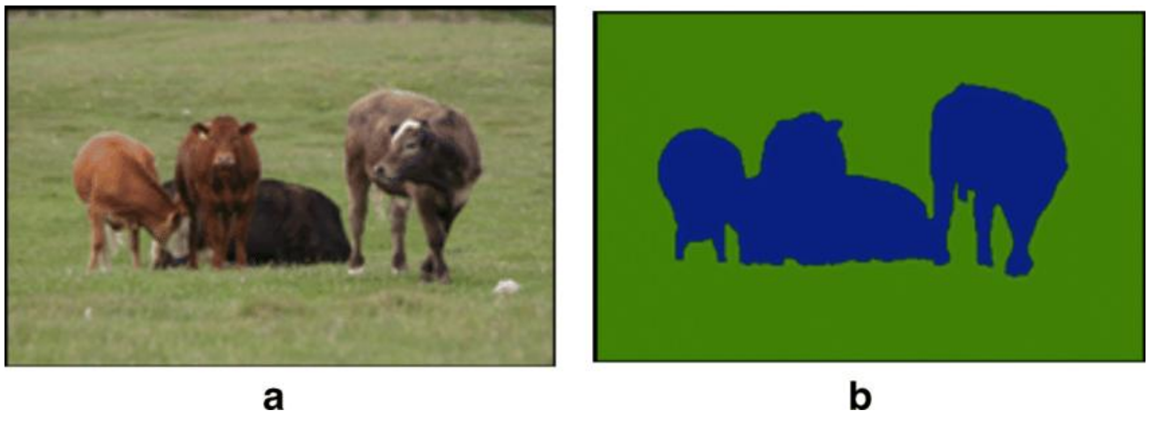


Fig.2.1.4. Example of Clustering-based segmentation.

a) – Original image; b) – Segmented image (the image is divided into two separate segments: the object and the background)

Region-based segmentation [19;22] is used for tasks where it is necessary to highlight an irregularly shaped object (for example, in medicine, where you need to separate a tumor from healthy tissue in an image). The main principle in region-based segmentation is to divide images into regions, taking into account their local and global features. The most popular methods in region segmentation are Watershed [23] and Graph Cuts [24]. In watershed segmentation, the pixel intensities are viewed as a topographic surface, and the process simulates flooding to delineate regions. In contrast, Graph Cuts employs graph theory principles to minimize an energy function that considers pixel similarities and dissimilarities (Fig.2.1.5).

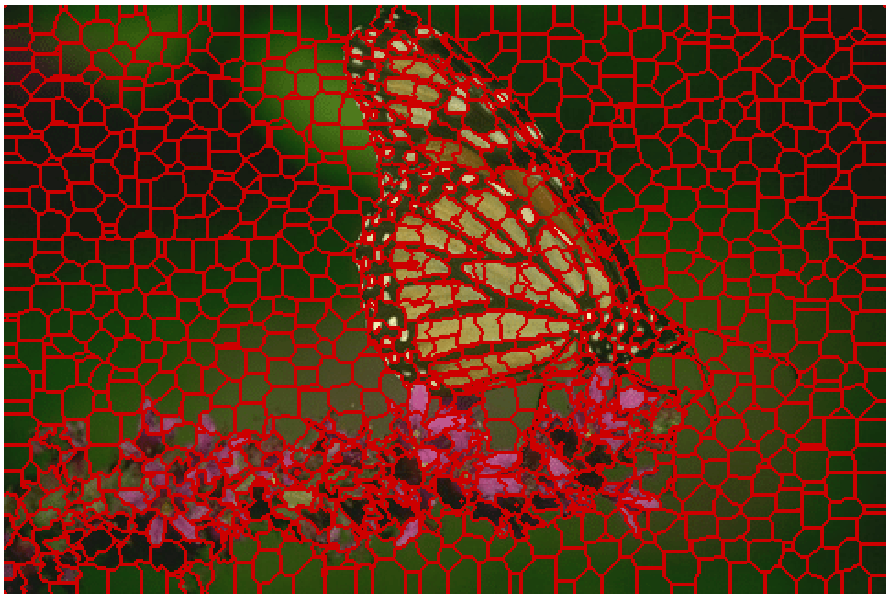


Fig.2.1.5. Illustration of region-based segmentation

Semantic segmentation [19] is the process of dividing an image into regions, where each region corresponds to a semantic class from a predefined list. The appearance of these classes is learned using labeled images. Distinguishing itself from alternative segmentation methodologies, it directs attention towards labeling at the object level as opposed to individual regions. Deep learning methodologies, notably Fully Convolutional Networks (FCNs), have made notable strides in semantic segmentation. Illustration of this type of segmentation is presented in Fig.2.1.6.

Instance Segmentation [19] Instance segmentation is essentially an extended semantic segmentation technique that not only separates pixels into classes, but also separates individual object instances within the class itself. For example, if there are several objects of different classes in an image, semantic segmentation will simply select the region where the objects of each class are located, while instance segmentation will select each element from a class as a separate object. This type of segmentation is widely used in computer vision tasks such as object detection, object counting and instance-level analysis. The method is used in robotics and applications requiring precise localization of an object. The main disadvantages of this approach are high computational costs of hardware.

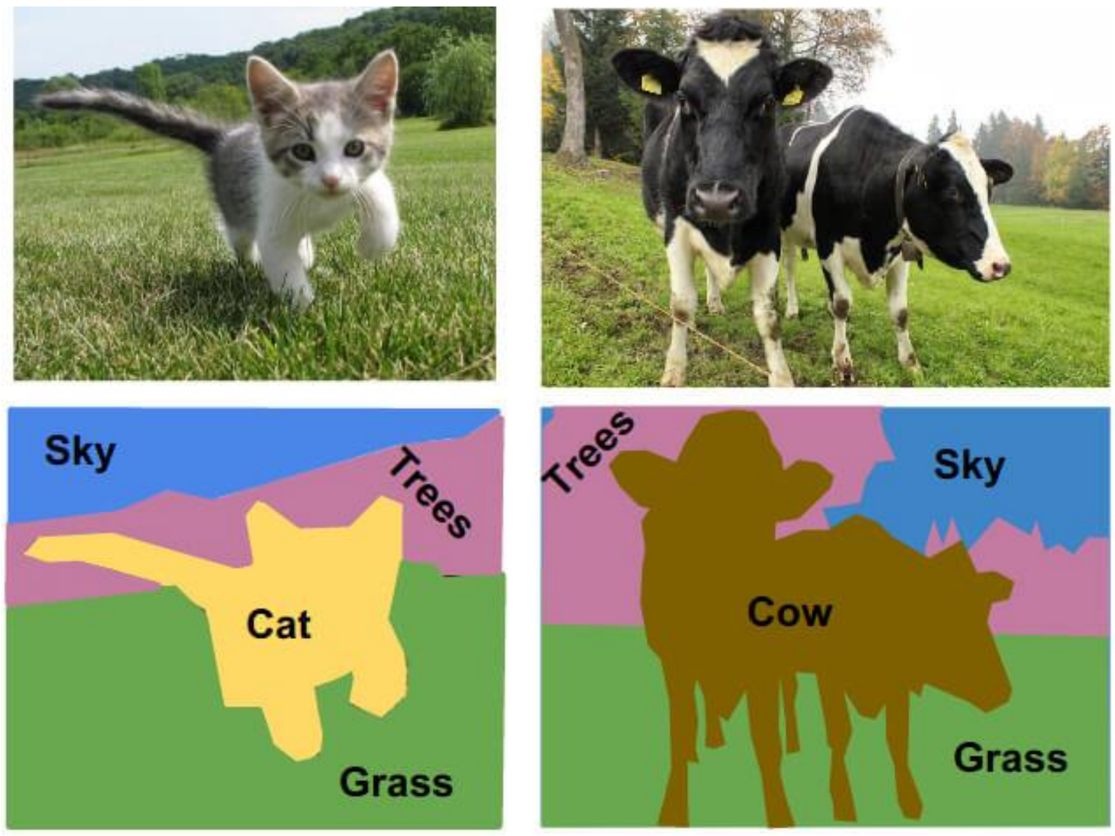


Fig.2.1.6. Semantic segmentation divides pixels by classes from a pre-sorted dataset

Panoptic segmentation [19] represents a synthesis of semantic and instance segmentation, offering a holistic comprehension of both "scene" classes (for example ocean, sky or road) and "thing" classes (like people, animals or cars) within an image. It assigns a distinct label to each pixel while also discerning between individual object instances. Hybrid methodologies that amalgamate deep learning-based semantic segmentation with instance segmentation techniques are frequently employed for achieving panoptic segmentation. Panoptic segmentation is mainly used in augmented reality applications (video games, flight simulators), and in autopilot systems for cars (Fig.2.1.7). as it allows to obtain holistic information about what is happening in space at both object and scene levels.

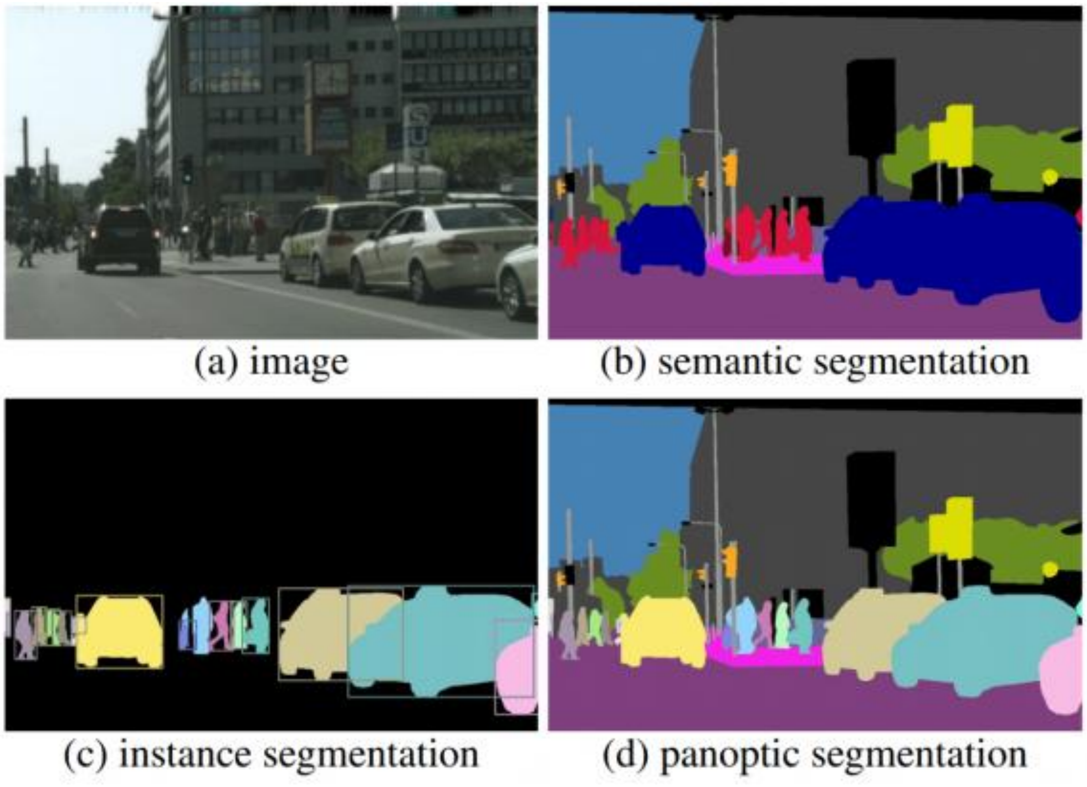

(a) image
(b) semantic segmentation
(c) instance segmentation
(d) panoptic segmentation

Fig.2.1.7. Panoptic segmentation and its comparison with other types of segmentation

5) Object detection: Object detection constitutes a computational technology within the realm of computer vision and image processing, addressing the identification of occurrences of semantic objects belonging to specific classes (e.g., humans, buildings, cars) within digital images and videos. There are many different object detection algorithms, but we will consider the face detection algorithm developed by Paul Viola and Michael Jones in "Rapid Object Detection using a Boosted Cascade of Simple Features" [25], as this algorithm is the basis of the subject of our study.

The Viola-Jones algorithm was developed in 2001 as a simple, fast and reasonably accurate method for object detection (mainly for face detection). Despite the fact that modern algorithms are more accurate and faster, the Viola-Jones method is still relevant in a number of tasks, due to its simplicity and the ability to run on hardware with limited resources.

In order to fully understand how the algorithm works, it is necessary to understand such concepts as Haar-like Features, Integral Images, the AdaBoost Algorithm, and the Cascade Classifier to create a system for object detection.

Haar-like Features [25-26]

Initially, to calculate the features we worked only with the intensity of the image, i.e. with RGB values in each pixel of the whole image. This approach was very resource-intensive. Then in 2001, based on Papageorgiou's publication, Paul Viola and Michael Jones decided to use Haar wavelets to develop their own algorithm for object detection and called this concept Haar-like features (Fig.2.1.8).

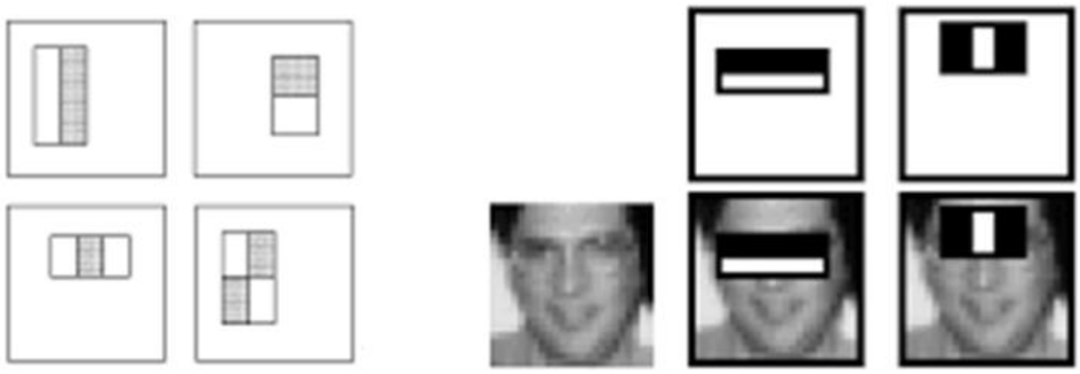


Fig.2.1.8. Haar-like features representation

Haar-like feature explores adjacent rectangular regions located at a specific position in the detection window by moving pixel-by-pixel across the image from the top-left edge to the bottom-right edge. Its main function is to quickly and accurately detect edges, lines, or sharp variations in pixel intensity in the image, which may indicate the edges of an object (a face, for example).

The principle of operation is described as follows: Suppose we have an image with different pixel intensities from 0 to 1, which means light and dark pixels respectively (Fig. 2.1.9). We also have a so-called Haar kernel - a rectangle divided into two parts, with all light pixels on the left and dark pixels on the right (Fig.2.1.10). Then this rectangle is superimposed on the image, forming a zone of feature search. Thus, each pixel of the image will be analyzed to search for features. For example, an edge, line or any structure in the image where there is a sharp change in intensity. To detect the edge of the image, we need to perform simple calculations: count the sum of all pixels in the dark area of the Haar feature. Do the same for the light area of the feature and count their difference. If the difference is closer to 1, it means that we have found an edge.

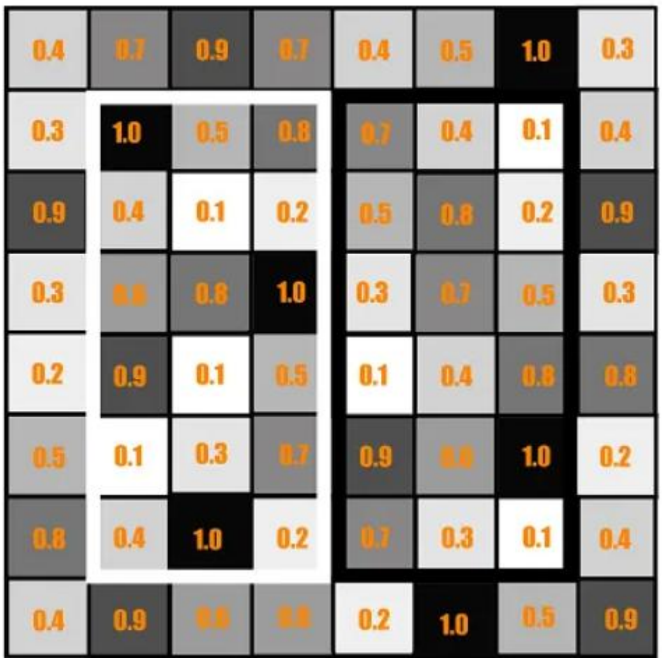
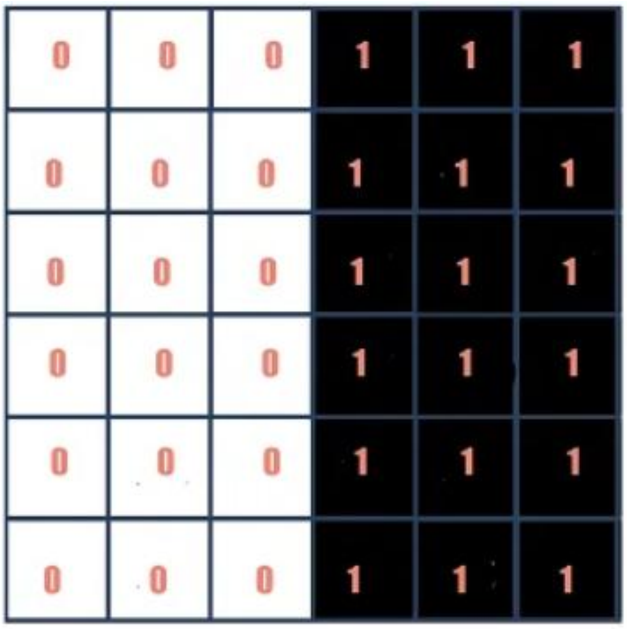


Fig.2.1.9. Original image representation with pixels intensity from light (0.0) to dark (1.0)



0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1

Fig.2.1.10. Haar-like feature kernel

A mathematical representation is shown below

– –

where:

– dark pixels;

– number of dark pixels;

– light pixels;

– number of light pixels.

