Hüsnü Halid Alabay

A Master's Thesis

AGU 2022

COORDINATED TARGET DETECTION AND TRACKING BY DRONES USING DISTANCE AND VISION

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE OF ABDULLAH GUL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Hüsnü Halid Alabay

June 2022

COORDINATED TARGET DETECTION AND TRACKING BY DRONES USING DISTANCE AND VISION

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE OF ABDULLAH GUL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Hüsnü Halid Alabay

June 2022

SCIENTIFIC ETHICS COMPLIANCE

I hereby declare that all information in this document has been obtained in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name-Surname: Hüsnü Halid Alabay
Signature:

REGULATORY COMPLIANCE

M.Sc. thesis titled Coordinated Target Detection And Tracking By Drones Using Distance And Vision has been prepared in accordance with the Thesis Writing Guidelines of the Abdullah Gül University, Graduate School of Engineering & Science.

Prepared By	Advisor
Hüsnü Halid Alabay	Asst. Prof. Samet Güler
Signature	Signature

Head of the Electrical and Computer Engineering Program

Assoc. Prof. Kutay İçöz

Signature

ACCEPTANCE AND APPROVAL

M.Sc. thesis titled Coordinated Target Detection And Tracking By Drones Using Distance And Vision and prepared by Hüsnü Halid Alabay has been accepted by the jury in the Electrical and Computer Engineering Graduate Program at Abdullah Gül University, Graduate School of Engineering & Science.

…….. /…….. / ……..

JURY:

Advisor : Asst. Prof. Samet Güler

Member : Asst. Prof. Burak Tekgün

Member : Asst. Prof. Harun Yetkin

APPROVAL:

The acceptance of this M.Sc. thesis has been approved by the decision of the Abdullah Gül University, Graduate School of Engineering & Science, Executive Board dated ….. /….. / …….. and numbered ……..……. .

…….. /…….. / ……..

(Date)

Graduate School Dean

Prof. Dr. İrfan Alan

ABSTRACT

COORDINATED TARGET DETECTION AND TRACKING BY DRONES USING DISTANCE AND VISION

Hüsnü Halid Alabay
M.Sc. in Electrical and Computer Engineering
Advisor: Asst. Prof. Samet Güler

June 2022

Robot autonomy refers to the ability to carry out objectives by perceiving the environment and deciding on the actions required without human interruption. Although autonomous aerial robots offer big advantages in our daily life, online localization and control remain the biggest challenge lying ahead of aerial robot implementations. For single robot applications, GPS, and motion capture (mocap) systems can be utilized for outdoor and indoor applications, respectively. However, when it comes to multi-robot systems, the relative localization problem needs to be solved beyond the single robot localization problem. Furthermore, GPS signals are not available everywhere, and mocap systems limit the application space of multi-robot systems. Motivated by the industrial application scenarios, we address the relative localization and docking problem in multi-drone systems where drones do not utilize any external infrastructure for localization. We consider a two-drone system that aims at docking a target object which consists of an ultrawideband (UWB) distance sensor. The drones are equipped with UWB sensors and cameras and try to localize the target object and dock around it in a pre-defined configuration in the absence of GPS and magnetometer sensors and external infrastructures. We design an extended Kalman filter based on the dynamic model of the drone-target configuration that fuses the distance and vision sensor outputs. Particularly, we use the YOLO algorithm for the bearing detection between the drones and the target. Next, we devise and implement a switching-based distributed formation control algorithm and integrate it into the estimation algorithm. We demonstrate the performance of our algorithm in several simulation studies in a realistic Gazebo environment. Finally, we provide primary experimental results and a roadmap to the full implementation of the system.

Keywords: Multi-drone systems, Image processing, Estimation algorithms, Docking

ÖZET

MESAFE VE GÖRÜNTÜ KULLANAN DRONLAR İLE KOORDİNE HEDEF TEŞHİSİ VE TAKİBİ

Hüsnü Halid Alabay
Elektrik ve Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans
Tez Yöneticisi: Dr. Öğr. Üyesi Samet Güler
Haziran 2022

Robot otonomisi, çevreyi algılayarak hedefleri gerçekleştirme ve insan müdahalesi olmadan gerekli eylemlere karar verme yeteneğini ifade eder. Otonom hava robotları günlük hayatımızda büyük avantajlar sunsa da çevrimiçi yer tespiti ve kontrol, hava robotu uygulamalarının önündeki en büyük zorluk olmaya devam etmektedir. Tek robot uygulamaları için, GPS ve hareket yakalama (mocap) sistemleri, sırasıyla dış mekân ve iç mekân uygulamaları için kullanılabilir. Ancak çok robotlu sistemler söz konusu olduğunda, göreli lokalizasyon probleminin tek robot lokalizasyon probleminin ötesinde çözülmesi gerekmektedir. Ayrıca, GPS sinyalleri her yerde mevcut değildir ve mocap sistemleri, çoklu robot sistemlerinin uygulama alanını sınırlar. Endüstriyel uygulama senaryolarından motive alarak, drone'ların yerelleştirme için herhangi bir harici altyapı kullanmadığı çoklu drone sistemlerinde göreceli lokalizasyon ve yerleştirme problemini ele alıyoruz. Ultra geniş bant (UWB) mesafe sensöründen oluşan bir hedef nesneye kenetlemeyi amaçlayan iki dronlu bir sistem düşünüyoruz. İHA'lar UWB sensörleri ve kameraları ile donatılmış olup, GPS, manyetometre sensörleri ve harici altyapıların yokluğunda hedef nesneyi lokalize etmeye ve önceden tanımlanmış bir konfigürasyonda etrafına kenetlenmeye çalışırlar. Mesafe ve görüş sensörü sonuçlarını birleştiren drone-hedef konfigürasyonunun dinamik modeline dayalı genişletilmiş bir Kalman filtresi tasarlıyoruz. Özellikle insansız hava araçları ile hedef arasındaki açı tespiti için YOLO algoritmasını kullanıyoruz. Ardından, anahtarlama tabanlı bir dağıtılmış formasyon kontrol algoritması tasarlayıp uyguluyor ve bunu tahmin algoritmasına entegre ediyoruz. Algoritmamızın performansını gerçekçi bir Gazebo ortamında çeşitli simülasyon çalışmaları üzerinde gösteriyoruz. Son olarak, sistemin tam olarak uygulanması için birincil deneysel sonuçlar ve bir yol haritası sunuyoruz.

Anahtar Kelimeler: Çoklu drone sistemleri, Görüntü işleme, Tahmin algoritmaları, Kenetlenme

Acknowledgments

I would like to express my gratitude to Dr. Samet GÜLER, who helped me to complete this study by giving me a great deal of time and valuable contributions.

GCRIS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

BEV	Bird's Eye View
CNN	Convolutional Neural Network
DOF	Degree of Freedom
IOT	Internet of Things
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
GPS	Global Positioning System
HIL	Hardware-in-the-Loop
HMM	Hidden Markov Model
HOGS	Histogram of Oriented Gradients
IMU	Inertial Measurement Unit
LIDAR	Light Detection and Ranging
MAP	Maximum A Posteriori
MRS	Multi-Robot System
ROS	Robot Operating System
RSS	Received Signal Strength
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SURF	Speeded Up Robust Features
SVR	Support Vector Regression
ToF	Time-of-Flight
UAV	Unmanned Aerial Vehicle
UHFRFID	Ultra-High Frequency Radio-Frequency Identification
UWB	Ultra-Wideband
VINS	Visual Inertial Navigation System
VTOL	Vertical-Take-Off-and-Landing
WFF	Wall-Floor Features
YOLO	You Only Look Once

To my family, friends, and all loved ones.

Chapter 1

Introduction

To localize an autonomous robot in an unknown environment, the Global Positioning System (GPS) is used which gathers position data of the sensor from at least four different satellites around the world. GPS data is one of the best ways to localize robots in an unknown environment, but this system has some problems which avoid us to gather position data. In this system, we are getting GPS signals from at least four different satellites, which compute the position and distance between the GPS sensor and satellites. Those signals are mainly radio signals and because of that, they cannot penetrate dense structures such as mountains or concrete buildings. So, in urban or indoor environments, which are surrounded by structures, we cannot use GPS signals efficiently. Because of this phenomenon, we need different localization systems for urban and indoor environments. To be able to localize our agents in these environments, we need to create a system that can gather different sensor data which help us to sense our environment and an estimator which uses sensor data to estimate our position to localize ourselves. For this kind of system, which does not use GPS signals, distance measurements and visual data are one of the best sensor data to help us to localize ourselves. With by using a specific sensor, we can get different distance data from different objects around us and by using visual data, we can use image processing techniques to detect a pre-defined object to localize ourselves with respect to the detected object. To create a system that works as mentioned, we can combine different sensors and algorithms to create a localization system. With the proposed study, a localization technique that uses distance measurements and visual data from their respective sensors to analyze their environment autonomously create and estimate their position with respect to objects in the environment. To be able to enhance this study's effectiveness, a harsh environment is accepted to mimic GPS signal losses which are realized by not using magnetometer sensors that create rotational data for drones. In this chapter, state-of-art works are summarized to get a better understanding of the application of visual navigation.

1.1 Related Works

Within current state-of-art works on Visual navigation, many of them focus on drone localization that combine distance measures and computer vision techniques with autonomous control algorithms to navigate robots in an environment. To gather environmental data, camera systems & distance measurements used and those data matched with pre-installed maps. To be able to summarize general concept in state-of-art, we can separate them with some main topic as follows.

To devise a localization system without GPS signal aid, many previous works focused on fusion of several sensor data with different characteristics. Vision sensors provide extensive information about the environment, which can be processed on a robot's computational boards to compute feasible paths. Recently, localization objective was separated from the mapping and path planning objectives to reduce the onboard computational complexity. Accordingly, vision sensors have been implemented heavily for localizing robots in a multi-robot system. Here, we provide a summary of the recent vision-based localization frameworks in the literature. We start with the general perspective where a robot tries to self-localize in unknown environments. Then, we narrow down to vision-based relative localization approaches for multi-robot systems.

A typical approach for global localization is to acquire onboard visual data and identify the robot location in a Geographic Information Systems dataset such as Google Maps or BING Maps. In [1], on-board visual data and Google maps aerial images is combined with Histograms of Oriented Gradients (HOGS) to create Histogram data of the landscape which is used to match between geo-tagged images. Like that work, in [2-4], Circular-HOG features is used to create matches between geo-tagged aerial images and on-board visual data. After the matching process, drone localization is completed by removing poor matches with a filtering process. In [5-7], visual data are compared with pre-installed geo-tagged images by using Scale-invariant feature transform (SIFT) algorithm, which detect features in images and check for comparison, to localize aerial vehicle in the map. Also, the localization results are tested by creating 3D maps and building facades. Similarly, in [8], oblique map is created by taking a large number of pictures in pre-defined patterns and fusing.

References [9,10] processed the on-board visual data with Kanade- Lucas-Tomasi feature detection algorithms to detect edges and corners of the paths and pavements. In this approach, gyro, accelerometer, and barometric pressure data are fused with vision

data in a Kalman filter to estimate an aerial vehicle's position. Sharing obtained data is similar approach to fusing for localization. In [11-13], obtained visual data from different cameras on different aerial vehicles shared between each aerial vehicle in the system to create general map which can be used for localization.

Several works take advantage of designated objects in the environment in localizing aerial vehicles. In [14,15], Quick Response (QR) code and QR code-like patterns with known positions are used for localizing aerial vehicles. A vehicle can extract its global position when it detects such patterns or objects with its onboard camera. As an alternative to QR detection, Convolutional Neural Network (CNNs) are applied in [16,17] to detect patterns on urban districts or environment such as people, animals, or cars. By detecting those features from visual data, localization can be extracted from detected features' positions. Such computer vision methods can be used for entertainment and educational activities with drones as well. For instance, human pose is used to localize and lock an aerial vehicle around the detected pose in [18]. Human face recognition technique is used in [19], which can be used in a classroom so that the aerial vehicle locks itself with respect to teacher's position and record the lecture.

In [20] visual sensor on a drone is used to scan an environment to detect intruder aerial vehicles. Localization is performed by detecting the intruder vehicle and calculating its relative pose and distance. A similar concept is demonstrated in [21] where a binocular like gadget with visual sensor is used to detect aerial vehicles with deep neural networks. In [22,23], the same concept is studied by using multiple visual sensors which help us to get distance measure by using angle difference in views.

Primarily, simultaneous localization and mapping (SLAM) dominated the localization research track with its complete solution methodology to both the localization and mapping problems. In [24,25], SLAM algorithm is used to locate vehicles by detecting environmental features such as streets and building in outdoor environments. In [24] by saving local position of the vehicle in each step and matching it with Google Street Maps' database to reduce the drifts in localization caused by the visual approach. In [25], by using on-board visual data, buildings detected and matched with Google Maps images and vehicle localized by calculating angle change in detected features.

To be able to use Google Street maps' images in localization, top of the buildings of facades of the building can be used as visual data. Many satellite images have defined angle and oblique features which make them enough to use in visual image comparison. In [26], position data of a vehicle is estimated with onboard visual data and estimation is

combined with Hidden Markov Model (HMM) to localize the vehicle. Google Street images is modified and used along with on-board visual data with SIFT algorithm to create noisy position data which has 4 - 16-meter error. By combining these data with HMM, localization of the vehicle can be estimated. Similarly, in [27], facades of the buildings in Google Maps images are used for localization. First, outlines of the buildings' roofs extracted in both satellite images and Bird's eye view (BEV) images which can be matched and indicate buildings positions. Then facades of the buildings extracted from top to bottom in BEV images. After that on-board visual data and extracted facades compared with Self-similarity descriptor computation which results with estimated position of the vehicle.

On out-door localization, images are generally used to calculate location of the vehicle or position estimation. Image or video sequences are not used usually because of occlusion, fast camera motion, and pose variation problems. In [28], multi-object tracking, and 3D localization scheme are used to estimate aerial vehicle's position by using deep learning algorithms. The proposed system can calculate 3D coordinates of other objects in the environments while detecting and tracking them. Moreover, view-point changes can affect visual localization and make it hard to detect predefined images in a dataset. In [29], a novel generative model for descriptor learning is proposed which can handle seasonal changes in the environment. Semantic localization technique is used which can handle extreme appearance changes which include weather, season, and illumination. By capturing high-level geometric and semantic information from visual data, accurate camera pose estimation is performed in that work.