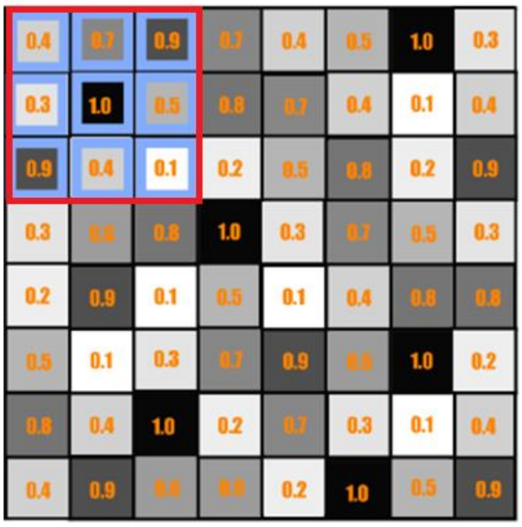
After calculations we have:

From results we can see that the value of Haar feature is far from 1, which means there is no edge in the image.

However, this algorithm has a significant drawback. Since the algorithm is executed from pixel to pixel, and in a real algorithm Haar features can have different sizes, this method turns out to be computationally expensive even for very powerful processors, since a large number of calculations are performed. To solve this problem, Viola and Jones proposed the concept of integral image.

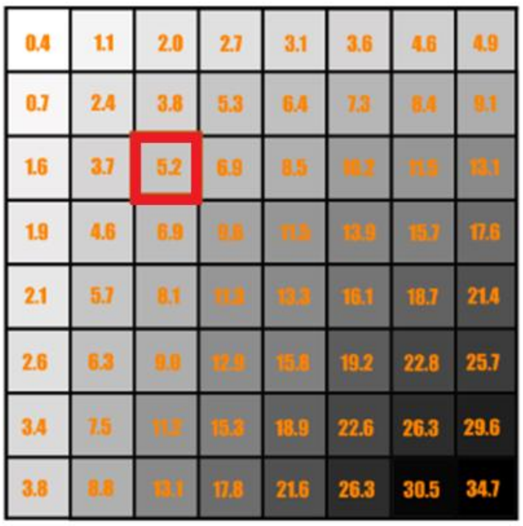
Integral Image [25-26]

This concept does the same thing, but much faster and saves CPU processing power for other important tasks. The algorithm is slightly different from the previous one: each next pixel of the integral image is calculated from the sum of all pixels to the left and above the selected one in the original image. The final pixel in the integral image shows the sum of all pixels in the original image. Illustration of the integral image is presented in Fig.2.1.11.



0.4	0.7	0.9	0.7	0.4	0.5	1.0	0.3
0.3	1.0	0.5	0.8	0.7	0.4	0.1	0.4
0.9	0.4	0.1	0.2	0.5	0.8	0.2	0.9
0.3	0.6	0.8	1.0	0.3	0.7	0.5	0.3
0.2	0.9	0.1	0.5	0.1	0.4	0.8	0.8
0.5	0.1	0.3	0.7	0.9	[illegible]	1.0	0.2
0.8	0.4	1.0	0.2	0.7	0.3	0.1	0.4
0.4	0.9	[illegible]	[illegible]	0.2	1.0	0.5	0.9

a) Original Image

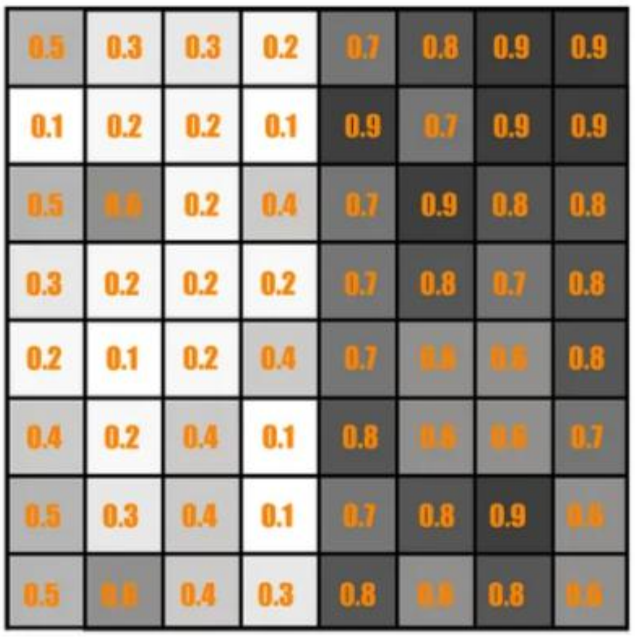


0.4	1.1	2.0	2.7	3.1	3.6	4.6	4.9
0.7	2.4	3.8	5.3	6.4	7.3	8.4	9.1
1.6	3.7	5.2	6.9	8.5	10.2	11.5	13.1
1.9	4.6	6.9	9.8	11.5	13.9	15.7	17.6
2.1	5.7	8.1	[illegible]	13.3	16.1	18.7	21.4
2.6	6.3	9.0	12.9	15.8	19.2	22.8	25.7
3.4	7.5	11.2	15.3	18.9	22.6	26.3	29.6
3.8	8.8	13.1	17.8	21.6	26.3	30.5	34.7

b) Integral Image

Fig.2.1.11. The process of transition from the original image to the integral image. Labeled pixel in the integral image is equal to the sum of labeled pixels in the original image.

Let us consider how the Haar value for the integral image is calculated. First, let us take the original image with a sharp increase in pixel intensity in the vertical direction and from left to right (Fig. 2.1.12), calculate the integral image from it and superimpose the Haar kernel shown above (Fig. 2.1.10).



0.5	0.3	0.3	0.2	0.7	0.8	0.9	0.9
0.1	0.2	0.2	0.1	0.9	0.7	0.9	0.9
0.5	0.6	0.2	0.4	0.7	0.9	0.8	0.8
0.3	0.2	0.2	0.2	0.7	0.8	0.7	0.8
0.2	0.1	0.2	0.4	0.7	0.6	0.6	0.8
0.4	0.2	0.4	0.1	0.8	0.6	0.6	0.7
0.5	0.3	0.4	0.1	0.7	0.8	0.9	0.6
0.5	0.6	0.4	0.3	0.8	0.6	0.8	0.6

Fig.2.1.12. Original image with sudden change of pixel intensities

To calculate the features of the light region, we need only 4 pixel values at the edges of the region of interest. After we calculate the value of the sum of pixels in the light region of the integrated image as follows:

Let us use formula (1) for the integral image shown in Fig. 2.1.13:

do the same for the dark area:

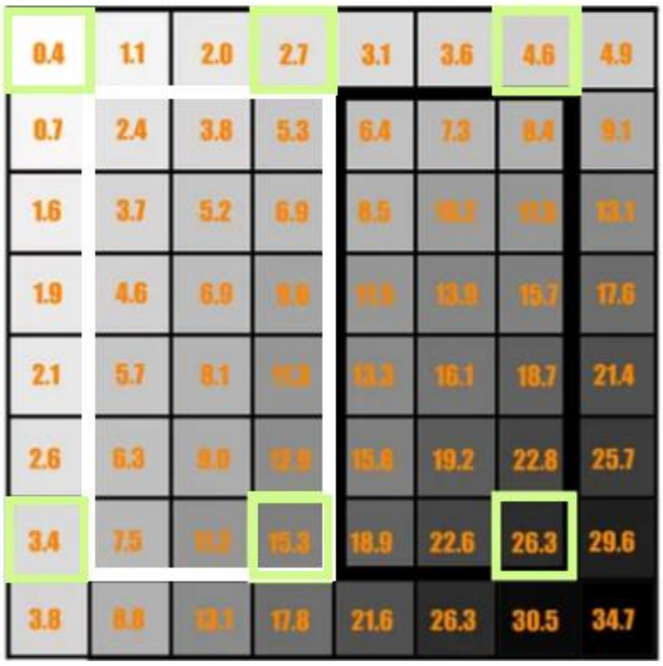


Fig.2.1.13. Integral image with calculated Haar values

At the end we calculate the Haar value:

Since the obtained result is closer to 1, we can assume that most likely an edge or a line has been seen in this image.

The AdaBoost Algorithm [25-27]

In order to properly train the object detection model, we need to have two sets of high-quality images with the object of interest and a set of images where this object is absent. The algorithm applies a set of Haar features to them and learns to predict the presence of the object in the image. However, in the process of work it turned out that most of the features were not suitable or gave irrelevant results for facial feature detection. Therefore, Viola and Jones decided to use an AdaBoost technique to select the most appropriate features and weed out the inappropriate ones.

The AdaBoost (Adaptive Boosting) algorithm is a machine learning technique aimed at selecting the optimal set of features from the available data. Its result is a strong classifier, which is a composition of weak classifiers, where each weak classifier is based on one of the features. To form a strong classifier, the algorithm must perform a number of iterations equal to the number of weak classifiers defined by the user. At each iteration, the algorithm calculates the classification error for each feature and selects the feature with the minimum error for the current iteration. By implementing this algorithm, it was possible to reduce the number of features from 180000 to 6000. A simplified illustration of this technique is shown in Fig.2.1.14.

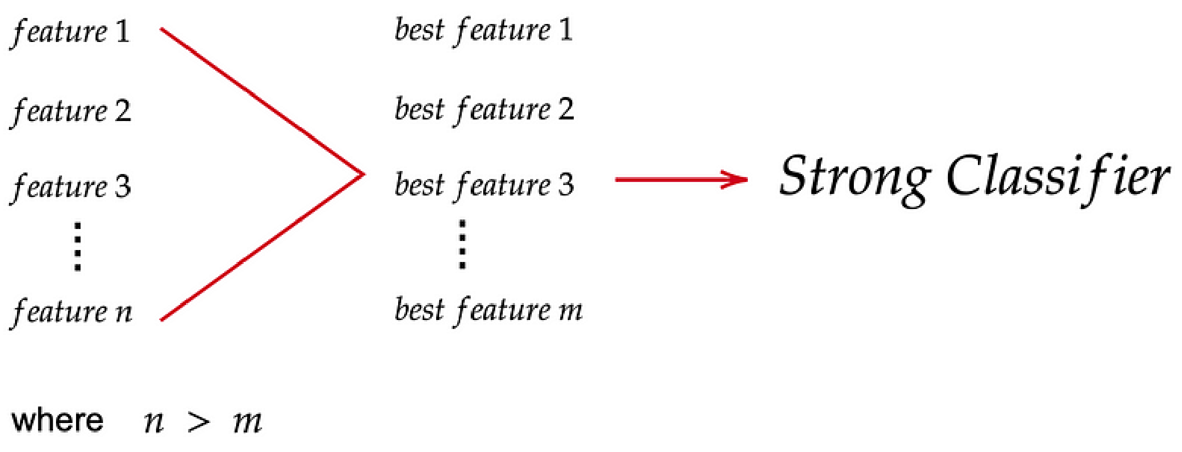


Fig.2.1.14. AdaBoost algorithm. The objective of employing the AdaBoost algorithm is to discern the best features from a pool of n features.

Cascade classifier [25-27]

A cascade classifier is a structure with multi-level classification that provides fast and accurate detection. Each level uses a strong classifier built using the AdaBoost algorithm. The number of weak classifiers in the strong classifier increases with each new stage. The data is evaluated sequentially and if the classifier produces a negative result at any stage, the data is immediately discarded. The next stage of applying new features does not start until the first stage produces all positive results (facial features). This method allows for the development of simpler classifiers and ensures that negative data is quickly weeded out, thus increasing the efficiency of positive data processing. Diagram of cascade classifier algorithm is presented in Fig.2.1.15.

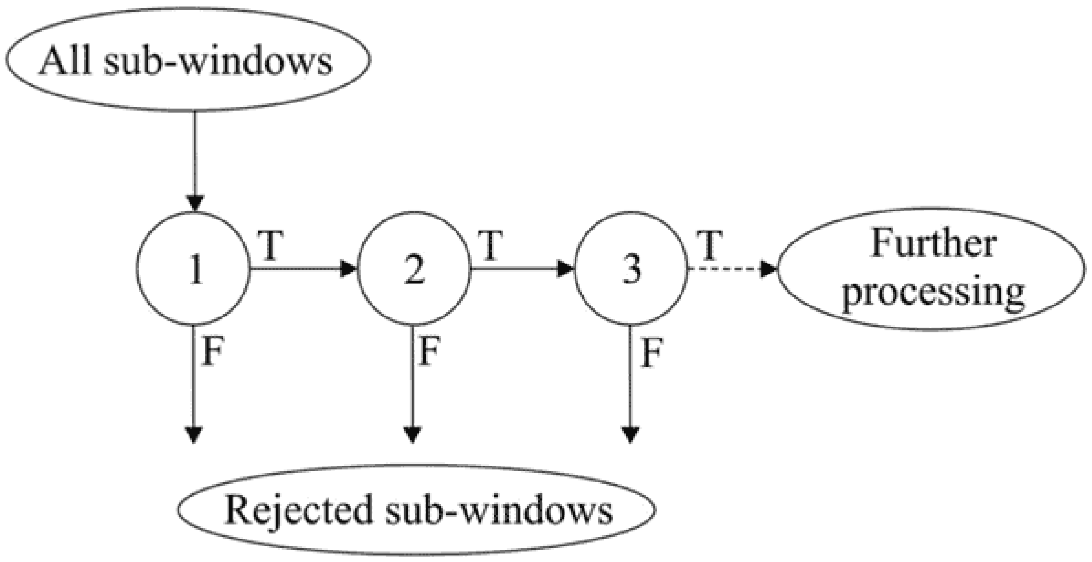


Fig.2.1.15. Representation of Viola Jones cascade classifier algorithm.

Viola-Jones Framework [25-27]

After familiarizing ourselves with the basic concepts, we can look at how the face detection algorithm proposed by Viola and Jones works. The algorithm can be divided into two main phases: Training and Application. The training phase in turn is divided into data preparation and cascade classifier construction. Let us describe the data preparation phase.

If we already have a training set of images including both positive (faces) and negative (non-faces) examples, the first step is to extract features from these images. Viola and Jones' recommendations indicate the use of 24 x 24 pixel images. This is because each type of Haar-like features can vary in size and position within this window. Next, we will use integral images, which will speed up the feature extraction process. The preparation process is illustrated in Fig. 2.1.16.

After that the stage of cascade classifier construction follows, the simplified algorithm of which is shown in the block diagram in Fig. 2.1.17.

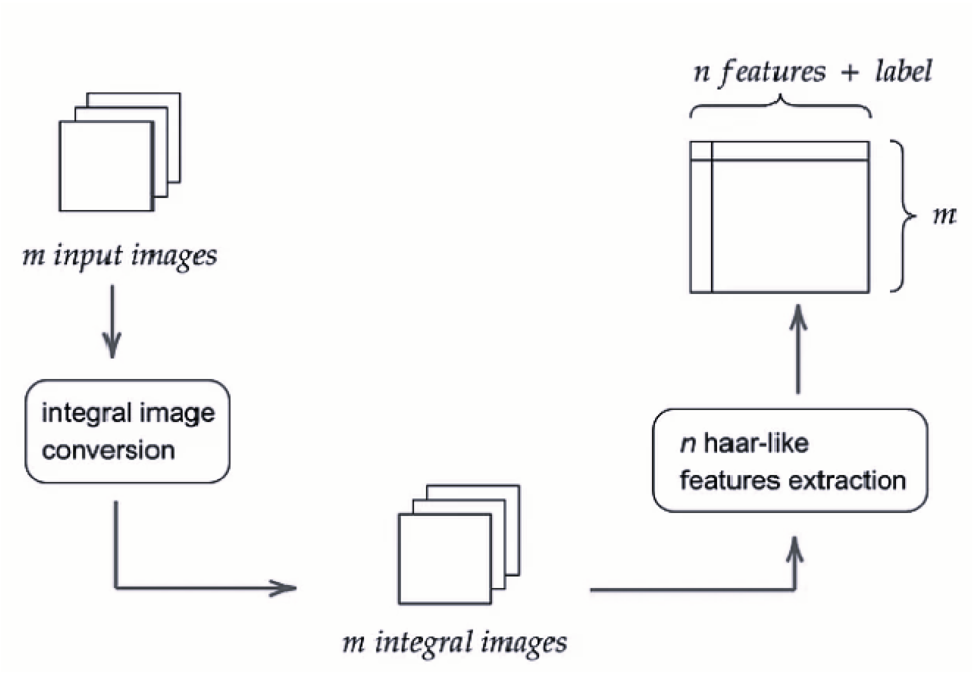


Fig.2.1.16. Preparation of Dataset

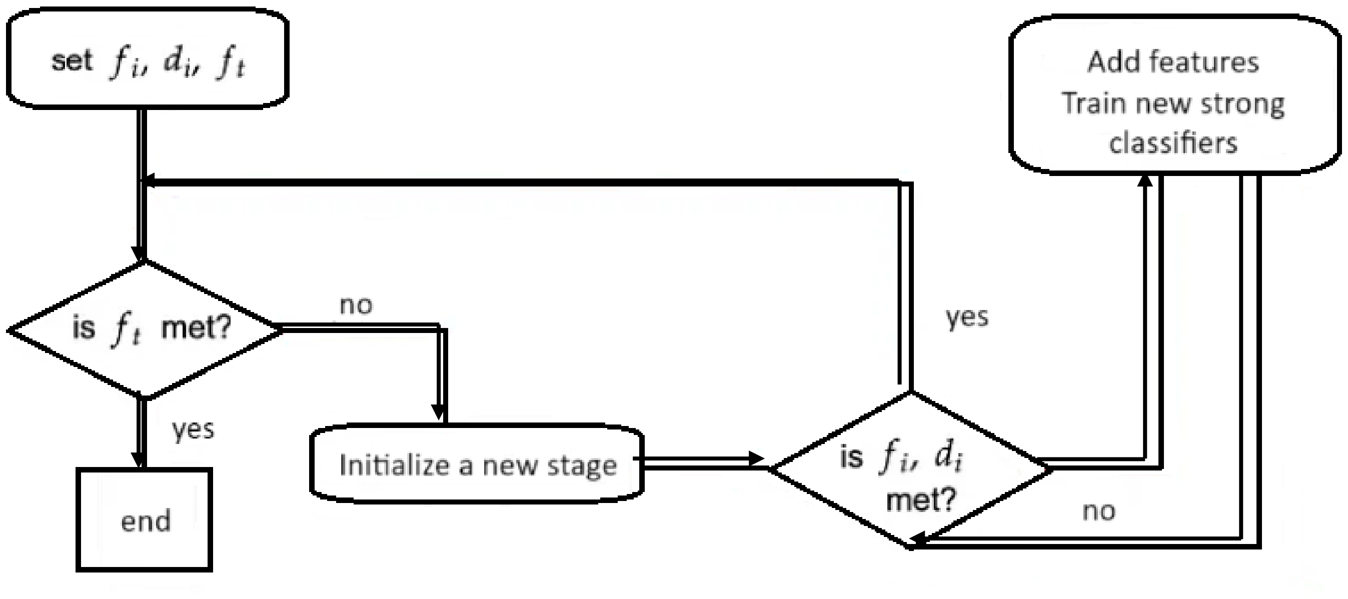


Fig.2.1.17. Cascade classifier with AdaBoost algorithm

where:

- maximum acceptable false positive rate per stage

- minimum acceptable true positive rate per stage

- target overall false positive rate

The face detection algorithm proceeds as follows: A sliding window approach is used on the selected image (windows with a set of features are passed in several steps through the image from left to right and from top to bottom). The window size coefficient and the sliding step are set by the user. For each subwindow, including the image, the framework resizes the subwindow image to a base size corresponding to the training data, converts it to an integral image, and passes it through the Cascading Classifier created during training. A face is detected if the subwindow passes through all stages in the Cascading Classifier. Once a face is detected, a bounding box is formed to highlight the face in the image for the user's visual perception. The output image is presented in Fig.2.1.18.

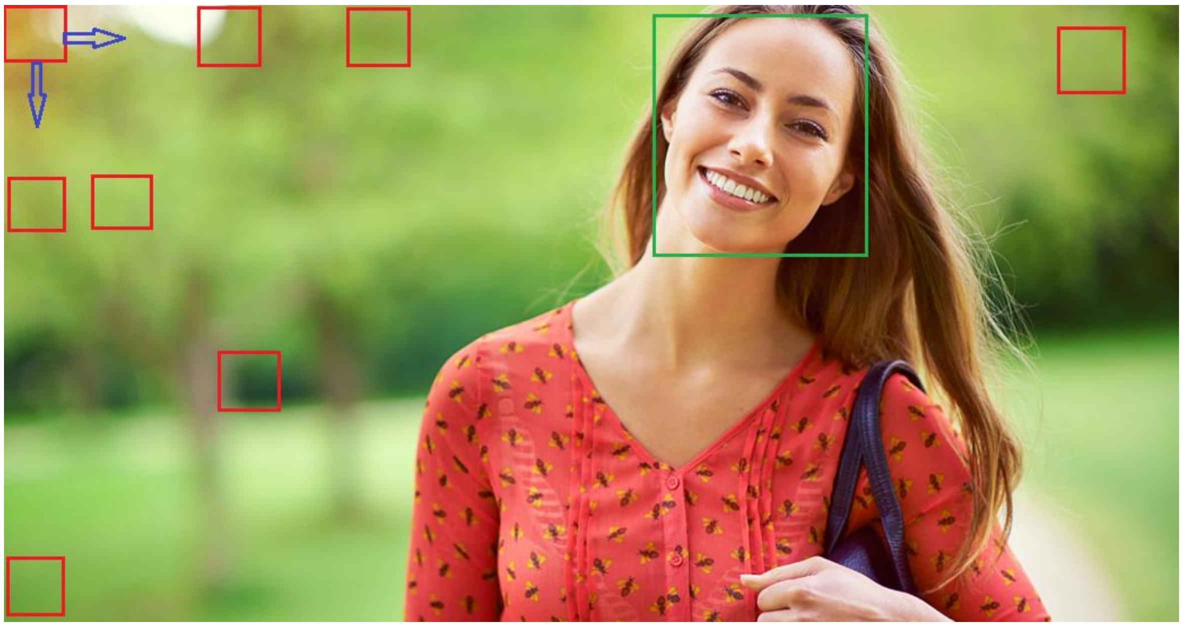


Fig.2.1.18. Sliding window detection process in object detection system. The arrows indicate the direction of movement of windows. Red "boxes" visualize the process of sliding windows. Green "box" shows the detected face

2.2. Types of controllers applicable to quadcopters and their design

Building a control system for a quadcopter is a very important step. Since the quadrocopter itself has an unstable aerodynamic scheme and will fall at the slightest disturbance, it is extremely important to provide it with precise controllability to keep it in flight.

There are many approaches to the realization of control systems and their choice depends on the control object, specifically set tasks that the control object must perform, as well as depends on the impact of disturbances of the environment in which the object is located.

Linear and non-linear control systems are used to control the quadcopter. Define each of them:

- Linear control systems governed by linear differential equations, where devices adhere to the superposition principle. A primary subset comprises linear time-invariant (LTI) systems featuring constant parameters.

Frequency domain techniques, including Laplace transform, Fourier transform, Z-transform, Bode plot, root locus, and Nyquist stability criterion, are employed to analyze such systems.

- Nonlinear control theory applies to real systems that are described by nonlinear differential equations and do not use the superposition principle. The mathematical methods for their solution are often more difficult and rigorous compared to linear systems. For example, nonlinear control systems can be solved using Poincaré maps and Lyapunov stability theory. However, when accuracy is not critical and results in the neighborhood of a stable point are of interest, it is often possible to linearize nonlinear systems. This is achieved by approximating the system with a linear model obtained by decomposing the nonlinear solution into a series and then using linear methods.

Possible approaches for realizing linear and non-linear control systems for quadcopter are presented in the table below [12].

Table 2.2.1

Quadcopter control system approaches

Type of controller	Linearity type	Stability and time response	Computation time
Proportional-Integral-Derivative (PID) controller	Linear	Low stability in higher disturbances, acceptable response time	Low
Linear quadratic regulator (LQR)	Linear	Good stability, slow response	Low
optimal control	Linear	High stability, medium response	Medium
Sliding mode	Nonlinear	Robust under disturbances	High

control (SMC)			
Adaptive control	Nonlinear	Good stability under perturbations	High
Model predictive control (MPC)	Linear and Nonlinear	No guarantee of stability, but high optimization for different constraints is achieved.	High

For the synthesis of the control system, the PID controller approach was chosen because it is simple to implement, has sufficient time response and requires low computational cost, and thus is well suited for implementation on microcontrollers.

2.3. PID controller structure and its applications

PID controller is a control device used in feedback control systems [28-30]. The basic principle of the regulator operation is to maintain the set level of the output signal and generate the corresponding control signal for the control object. For example, in the quadcopter control system, the PID controller adjusts the motor rotation speed to perform maneuvers in the air. When the operator moves the control stick on the joystick, the measured physical quantity (thrust level) is input to the controller. The controller, according to its algorithm, generates a control action that causes a change in the controlled variable. After the control signal is transmitted from the controller to the actuator, the feedback is returned to the sensor input. In a subsequent step, the controller again measures the controlled parameter and compares it with the set point, calculating the control error. The control error is calculated according to the formula:

where:

R(s) – reference signal;

Y(s) - output signal from plant.

After that the cycle is repeated. At each stage a new control action is generated, taking into account the control error. The structural diagram of the regulator is shown in Fig. 2.3.1.

The equation of output in the time domain is

The PID controller, as its name suggests, comprises three terms: proportional (K_P), integral (K_I), and derivative (K_D). Each term contributes uniquely to the control action.

Let look at each component separately and how they affect the object control system.

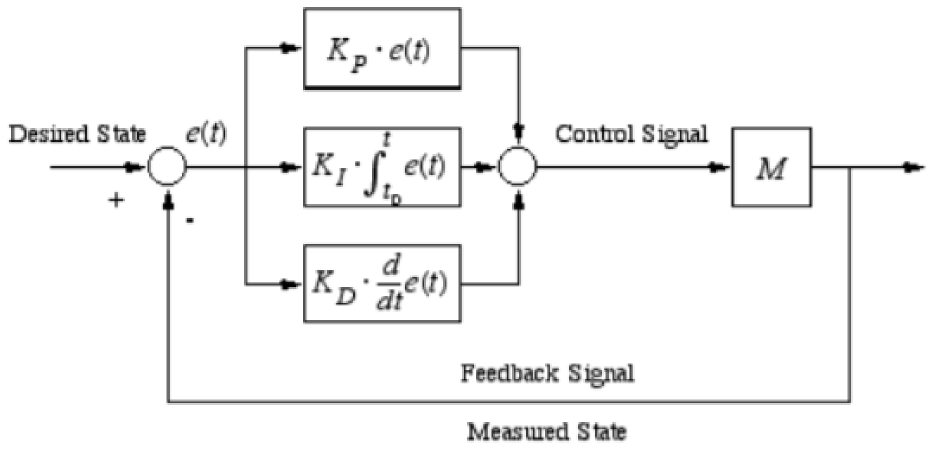


Fig.2.3.1. Structure of PID controller

The **proportional component** (P) [28-30] generates a control signal proportional to the current error, defined as the difference between the setpoint and the actual value of the variable. This component reduces the rise time of the correction signal, bringing the system output back to the set value due to the correction proportional to the magnitude of the error (see Fig.2.3.2). The larger the error, the greater the correction introduced by the proportional component.

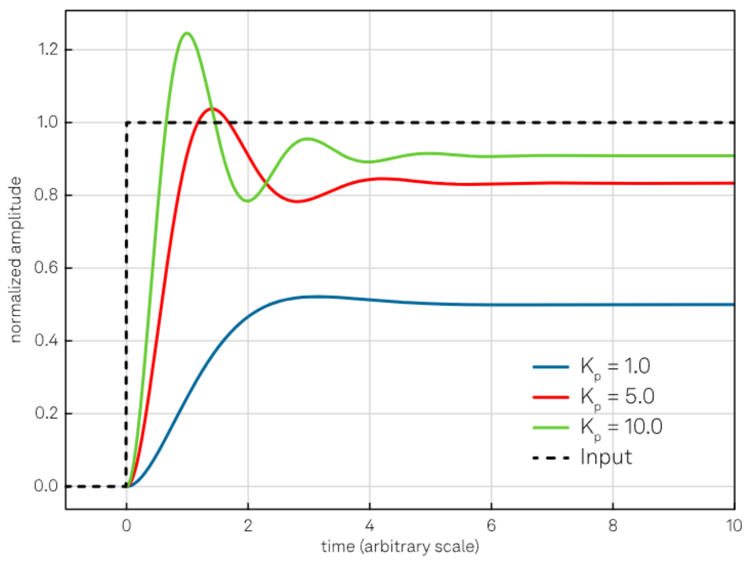


Fig.2.3.2. Effect of proportional regulation. The higher the Kp coefficient, the lower the rise time and the more oscillations in the output. In addition, the error will never approach zero.

A proportional controller is capable of controlling any stable system, but its effectiveness is limited and it cannot completely eliminate the steady-state error. This is because its frequency response does not allow it to achieve zero error at all frequencies.

The **integral component** (I) [28-30] applies a correction proportional to the integral of the error over time, i.e., the increment of the error. For example, if the error persists over time, the integral term continues to increase, resulting in an increase in the correction applied to the system output. Even in the presence of zero error in a given time interval, the integral term allows the controller to generate a non-zero control signal, which the proportional term cannot. This property allows the controller to generate a sufficiently accurate signal for the system to reach the set point (see Fig. 2.3.3).

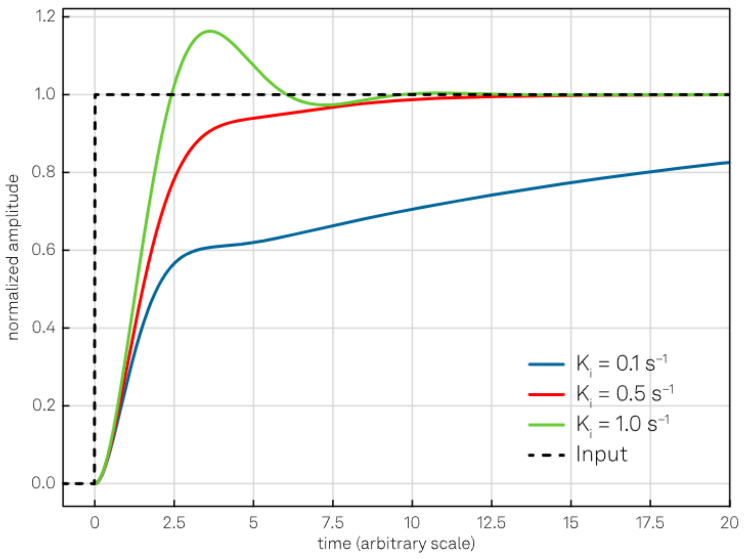


Fig.2.3.3. Integral regulation effect. Increasing the integral gain coefficient K_I decreases the response time of the system, but at the same time excessive gain can lead to overshoot of the system.

Differential component (D) [28-30] controls the variation of the error over time by implementing a correction which is proportional to the derivative of the error. This helps to minimize the rate of change of the error, improving the stability and responsiveness of the control system. The key objective is to predict changes in the error signal: if the error starts to increase, the derivative component seeks to compensate for it before the error grows large, which is typical of proportional action, or remains for a long time like in the integral action.. The main advantage of using the differential component is that it reduces overshoot and system setup time. On the other hand, an excessive Kd coefficient can lead to strong oscillations, which affects the stability of the system as a whole. The effect of the differential component on the system characteristics is shown in Fig. 2.3.4.

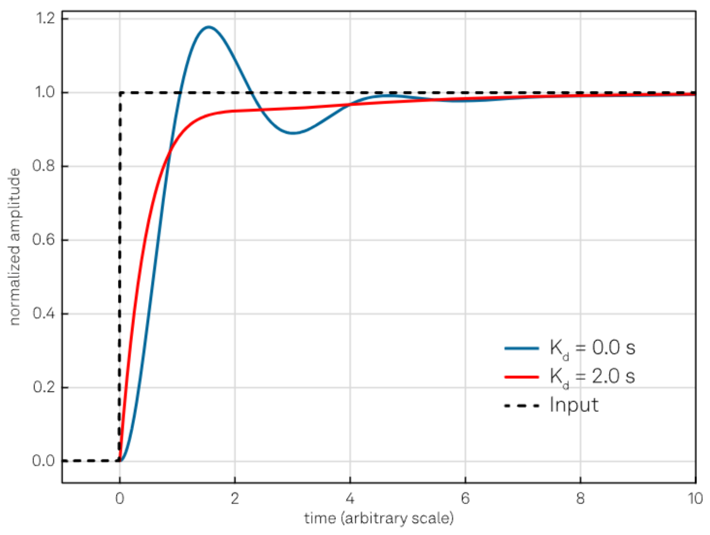


Fig.2.3.4. Effect of derivative action. The diagram shows that increasing the Kd coefficient changes the characteristic of the transient process: the overshoot is reduced and the settling time is decreased.

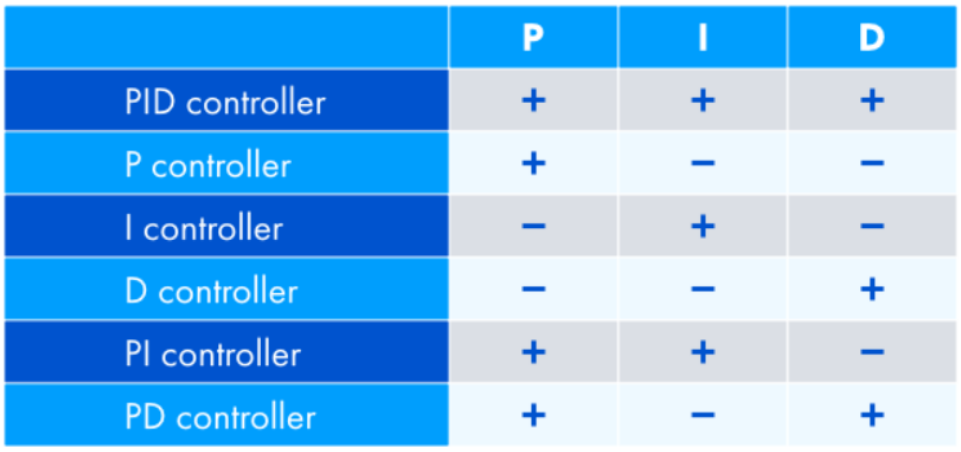
A comparative characterization of the effect of each component on the closed-loop control system is given in Table 2.2.2 [31].

Table.2.2.2

PID controller coefficients characteristics comparison

Control Response	Overshoot	Settling Time	Rise Time	Steady state error
K_p	Increase	Minor change	Decrease	Decrease
K_i	Increase	Increase	Decrease	Eliminate
K_d	Decrease	Decrease	Minor change	Not changing

Sometimes, in fairly simple systems, only one or two of the three control components of a PID controller may be required. In such cases, the unused components can be set to zero, thus creating various combinations of controllers (see Fig. 2.3.5) [32].



	P	I	D
PID controller	+	+	+
P controller	+	−	−
I controller	−	+	−
D controller	−	−	+
PI controller	+	+	−
PD controller	+	−	+

Fig.2.3.5. PID controller modifications

For example, the PD controller is used in situations where we need a fast response without loss of stability. This type of controller is widely used in aerospace applications such as flight dynamics control, where it is important to quickly minimize deviations and oscillations when maneuvering the aircraft.

PI controllers perform well in applications with slow dynamics such as heating and air conditioning systems. In these systems, fast response times are not important, because the same ambient temperature changes only after a certain period of time. In order to maintain the set temperature level, it is more important to accurately handle steady-state errors and maintain the stability of the system as a whole.

2.4. PID controller tuning methods

The stability of the entire system depends on the correct setting of the controller. Therefore, it is very important to set each component in such a way that the controller meets the stability requirements of the control signal and does not overshoot the system.

The tuning of PID controllers for relatively simple systems is mostly done in a heuristic way, which is a significant advantage of PID controllers compared to other ways of controlling the system, because the tuning of the controller coefficients is so simple that the user can ensure sufficient stability of the system without a detailed model [28].