In indoor environments, different practices or modified versions of outdoor techniques can be used to gather position data of an aerial vehicle. Several works combine map creation with localization techniques to create position data for aerial vehicles. In [30,31], CNNs are used to detect interior features such as doors, dead-ends, and edges of walls to create 3D semantic map of indoor environments. To reduce pose estimations, Octree method and Support Vector Regression (SVR) methods are used. In [32], vistas, distance features gathered from parallel tracking, are generated to localize aerial vehicles. By detecting and tracking Wall-Floor Features (WFFs), odometry data and visual data can be fused to create position data for the system. In [33], a monocular visual-inertial navigation system (VINS) is created which has small footprint due to only consisting inertial measurement unit (IMU) and a camera, used to create maps for indoor environment. Online trajectory planner which operates on three-dimensional map, used

to guarantee safe navigation for aerial vehicle. In [34], visual SLAM technique is used to create global map with multiple cameras located at different points. The acquired vision data are compared to estimate the inter-camera poses. In [35], visual data is combined and used with Ultra-wideband (UWB) sensors to localize aerial vehicles in tight environments. As an alternative solution, a new camera system based on a Time-of-Flight (ToF) concept is used to localize an aerial vehicle in in-door environments in [36]. ToF camera can acquire dense depth maps with high frame rates which is suitable for localization.

Despite their several remarkable advantages, vision sensors may not be sufficient for self- or relative localization of aerial vehicles. Notably, vision sensors are prone to adversarial ambient conditions such as poor lighting and limited field-of-view. Therefore, vision sensors are usually applied with another sensor type to exploit the practicality and robustness of both worlds. A typical approach is to fuse the vision data with distance sensor data. We now summarize some important results about the distance-based localization solutions.

In [37], a multi-robot Cooperative Localization (CL) system is proposed with custom covariance intersection-based algorithm. Each robot in the system has proprioceptive sensors for ego-motion, exteroceptive sensors for other robots' relative pose and communication device that help robots to share information. By sharing and receiving information with other robots, relative position between the robots is calculated. In [38], a CL system is devised with a distributed Maximum A Posteriori (MAP) estimator which helps system to reduce the memory and processing requirements. Furthermore, fusion of all data in a team of robots are used in [39] to overcome single robot's calculation error over time. Each robot in the system try to localize itself in the environment by fusing on-board sensors' data such as cameras or IMU but as time passes, error rate will increase and acquired location will be corrupted. To overcome this problem, on top of fusing on-board sensors together, measurements from robot couples will be added which can be used to improve localization accuracy. Similarly, in [40], localization of a team of robot is done by fusing proprioceptive sensors, exteroceptive sensors, and data from other robots in the team. By using Extended Kalman Filter (EKF), relative observation between robots can be acquired. Reference [41] eliminates the need for prior knowledge about the robots' initial relative poses. In that system, robots share their sensor data with other robots to match any common observation which help them to calculate pose transformation in the robots' reference frames.

Trilateration techniques can be used for localization too. In [42,43], by applying a single trilateration for each sensor in a deployment area, localization of an aerial vehicle with distance measures can be achieved. In the system, all the sensors measured at least three times with different orientations to plan a static path for the aerial vehicle.

For outdoor localization with distance data, UWB sensors are used to get distance measurement between vehicles and an anchor point. Because in outdoor environments, we do not have dense obstacles, the way of gathering measurements will be different. In [44], Drone Range-Free localization algorithm is created for aerial vehicle localization. Drone Range-Free algorithm try to localize Internet-of-Things (IOT) device, that fixed on aerial vehicle, on the intersection of the UWB sensors measurement. To overcome noisy measurement from UWB sensor, range-free assumption for UWB relaxed and nearly perfect measurement circles achieved. In [45], localization of single aerial vehicle is done by using UWB sensors. Location of single aerial vehicle is detected with one fixed anchor points and for swarm aerial vehicles, active swarm members change their role to enhance the localization performance.

In indoor environments, we can have many different objects or features which can be used to get distance data. To be able to localize itself and fly in indoor environment, aerial vehicles should be able to determine collision-free path. In [46], ultra-high frequency radio-frequency identification (UHFRFID) tags data are fused in Kalman Filters to localize vehicle in an indoor environment. Position of the tags are known in the environment and by calculating reckoning of the data, position of the vehicle is estimated. In [47], to generate collision-free path, SLAM is used to detect the environment. By gathering data in different altitude levels, environment that aerial vehicle flies identified and gathered data is used to get a match for estimation of the position. In each iteration, estimated pose will come closer to true position of the aerial vehicle and localization can be done. In [48], depth maps are used to localize aerial vehicle in an indoor environment.

Although several vision- and distance-based approaches are proposed for indoor environments, a complete docking strategy for collaborative aerial vehicles remains an open problem. In this thesis, we aim at designing and implementing a coordinated target detection and tracking framework for two vertical take-off and landing (VTOL) aerial vehicles by employing a monocular camera and UWB distance sensors. Our approach differs from the previous work in several points. First, the proposed approach relies on the onboard exteroceptive sensors (camera and UWB) and does not take advantage of magnetometers or external infrastructure. Second, we propose an EKF algorithm which

contains two measurement modes based on the availability of the target object detection by the onboard camera sensor. Thus, the proposed method inherits robustness and resiliency. Finally, as opposed to the SLAM approaches which may fail if the environment does not contain sufficiently rich features, the proposed solution does not require dense feature environment.

1.2 Organization

The following parts of the thesis are organized as follows. In Chapter 2, the proposed system is explained by analyzing each system and algorithm used and the connection between them. In Chapter 3, the General framework designed for drones is explained. Extended Kalman Filter which is used to estimate distance measurements acquired from the environment explained and the control algorithm is analyzed. In Chapter 4, simulation results are provided to show how estimated distance measurements follow ground truth values. analyzation of the simulation results discussed. In Chapter 5, the setup of lab experiments of the proposed method is explained. Hardware, software, and operational procedures are described, and results are illustrated. After that, Chapter 6 conclude the work and future studies discussed.

Chapter 2

System Definition

We consider two drone vehicles in an indoor environment with a target object that can be detected and evaluated by the drones. The main goal is to design and evaluate a perception-decision mechanism for the drones to plan and execute a given mission in the environment. In this section, we define the vehicle model used, formulate the objectives in general terms, and provide a general picture for the desired characteristics of the system to be designed.

2.1 Drone Team Model

We consider a team of two vertical-take-off-and-landing (VTOL) aerial vehicles, specifically quadrotors or drones, named D_1 and D_2. Each drone contains a four-motor structure that provides an agile structure to fly in various environments with defined motion characteristics such as altitude and velocity. The drones sense their environment by omnidirectional ultrawideband (UWB) distance sensors and monocular camera sensors. We define the motion models and characteristics of the individual drones in the following section.

2.1.1 Drone Model

A typical quadrotor comprises the following components: an autopilot (flight controller), DC motors connected through electronic speed controllers (ESC), a battery, a complementary computer, and a variety of onboard sensors mounted based on the objective. Each of these components takes care of a different part of the system such as the flight controller applying basic control on the drone. To be able to fly with a quadrotor

system, we have four motors that two of them rotate counterclockwise and the other two rotate clockwise with their respective propellers which create enough lifting force for all the weight. In this system, to be able to analyze the motion model, we can start creating it by motor rotations, the center of gravity, etc.

In our system, we arranged our drones in a way that, they maintain their altitude constant on any surface, and they only move freely on the plane. There are two basic motion models for mobile robots defined based on the degree of freedom (DOF): Holonomic and non-holonomic. In non-holonomic models, a robot's total number of DOF is bigger than controllable DOF. For instance, in a four-wheeled ground robot, the total DOF can be counted as the capability of moving on the x-axis, y-axis, and the heading direction. However, those kinds of robots are mainly controlled with two parameters which are forward/backward acceleration and the angle of the rotation, which create two DOF. On the other hand, in holonomic systems, one can control movement on the x and y axes and the direction of the robot which creates the same amount of DOF with a total DOF, three.

A VTOL drone has a holonomic motion model which enables motion in all x, y, z axes. However, as we explain in detail in the next chapter, to alleviate the design of the localization algorithm in the absence of the magnetometer sensor, we restrict the motion capability of the drones so that its motion resembles the non-holonomic model. Essentially, such a restriction stems from the inability of measuring the heading angle of the drones.

2.1.2 Kinematic Model

In our system, we have several coordinate frames in the environment which define the coordinates for our drones. In total, we have two coordinate frames which are the ground coordinate frame, $\mathscr{E} = \{Ex, Ey, Ez\}$, and the body coordinate frame, $\mathscr{B} = \{Bx, By, Bz\}$. For a proper representation of the drones' motion, we need to create a connection between these coordinate frames. The body coordinate frame is attached to our drones, and it moves with the drones' motions while the ground coordinate frame does not change with motion and stays as a reference point for the rest of the system. Because of these differences in motion, the transformation between them should be calculated and used for drones' motion.

To find the transformation between those coordinate frames, we define the relative position $r = [x, y, z]^T$ between the ground (fixed) coordinate frame $\mathscr{E}$ and the drone's body coordinate frame $\mathscr{B}$. By creating a transformation matrix, R_B^E, drone's relative orientation to ground coordinates can be defined. While creating this transformation, quaternion or Euler angles can be used. In Euler transformation, three rotation angles, yaw, pitch, and roll angles of the drone can be used as follows:

$$\boldsymbol{R_i^b = R(\phi, x) * R(\theta, y) * R(\psi, z) \quad =} \tag{2.1}$$

$$\begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{cos(\phi)} & \boldsymbol{sin(\phi)} \\ \mathbf{0} & \boldsymbol{sin(\phi)} & \boldsymbol{sin(\phi)} \end{bmatrix} \begin{bmatrix} \boldsymbol{cos(\theta)} & \mathbf{0} & \boldsymbol{sin(\theta)} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \boldsymbol{-sin(\theta)} & \boldsymbol{cos(\theta)} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{cos(\psi)} & \boldsymbol{-sin(\psi)} & \mathbf{0} \\ \boldsymbol{sin(\psi)} & \boldsymbol{cos(\psi)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$$= \begin{bmatrix} cos(\psi)cos(\theta) & cos(\psi)sin(\theta)sin(\phi) - sin(\psi)cos(\phi) & cos(\psi)sin(\theta)cos(\phi) + sin(\psi)sin(\phi) \\ sin(\psi)cos(\theta) & sin(\psi)sin(\theta)sin(\phi) + cos(\psi)cos(\phi) & sin(\psi)sin(\theta)cos(\phi) - sin(\phi)cos(\psi) \\ -sin(\theta) & cos(\theta)sin(\phi) & cos(\theta)cos(\phi) \end{bmatrix} \tag{2.2}$$

2.1.3 Dynamic Model

To create dynamic model of the drone, we need to focus on two equations which are transformation equations and rotational motion equations. Rotational motion equations are coming from drone frame and motion of the motors that create rotation. By using Newton-Euler method, we can define forces on the drone frame axes as noted below:

$$M_b = J\dot{\omega} + \omega x J\omega + \omega x \begin{bmatrix} 0 \\ 0 \\ J_r\Omega_r \end{bmatrix} \tag{2.3}$$

where M_b is the total moment on all axes on drone frame, B, J is stand for diagonal inertia matrix, ω is angular velocity vector, $\dot{\omega}$ is the drone's angular acceleration vector, J_r is the inertia of the rotor and ω_r is the drone's z-axis rotational imbalance.

Thrust created by motors generate moment on drone's arm that help drone to rotate. To be able to calculate these moments, we can use following equations:

$$F_i = b\Omega_i^2 \tag{2.4}$$

$$M_i = bl\Omega_i^2$$

Which F_i is the motor's thrust, b is the aerodynamic force constant, Ω is the motor's angular velocity, M_i is the motor's torque and l is the length of the drone's arm which is the distance between motor and the center of the drone. So, to calculate all the moments that effect drone from each axis, formulation can be written as follows:

$$M_B = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} lb(-\Omega_2^2 + \Omega_4^2) \\ lb(\Omega_2^2 + \Omega_2^2) \\ d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{bmatrix} \quad \textbf{(2.5)}$$

Which M_x , M_y , M_z are the total moment on x, y and z axis respectively and d is the rotational unbalance moment constant. After we define total moment on each axis like that, by using Newton's second law, we can update transformation equation from body coordinate frame to ground coordinate frame.

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + RF_B \quad \textbf{(2.6)}$$

Which F_B is the drone rotors' total thrust without gravitational force, m is the drone's mass, $\ddot{r}$ is the Newton's second law's acceleration and R is our transformation matrix. After those definitions, our total thrust force can be written as:

$$F_B = \begin{bmatrix} 0 \\ 0 \\ -b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \quad \textbf{(2.7)}$$

2.2 Problem Formulation

We address the problem of indoor navigation by a team of VTOL aerial vehicles. From a wider perspective, we are interested in steering robots toward a goal location in an indoor environment by utilizing onboard sensing mechanisms only, without any external sensing aids. Typically, a multi-robot system (MRS) takes advantage of external infrastructures in an indoor environment such as motion capture (mocap) systems or a set of distance

sensors. A mocap system consists of a set of infrared cameras that can detect markers onboard of a vehicle and generate the pose (position and orientation) information with high precision and accuracy. Although mocap systems can be a viable solution for small lab environments to aid in producing prototype solutions, they have two main drawbacks. First, the MRS motion is bound to the environment covered by the mocap system. As the environment size increases, e.g., in a large industrial environment, excessive number of cameras together with a ground station with high computational power would be needed. For instance, to cover a 10-meter by 10-meter room precisely, one may need at least 30 IR cameras. The second downside is the high cost.

The second viable option for localization in indoor environments is installing a set of distance sensors on the roof of a room and a sensor of the same type on each robot. Such sensor types include wi-fi access points, Bluetooth modules, and UWB sensors. The stationary sensors mounted at a high level are called anchors and are typically mounted at a safe distance apart from each other. One then measures the distances between the anchors and sensors and generate the 2-D or 3-D location estimates of the robots by using several geometric methods such as trilateration or optimal methods. The communication between Wi-fi and Bluetooth modules usually provide the received-signal-strength (RSS) measurements which can be converted to distance values, while UWB sensors use the time-of-flight (TOF) method to measure distance. UWB sensors are proved to be accurate, precise, and reliable [49]. With several ranging schemes such as single-sided, double-sided, two-way ranging, and time-difference-of-arrival, most commercial UWB sensors can generate accurate omni-directional distance measurements of up to 100 meters. Also, they can generate distance measurements even in non-line-of-sight cases where an object occludes the line between the anchor and tag sensors. Furthermore, the UWB scheme can be scaled up to multiple sensor configurations so that the distances between multiple sensors can be acquired.

In contrast to the traditional localization schemes with multiple anchors located at customized positions, we opt for mounting a UWB sensor onboard of the drones and on the goal location (which will be referred as target). In this scheme, each drone $D_i, i \in \{1,2\}$, and the target T consists of a UWB sensor which communicate to generate the three distances between themselves. Particularly, if we denote the positions of the drones $D_i, i \in \{1,2\}$, and the target T by $p_1 = [x_i, y_i, z_i]^{\mathrm{T}}, i \in \{1,2\}$, and $p_T = [x_T, y_T, z_T]^{\mathrm{T}}$, respectively, then the UWB sensors generate the measurements.

$$\rho_i = ||p_i - p_T||, i \in \{1,2\}, \quad \textbf{(2.8)}$$
$$\rho_0 = ||p_1 - p_2||,$$

where $||.||$ denotes the 2-norm.

Typically, a drone makes use of the magnetometer sensor in the autopilot to aid in finding its heading angle with respect to the Earth north. Generally, magnetometer sensors provide sufficiently accurate heading measurements in outdoor environments. However, in some indoor environments, magnetometer readings can be affected by external magnetic variations caused by several materials such as metals. Although magnetic shielding can prevent the variations to some degree, it may not suffice to eliminate the entire variation. Such variations may be tolerated in some applications, e.g., slowly moving ground robots, but in drone applications, they can cause deviations from the drone's desired path or even collisions. Therefore, providing solutions without using magnetometers brings a great advantage in most indoor applications. Hence, we aim at providing a navigation solution without a magnetometer, which prevents us from applying the common localization and path planning techniques.

We assume that each robot has a monocular camera sensor, and the target T contains a UWB sensor and has a pre-defined shape which can be discriminated by the robots by computer vision techniques. For instance, in an industrial environment, the drone team can have the capability of collaborative transportation of objects, and the target can be a robot manipulator at a fixed location which waits for the drone team for transferring an object (Figure 2.1). In such a scenario, the drone team should localize the target, approach it, and finally dock in front of the manipulator at certain configuration. Another scenario could be a worker with a helmet equipped with a UWB sensor waiting for the drone team to accomplish further human-robot collaborative tasks (Figure 2.2). The drones first need to approach the worker and dock around it, in the absence of external sensing aids.

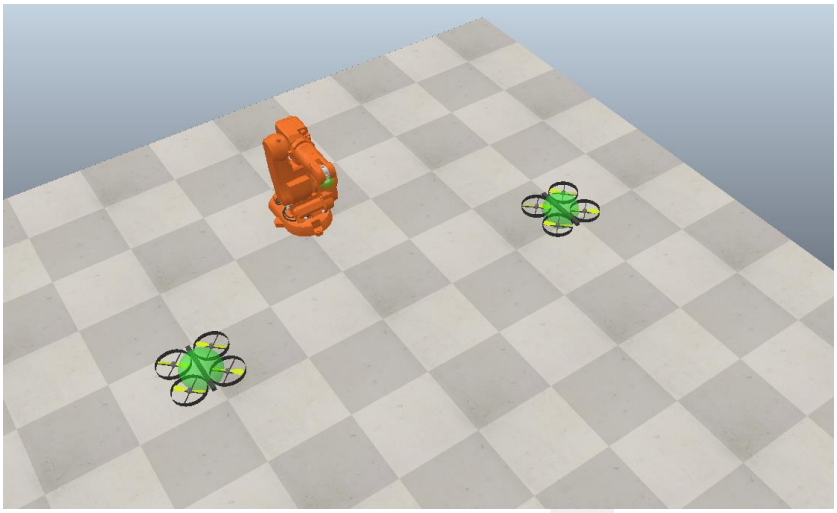

Figure 2.1 Scenario that shows drone team localize robot manipulator

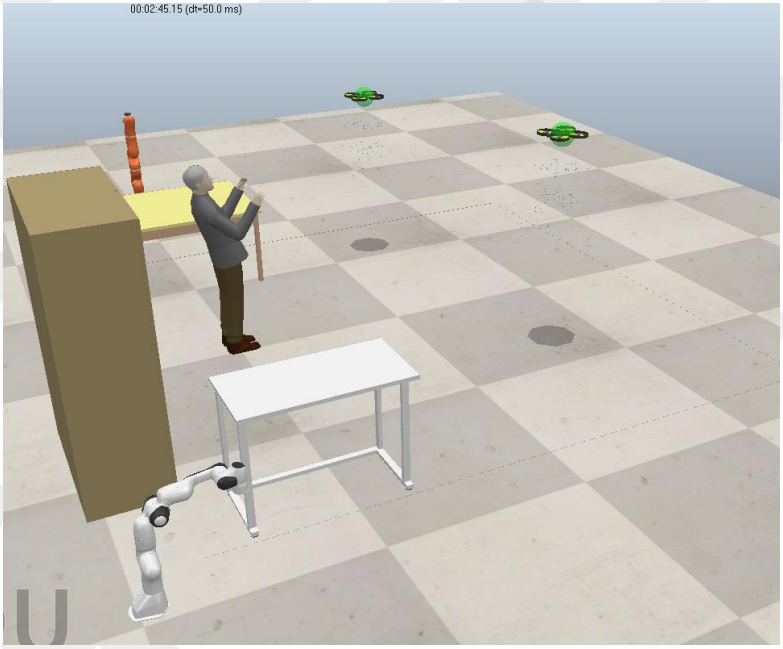

Figure 2.2 Scenario that shows drone team docks worker

We pose our main objective as follows. The drone team $\{D_1, D_2\}$ aim at searching and navigating through the target T by utilizing its onboard sensing and computational units solely, without using magnetometer or any external sensing aid such as mocap systems. Under the assumptions set forth thus far, we decompose the main objective into two parts:

- **Objective 1:** Design and implement a localization algorithm to generate the relative positions between the robots and the target,
- **Objective 2:** Design and implement a high-level motion controller to steer the robots toward the target and dock them at a certain relative configuration.

Objective 1 includes the design of a relative localization mechanism between the drones D_1, D_2 and the target T without any magnetic sensing aids. Thus, the scenario imposed in Objective 1 can be interpreted that the drones cannot measure their positions and heading angles with respect to a fixed frame. This constraint poses a big challenge which we tackle by exploiting the capabilities of the UWB sensors and computer vision techniques. Moreover, Objective 2 requires solving a demanding control task for a drone in the absence of a fixed frame, which poses another challenge since most control algorithms require the knowledge of a fixed frame. In the next chapter, we present our solution method for both objectives.

Chapter 3

Indoor Navigation

This chapter proposes a solution to the objective given in the previous chapter. First, we design a relative localization algorithm by utilizing the onboard capabilities of the drones. Then, we present our control framework which aims at driving the drones to a pre-defined configuration around the target. We follow a systematic approach so that the two modules are integrated into one framework directly.

3.1 General Framework

We address the problem of indoor navigation in two sub-parts: Relative localization and control. We demonstrate the block diagram of the proposed framework in Figure 3.1. The system is composed of two drones and a stationary target. Drone D_1 includes a sensing module, localization module, and a control module whereas drone D_2 includes a sensing module and a control module. The drones share the same wi-fi network over which some estimation parameters are communicated. We note that such a common network is not necessary for the framework; the onboard communication can be handled by the UWB sensors as well.

The sensing module consists of the onboard camera and UWB sensors and generates the camera image frame and the distances ρ_i. Since all UWB distance data can be acquired in one UWB sensor, we design drone D_1 as the master so that it runs the estimation algorithm and transmits the estimation results to drone D_2. Thus, we avoid the extra computational burden on the drone D_2. Furthermore, each drone implements a computer vision algorithm for object detection and relays the processed image information to its high-level controller.

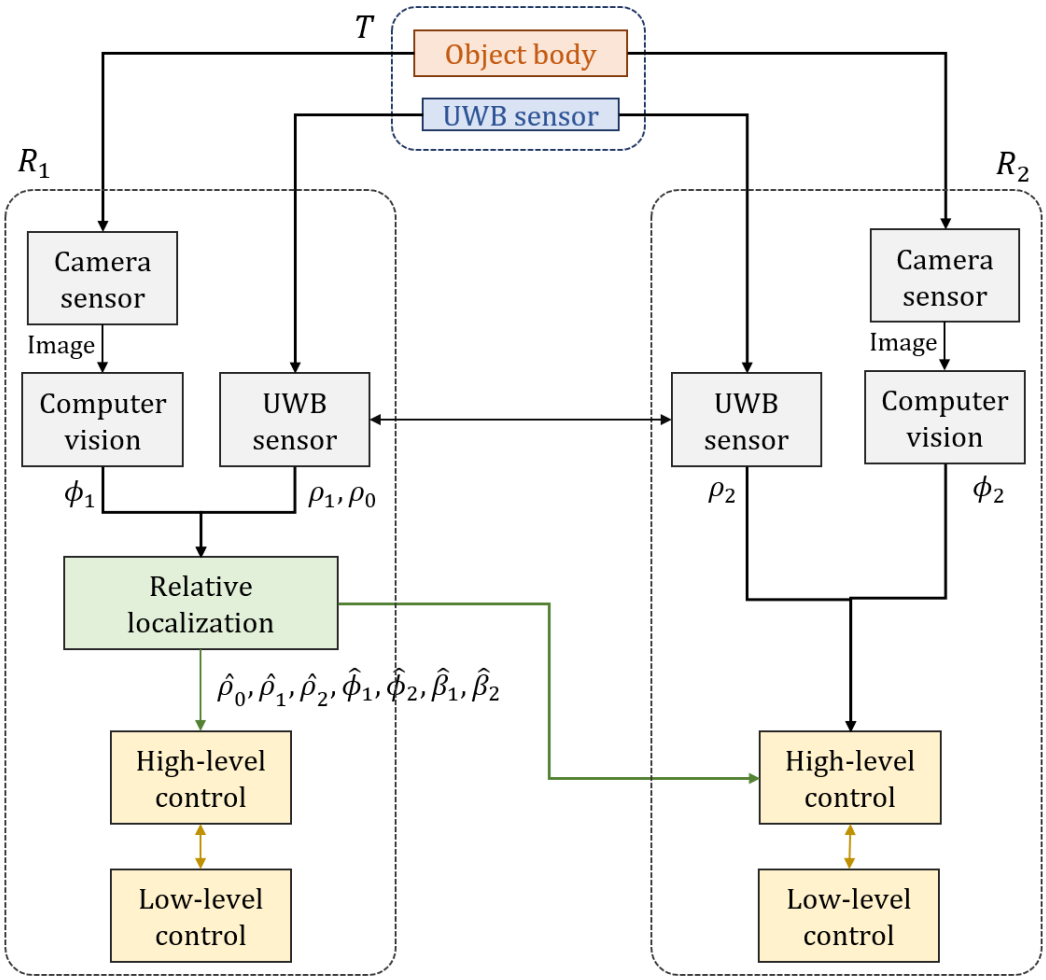

Figure 3.1 Block diagram for the system

The high-level control unit implements the path planning control algorithm and generates the high-level motion control commands, which are then relayed to the low-level motion controller. Assuming planar motion, the high-level controllers generate the desired velocity commands. The low-level controller block is responsible for receiving the velocity commands and generating the required torques accordingly by fusing the IMU and optical flow sensor data. We detail the localization and the control modules in the following sections.

3.2 Localization Algorithm

We now derive the system kinematics for the realization of the localization algorithms. We assume that the drones fly at a fixed altitude $\bar{h}$, i.e., $z_i = \bar{h}, i \in \{1,2\}$. Thus, we

assume that the drones move on a plane. Consider Figure 3.2 where drones D_1, D_2, the target T, and the coordinate frames are seen. We focus on the geometry of the triangle $\{D_1, D_2, T\}$. Let the line between target T and drone D_i be denoted by $l_i, i \in \{1,2\}$. We denote the angle between the line l_i and the y-axis of drone D_i measured counter-clockwise by $\phi_i \in [-\pi, \pi)$. Notably, the y-axis direction of a drone corresponds to the drone's forward direction. Also, denote the internal angles of the triangle $\{D_1, D_2, T\}$ on the edge D_i by $\alpha_i \in [-\pi, \pi), i \in \{1,2\}$, and on the edge T by $\alpha_0 \in [-\pi, \pi)$. Finally, denote the line between D_1 and D_2 by l_0 and the angle measured from line l_0 to the y-axis of drone D_i measured counter-clockwise by $\gamma_i \in [-\pi, \pi)$. Then, we have the following relations:

$$\begin{aligned} \gamma_1 &= \phi_1 + \alpha_1, \\ \gamma_2 &= 2\pi - (\phi_2 + \alpha_2), \\ \phi_i &= \theta_i - \operatorname{atan}(x_T - x_i, y_T - y_i), \end{aligned} \tag{3.1}$$

where $\theta_i \in [-\pi, \pi)$ is the heading angle of drone D_i measured with respect to a global frame Σ_G.