Basically, the procedure of PID controller coefficient search consists of 3 steps:

1. Get the open loop response of the system and measure the required parameters such as the oscillation period at the system output and the process delay.
2. On the basis of the obtained measurements we calculate the initial gain coefficients Kp, Ki and Kd.
3. Adjusting the coefficients of the regulator to achieve optimal robustness characteristics.

There are widespread methods for synthesizing the coefficients of the PID regulator such as the Ziegler-Nichols method [33] and the Cohen-Coon method [34]. We will focus on the Ziegler-Nichols method because it is simple enough to implement and equally applicable to both open-loop and closed-loop systems, unlike the Cohen-Coon method.

To realize the Ziegler-Nichols method, we need to calculate two parameters experimentally: the ultimate gain coefficient (Ku) and the oscillation period of the system (Tu).

At the first stage we set coefficients Kp, Ki and Kd to zero. After that we start to increase the proportional coefficient (P) until the moment when the impulse response of the system reaches stable oscillations [28;33]. If we choose too small gain coefficient - the output signal will damp and come to equilibrium under the influence of the disturbance (Fig.2.4.1).

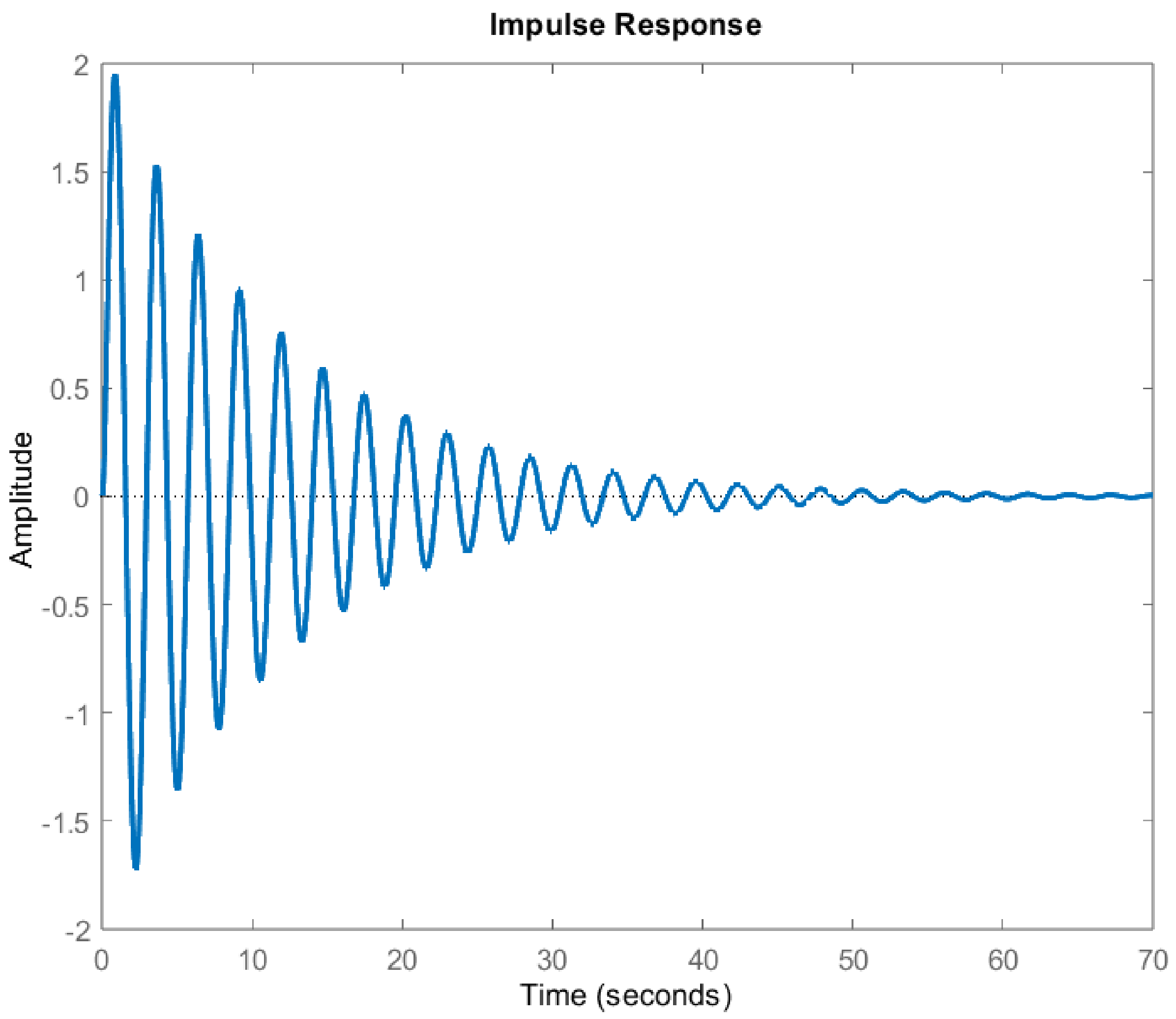


Fig.2.4.1. Impulse response of the system at small proportional coefficient P

And vice versa - if we take too high gain value, the system will start to swing infinitely over the whole time interval (Fig.2.4.2), which is not satisfactory for us.

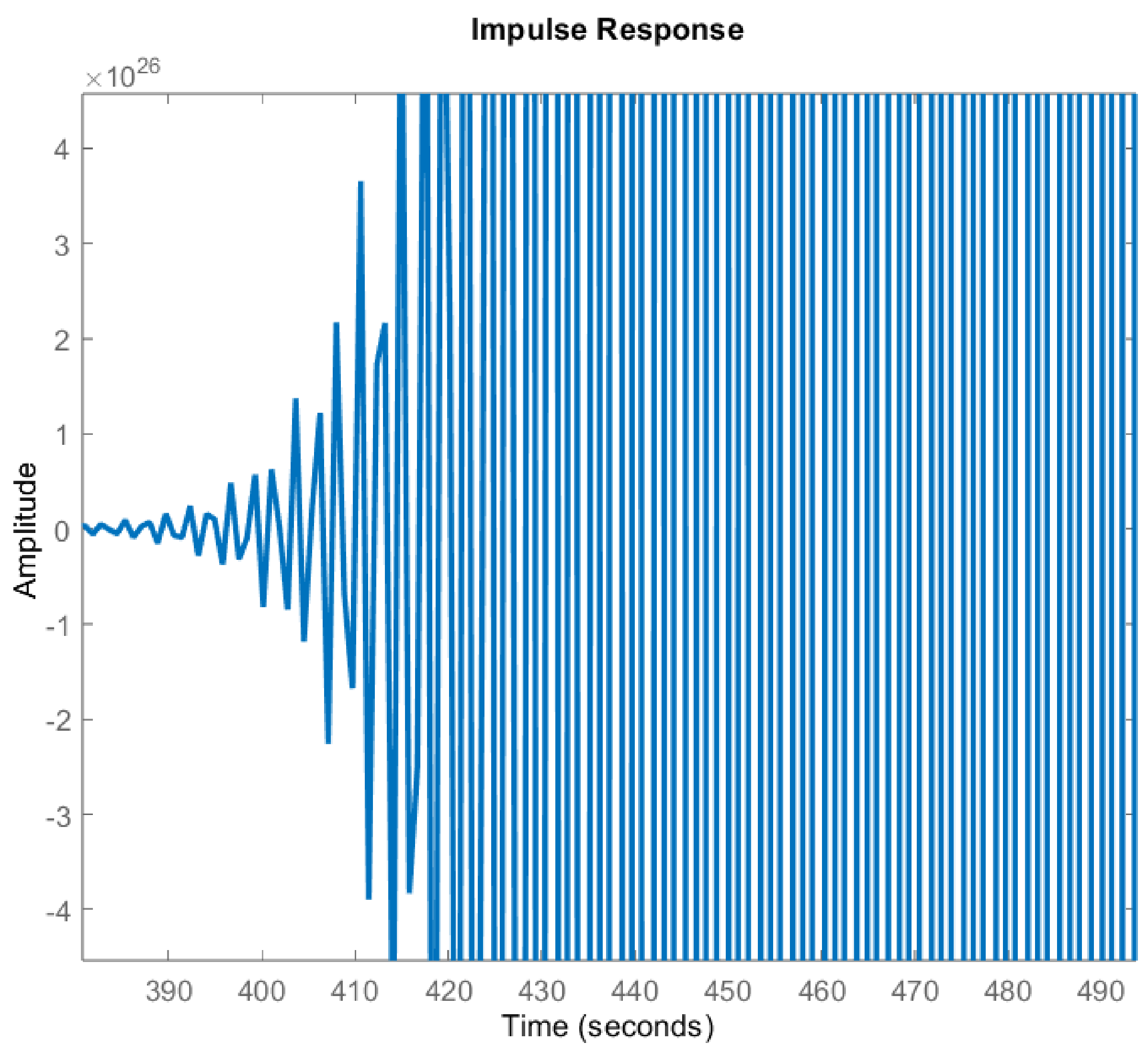


Fig.2.4.2. Impulse response of the system at high proportional coefficient P

The process of optimal selection of the ultimate gain coefficient (Ku) is shown in Fig.2.4.3. We can see that at the optimally chosen coefficient the system became oscillating with regular and stable oscillations.

Also on this graph we determine the oscillation period (Tu), which will be useful in calculating the initial coefficients of the PID controller.

After determining the limiting gain and oscillation period of the closed-loop control system, we can proceed to adjust the coefficients Kp, Ki, Kd depending on the desired behavior of the controller. To do this, we need to combine simple calculations with the values of Ku and Tu in the table derived by Ziegler and Nichols (Table.2.4.1) [28;33].

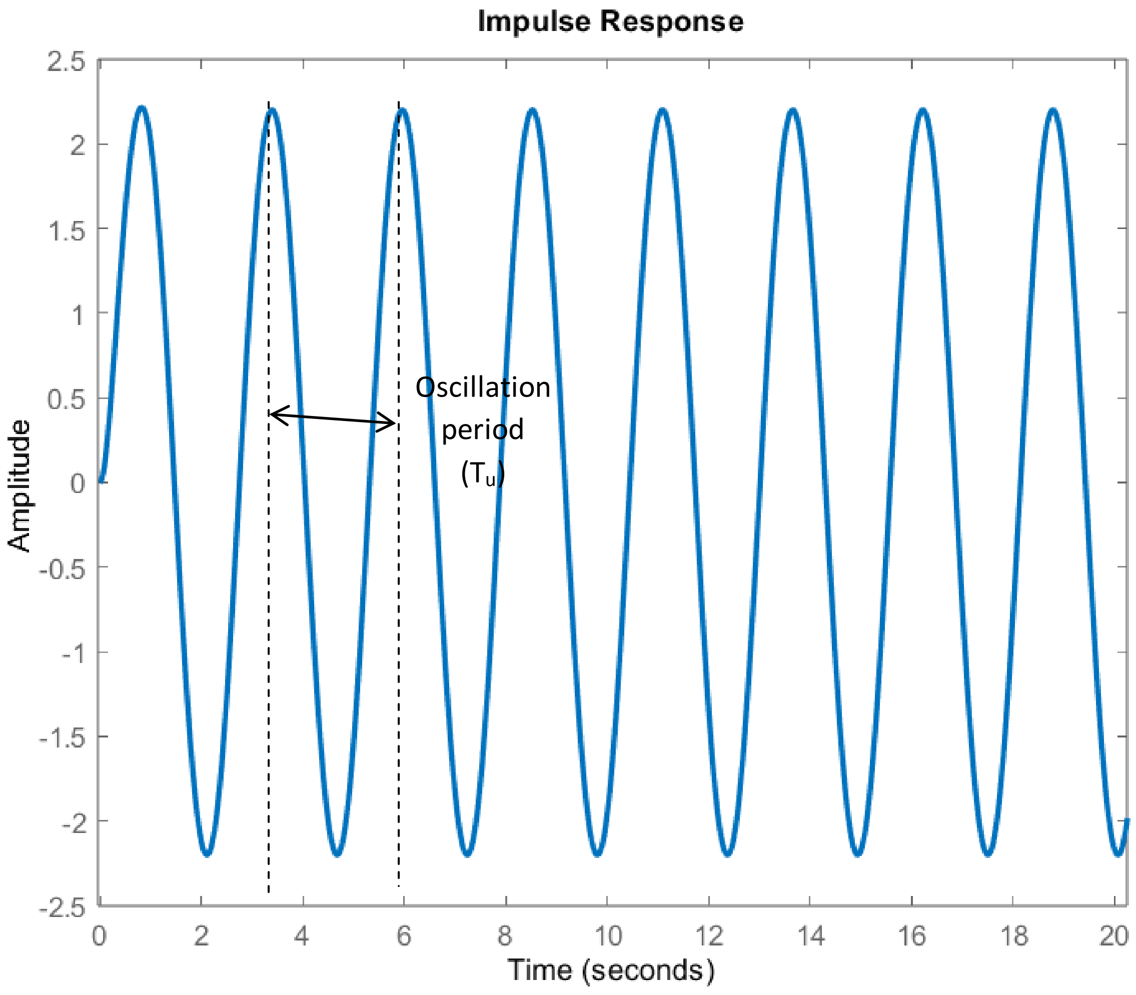


Fig.2.4.3.Impulse response with optimal proportional coefficient P.

Table.2.4.1

Ziegler-Nichols tuning using the oscillation method

Control Type					
P		-	-	-	-
PI			-		-
PD		-		-	
PID					

where:

- the parameter that scales the integral controller;

- the parameter that scales the derivative controller.

The obtained 3 coefficients are substituted into the equation for the correction of the input signal u(t) from the error e(t) [29-30]:

After the coefficients are calculated, they are adjusted based on the system response. If the response is too slow, increase the P or I gain. If the response is too fast or too oscillating, reduce the P or I gain. To eliminate system overshoot, increase the D gain [28].

2.5. Advantages and Disadvantages of PID controller usage

In general, the PID controller is a universal tool that is used in many industrial applications. Let consider the main advantages of this regulator.

Advantages of PID control [32]:

- Sufficient control accuracy
- Energy efficiency
- Low hardware requirements
- Variety of implementation (Can be both analog and digital)
- Intuitive and easy setup:
- Can be reconfigured without stopping the process:
- Excellent price-performance ratio
- Better response to unforeseen disturbances

However, like any other method, the PID controller has a number of disadvantages.

Disadvantages of PID control [32]:

- Sensitivity to mis-tuning
- Increased high-frequency noise
- Problems with response delay

2.6. Conclusion

In the second section it was reviewed the concept of computer vision, what it consists of, what it is needed for, what methods are used to realize computer vision approaches.

Special attention was paid to the method of cascade clustering and Viola Jones algorithm which is based on this method. Despite its antiquity, this method is still in demand in tasks related to detection, clustering and tracking of objects.

Also, various ways of realization of control system realization of pilotless flying apparatus of quadcopter type were considered. We defined the proportional-integral-differential (PID) controller, analyzed the principle of its operation and also found out how the coefficients of PID controller influence.

It was considered the simplest and most popular method of tuning PID regulators - the Ziegler-Nichols method.

In the end it was identified the main advantages and disadvantages of this approach to controlling moving objects. Despite its popularity and ease of synthesis and tuning, this type of controller tends to raise the noise level of the signal at high frequencies, which may require pre-filtering of the signal before it enters the PID controller.

SECTION 3
OBJECT TRACKING SYSTEM DESIGN

3.1. Tracking algorithm description

The design of the system that is described in this part of the qualification work was presented at the POLIT conference in april 2024 [35].

The algorithm of object detection and tracking is as follows: The quadcopter is lifted to the altitude set by the user, then the camera is turned on and the picture is transmitted to the workstation. As soon as a person's face appears in the frame, the system finds the face at the same moment via the cascade classifier. The face is detected and a bounding box is plotted around it. After the image processing, the system sends a command to the quadcopter's microprocessor to start moving towards the object. The main stages of the algorithm are presented in the flowchart (see Fig.3.1.1)

To implement tracking, it is necessary to have a set point to which the position of the object in the image will be compared. In this work it was decided to use the AABB (Minimum bounding box) method [36].

A bounding box is created, which is fixed strictly in the center of the scene with arbitrary dimensions set by the user. This will be our starting position. Then, the center of the object bounding box is compared to the center of the main bounding box.

The main idea of yaw and height control is to calculate the difference between the coordinates of the center point of the object and the center point of the main frame, i.e. the starting point of the coordinate plane. The calculated difference forms the value of the control error, which will then be used to realize the PD control of the quadcopter.