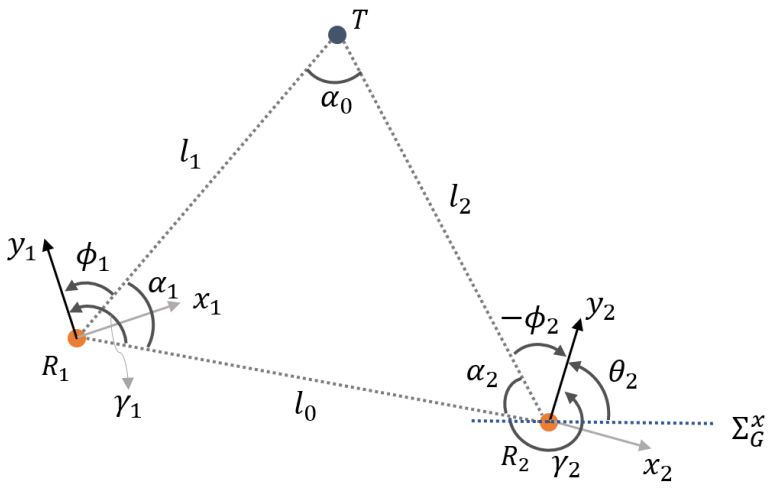

Figure 3.2 The triangle formed by the drones and the target projected onto a plane.

We emphasize that the drones do not have access to their actual heading angles θ_i during operation. Due to this fact, although a drone has holonomic motion kinematics if the roll and pitch motions are stabilized around the zero equilibrium, we cannot take full advantage of its holonomic motion freedom. In other words, the drone does not have a sense of a fixed reference frame, and thus it cannot have the odometry capability.

Therefore, we assume that the motion in the body x-axis is maintained at zero, and the drones move only in their body y-axis and rotate their heading angles, i.e.,

$$v_i^x = 0, \qquad v_i^y \in [0, \bar{v}], \qquad \omega_i \in [-\bar{\omega}, \bar{\omega}], \tag{3.2}$$

where $\bar{v}, \bar{\omega} > 0$ are design constants. It then follows that $||v_i|| = v_i^y = v_i$.

We resemble this motion characteristics to the non-holonomic kinematics which enables the derivation of the relative motion dynamics. The entire system model **(3.3)** consists of the distances and bearing angles inside the triangle $\{D_1, D_2, T\}$. Particularly, the model between each drone and the target can be derived by using the polar dynamics as follows:

$$\begin{aligned} \dot{\rho}_i &= -v_i \cos(\phi_i), \\ \dot{\phi}_i &= \frac{1}{\rho_i} v_i \sin(\phi_i) + \omega_i, \end{aligned} \tag{3.3}$$

where $i \in \{1,2\}$. Furthermore, the polar dynamics between the drones can be written as:

$$\begin{aligned} \dot{\rho}_0 &= -v_1 \cos(\gamma_1) - v_2 \cos(\gamma_2), \\ \dot{\gamma}_1 &= \frac{1}{\rho_0} (v_1 \sin(\gamma_1) + v_2 \sin(\gamma_2)) + \omega_1, \\ \dot{\gamma}_2 &= \frac{1}{\rho_0} (v_1 \sin(\gamma_1) + v_2 \sin(\gamma_2)) + \omega_2. \end{aligned} \tag{3.4}$$

Noting that the input vector is defined as

$$\mathbf{u} = [v_1, v_2, \omega_1, \omega_2]^{\mathrm{T}}, \tag{3.5}$$

we define the state vector as follows:

$$\mathbf{x} = [\rho_0, \rho_1, \rho_2, \phi_1, \phi_2, \gamma_1, \gamma_2]^{\mathrm{T}}. \tag{3.6}$$

Therefore, the entire system model can be represented as follows:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} -\mathbf{u}_1 \cos(\mathbf{x}_6) - \mathbf{u}_2 \cos(\mathbf{x}_7) \\ -\mathbf{u}_1 \cos(\mathbf{x}_4) \\ -\mathbf{u}_2 \cos(\mathbf{x}_5) \\ \frac{1}{\mathbf{x}_2} \mathbf{u}_1 \sin(\mathbf{x}_4) + \mathbf{u}_3 \\ \frac{1}{\mathbf{x}_3} \mathbf{u}_2 \sin(\mathbf{x}_5) + \mathbf{u}_4 \\ \frac{1}{\mathbf{x}_2} (\mathbf{u}_1 \sin(\mathbf{x}_6) + \mathbf{u}_2 \sin(\mathbf{x}_7)) + \mathbf{u}_3 \\ \frac{1}{\mathbf{x}_2} (\mathbf{u}_1 \sin(\mathbf{x}_6) + \mathbf{u}_2 \sin(\mathbf{x}_7)) + \mathbf{u}_4 \end{bmatrix} + \boldsymbol{\epsilon}_x, \tag{3.7}$$

where $\boldsymbol{\epsilon_x} \sim N(\mathbf{0}, \mathbf{R})$ is the additional Gaussian noise vector describing the process noise and disturbance. Having defined the motion model, we continue with the measurement model definition. The drones can always measure the distances ρ_0, ρ_1, ρ_2 with the UWB sensors. By using the law of cosines, we can always calculate the internal angles $\alpha_0, \alpha_1, \alpha_2$. By using the relation in **(3.1)**, we then measure the angles $\gamma_1 - \phi_1$ and $\gamma_2 - \phi_2$ as well. However, the drones can measure the bearing angles ϕ_1, ϕ_2 only when the computer vision algorithms can produce detection results, i.e., when the drones can detect the target body. We explain the detection algorithms applied in Section in detail. The object detection can only occur within a certain radius of the object. We define that region as a disc $B(p_T, \bar{\rho})$, where $\bar{\rho}$ is the detection radius. Also, inside this disc, detection may not occur. Therefore, we consider two measurement models. The first model does not include the bearing angles ϕ_1, ϕ_2, and is defined as follows:

$$\mathbf{y_1} = [\rho_0, \rho_1, \rho_2, \gamma_1 - \phi_1, \gamma_2 - \phi_2]^{\mathrm{T}} + \boldsymbol{\epsilon_y}. \quad \textbf{(3.8)}$$

The second model consists of the bearing angles and is defined as follows:

$$\bar{\mathbf{y}}_\mathbf{2} = [\rho_0, \rho_1, \rho_2, \phi_1, \phi_2, \gamma_1 - \phi_1, \gamma_2 - \phi_2]^{\mathrm{T}}. \quad \textbf{(3.9)}$$

Or equivalently,

$$\mathbf{y_2} = [\rho_0, \rho_1, \rho_2, \phi_1, \phi_2, \gamma_1, \gamma_2]^{\mathrm{T}} + \boldsymbol{\epsilon_y}. \quad \textbf{(3.10)}$$

We note that all measured angles reside in the interval $[-\pi, \pi)$, and otherwise, they need to be wrapped to this interval.

Having defined the motion model and measurement models for the entire system, we design a multi-rate extended Kalman filter in discrete time with the motion model **(3.14)** and the measurement models **(3.15)** and **(3.17)**. The prediction update part includes the propagation of the motion model with the given input vector:

$$\bar{\boldsymbol{\mu}}_\mathbf{k} = \mathbf{f}(\mathbf{x}, \mathbf{u}),$$
$$\bar{\boldsymbol{\Sigma}}_\mathbf{k} = \mathbf{G_k} \boldsymbol{\Sigma_k} \mathbf{G_k^T} + \mathbf{R} \quad \textbf{(3.11)}$$

where $\boldsymbol{Q}$ is the motion model covariance matrix and

$$\mathbf{G_k} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\mathbf{x_k}=\boldsymbol{\mu}_\mathbf{k}} \tag{3.12}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \mathbf{u}_1 \sin(\mathbf{x_6}) & \mathbf{u}_2 \sin(\mathbf{x_7}) \\ 0 & 0 & 0 & \mathbf{u}_1 \sin(\mathbf{x_4}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{u}_2 \sin(\mathbf{x_5}) & 0 & 0 \\ 0 & \mathbf{g_{42}} & 0 & \mathbf{g_{44}} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{g_{53}} & 0 & \mathbf{g_{55}} & 0 & 0 \\ \mathbf{g_{61}} & 0 & 0 & 0 & 0 & \mathbf{g_{66}} & \mathbf{g_{67}} \\ \mathbf{g_{71}} & 0 & 0 & 0 & 0 & \mathbf{g_{76}} & \mathbf{g_{77}} \end{bmatrix},$$

$$\mathbf{g_{42}} = \frac{-1}{\mathbf{x_2^2}} \mathbf{u}_1 \sin(\mathbf{x_4}), \qquad \mathbf{g_{53}} = \frac{-1}{\mathbf{x_3^2}} \mathbf{u}_2 \sin(\mathbf{x_5}),$$

$$\mathbf{g_{44}} = \frac{1}{\mathbf{x_2}} \mathbf{u}_1 \cos(\mathbf{x_4}), \qquad \mathbf{g_{55}} = \frac{1}{\mathbf{x_3}} \mathbf{u}_2 \sin(\mathbf{x_5}),$$

$$\mathbf{g_{61}} = \mathbf{g_{71}} = \frac{-1}{\mathbf{x_1^2}} (\mathbf{u}_1 \sin(\mathbf{x_6}) + \mathbf{u}_2 \sin(\mathbf{x_7})),$$

$$\mathbf{g_{66}} = \mathbf{g_{76}} = \frac{1}{\mathbf{x_1}} \mathbf{u}_1 \cos(\mathbf{x_6}), \qquad \mathbf{g_{67}} = \mathbf{g_{77}} = \frac{1}{\mathbf{x_1}} \mathbf{u}_2 \cos(\mathbf{x_7}).$$

The measurement update step includes finding the Kalman gain and updating the predicted state by using the measurement. For the first measurement model, we have

$$\mathbf{H^1} = \frac{\partial \mathbf{h^i}}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}. \tag{3.13}$$

For the second measurement model, we have

$$\mathbf{H^2} = \mathbf{I_7}, \tag{3.14}$$

where $\mathbf{I_n}$ is an identity matrix of size n-by-n. Therefore, the measurement update step consists of the following:

$$\mathbf{K_k} = \bar{\boldsymbol{\Sigma}}_\mathbf{k} \left(\mathbf{H_k^i}\right)^\mathbf{T} \left[\mathbf{H_k^i} \bar{\boldsymbol{\Sigma}}_\mathbf{k} \left(\mathbf{H_k^i}\right)^\mathbf{T} + \mathbf{Q}\right]^{-\mathbf{1}}, \tag{3.15}$$

$$\boldsymbol{\mu}_\mathbf{k} = \bar{\boldsymbol{\mu}}_\mathbf{k} + \mathbf{K_k} \left(\mathbf{y_k^i} - \mathbf{h^i}(\bar{\boldsymbol{\mu}}_\mathbf{k})\right)$$

$$\boldsymbol{\Sigma}_\mathbf{k} = \left(\mathbf{I}_{\mathbf{n_i}} - \mathbf{K_k} \mathbf{H_k^i}\right) \bar{\boldsymbol{\Sigma}}_\mathbf{k},$$

where $\mathbf{n_1} = 5, \mathbf{n_2} = 7$, $\boldsymbol{y_k^i}$ is the measurement vector at time k, $\mathbf{H_k^i}$ is as in **(3.14)**, **(3.15)**, and $\mathbf{K_k}$ is the Kalman gain.

3.3 Control Algorithm

In this part, we model the high-level control units of the two drones. First, we design a fictitious control law as if the controlled variables are measured perfectly. Then, we integrate the estimation module into the formation control module to replace the fictitious variables with the estimated variables.

The high-level controller aims at approaching the target and docking at a suitable distance. Thus, we decompose the control action into two parts: Approach and dock. We assume that the drones are initially located far distance apart from the target, i.e., $\rho_i(0) > \rho^{\text{des}}, i \in \{1,2\}$, where ρ^{des} is the radius of the detection disc $B(p_T, \rho^{\text{des}})$ defined in the previous section. Our method comprises a series of action modes for the drones. Thus, we define a state machine to manage the high-level control actions. The state machine consists of four modes: Approach, Converge, Rotate, and Orient. The state machine operates based on the following flow diagram:

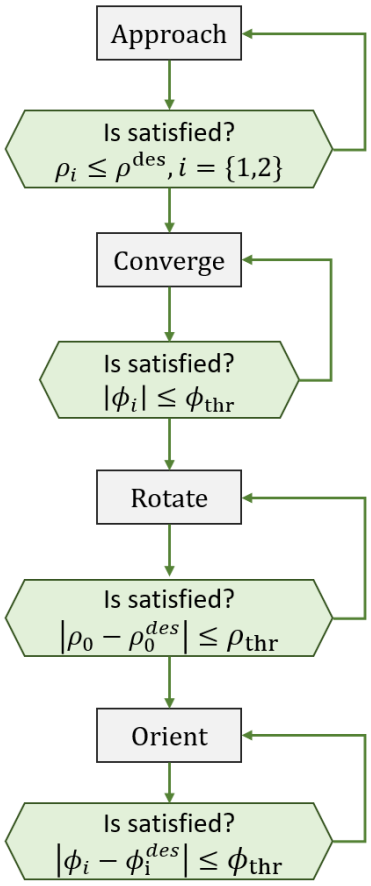

Figure 3.3 State machine

First, we design the control law for the Approach mode. The main goal for drone D_i is to reduce the distance ρ_i toward the target and maintain the distance toward the other drone D_j at a suitable value. Since the drones do not use a magnetometer sensor, they cannot take full advantage of their holonomic motion capabilities. Instead, every drone moves only in the y-axis and performs rotation in the z-axis of its body frame Σ_b while maintaining the speed in the x-axis of its body frame Σ_b at zero as in **(3.2)**. Such a modification results in a non-holonomic behavior and puts an important constraint in the motion control algorithm design. We propose the following control law for D_i in the Approach mode:

$$
\begin{aligned}
&v_i(t) = K_v e_{\rho_i}(t), \\
&\omega_i(t) = K_\omega e_{\omega_i}(t), \\
&e_{\rho_i}(t) = \rho_i(t) - \rho^{\text{des}}, \\
&e_{\omega_i}(t) = \phi_i(t) - \phi_i^{\text{des}}(t), \\
&\phi_i^{\text{des}}(t) = \arctan\left(D, \sqrt{1 - D^2}\right), \\
&D(t) = x_{des}/\rho_i(t),
\end{aligned}
\qquad \textbf{(3.16)}
$$

where $K_v, K_\omega > 0$ are design constants, and ϕ_i^{des} is the desired bearing angle between the drone R_i and the target T. We assume that the drones are initiated such that the initial inter-drone distance is close to its desired value, i.e., $|\rho_0(0) - \rho_0^{\text{des}}| < c$, where ρ_0^{des} is the desired inter-drone distance at steady-state and c is an arbitrary constant. The linear speed controller v_i aims at approaching the target by reducing the distance error e_{ρ_i} whereas the angular speed controller ω_i aims at regulating the bearing error toward the target. Referring to Figure 3.4, we denote the desired bearing of drone R_i toward T at time t by $\phi^{\text{des}}(t)$ which is calculated by using the distance $\rho_i(t)$ and the desired separation x_{des} between the R_i and T in the x-axis of the body frame Σ_{bi}. In other words, the drone tries to direct its heading toward the point which is x_{des} unit away from T by using the error term e_{ω_i}.

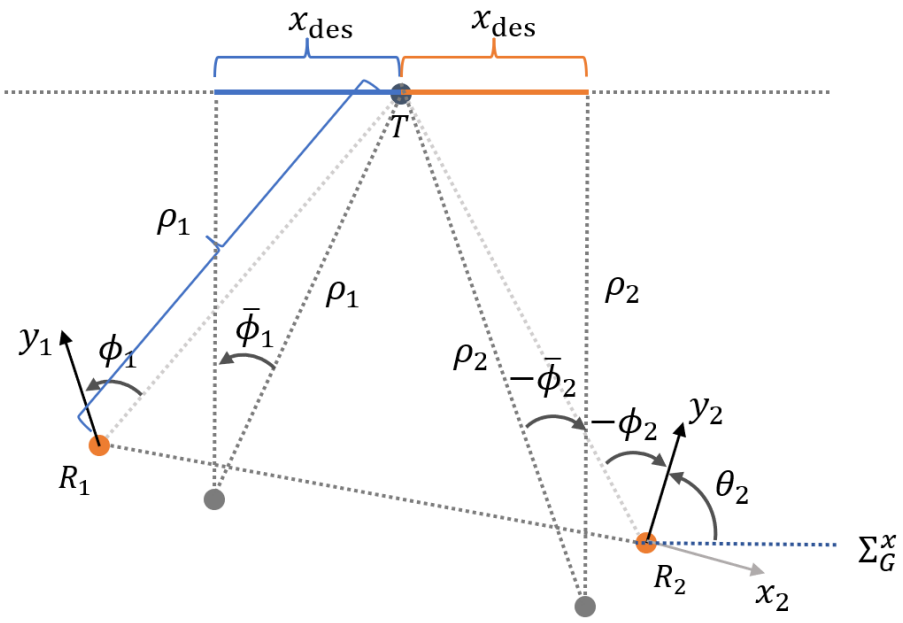

Figure 3.4 Approach control mode parameters

Once both drones enter the disc $B(p_T, \rho^{\text{des}})$, which is the disc with center p_T, the target location, and with radius ρ^{des}, both drones switch their states to Converge. In the Converge state, both drones aim at regulating their distances and bearings to their desired values so that

$$\begin{aligned} v_i(t) &= K_v e_{\rho_i}(t), \\ \omega_i(t) &= -K_\omega \phi_i(t), \\ e_{\rho_i}(t) &= \rho_i(t) - \rho^{\text{des}}. \end{aligned} \tag{3.17}$$

The drones aim at stabilizing the drones on the boundary of the disc $B(p_T, \rho^{\text{des}})$ as in Figure 3.3. Since the drones do not have any sense of their heading angles due to the absence of a fixed reference frame, they align themselves around the target T by using the available measurements. However, the drones may drift in the x-axis of their body frames because no external infrastructure can aid in the localization procedure. Therefore, in the Converge state, the drones can move around the boundary of the disc $B(p_T, \rho^{\text{des}})$.

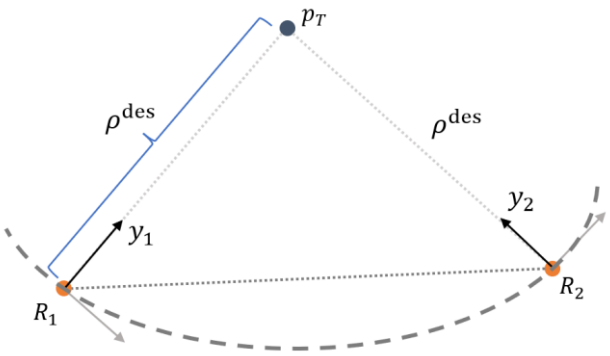

Figure 3.5 Converge mode parameters. In the converge mode, the drones aim at stabilizing themselves on the boundary of the disc $\boldsymbol{B(p_T, \rho^{\text{des}})}$.

Once both drones regulate the distances ρ_i and the bearings ϕ_i around the circle within a threshold, they switch to the `Rotate` state where they aim at regulating the inter-drone distance ρ_0. At this stage, the drones apply the following control law:

$$
\begin{aligned}
&v_i(t) = K_v e_{\rho_i}(t), \\
&v_1^x(t) = K_v e_{\rho_0}(t), \qquad v_2^x(t) = -K_v e_{\rho_0}(t), \\
&\omega_i(t) = -K_\omega \phi_i(t), \\
&e_{\rho_i}(t) = \rho_i(t) - \rho^{\text{des}}, \\
&e_{\rho_0}(t) = \rho_0(t) - \rho_0^{\text{des}}.
\end{aligned}
\tag{3.18}
$$

Chapter 4

Simulation

We created a simulation environment which can mimic physical properties of real experiment setup and help us to perform and analyze system setup. For simulation, Gazebo simulation environment is used and Robot operating System (ROS) is combined with it to have autonomous control in simulated drone models. For robot model, Iris drone model is used and warehouse items such as boxes, pallets and shelves are used to create indoor warehouse environment. In this chapter, the simulation environment, drone and system setup, and simulation procedure are expalined and analyzed.

4.1 Simulation Environment

In robotic area, ROS is used in many studies because of many benefits that it provides. In robotic applications, we may need to control many different objects and instances such as robots, manipulators, or simulation physics. Normally by using only one code and terminal, controlling all these parts is hard to implement. To be able to control them from one place and create a common communication plane, ROS is used. With ROS, we can create nodes in our network and connect each instance in our project to our system. In ROS, we have different type of messages that can contain different type of data such as Twist message that can carry 6 different data which are Linear x – y – z and Angular x – y – z values. Each node in this network can carry multiple data and messages (which called topics) which published from robots and anytime, by subscribing to these topics, we can reach these data and messages and use them for any purposes. By using this system, we can send any kind of data such as sensor data or images from cameras and we can create multi-robot system that each robot can share the same data from same node and realize their objective.

Several simulation environments can be built to simulate the performance of our algorithm such as Gazebo, AirSim, and DJI simulation tools. Robot Operating System (ROS) and its new version ROS2 work in different operating systems such as Windows, MacOS and Linux distros but to be able to run these systems with an appropriate simulator, Linux distros are the best options because of their flexibility and easy to run all these programs. For these purposes, we used Ubuntu 18.04 operating system as our main operating system which can run our simulator program, Gazebo Simulator, can run essential libraries and also can work with ROS very well. For our study, we created our simulation environment in Gazebo 9 robot simulator which installed on a desktop computer with Ubuntu 18.04 Operating system. Hardware setup is shown in Figure 4.1.

Figure 4.1 Hardware setup that contains Ubuntu 18.04 desktop computer and 2 Jetson Nano computers for simulation

In the Gazebo simulation, we gathered necessary models such as boxes, shelves, and pallets to create our warehouse environment. As the drone simulation, Iris drone model is used. Within this simulator application, we can control many different properties of our objects and environments such as the size of the objects or environmental properties, wind, and light. A sample perspective view of our environment can be seen in Figure 4.2.

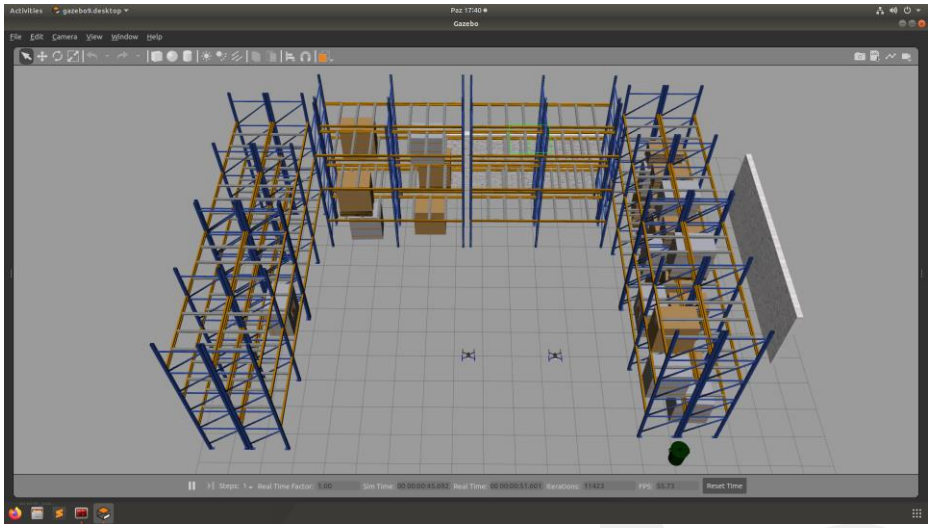

Figure 4.2 The simulated warehouse environment in Gazebo 9 robot simulator

To control the drone's motion as in real-world applications, ROS and Px4 systems are wrapped on it which help us to get important topics and services for our drones such as drone's velocity values or taking off and landing commands. The Iris drone model is created by using 3D body parts, propellers and simulated motors. Also, to acquire the images in simulation environment, forward-facing 640x480 pixel RGB camera module implemented on Iris drone model (Figure 4.3). To be able to implement image processing techniques, we need to have an object of interest. In simulation, a television image used for that purpose. By using this drone model, we can mimic every possible action that any normal drone can realize in real life.

Figure 4.3 The Iris drone model that used in the simulation for drone modeling

4.2 Formation Model

In our work, formation of our drones is set like real world. After simulation environmment created, our system works step by step to localize our drones. To be able to control our simulated drone models, we design our simulation as Hardware-in-the-Loop (HIL) which we use real hardware setup, Jetson Nano computers to process data from simulation. As in a typical real experiment, Jetson Nano computers acquire the required data from simulated drone models and all calculations are performed onboard. The simulation operations start with arming both drones at 1.5 meter altitude and continue with forward movement towards the object. While they are moving, by using camera sensor's data, image processing is done and we try to detect television image in our image frames. Until we detect television image in our image frame, drones continue their forward movement and by using UWB sensors, we determine distances between drones and object, which are fused in the EKF function to estimate the relative positions. After the television is detected by the Jetson boards, we put bounding box around the objects in the image frames and calculate the pixel distances between x axis of image frame and center of the bounding box. Afterwards, we translate pixel distance to metric unit such as meter, and we can measure angle between object and our forward axis and use the bearing data in the estimation algorithm.

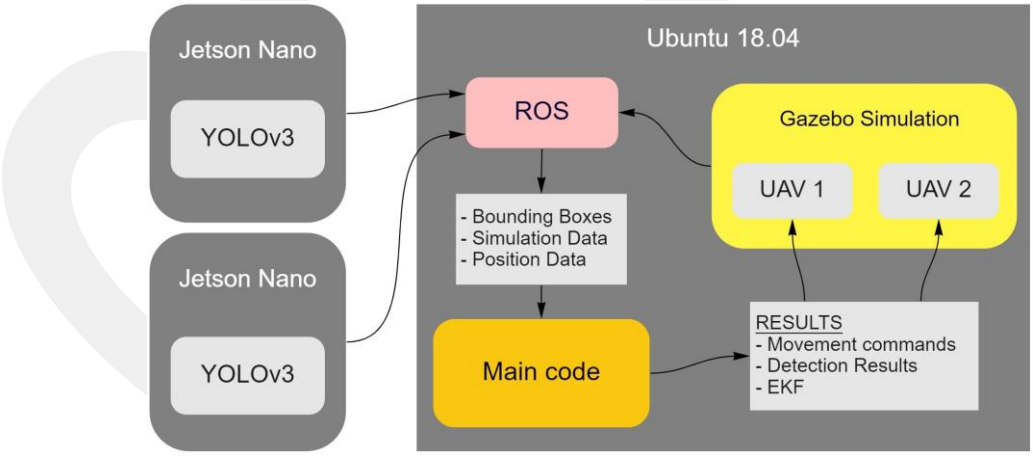

Figure 4.4 Block diagram of the simulated system

4.3 Object Detection

In image processing, there are several different methods and algorithms for object detection and tracking in given images or video feeds such as SIFT, Speeded up Robust Features (SURF) and You Only Look Once (YOLO). The main purpose of these methods is to detect any given object in a given source of images or video feeds by applying comparison and finding similarities between them. Algorithms such as SIFT and SURF mainly try to find similar features between those images and try to match given images on given image or video sets. On the other hand, YOLO algorithm uses neural network, which is a series of algorithms that can create relationship between set of data and mimic human brain. Unlike other detection methods, instead of checking every pixel in the given image over and over again, YOLO separate given image into grids for detection. For each grid in an image, neural networks can analyze relationships between them and after that analyze, it can suppress grids that has lower relationship value to create bounding boxes around the object. In this work, we use YOLO algorithm to detect the target object by the drones' onboard cameras.

In YOLO, they are many different versions varying from original updated versions to seperately personalized versions for specific tasks that can expand its usage area and can enhance its detection power. Currently, YOLO v5 is the latest published version which has better detection power than previous updates at the expanse of more processing power requirements. If we want to use YOLO algorithms in any work, we need to consider its processing power requirements to be able to use them well and without any problem. YOLO can run on the main processor or on a separate graphics card. Usually, implementing YOLO on a separate dedicated board improves the detection quality. We implemented YOLO on the dedicated Jetson Nano boards for improved detection rates and quality.