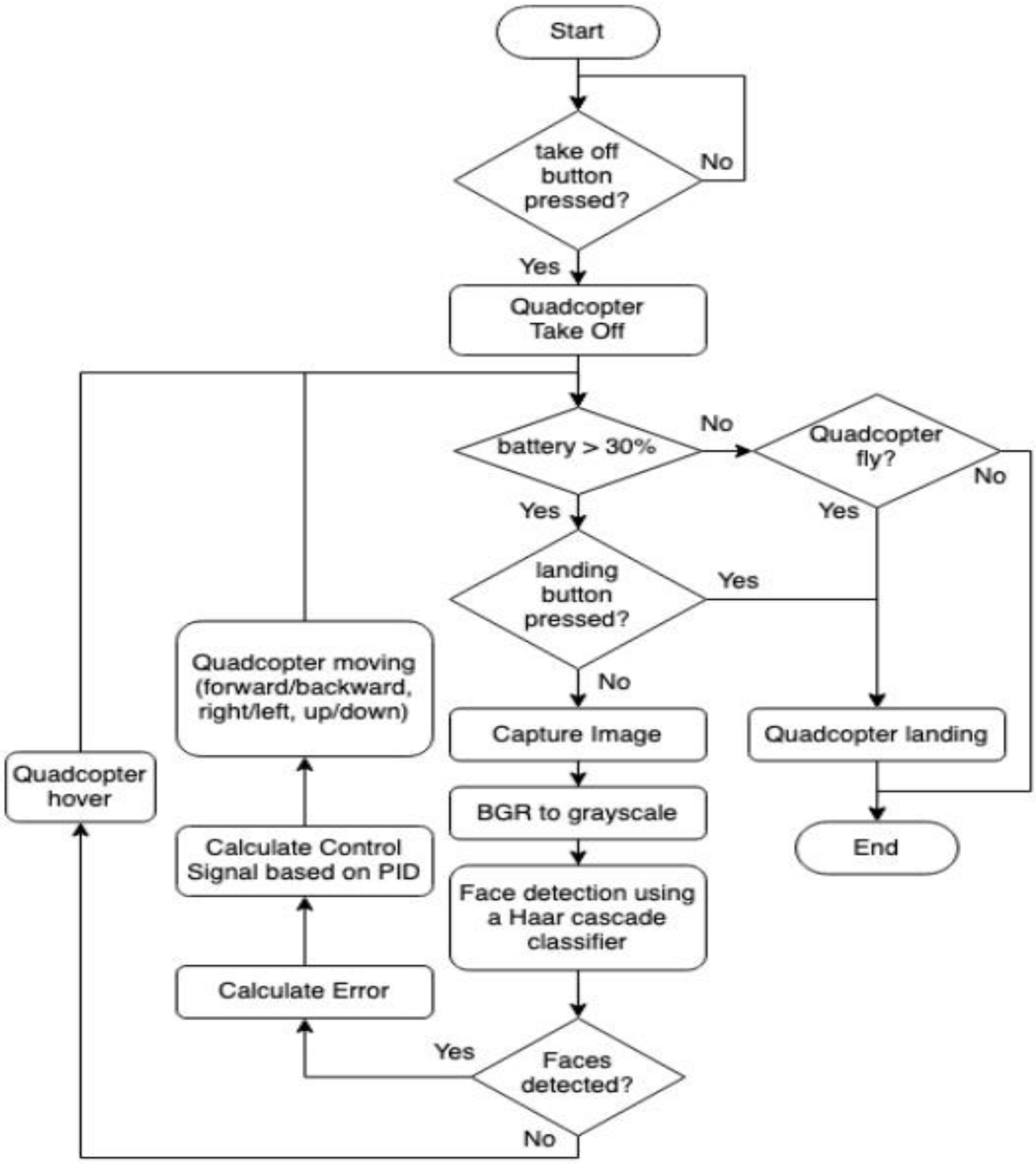


Fig.3.1.1. Face tracking algorithm for Tello quadcopter flowchart

In other words, the quadcopter will always try to keep the object in the center of the image. If the object deviates from the center in the x, y axes, the control system will compensate for the deviation and turn the quadcopter to the tracked object. Visualization of the yaw motion algorithm is shown in Fig.3.1.2 and height movement algorithm is shown in Fig.3.1.3

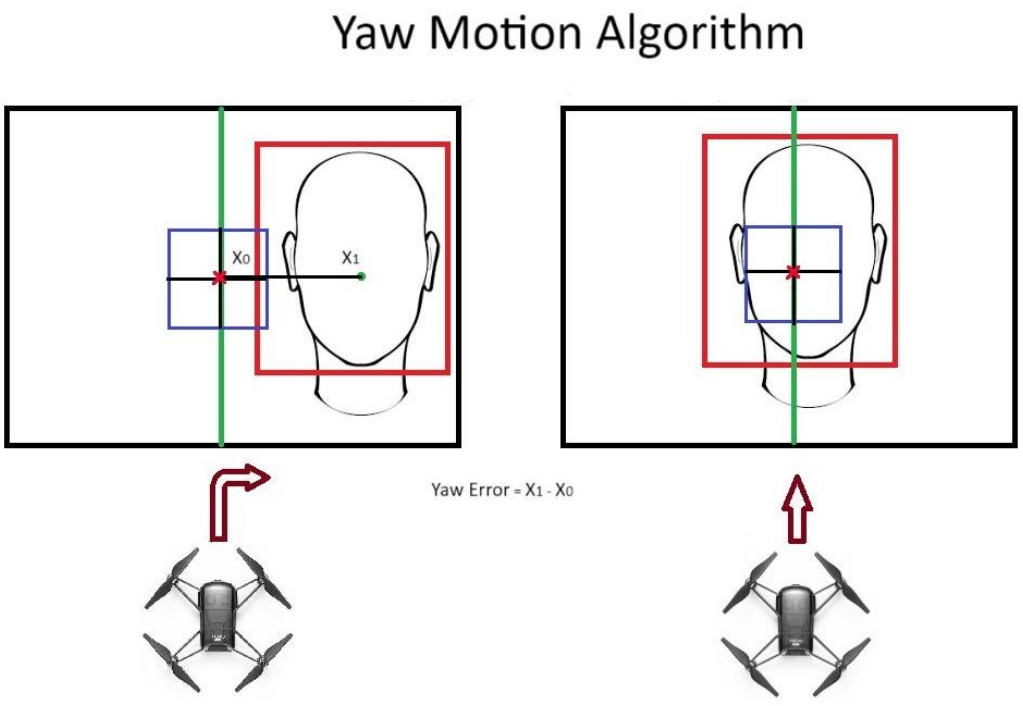


Fig.3.1.2. Illustration of yaw motion algorithm

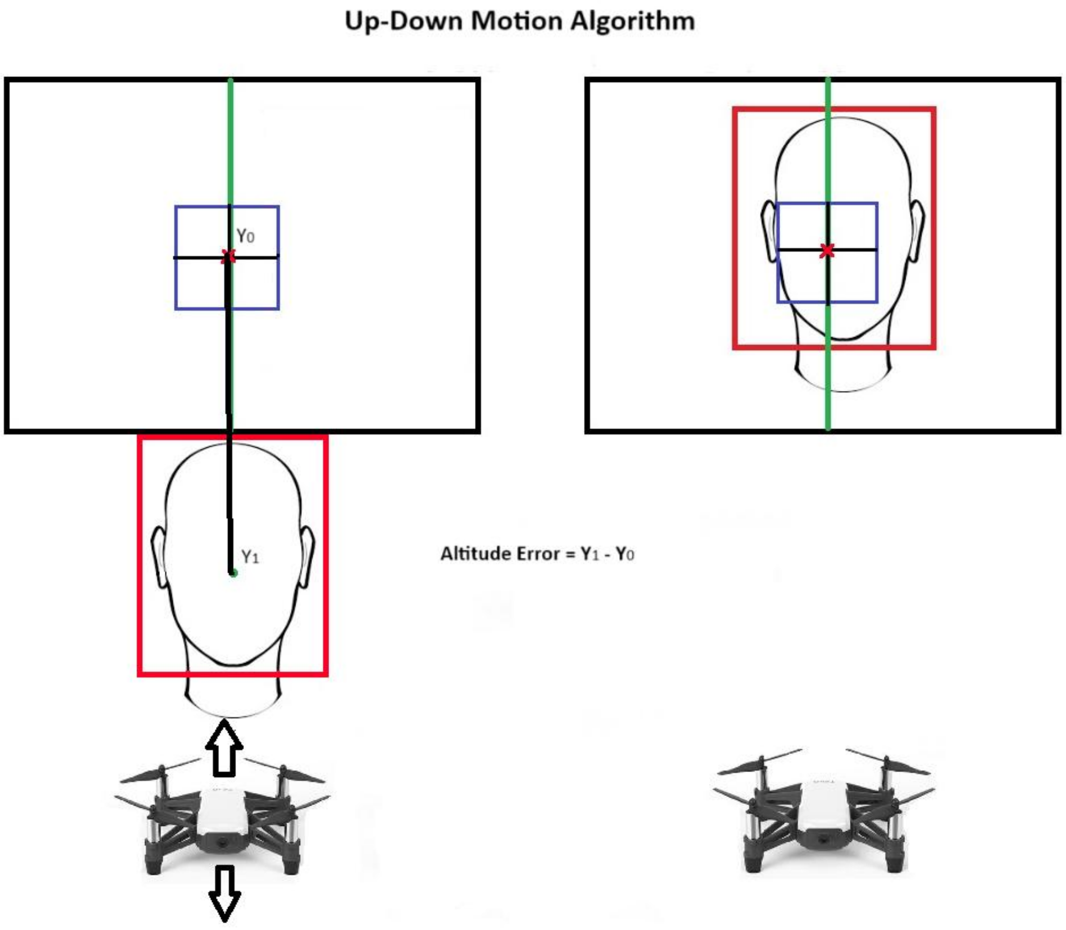


Fig.3.1.3. Illustration of “Up-Down” motion algorithm

The forward-backward motion algorithm is implemented on a different principle. In contrast to the previous two algorithms, where the displacement of the object can be calculated simply enough by Euclidean distance, the forward-backward movement in two-dimensional space is realized by the method of frame scale difference.

First, the areas of the bounding frames are calculated and depending on the distance of the object to the camera, the area of the object frame changes. The closer the object - the larger the area of its bounding box and vice versa.

The next step is to create a notional "safe zone", i.e. the range of areas in which the quadcopter hovers in front of the object (Fig.3.1.4).

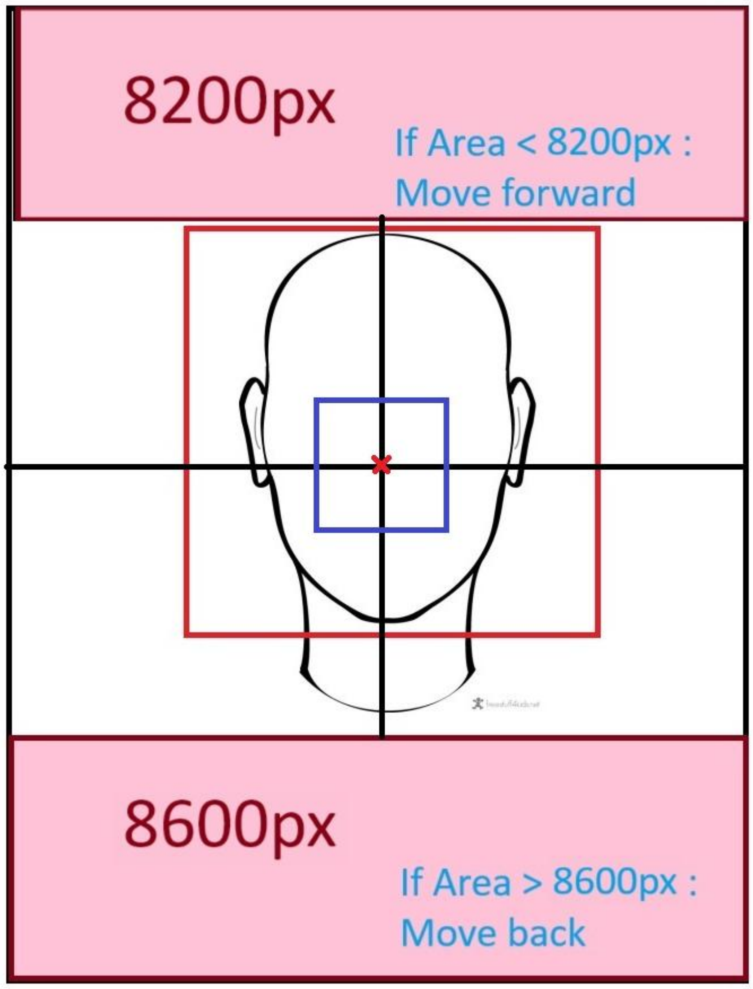


Fig.3.1.4. Safe zone illustration

If the object approaches or moves away from the quadcopter's camera lens, the first one goes out of the zone limits, the quadcopter is given instruction to change its position in order to return to the zone limits.

The error signal for longitudinal motion is calculated as follows:

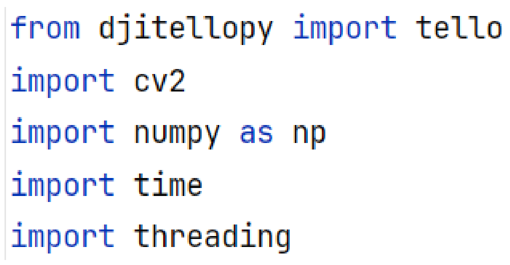
where: – minimum and maximum values of safe zone;

Area of object's boundary box.

3.2. Program code description

All code is written in Python programming language version 3.11, as Tello supports this language.

To get started, it is important to add all the necessary libraries (Fig.3.2.1).



```
from djitellopy import tello
import cv2
import numpy as np
import time
import threading
```

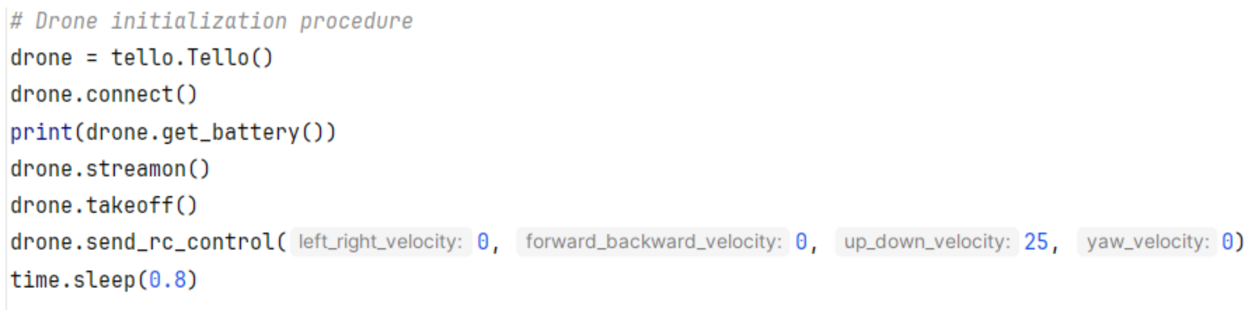
Fig.3.2.1. Import of required libraries

Brief overview of libraries:

- djitellopy - This is a custom library that adds advanced functionality based on the standard features of Tello SDK 2.0. Used for q quadcopter's control.
- cv2 – OpenCV (Open Computer Vision) An open source library for computer vision and image processing. Used for processing and analyzing video stream, object recognition, etc.
- numpy - A library for working with multidimensional arrays and matrices, as well as a wide range of math functions. Used to perform various numerical operations such as scaling and image transformations.

- time - Python's built-in time module. Used to manage code execution delays and measure the execution time of operations.
- threading - Python's built-in time module. Used to perform multiple operations simultaneously.

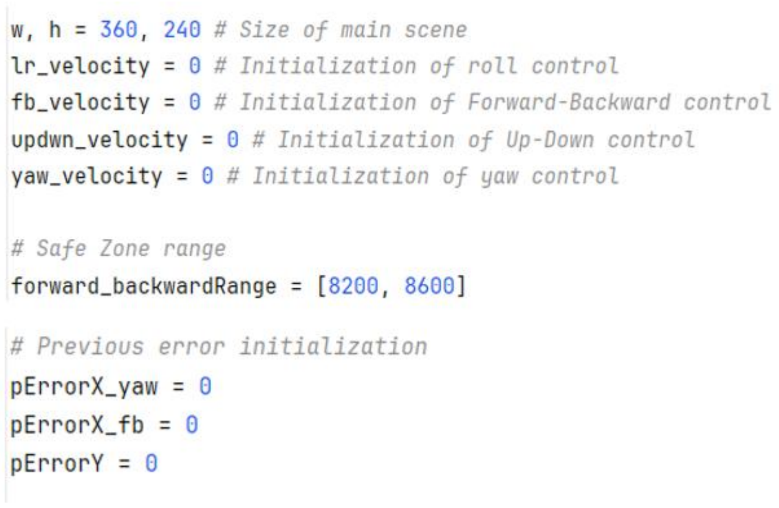
After the libraries are connected, the quadcopter is initialized. Here, Fig.3.2.2 shows the commands for connecting to the quadcopter, after which it rises to the set altitude and waits for further commands.



```
# Drone initialization procedure
drone = tello.Tello()
drone.connect()
print(drone.get_battery())
drone.streamon()
drone.takeoff()
drone.send_rc_control( left_right_velocity: 0, forward_backward_velocity: 0, up_down_velocity: 25, yaw_velocity: 0)
time.sleep(0.8)
```

Fig.3.2.2. Tello quadcopter Initialization

Here we declare all the necessary variables (Fig.3.2.3).



```
w, h = 360, 240 # Size of main scene
lr_velocity = 0 # Initialization of roll control
fb_velocity = 0 # Initialization of Forward-Backward control
updwn_velocity = 0 # Initialization of Up-Down control
yaw_velocity = 0 # Initialization of yaw control

# Safe Zone range
forward_backwardRange = [8200, 8600]

# Previous error initialization
pErrorX_yaw = 0
pErrorX_fb = 0
pErrorY = 0
```

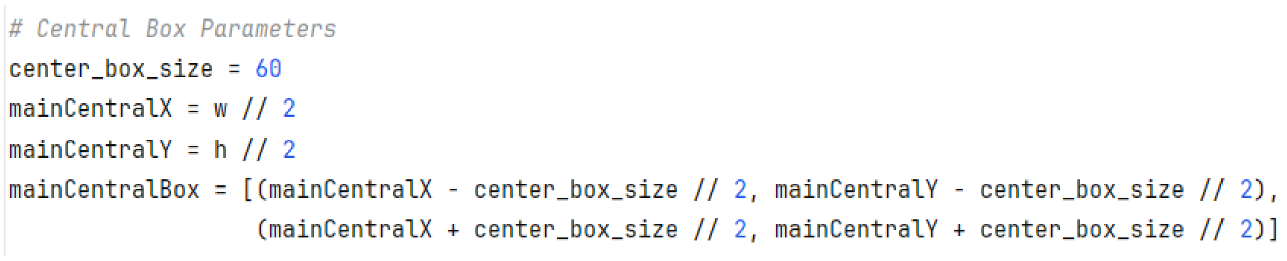
Fig.3.2.3. Variables declaration

Create a static center boundary box in the middle of the image. The coordinates of the box are calculated in the following way: First we calculate the coordinates of the center of the image:

where *w* and *h* are the width and the height of the image.

Then we calculate the coordinates of the extreme corners of our box: top left and bottom right corners:

where «*center box size*» indicates the size of a box with an arbitrary user-defined side length. In the code it looks as follows (Fig.3.2.4).



```
# Central Box Parameters
center_box_size = 60
mainCentralX = w // 2
mainCentralY = h // 2
mainCentralBox = [(mainCentralX - center_box_size // 2, mainCentralY - center_box_size // 2),
                  (mainCentralX + center_box_size // 2, mainCentralY + center_box_size // 2)]
```

Fig.3.2.4. Coordinates of central boundary box

The coordinates of the face bounding box are counted according to the same principle.