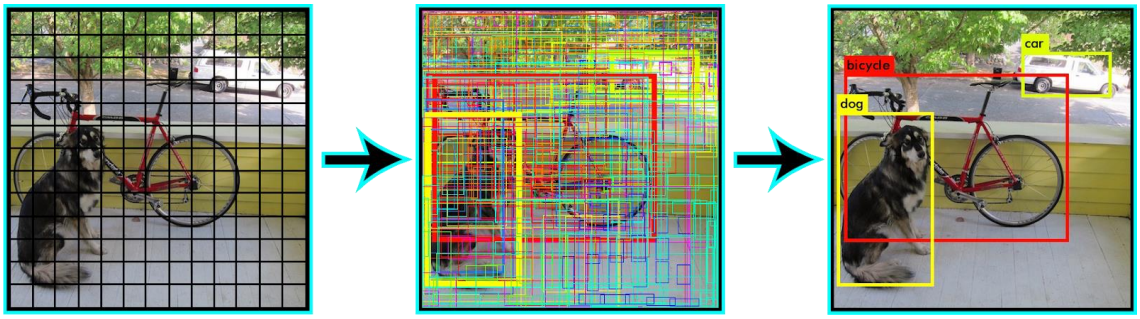

Figure 4.5 Figure shows how the YOLO algorithm and neural network griding and detection works

For our system, we decided to use YOLO version 3 which has moderate power consumption and also can detect required objects very well. To be able to use our installed camera module's image data with YOLOv3, we should create a connection between our system and the YOLO algorithm. To be able to attach YOLOv3 to ROS, we used Darknet-Ros [50] ROS packages that run YOLO algorithms within the ROS environment so we can get results of YOLO algorithms as a ROS topic in our network. After installation of these packages, to be able to use them within our system, we need to change some parameters and files. In YOLOv3, we have two different versions of it for different purposes which are normal YOLOv3 and tiny versions of it. In YOLOv3-tiny, we have the same algorithms as YOLOv3 but we have less power consumption and also it requires less processing power. So, this version is a better option for our robotic system which has onboard computers. After deciding on our YOLO algorithm's version, we need to find a proper data set that will include all the objects that we want to detect with YOLOv3-tiny. For that purpose, we used COCO [51] dataset from Microsoft, which is a pre-trained dataset and has eighty different objects varying from foods to daily use objects such as television or car. By using this data set with YOLOv3-tiny, we can detect our object of interest in the warehouse environment and get bounding box data to analyze it and define our position in the environment.

In our simulation, after we run Gazebo with ROS environment together, we start Darknet-Ros package within ROS to start image processing. From our pre-installed camera sensor on Iris drone model, we can get onboard image data and transfer this data to our Darknet-Ros topics created by Darknet-Ros package. While the drones are running, YOLOv3-tiny can process transferred image data and detect the object of interest. After the detection occurs, YOLOv3-tiny can create a bounding box, which cover detected

object with 4 corner points around it, then it can send position of this bounding box by using specific topic in our network so we can use it in our code as we want.

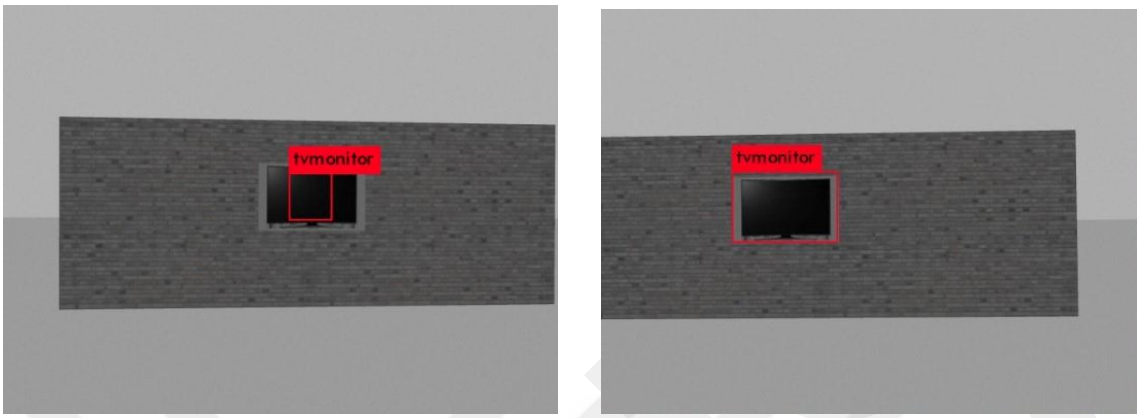

Figure 4.6 Image frame and detection of object of interest in simulated drones

After detection, our image frame and bounding boxes looks like in Figure 4.6. Created bounding boxes can be in different shapes because of our detection quality and movement of the drones and image frames but these differences will not effect our calculation because after detection happened, center point of created bounding box is used for pixel distance calculation, so even if we have different sized bounding boxes, our calculation will not get effected.

4.4 Simulation Results

We carried out a comprehensive simulation study to analyze the performance of the proposed framework. A sample simulation environment is seen in Figure 4.2 where the two drones reside in an industrial environment. We placed the object to be detected at the position $p_T = [0,7.35]^T$ meters in the Gazebo frame at 2 meter high from the ground. The drones were started from various initial locations and aimed at docking around the target T. Our simulations included an initialization phase and an operation phase. We assumed that the drones were at rest on the ground. In the initialization phase, the drones were commanded to take off and hover at the altitude $z = 2$ meters at their initial locations on the xy-plane by utilizing the px4 low-level controllers. Then, the EKF code was ran and the drones started moving in the operation phase.

We demonstrate a simulation result in Figure 4.7. In this simulation, the drones started from the initial locations $p_1 = [-2.25, 0.45]^T, p_2 = [2.25, 0.15]^T$ meters on the Gazebo xy-plane. In Figure 4.7, the target object location is shown with the blue diamond, while the red and yellow lines depict the traces of the drones R_1, R_2, respectively. The purple and green diamonds show the initial locations of the drones. The blue and purple circles denote the discs $B(p_T, \rho^{\text{des}} - \rho^{\text{th}})$ and $B(p_T, \rho^{\text{des}} + \rho^{\text{th}})$, where ρ^{th} is the pre-defined threshold value to avoid chattering. Thus, the aim of the drones is to enter the zone between these discs, align their headings toward the target, and adjust the inter-drone distance at its desired value ρ_0^{des}. We illustrate the start times of the four phases (Approach, Converge, Rotate, and Orient) in the simulation in Figure 4.7.

The Approach mode took 14.6 seconds in which the drones move toward the target by using the control law **(3.16)**. When both drones enter the disc $B(p_T, \rho^{\text{des}})$ at $t = 14.6$ seconds, the drones switch to the Converge mode where they try to align their headings toward the target and regulate their distances ρ_1, ρ_2 toward the target at its desired value ρ^{des} by using the control law **(3.17)**. The drones' headings are shown with the cyan arrows. It is observed that the drones aligned their headings toward the target successfully by adjusting the angular velocity control term ω_i. Also, since the bearings $\phi_i \in (-\pi, \pi), i \in (1,2)$, the control law $v_i(t) = K_v e_{\rho_i}(t)$ applied on the y −axes of the drones' body frames regulated the error $e_{\rho_i}(t)$, steering the drones on the circle $C(p_T, \rho^{\text{des}})$. In the Converge mode, the drones utilized the second measurement modal, i.e., $\mathbf{y_2} = [\rho_0, \rho_1, \rho_2, \phi_1, \phi_2, \gamma_1, \gamma_2]^{\text{T}} + \boldsymbol{\epsilon_y}$. That is, the drones used the bearing angles produced by the deep learning method. In the lower-left figure in Figure 4.7, it is observed that at the end of the Converge mode ($t = 14.6$ seconds), the drones entered the zone defined by the discs $B(p_T, \rho^{\text{des}} - \rho^{\text{th}})$ and $B(p_T, \rho^{\text{des}} + \rho^{\text{th}})$ and aligned their headings toward the target with a small error.

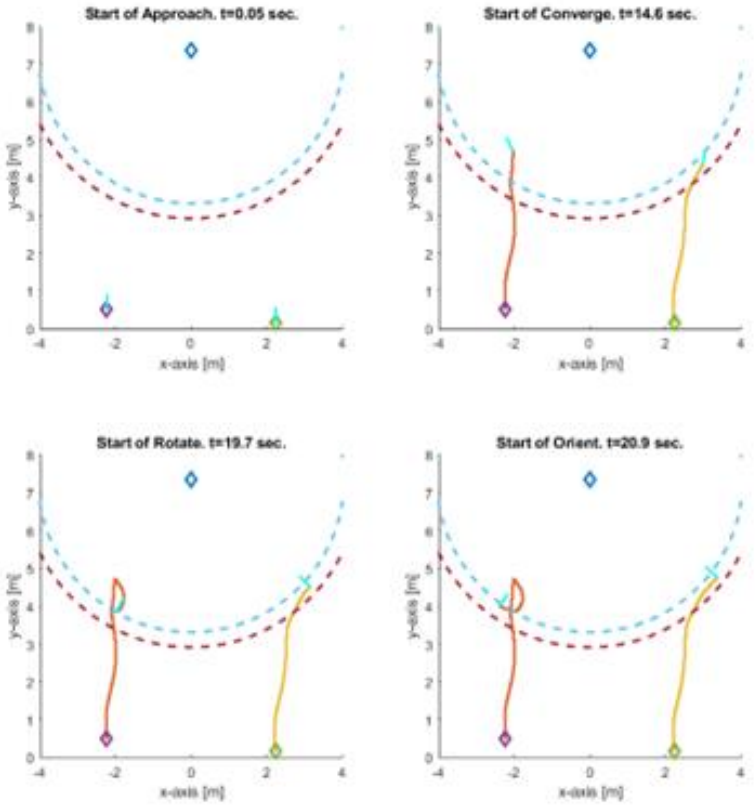

Figure 4.7 The modes of the drones in a simulation. The solid lines and the cyan arrows show the traces and the heading angles of the drones, respectively

After satisfying the conditions in the Converge mode, both drones switched to the Rotate mode at $t = 19.7$ seconds, where the aim was to regulate the inter-drone distance ρ_0 at its desired value ρ_0^{des}. In this state, the drones use the controller **(3.18)** where they move in the $x-$axes of their body frames, which correspond to the axes which are perpendicular to the lines combining the drones with the target. Meanwhile, they continue regulating their bearing angles toward the target. Therefore, the drones move on the circle $C(p_T, \rho^{\text{des}})$ while maintaining the bearings at $\phi_i = 0$. Once the inter-drone distance is within the threshold $|\rho_0 - \rho_0^{\text{des}}| \leq \rho_0^{\text{th}}$, where $\rho_0^{\text{th}} > 0$ is a design parameter to avoid chattering, the drones switch to the Orient mode where they maintain their distances and bearing angles toward the target at their desired values. In this simulation, we observed that the drones spent 1.2 seconds in the Rotate mode.

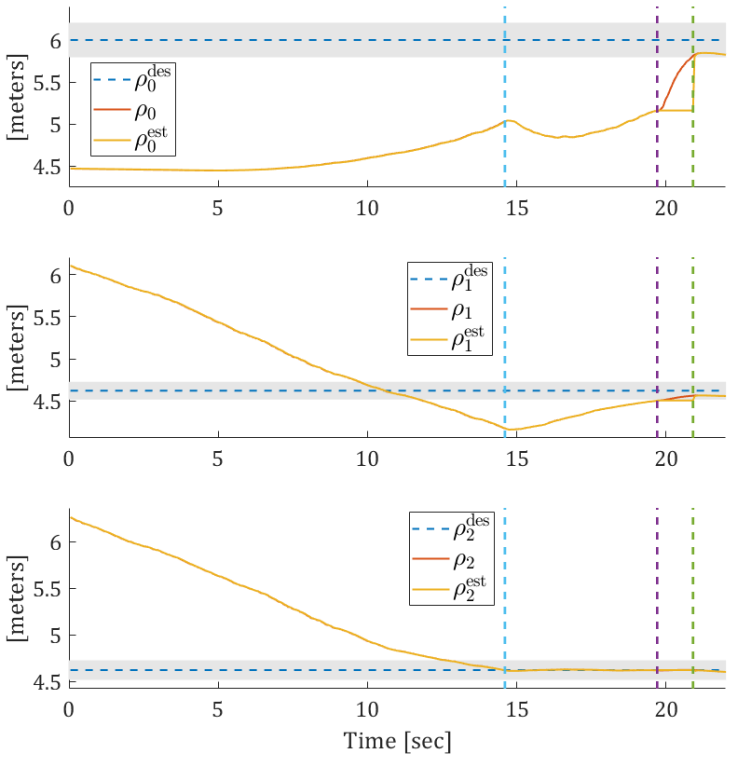

Figure 4.8 Ranges and their estimations in the simulation: The vertical lines denote the start times of the states: Blue: Converge; Purple: Rotate; Green: Orient

We present the estimation results of the same simulation in Figure 4.8 and Figure 4.9. Figure 4.8 demonstrates the range values, their estimations, and the desired values. The grey background areas denote the threshold zones for the ranges allowed by the threshold value ρ_0^{th} and ρ^{th}. For instance, in the upper figure, the actual range ρ_0 is desired to be in the in the range $[\rho_0 - \rho_0^{\text{th}}, \rho_0 + \rho_0^{\text{th}}]$ depicted by the grey zone. The vertical lines denote the start time of the states. We observed that the drones approached the target with monotonically decreasing range values to ρ_1, ρ_2 until the end of the Approach mode. Then, once both drones entered the disc $B(p_T, \rho^{\text{des}})$ at $t = 14.6$ seconds, the drones controlled their ranges ρ_1, ρ_2 to be in the grey zone in the middle and lower figures. We see from these figures that during this period, it took some time to regulate ρ_1 while ρ_2 was already in the grey zone.

Next, the drones switched to the Rotate mode at $t = 19.7$ seconds, where they utilized the actual range and bearing measurements instead of their estimations to adjust ρ_0. Thus, the range estimations remained constant during the Rotate mode. We remind

that the main reason why we did not utilize the EKF during the Rotate mode is because the system model **(3.10)** was no longer valid when we moved the drones in their body x-axes. The drones brought ρ_0 to its desired range $\left[\rho_0 - \rho_0^{\text{th}}, \rho_0 + \rho_0^{\text{th}}\right]$ (inside the grey zone in the upper figure in 1.2 seconds. Afterward, both drones switched to the Orient mode (the green vertical line) in which they maintained their distances and bearing angles toward the target.

We demonstrate the bearing angle estimations with their ground truth values obtained from the Gazebo environment in Figure 4.9. We note that in the Approach mode, none of these variables are measured directly, the drones can measure $\phi_i - \gamma_i$ only. Therefore, until the Converge mode starts, these variables are estimated to some degree of error only. Once the Converge mode starts, the drones can measure the bearing angles ϕ_i and γ_i directly by utilizing the deep learning method thus the estimation outcomes converge to their ground truth values. However, it took some time for the estimated values to converge at the beginning of the Converge mode. The main reason for this issue is that the drones rotated their heading angles during the Approach mode in order to satisfy the control objective, which caused the bearing angles ϕ_i to increase in magnitude. Since the target was close to the sides of the image frames on both drones, the bearing angle estimation based on the detected bounding box performed poorly. Nevertheless, since the drones estimated the signs of the bearing angles, the angular velocity control was executed correctly, resulting in the decrease in the angles in magnitude. This motion brought the target object toward the center of the image frames, and at $t = 17$ seconds, the correct bearing angles are estimated by the EKF. Since the EKF was not executed during the Rotate mode, the estimations diverged from their actual values in that mode. However, this divergence did not affect the convergence of the drones because they used the raw measurements coming from the sensors. We illustrate the entire traces of the drones with the final range and bearing angle values in Figure 4.10. We observe that the drones entered the desired zone defined by the discs $B\left(p_T, \rho^{\text{des}} - \rho^{\text{th}}\right)$ and $B\left(p_T, \rho^{\text{des}} + \rho^{\text{th}}\right)$, aligned their headings toward the target (shown by the purple arrows), and regulated the inter-drone distance ρ_0 to its desired range $\left[\rho_0 - \rho_0^{\text{th}}, \rho_0 + \rho_0^{\text{th}}\right]$, meeting all conditions set in Chapter 3.

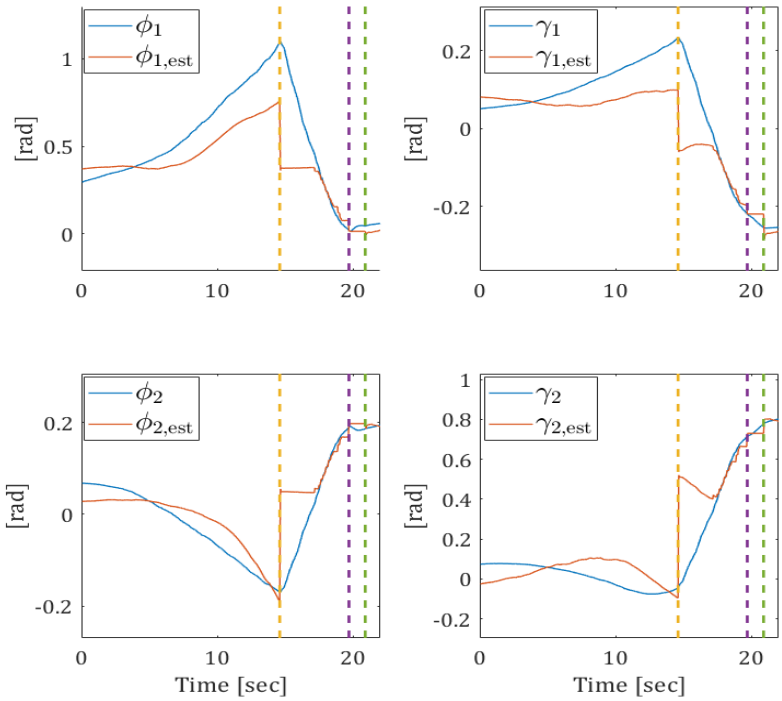

Figure 4.9 Bearing angles and their estimations in the simulation: The vertical lines denote the start times of the states: Blue: Converge; Purple: Rotate; Green: Orient

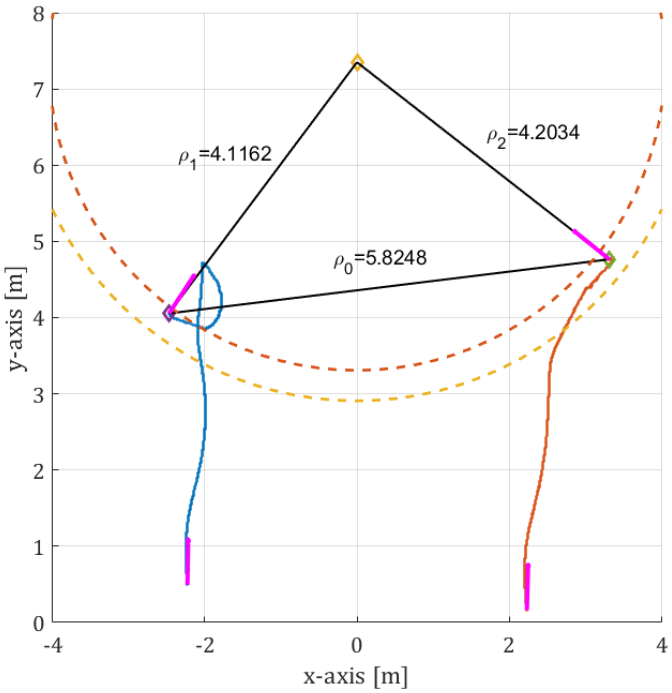

Figure 4.10 The initial and final locations and orientations of the drones and the final range values

We illustrate another simulation result in Figure 4.11-Figure 4.13, where the desired drone-target distances are increased to 5.65 meters. We observed that the drones approach the target and dock around it successfully. Also, the estimation performance was sufficient. In Figure 4.13, the estimated variables are represented where they converge to their actual values in a short time in the Converge mode (between the yellow and purple vertical lines).

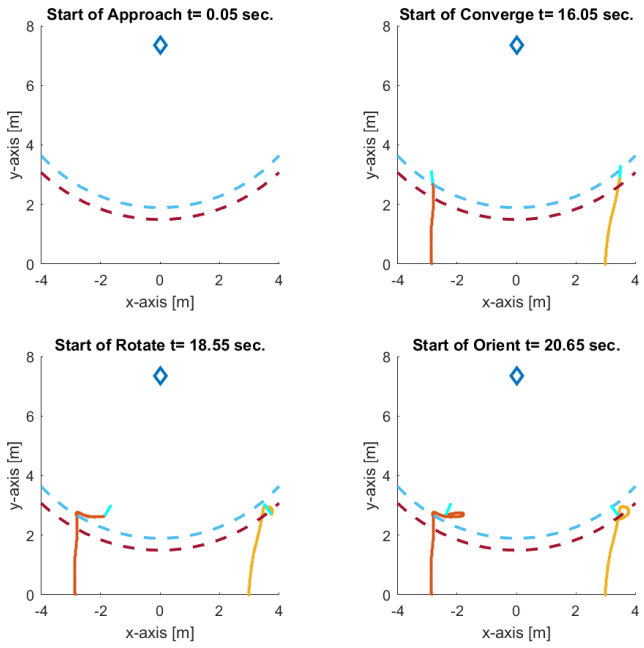

Figure 4.11 The drones' modes in the second experiment with image processing.

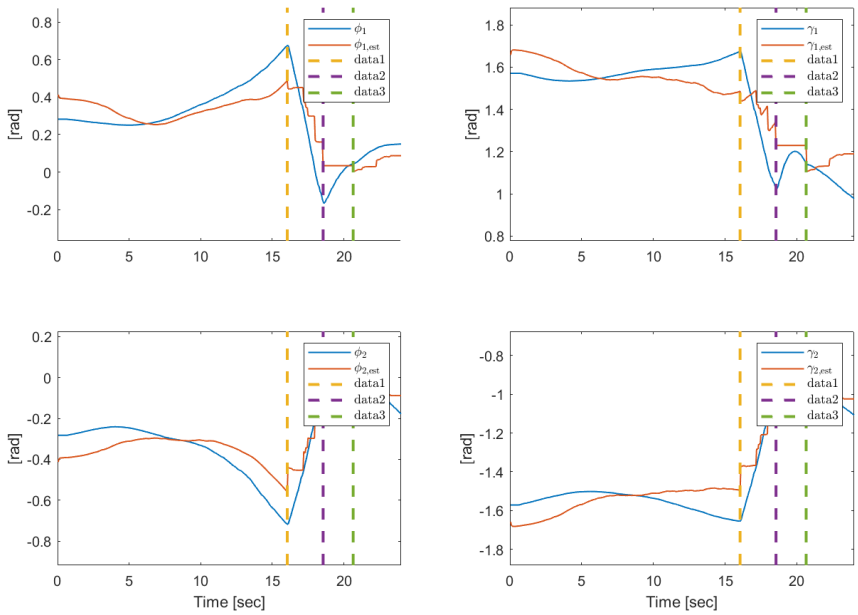

Figure 4.12 Bearing angles and their estimations in the second simulation with image processing.

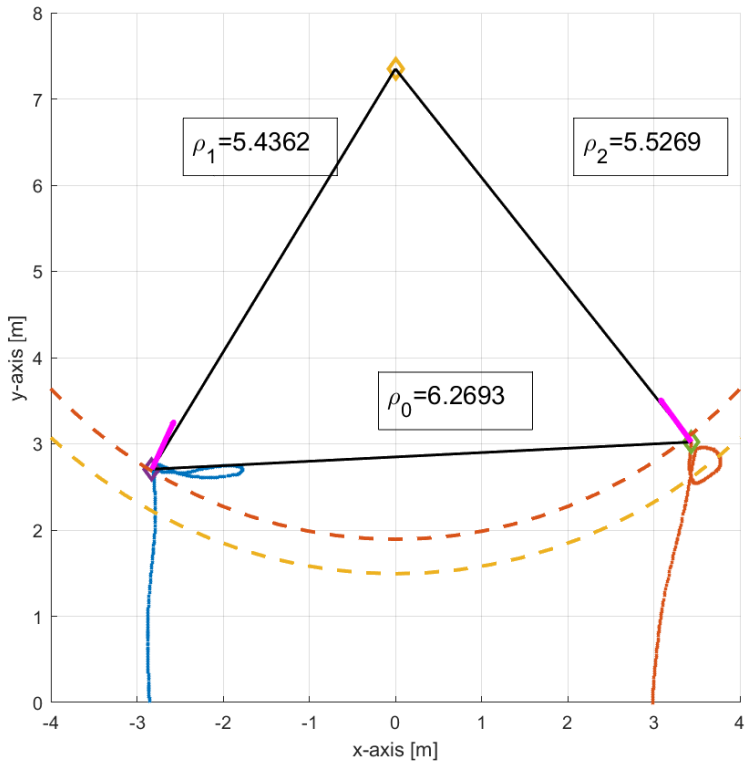

Figure 4.13 The initial and final locations and orientations of the drones in the second simulation with image processing.

4.5 Analysis

In this part, we analyze the system performance of each stage (mode). In the Approach mode, the drones are controlled with purely distance measurement data. The drones need to start from a suitable initial condition to realize this mode successfully. Particularly, the drones need to start their heading angles such that $\gamma_1 \in \left(\frac{\pi}{2}-\epsilon,\frac{\pi}{2}+\epsilon\right), \gamma_2 \in \left(\frac{-\pi}{2}-\epsilon,\frac{-\pi}{2}+\epsilon\right)$ with an arbitrarily small ϵ so that they can approach to the target in the first few seconds. Distance-based formation control with global convergence guarantees is an open research problem in the literature, and a more advanced control technique can be designed for the Approach mode of our proposed framework. Nevertheless, the proposed distributed control algorithm here sufficed to obtain small errors in the Approach mode.

Once both drones detect the target object with the YOLO method, they enter the Converge mode where they utilize the bearing data together with the distance measurements in the control algorithm. Since we design the drones to switch to the

Converge mode after both drones detect the target object simultaneously, the EKF algorithm is guaranteed to obtain both bearing data as the input. As can be observed from the estimation figures of the previous section, the drones can detect the relative positions toward the target up to a small error in the Converge mode. Since we use a multi-rate EKF algorithm with two measurement models, the drones continue estimating the relative position in the absence of the visual target detection, which enhances the performance. We emphasize that the drones may drift in their body x-axis in the Converge mode, which can be compensated with a proportional controller in that axis.

In the Rotate mode, the drones aimed at regulating the inter-drone distance by moving on the circle around the target. In the simulations, we observed that the drones were able to detect the target object in the Rotate mode, which helped regulating the heading angle toward the target all the time. Notably, this behavior is expected because the drones enter the Rotate mode after the Converge mode where the drone-target distances allow the target object detection. Since the drones moved on their body x-axes, the non-holonomic motion behavior was not satisfied in this mode. Thus, EKF was not running during this mode, and the drones had to move based on the distance measurements and visual detection results. We observed sufficient accuracy in the Rotate mode, and the drones maintained the inter-drone distance around the desired values.

Chapter 5

Experiment

After our simulation results analyzed, to be able to test our designed system in real world, we created an experimental testbed. While carrying out our simulation setup to the real-world experiments, we had to tune some parameters and functions in the algorithm to fit the real drone requirements. In this chapter, we describe the complete experimental setup to realize the proposed system and our primary results on the object detection.

5.1 Experiment Setup

To realize our study in a real-world experiment, we need to set up some hardware and software parts that should be different than our simulation setup. In the simulation, by fixing pre-defined parameters in Gazebo simulation files and our code, we can summon our drones in a simulation environment, change parameters on them, use services such as take-off and land that our drones have them and we can control them in off-board mode, which is a mode that helps us to send control commands to make them follow. So, first, to be able to create an experiment setup, those parameters should be fixed.

For this experiment, we used Mavic 2 rpo and Mavic mini 2 drones from DJI company. These drones have smaller size with respect to other same level drones, also they can fly smootly without so much vibration and shifting, they can fly at least 20 minutes and also they can takeoff and land automatically. In normal use, we can only control these drones by using their remote controller or mobile phone applications which connect drones' wifi signals for connection. So, by using our system, which we created for our simulation environment, we cannot send movement command to our drones to fly with them and apply our study. DJI company created a software development kit, which is in a form of mobile phone application, that can modified and can control drones as customer want. So, for this study, we modified this published application for our case which can send movement command to our drones by runnig our code.