Now we need to initialize our PD regulators and their coefficients. (Fig.3.2.5).

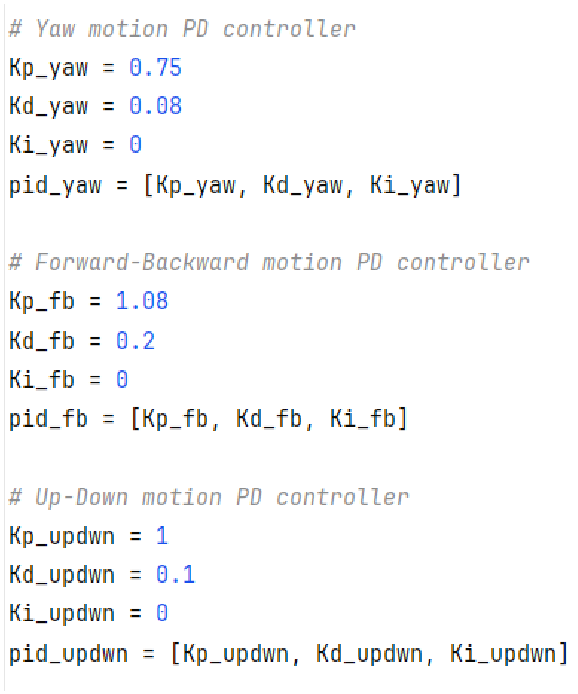
Based on the theory described in Section 2 and using the object, by iterative selection of coefficients for each of the 3 regulators separately, the coefficients of the regulators were determined, the values of which are given in Table 3.2.1.

Table 3.2.1

PD regulators coefficients

	0.75
	0.08
	1.08
	0.2
	1
	0.1

Since we decided to use a PD controller, the integral component K_i is set to zero.



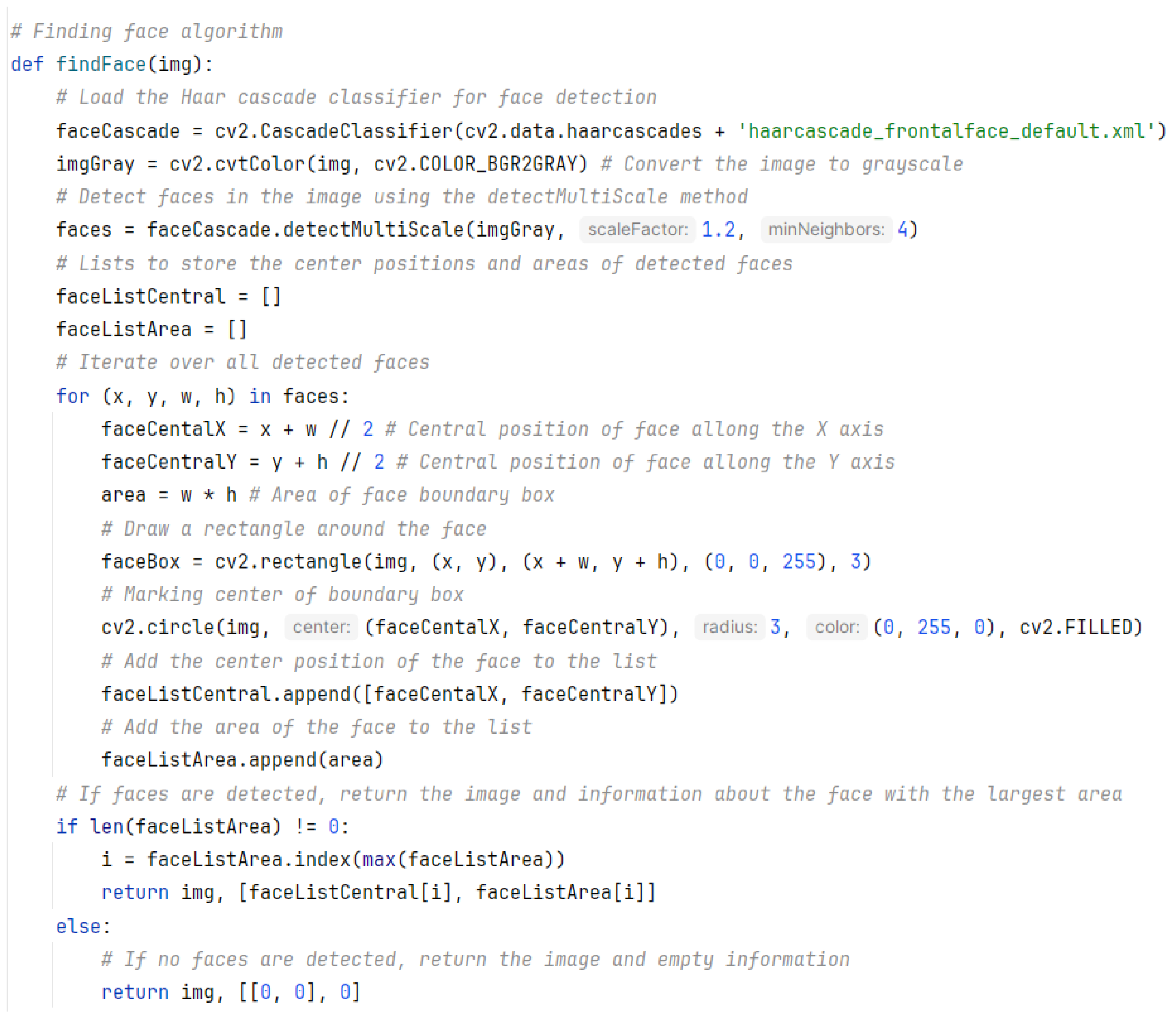
```
# Yaw motion PD controller
Kp_yaw = 0.75
Kd_yaw = 0.08
Ki_yaw = 0
pid_yaw = [Kp_yaw, Kd_yaw, Ki_yaw]

# Forward-Backward motion PD controller
Kp_fb = 1.08
Kd_fb = 0.2
Ki_fb = 0
pid_fb = [Kp_fb, Kd_fb, Ki_fb]

# Up-Down motion PD controller
Kp_updwn = 1
Kd_updwn = 0.1
Ki_updwn = 0
pid_updwn = [Kp_updwn, Kd_updwn, Ki_updwn]
```

Fig.3.2.5. Initialization of PD regulators

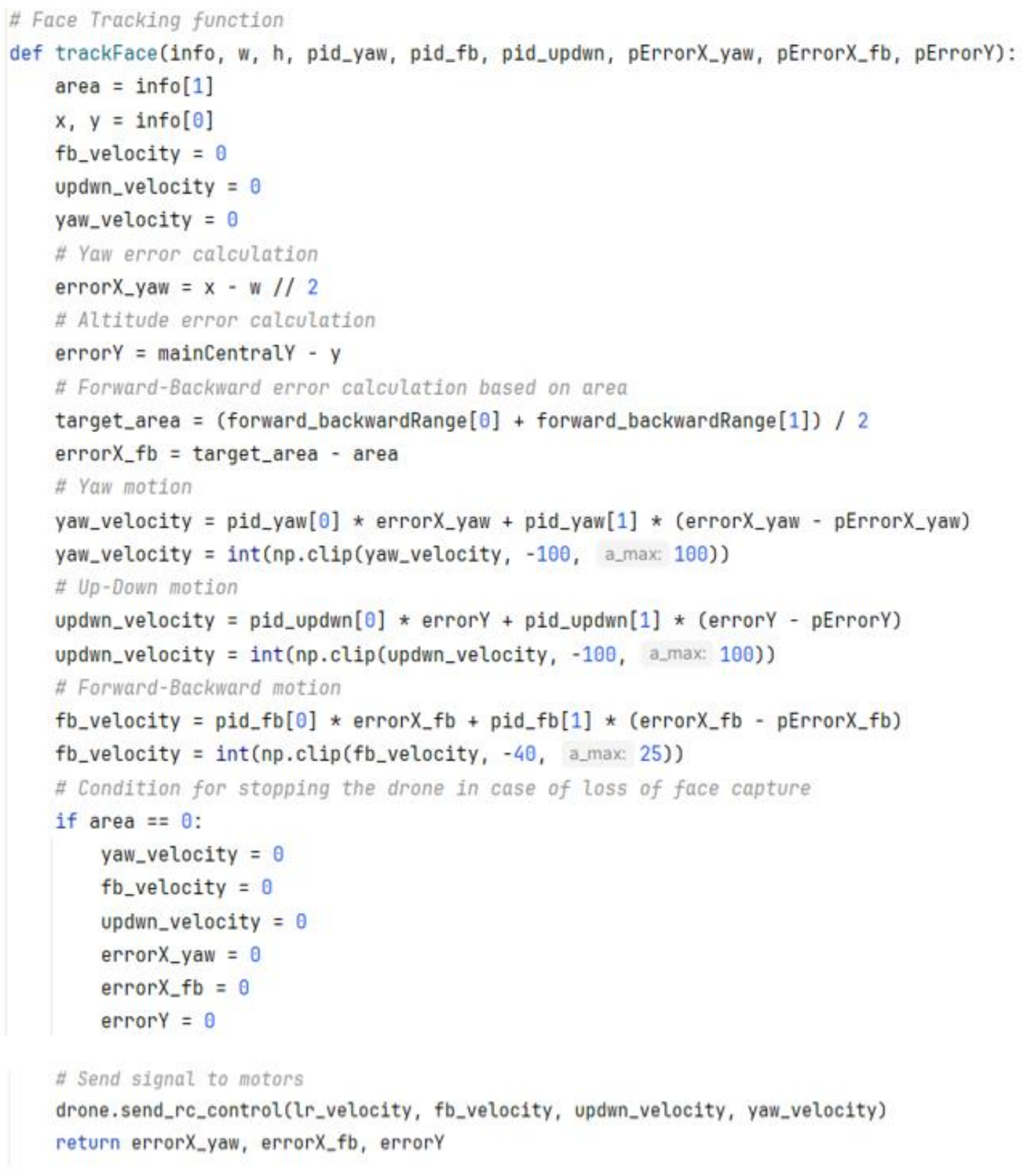
The face retrieval algorithm is implemented by creating a trained cascade classifier. To improve the performance of the classifier, the image is converted to grayscale, because in this case the image loses 3 color channels, but does not lose information about contrast, on the basis of which Haar cascades work. in this way the computational complexity of the algorithm is reduced. After that, we detect faces using the MultiScale method [37] and create bounding boxes for the found face (Fig.3.2.6).



```python
# Finding face algorithm
def findFace(img):
    # Load the Haar cascade classifier for face detection
    faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # Convert the image to grayscale
    # Detect faces in the image using the detectMultiScale method
    faces = faceCascade.detectMultiScale(imgGray,  scaleFactor: 1.2,  minNeighbors: 4)
    # Lists to store the center positions and areas of detected faces
    faceListCentral = []
    faceListArea = []
    # Iterate over all detected faces
    for (x, y, w, h) in faces:
        faceCentalX = x + w // 2 # Central position of face allong the X axis
        faceCentralY = y + h // 2 # Central position of face allong the Y axis
        area = w * h # Area of face boundary box
        # Draw a rectangle around the face
        faceBox = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 3)
        # Marking center of boundary box
        cv2.circle(img,  center: (faceCentalX, faceCentralY),  radius: 3,  color: (0, 255, 0), cv2.FILLED)
        # Add the center position of the face to the list
        faceListCentral.append([faceCentalX, faceCentralY])
        # Add the area of the face to the list
        faceListArea.append(area)
    # If faces are detected, return the image and information about the face with the largest area
    if len(faceListArea) != 0:
        i = faceListArea.index(max(faceListArea))
        return img, [faceListCentral[i], faceListArea[i]]
    else:
        # If no faces are detected, return the image and empty information
        return img, [[0, 0], 0]
```

Fig.3.2.6. Face recognition algorithm

The control function is very simple in its implementation. It is necessary to calculate the difference between the actual location of the object in the image and the desired one, i.e. the center of the image. With the help of PD controller the quadcopter will try to quickly reduce the error to zero and thus turn in the right direction depending on the position of the object (Fig.3.2.7).

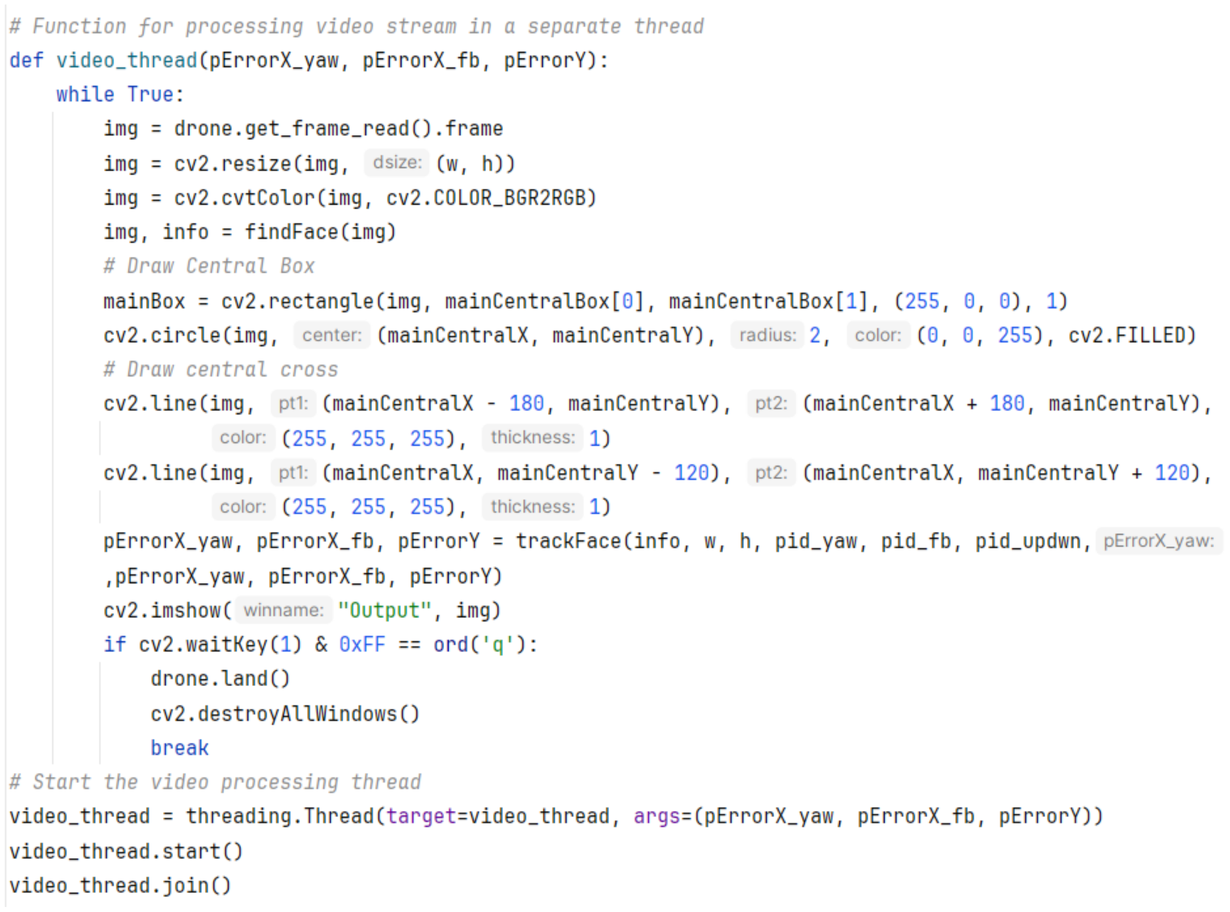


```python
# Face Tracking function
def trackFace(info, w, h, pid_yaw, pid_fb, pid_updwn, pErrorX_yaw, pErrorX_fb, pErrorY):
    area = info[1]
    x, y = info[0]
    fb_velocity = 0
    updwn_velocity = 0
    yaw_velocity = 0
    # Yaw error calculation
    errorX_yaw = x - w // 2
    # Altitude error calculation
    errorY = mainCentralY - y
    # Forward-Backward error calculation based on area
    target_area = (forward_backwardRange[0] + forward_backwardRange[1]) / 2
    errorX_fb = target_area - area
    # Yaw motion
    yaw_velocity = pid_yaw[0] * errorX_yaw + pid_yaw[1] * (errorX_yaw - pErrorX_yaw)
    yaw_velocity = int(np.clip(yaw_velocity, -100,  a_max: 100))
    # Up-Down motion
    updwn_velocity = pid_updwn[0] * errorY + pid_updwn[1] * (errorY - pErrorY)
    updwn_velocity = int(np.clip(updwn_velocity, -100,  a_max: 100))
    # Forward-Backward motion
    fb_velocity = pid_fb[0] * errorX_fb + pid_fb[1] * (errorX_fb - pErrorX_fb)
    fb_velocity = int(np.clip(fb_velocity, -40,  a_max: 25))
    # Condition for stopping the drone in case of loss of face capture
    if area == 0:
        yaw_velocity = 0
        fb_velocity = 0
        updwn_velocity = 0
        errorX_yaw = 0
        errorX_fb = 0
        errorY = 0

    # Send signal to motors
    drone.send_rc_control(lr_velocity, fb_velocity, updwn_velocity, yaw_velocity)
    return errorX_yaw, errorX_fb, errorY
```

Fig.3.2.7. Tracking function

At the end follows the main loop of the program where the functions of receiving the image from the quadcopter camera are called, and the main function *trackFace()* from which the values for calculating the previous error are taken (Fig.3.2.8).



```
# Function for processing video stream in a separate thread
def video_thread(pErrorX_yaw, pErrorX_fb, pErrorY):
    while True:
        img = drone.get_frame_read().frame
        img = cv2.resize(img,  dsize: (w, h))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img, info = findFace(img)
        # Draw Central Box
        mainBox = cv2.rectangle(img, mainCentralBox[0], mainCentralBox[1], (255, 0, 0), 1)
        cv2.circle(img,  center: (mainCentralX, mainCentralY),  radius: 2,  color: (0, 0, 255), cv2.FILLED)
        # Draw central cross
        cv2.line(img,  pt1: (mainCentralX - 180, mainCentralY),  pt2: (mainCentralX + 180, mainCentralY),
                  color: (255, 255, 255),  thickness: 1)
        cv2.line(img,  pt1: (mainCentralX, mainCentralY - 120),  pt2: (mainCentralX, mainCentralY + 120),
                  color: (255, 255, 255),  thickness: 1)
        pErrorX_yaw, pErrorX_fb, pErrorY = trackFace(info, w, h, pid_yaw, pid_fb, pid_updwn, pErrorX_yaw:
        ,pErrorX_yaw, pErrorX_fb, pErrorY)
        cv2.imshow( winname: "Output", img)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            drone.land()
            cv2.destroyAllWindows()
            break
# Start the video processing thread
video_thread = threading.Thread(target=video_thread, args=(pErrorX_yaw, pErrorX_fb, pErrorY))
video_thread.start()
video_thread.join()
```

Fig.3.2.8. Main loop of program.

You may also notice that the main loop is inside the video thread function, which means that the program is divided into two threads, one of which performs calculations concerning the video stream. This is done to save resources of the quadcopter processor, so that it could make calculations concerning the motion in parallel.

3.3. Laboratory tests

Laboratory testing is an integral part of new product development. The following is a list of the importance of this stage of development:

1. Safety: First of all, laboratory tests under controlled conditions allow the product to be tested without risk to others. In addition, laboratory testing allows to identify potential vulnerabilities of the system before the product is released for mass use by consumers.

2. Error Detection and Correction: This point is critical because a buggy design will either work poorly or not at all. Fixing bugs at the initial stage is much easier and cheaper than fixing them during production and delivery to potential users of the product.

3. Adjustment and calibration: At the stage of laboratory tests it is very likely that bugs will appear in the operation of the software or hardware part of the test object. Lab tests are great for testing sensors and various systems to ensure that the robot works according to the given specifications and can accurately perform its tasks.

4. Performance evaluation: Under laboratory conditions, it is possible to fully evaluate the performance of a new development, whether the actual results match the expected one, and whether the system can perform the intended task.

A large sport hall was chosen as the place for the quadcopter's laboratory tests. The tests were conducted in the daytime. This choice of place and time was made for two reasons:

1. The quadcopter has a small weight, which imposes its own limitations on the choice of the place of flight. Attempts to test on a soccer field in sunny weather with an average wind speed of 3 meters per second were unsuccessful. Immediately after takeoff, the quadcopter began to blow away and the control system was unable to handle the resulting environmental disturbances. Therefore, the sport hall turned out to be good protection from the wind and was perfect as a testing ground.

2. Since navigation in space is carried out using a camera and a visual positioning system, the quadcopter is very sensitive to lighting levels. During testing in low light, the quadcopter repeatedly lost facial lock and performed inappropriate maneuvers. Therefore, it was decided to conduct tests exclusively in sunny weather or in rooms with powerful lighting.

3. Despite the fact that the quadcopter is quite compact, conducting tests in a small apartment with many sharp corners and various furniture turned out to be not entirely safe for both the tester and the subject. In the initial stages, the quadcopter often crashed into walls, got stuck on the ceiling, and got tangled in curtains. One of these tests led to the destruction of one of the 4 propellers. The sports hall opposite has a large

and spacious interior space with a high ceiling. It also turned out to have much fewer obstacles and potential threats to the quadcopter, which made it possible to conduct full testing.

Fig.3.3.9 illustrates images from the quadcopter camera with the user interface implemented using OpenCV.

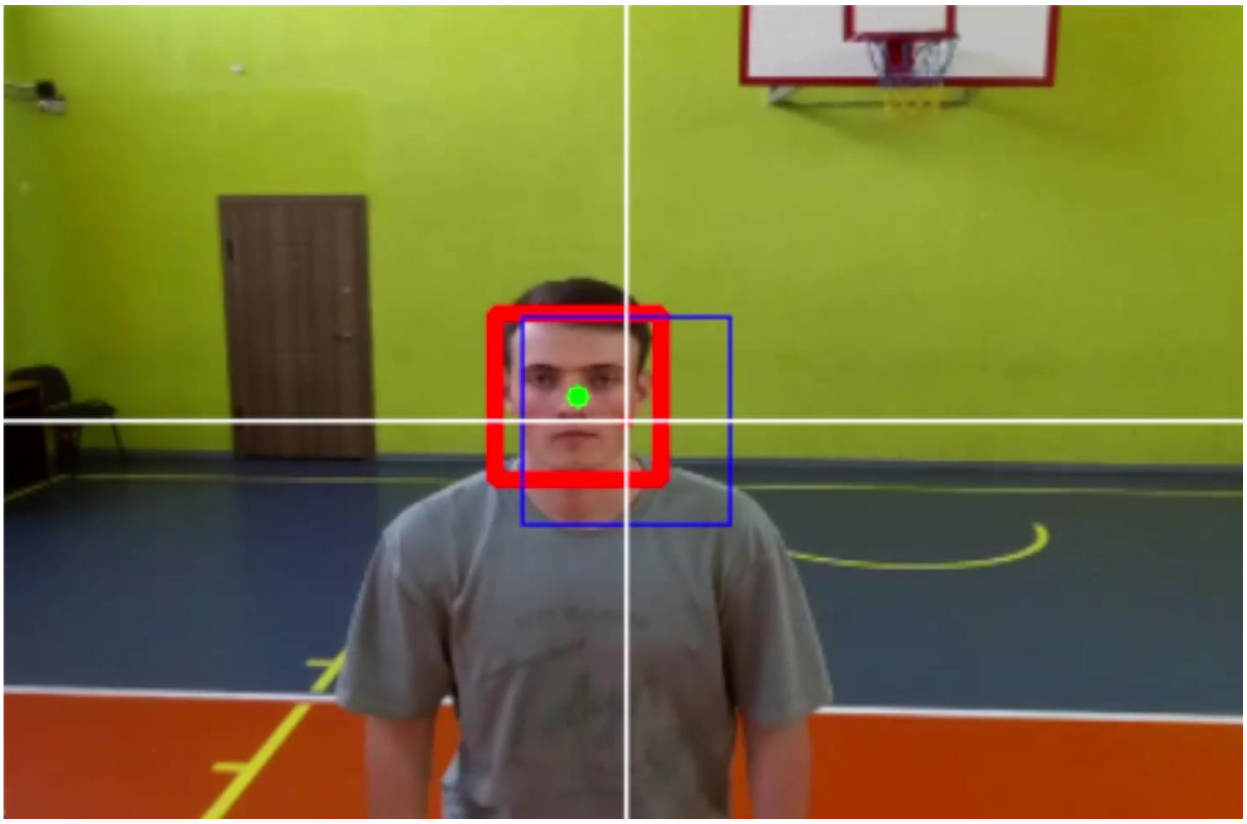
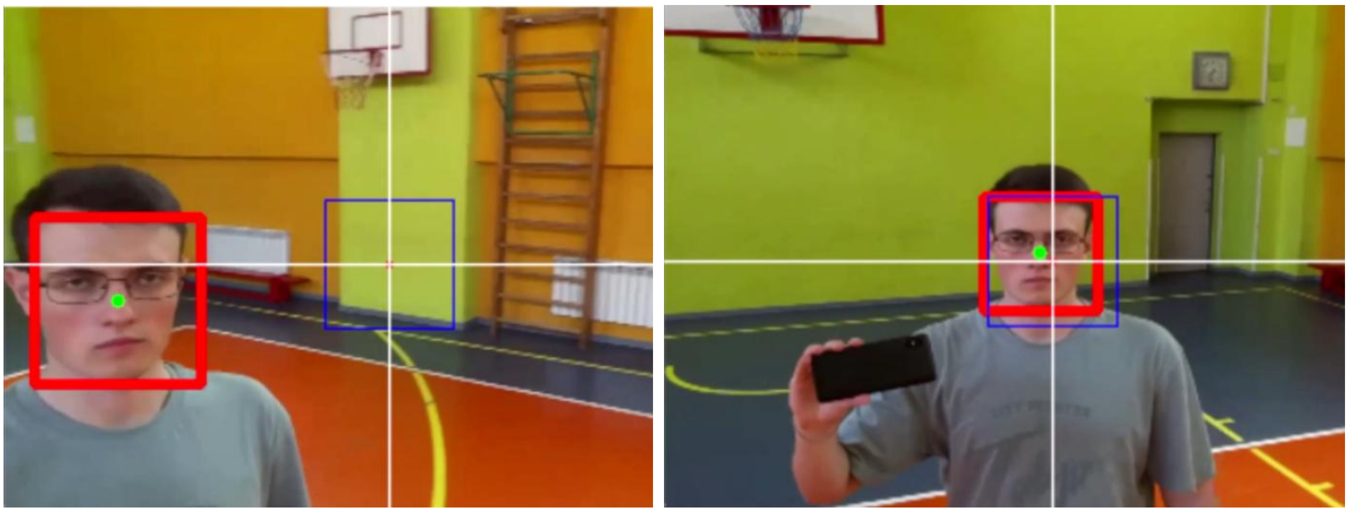


Fig.3.3.1. User Interface from Tello camera

Here you can see the main elements of the interface. In the center there is a landmark (Center Frame). To the left is a red bounding box that highlights the face, the green dot marks the center of the bounding box. Also, on the image there is a white marking in the form of a crosshair, which can be perceived as a two-dimensional coordinate system x and y. It is necessary for better visual perception of the person's movement in the frame.

The Left-Right movement is shown in Fig.3.3.2.

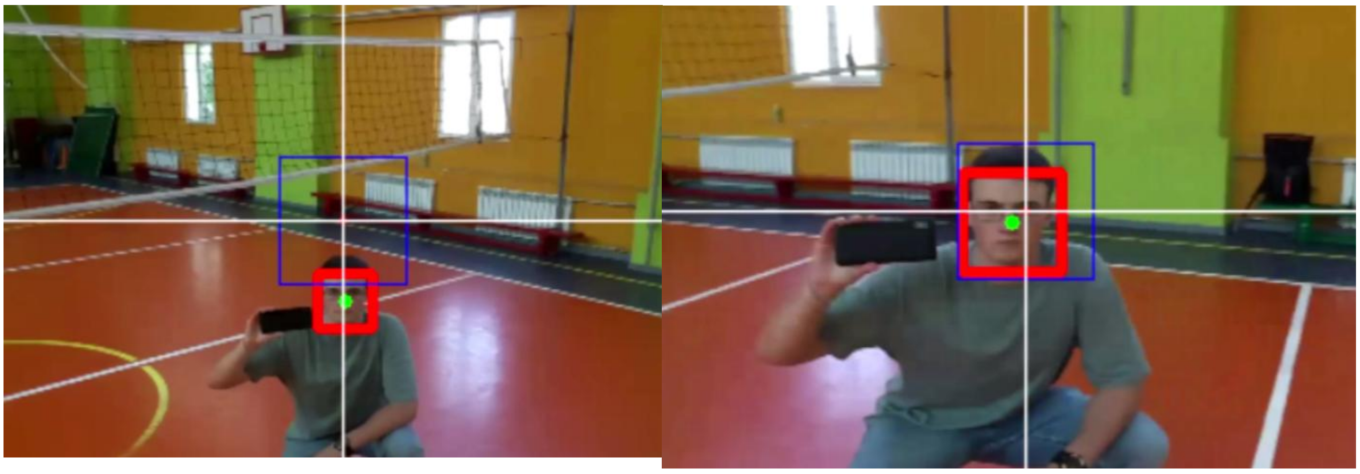


a) b)

Fig.3.3.2. Real time Left-Right movement illustration

a) - Change of the object position to the left; b) - Quadcopter response to object position change

The Up-Down movement is shown in Fig.3.3.3.

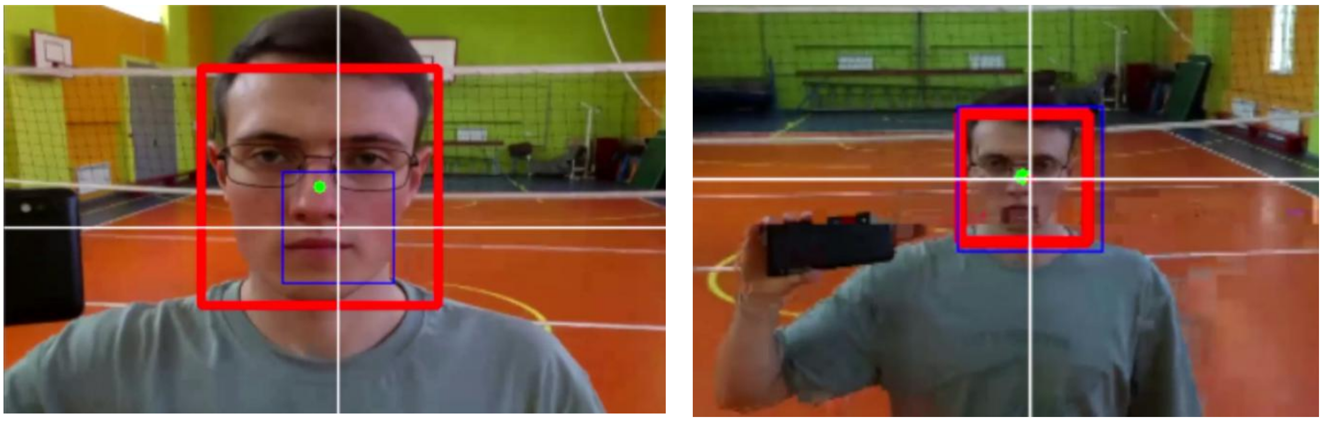


a) b)

Fig.3.3.3. Real time Up-Down movement illustration

a) - Change of object position along the y-axis; b) - Quadcopter response to object position change

The Forward-Backward movement is shown in Figure 3.3.4.



a) b)

Fig.3.3.4. Real time Forward-Backward movement illustration

a) - Change of the object position along x-axis; b) - Quadcopter response to object position change

In Fig.3.3.4 (a) we can see how much larger the face bounding box is compared to the center bounding box. During the real tests, the distance between the face and the quadcopter was about 20-30 cm during the approach.

Also, during the initial stages of testing, several flaws in the system were discovered. One of them is possible errors in the operation of the cascade classifier. In case of changeable illumination, the classifier can erroneously recognize a completely different object as a face. An example of this drawback is shown in Fig.3.3.5.

In such situations, it is very important to consider an emergency landing system to avoid an unwanted collision and potential injuries and breakdowns.

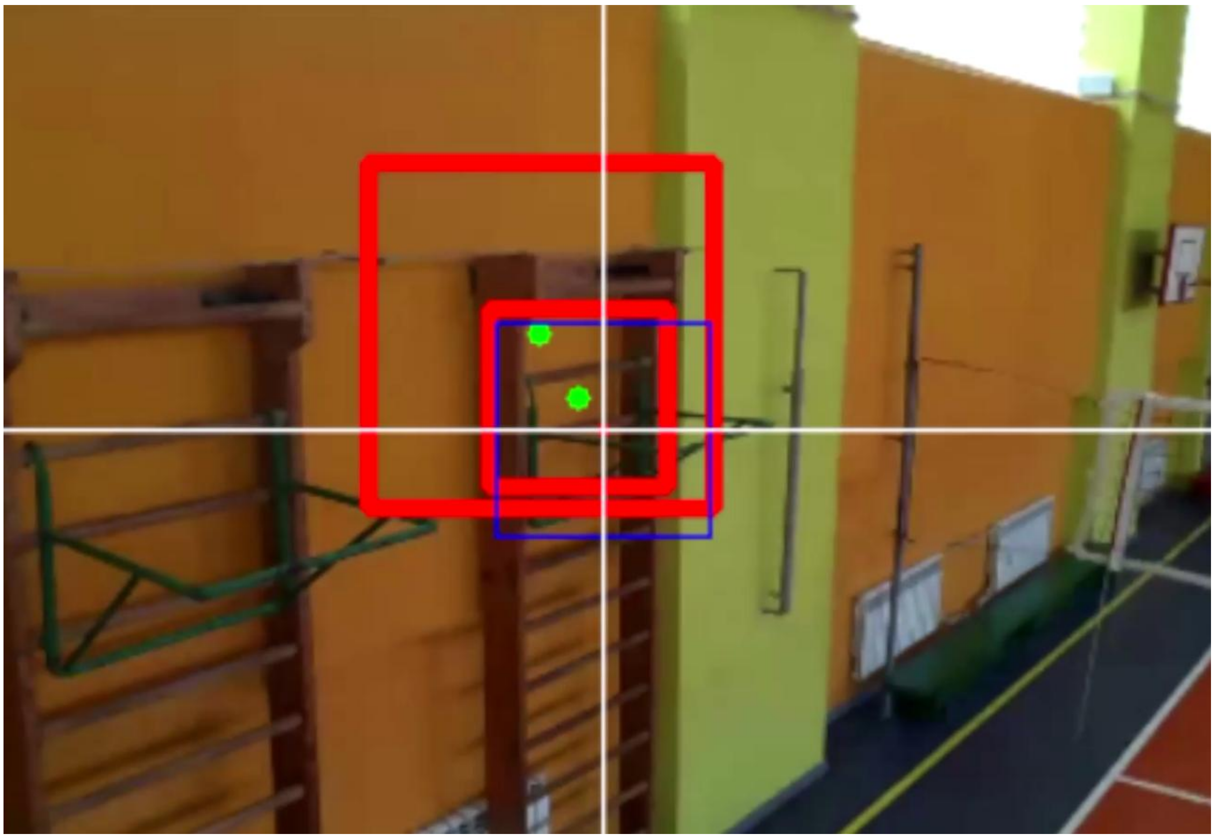


Fig.3.3.5. Object misclassification

Second disadvantage - Sluggish camera picture transfer and segmentation into pixels. This phenomenon can occur due to several reasons:

- insufficient network bandwidth, which may cause data packet loss and, as a result, video artifacts.
- Insufficient processor power causes slowdowns and video artifacts.
- Incorrect setting of video stream resolution, bitrate and compression often lead to frame loss.
- Due to insufficient caching, data does not arrive in time and video may "pixelize".

As a consequence, this problem causes face detection to fail, leading to loss of tracking. This problem you can see in Fig.3.3.6.

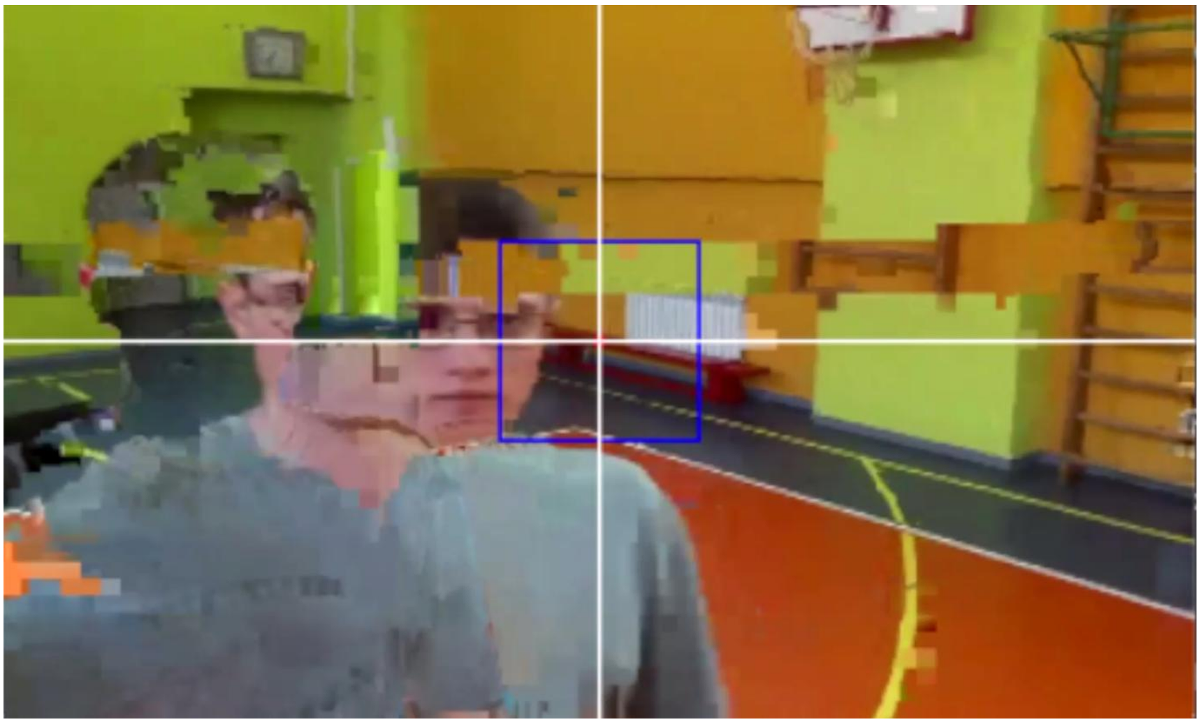


Fig.3.3.6. Video interference

Despite these weaknesses and childhood diseases, which are common to any prototype, the object tracking system for quadcopter copes with its main task, showing a satisfactory result.

3.4. Conclusion

In the third part we described the algorithm by which the object tracking system gives commands to control the quadcopter. We also implemented the program in Python programming language, using custom libraries for computer vision and quadcopter control.

In the end we conducted laboratory tests, during which we evaluated the results of the program. Although the obtained results were not very good, but the system fulfills the tasks set before it.

In the future, this prototype can be improved by optimizing the program and adding new features.

CONCLUSIONS

In the scope of the qualification work we were able to realize an object tracking system based on the Ryze Tello quadcopter, using knowledge obtained after studying the achievements of many scientists and engineers in the field of computer vision, machine learning and automatic control systems.

Based on the analysis of the computer vision topic, the main components of this technology and possible implementation approaches, including image processing algorithms, have been considered. The key method of realization of the object detection system is the Viola-Jones algorithm. The steps of its realization are described in detail, which allowed to achieve sufficient accuracy in real operating conditions.

The implementation of the quadcopter control system was also studied. The realization was based on the concept of control by a PID controller, particularly its special case - the PD controller. The principle of operation was described, as well as methods of coefficient adjustment using the Ziegler-Nichols method. This provided a sufficiently accurate control of the quadcopter flight in the process of tracking the object.

The practical part of the work included writing a program for the quadcopter and testing it in real conditions. Real-time testing allowed to evaluate the performance and efficiency of the developed system. According to the test results, it can be said that the system satisfactorily copes with the task, but it needs to be improved and optimized software.

Thus, the research and development of the object tracking system for quadcopter confirmed the relevance and prospects of the use of computer vision and automatic control systems in the field of unmanned aerial systems. Conclusions and results obtained in the process of research can be used both for further improvement of the existing project and for the development of new solutions in the field of automatic systems.

For example, for an already implemented project based on the Ryze Tello quadcopter, the following improvements can be implemented:

1. Expand the functionality of the quadcopter: Adding features such as obstacle avoidance or the function of building a room map can significantly improve the autonomy of the quadcopter, thereby expanding its areas of application.
2. Improve the control and object recognition system: The implementation of the control system using more advanced methods from modern control theory, such as MPC controllers, as well as the use of new approaches in the field of object detection and classification, for example, Convolutional Neural Networks, will improve the tactical and technical characteristics of the quadcopter and its object detection system.
3. Integration of smartphone management solutions: Despite the fact that there is an application from the official manufacturer of the quadcopter, it is very limited in its functionality. The development of software for smartphones with the addition of advanced functions described in paragraph 1 can significantly facilitate the experience of using a quadcopter with all the features.
4. Simultaneous use in cooperation with other quadcopters: The implementation of the "swarm" system will help to implement such tasks as patrolling the state border. Communicating with each other and exchanging the information received as a single centralized system, can significantly expand the patrol area and reduce the time of the mission.

These improvements will extend the functionality of the object tracking system for the quadcopter, but their implementation will require additional research and testing.

REFERENCES

1. Barnhart R. K. Introduction to Unmanned Aircraft Systems / R. K. Barnhart, S. B. Hottman, D. M. Marshall, E. Shappee. – 1st Ed. – Boca Raton, FL, USA: CRC Press, 2016. – 524 p.
2. The History of Drones in 10 Milestones [Electronic resource]. – URL: https://www.digitaltrends.com/cool-tech/history-of-drones/
3. The Quadrotor’s Coming of Age – USC Viterbi School of Engineering [Electronic resource]. – URL: https://illumin.usc.edu/the-quadrotors-coming-of-age/
4. Johnson W. Helicopter Theory / W. Johnson. – New York: Dover Publications, 1994. – 576 p.
5. Agho C. Dynamic Model and Control of Quadrotor in the Presence of Uncertainties / C. Agho. – 2017. – P. 1-4.
6. Leishman G. J. Principles of Helicopter Aerodynamics with CD / G. J. Leishman. – Cambridge: Cambridge University Press, 2000. – 826 p.
7. The Flying Platforms & Jeeps [Electronic resource]. – URL: https://web.archive.org/web/20111024155224/http://www.vectorsite.net/avplatfm.html
8. Parrot AR.Drones specs: ARM9, Linux, 6DoF IMU, Ultrasonics sensor [Electronic resource]. – URL: https://diydrones.com/profiles/blogs/parrot-ardrones-specs-arm9
9. FAA Says Drone Usage Will Triple by 2020 [Electronic resource]. – URL: https://www.businessinsider.com/faa-says-drone-usage-will-triple-by-2020-2016-3#:~:text=The%20FAA%20expects%20consumer%20drones%20to%20grow%20from,which%20it%20plans%20to%20release%20in%20the%20spring
10. Drone Technologies and Applications [Electronic resource]. – URL: https://www.intechopen.com/chapters/1154922

11. Ye J., Wang J., Lv P. Mutual Aerodynamic Interference Mechanism Analysis of an "X" Configuration Quadcopter / J. Ye, J. Wang, P. Lv // Aerospace. – 2021. – Vol. 8, No. 11.
12. Benotsmane R., Vásárhelyi J. Towards Optimization of Energy Consumption of Tello Quad-Rotor with MPC Model Implementation / R. Benotsmane, J. Vásárhelyi // Energies. – 2022. – Vol. 15, No. 23.
13. Tello [Electronic resource]. – URL: https://www.ryzerobotics.com/tello
14. Computer Vision Definition [Electronic resource]. – URL: https://deepai.org/machine-learning-glossary-and-terms/computer-vision
15. Shortis M., Seager J., Harvey E., Robson S. Influence of Bayer Filters on the Quality of Photogrammetric Measurement / M. Shortis, J. Seager, E. Harvey, S. Robson // Videometrics VIII. – 2005. – Vol. 5665. – P. 164-171.
16. Image Processing: Preprocessing Techniques with OpenCV [Electronic resource]. – URL: https://www.analyticsvidhya.com/blog/2023/03/getting-started-with-image-processing-using-opencv/
17. Comprehensive Guide to Edge Detection Algorithms [Electronic resource]. – URL: https://www.analyticsvidhya.com/blog/2022/08/comprehensive-guide-to-edge-detection-algorithms/
18. Feature Description - Hugging Face Community Computer Vision Course [Electronic resource]. – URL: https://huggingface.co/learn/computer-vision-course/unit1/feature-extraction/feature_description
19. Marcus Angella, “Mastering the Art of Image Segmentation: A Comprehensive Guide to Image Segmentation Techniques”, 2023. [Electronic resource]. – URL: https://medium.com/@1marcusangella/mastering-the-art-of-image-segmentation-a-comprehensive-guide-to-image-segmentation-techniques-545a75010a35
20. Senthilkumaran N., Vaithegi S. Image Segmentation by Using Thresholding Techniques for Medical Images / N. Senthilkumaran, S. Vaithegi // Computer Science & Engineering: An International Journal. – 2016. – Vol. 6, No. 1. – P. 1-13.

21. Mutneja V. Methods of Image Edge Detection: A Review / V. Mutneja // Journal of Electrical & Electronic Systems. – 2015. – Vol. 4, No. 1.
22. Gould S., Gao T., Koller D. Region-Based Segmentation and Object Detection / S. Gould, T. Gao, D. Koller // Advances in Neural Information Processing Systems. – 2009. – Vol. 22. – P. 655-663
23. Vincent L., Soille P. Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations / L. Vincent, P. Soille // IEEE Transactions on Pattern Analysis & Machine Intelligence. – 1991. – Vol. 13, No. 6. – P. 583-598.
24. Freedman D., Zhang T. Interactive Graph Cut Based Segmentation with Shape Priors / D. Freedman, T. Zhang // IEEE Conf. Computer Vision and Pattern Recognition CVPR 2005: 20-25 June 2005; San Diego. – 2005. – Vol. 1. – P. 755-762.
25. Viola P., Jones M. Rapid Object Detection Using a Boosted Cascade of Simple Features / P. Viola, M. Jones // Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. – CVPR 2001. – Vol. 1.
26. Girija S.B., Face Detection with Haar Cascade // Towards Data Science, 2020. [Electronic resource]. – URL: https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08
27. Socret L., Understanding Face Detection with the Viola-Jones Object Detection Framework // Towards Data Science, 2020. [Electronic resource]. – URL: https://towardsdatascience.com/understanding-face-detection-with-the-viola-jones-object-detection-framework-c55cc2a9da14
28. Principles of PID Controllers [Electronic resource]. – URL: https://www.zhinst.com/europe/en/resources/principles-of-pid-controllers
29. Goodwin G. C., Graebe S. F., Salgado M. E. Control System Design / G. C. Goodwin, S. F. Graebe, M. E. Salgado. – Vol. 240. – Upper Saddle River: Pearson Prentice Hall, 2001. – 920 p.
30. Dorf R. C., Bishop R. H. Modern Control Systems / R. C. Dorf, R. H. Bishop. – Pearson Education, Inc., Pearson Prentice Hall, 2008. – 1046 p.

31. PID Controllers in Control Systems [Electronic resource]. – URL: https://www.electrical4u.com/pid-control/
32. Basics of PID Controllers: Working Principles, Pros & Cons [Electronic resource]. – URL: https://www.integrasources.com/blog/basics-of-pid-controllers-design-applications/
33. Ziegler J. G., Nichols N. B. Optimum Settings for Automatic Controllers / J. G. Ziegler, N. B. Nichols // Transactions of the ASME. – 1942. – Vol. 64, No. 6. – P. 759-768.
34. Cohen H., Coon G. A. A Linear Operator Approach to the Analysis of Closed-Loop Servomechanisms / H. Cohen, G. A. Coon // Journal of the Franklin Institute. – 1953. – Vol. 256, No. 6. – P. 489-512.
35. Hladchenko V. Computer Vision Technology for Quadrotor / V. Hladchenko, A. Klipa // Політ. Сучасні проблеми науки: XXIV міжнар. наук.-практична конф. молодих учених і студентів, 2-5 квітня 2024 р.: тези доп. – К.:НАУ, 2024. – С. 124-125. (https://drive.google.com/file/d/10QBFWc6Xdpw_0F9MgeJgqcP2gHOWUEf2/view)
36. AABB Bounding Box and Item Restrictions [Electronic resource]. – URL: https://docs.sansar.com/untitled/creating-in-sansar/importing-things-to-sansar/aabb-bounding-box-and-item-restrictions
37. Tian L., Fan C., Ming Y. Multiple Scales Combined Principle Component Analysis Deep Learning Network for Face Recognition / L. Tian, C. Fan, Y. Ming // Journal of Electronic Imaging. – 2016. – Vol. 25, No. 2.

APPENDIX A.

```
from djitellopy import tello
import cv2
import numpy as np
import time

# Drone initialization procedure
drone = tello.Tello()
drone.connect()
print(drone.get_battery())

# Drone initialization
drone.streamon()
drone.takeoff()
drone.send_rc_control(0, 0, 25, 0)
time.sleep(0.6)

w, h = 360, 240 # Size of main scene
lr_velocity = 0 # Initialization of roll control
fb_velocity = 0 # Initialization of pitch control (Forward-Backward movement)
updwn_velocity = 0 # Initialization of thrust control
yaw_velocity = 0 # Initialization of yaw control

# Safe Zone range
forward_backwardRange = [6200, 6600]
# Yaw motion PD controller
Kp_yaw = 0.75
Kd_yaw = 0.08
Ki_yaw = 0
pid_yaw = [Kp_yaw, Kd_yaw, Ki_yaw]

# Forward-Backward motion PD controller
Kp_fb = 1.08
Kd_fb = 0.2
Ki_fb = 0
pid_fb = [Kp_fb, Kd_fb, Ki_fb]

# Up-Down motion PD controller
Kp_updwn = 1
Kd_updwn = 0.1
Ki_updwn = 0
pid_updwn = [Kp_updwn, Kd_updwn, Ki_updwn]

# Previous error initialization
pErrorX_yaw = 0
pErrorX_fb = 0
pErrorY = 0

# Central Box Parameters
center_box_size = 60
mainCentralX = w // 2
mainCentralY = h // 2
mainCentralBox = [(mainCentralX - center_box_size // 2, mainCentralY -
center_box_size // 2),
                  (mainCentralX + center_box_size // 2, mainCentralY +
center_box_size // 2)]

# Finding face algorithm
def findFace(img):
    faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
```

```
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(imgGray, 1.2, 4)

    faceListCentral = []
    faceListArea = []

    for (x, y, w, h) in faces:
        faceCentalX = x + w // 2 # Central position of face according to X axis
        faceCentralY = y + h // 2 # Central position of face according to Y axis
        area = w * h
        faceListCentral.append([faceCentalX, faceCentralY])
        faceListArea.append(area)

    if len(faceListArea) != 0:
        i = faceListArea.index(max(faceListArea))
        return img, [faceListCentral[i], faceListArea[i]]
    else:
        return img, [[0, 0], 0]

def trackFace(info, w, h, pid_yaw, pid_fb, pid_updwn, pErrorX_yaw, pErrorX_fb,
pErrorY):
    area = info[1]
    x, y = info[0]
    fb_velocity = 0
    updwn_velocity = 0
    yaw_velocity = 0

    # Yaw error calculation
    errorX_yaw = x - w // 2

    # Altitude error calculation
    errorY = mainCentralY - y

    # Forward-Backward error calculation based on area
    target_area = (forward_backwardRange[0] + forward_backwardRange[1]) / 2
    errorX_fb = target_area - area

    # Yaw motion
    yaw_velocity = pid_yaw[0] * errorX_yaw + pid_yaw[1] * (errorX_yaw -
pErrorX_yaw)
    yaw_velocity = int(np.clip(yaw_velocity, -100, 100))

    # Up-Down motion
    updwn_velocity = pid_updwn[0] * errorY + pid_updwn[1] * (errorY - pErrorY)
    updwn_velocity = int(np.clip(updwn_velocity, -100, 100))

    # Forward-Backward motion
    fb_velocity = pid_fb[0] * errorX_fb + pid_fb[1] * (errorX_fb - pErrorX_fb)
    fb_velocity = int(np.clip(fb_velocity, -20, 20))

    if area == 0:
        yaw_velocity = 0
        fb_velocity = 0
        updwn_velocity = 0
        errorX_yaw = 0
        errorX_fb = 0
        errorY = 0

    drone.send_rc_control(lr_velocity, fb_velocity, updwn_velocity, yaw_velocity)
    return errorX_yaw, errorX_fb, errorY

# Main loop
while True:
    img = drone.get_frame_read().frame
    img = cv2.resize(img, (w, h))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img, info = findFace(img)

    # Draw Central Box
    mainBox = cv2.rectangle(img, mainCentralBox[0], mainCentralBox[1], (255, 0,
0), 1)
    cv2.circle(img, (mainCentralX, mainCentralY), 2, (0, 0, 255), cv2.FILLED)

    # Draw central cross
    cv2.line(img, (mainCentralX - 180, mainCentralY), (mainCentralX + 180,
mainCentralY), (255, 255, 255), 1)
    cv2.line(img, (mainCentralX, mainCentralY - 120), (mainCentralX, mainCentralY
+ 120), (255, 255, 255), 1)

    pErrorX_yaw, pErrorX_fb, pErrorY = trackFace(info, w, h, pid_yaw, pid_fb,
pid_updwn, pErrorX_yaw, pErrorX_fb, pErrorY)

    cv2.imshow("Output", img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        drone.land()
        cv2.destroyAllWindows()
        break
```