We modified the MSDK applications by using Swift and Android Studio applications, which are mainly used to create applications for IOS and Android platforms. We add some control buttons that allows us to recieve data from our code and send it to

the drones as velocity commands. To create servers between our main computer and our applications, we mounted a Jetson Nano computer on each drone. So, basically, our code send required data to the our application over Jetson Nano's server to fly drones automatically as shown in Figure 5.1.

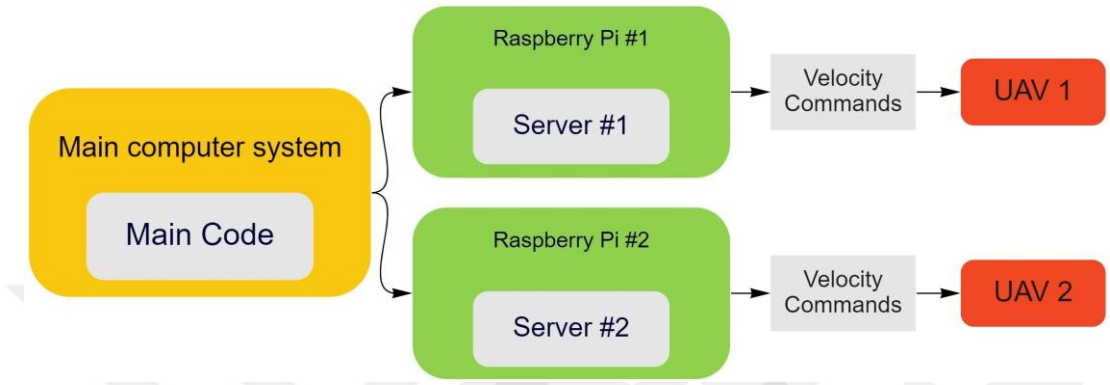

Figure 5.1 Block diagram shows how movement commands are sent to DJI drones

Our final drone setup is shown in Figure 5.2. Each drone has the identical components, which help them to fly autonomously with velocity input commands. We used Jetson Nano computers, as in the simulations, to realize the deep learning method and the image processing techniques which require a GPU and sufficient computational power. To be able to feed the Jetson Nano computer with enough power, we used a small LiPo battery, 3C1S 1100 mAh, and power regulater which gives 5 V and 3 A output. Also, we placed a RasPicam camera sensor to get image data and a UWB sensor to get distance data. After this setup, all hardware part for the experiment was ready.

As the initial step for the real experiments, we conducted detection tests with the Jetson Nano computer. As shown in Figure 5.4, we power up the Jetson Nano computer with a new power module and connect it to the monitor and start the image processing module within the ROS system. As we can observe in Figure 5.5, image processing was working fine and we can detect the monitor object shown in Figure 5.6 with our system.

Figure 5.2 DJI “Mavic 2” and “Air 2” drones that were modified for experiment purposes.

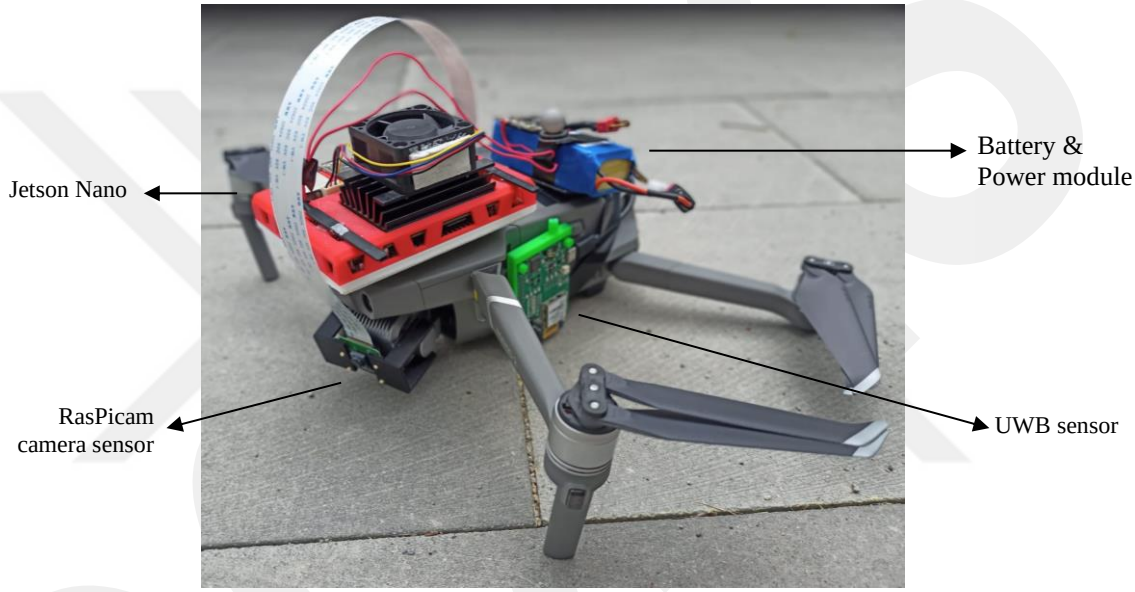

Figure 5.3 Image shows each component that was added to DJI drones

Figure 5.4 Image shows image processing test setup for DJI drones

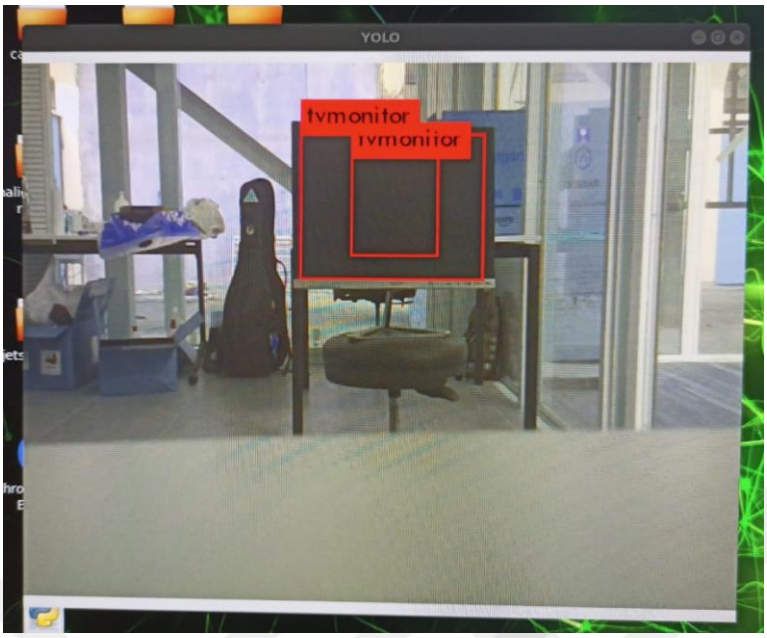

Figure 5.5 Monitor detection with Jetson Nano and onboard camera sensor

Figure 5.6 Detected monitor in our image processing system

After the sucessful initial detection tests with the individual Jetson boards, we set up the complete system, connected the two drones, and ran the detection code to detect our object of interest simultaneously. As demonstrated in Figure 5.7 and Figure 5.8, our drones can detect monitor objects with the YOLOv3 at the same time. We noticed that the camera FOV allows good detection rates at the bearing angle interval $\phi_i \in \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]$ radians. Notably, if a camera with a larger FOV is used (e.g., a fish-eye camera), bigger intervals

can be obtained. Also, the Jetson Nano boards yielded the detection rate at around 10 frame per second (fps) under default configurations, which was sufficient for our particular application.

Figure 5.7 Detection of the object in a straight direction with a) Left drone and b) right drone

Figure 5.8 Oriented yaw angle detection of the object with a) Left drone and b) right drone

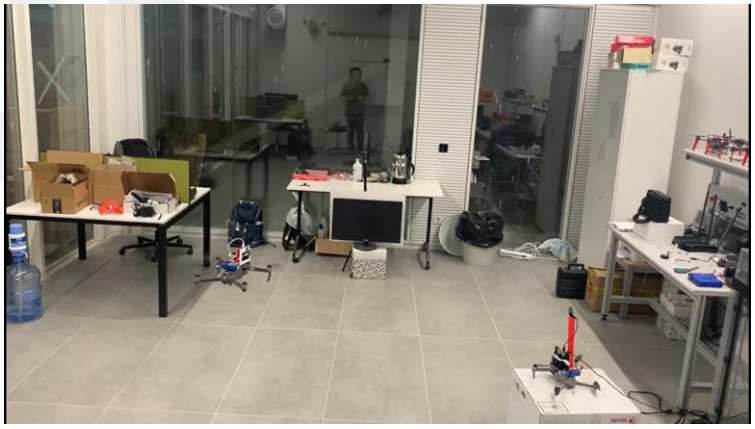

Figure 5.9 Experimental setup in a lab environment

Afterwards, we tested the Converge mode of the formation control algorithm with one drone while the other drone was at rest. As shown in Figure 5.9, we put the monitor object and our drones in a way that they can start the experiment from converge state. The first drone (left drone in Figure 5.9) aimed at maintaining the desired distance toward the target and regulating the bearing angle ϕ_1. We observed a successful performance in three tests. In the future, we plan to tune the system parameters and conduct a complete test of the proposed framework.

Chapter 6

Conclusions and Future Prospects

6.1 Conclusions

Localization methods are needed for many different applications such as search and rescue missions or automation. For two main different environments, outdoor and indoor environments, we have different types of data and sensors which help us to get the required knowledge on the position of the robots. To be able to locate our agent in the outdoor environment, we mainly rely on GPS data which we can get from GPS sensors and at least three or four satellites around the world. This signal is very helpful for finding our position in an outdoor environment but in indoor environments, we cannot use this signal clearly. In an indoor environment, we have many obstacles and objects that interfere with GPS signals that prevent us from using them clearly. With this study, we created a new approach for indoor localization by using image processing techniques and distance sensors. In this work, two identical drones can fly in their respective body frames, they can move freely in an indoor environment and be able to mimic harsh environments, these drones do not use a magnetometer sensor that gives rotation angle for drones which result in, drones do not know their yaw angles. To be able to give sense to drones for their environment, a forward-looking camera sensor is placed and used to receive image data for localization. With UWB sensors placed on them, they can get distance measurements between drones and objects of interest, which can be detected by the image processing system. Also, by using these distances, we can define the angle between drones and object which help us to define our yaw angle for drones. By using an Extended Kalman Filter, we can use these distance measurements and angle calculations to filter them and estimate our position in the environment. By using image data from camera sensors and image processing techniques, we can detect objects of interest within the image frame to estimate our relative position and improve our Extended Kalman Filter results. Drones will move

in 4 different states that in each state they will try to move and fix different data to make localization better.

6.2 Social Impact and Contribution to Global Sustainability

Localization systems are used in many different areas which robotic systems involve in our life. With new research, we started to use those systems in many different areas such as farming, agricultural spraying, and cinematography. In those systems, gathering position data of the agent is so important and needed to execute the requested objective, which will be helpful for humans and our societies in many different ways. With this study, we created a new localization method for indoor environments which robot swarms with two drones work together to detect a defined object to localize themselves around it to estimate their respective position. While realizing this mission, they can use image processing techniques that can detect pre-defined objects placed in the environment. By this detection, the position of this object can be defined in the image frame and this data can be used by autonomous systems to create position data. With this system, we can solve the indoor localization problem which resulted from interfering with GPS signals, by combining estimation and image processing techniques in a way that we can estimate our relative position by checking detected objects in our image frame. By using the proposed system, we can localize any robot in a different indoor environment which has different objects and shapes, and we can use this system for different purposes such as search and rescue missions, which we can detect a wounded person that needs to be rescued and define its respective position to send required help to them. In many situations where we cannot rely on GPS data, such as interference and signal blocage, we can use the proposed study to localize any agent in the system, and moreover, we can modify this work by adding different kinds of sensors or different robots to make it better and suitable for requested work. Also, by using this system, we can improve its working mechanism to speed up the ongoing process to make it better.

6.3 Future Prospect

For future prospects, this system can be improved in a way that we can add more drones or ground robots to improve our image data and detection power, which can be resulted in better detection and more possibilities. For future research, improving the estimation method and image data will be a good way to create a better system for future works.

BIBLIOGRAPHY

[1] Shan M, Wang F, Lin F, Gao Z, Tang YZ, Chen BM. Google map aided visual navigation for UAVs in GPS-denied environment. In: IEEE. ; 2015: 114–119.

[2] Patterson T, McClean S, Morrow P, Parr G. Utilizing geographic information system data for unmanned aerial vehicle position estimation. In: IEEE. ; 2011: 8–15.

[3] Lindsten F, Callmer J, Ohlsson H, Törnqvist D, Schön TB, Gustafsson F. Georeferencing for UAV navigation using environmental classification. In: IEEE.;2010: 1420–1425..

[4] Bansal M, Daniilidis K, Sawhney H. Ultrawide baseline facade matching for geo-localization. In: Springer. 2016 (pp. 77–98).

[5] Majdik AL, Albers-Schoenberg Y, Scaramuzza D. Mav urban localization from google street view data. In: IEEE. ; 2013: 3979–3986.

[6] Majdik AL, Verda D, Albers-Schoenberg Y, Scaramuzza D. Micro air vehicle localization and position tracking from textured 3d cadastral models. In: IEEE. ; 2014: 920–927.

[7] Majdik AL, Verda D, Albers-Schoenberg Y, Scaramuzza D. Air-ground matching: Appearance-based GPS-denied urban localization of micro aerial vehicles. Journal of Field Robotics 2015; 32(7): 1015–1039.

[8] Jiang S, JiangW. On-board GNSS/IMU assisted feature extraction and matching for oblique UAV images. Remote Sensing 2017; 9(8): 813.

[9] Conte G, Doherty P. An integrated UAV navigation system based on aerial image matching. In: IEEE. ; 2008: 1–10.

[10] Conte G, Doherty P. Vision-based unmanned aerial vehicle navigation using geo-referenced information. EURASIP Journal on Advances in Signal Processing 2009; 2009: 1–18.

[11] Indelman V, Gurfil P, Rivlin E, Rotstein H. Distributed vision-aided cooperative localization and navigation based on three-view geometry. Robotics and Autonomous Systems 2012; 60(6): 822–840.

[12] Vemprala SH, Saripalli S. Collaborative Localization for Micro Aerial Vehicles. IEEE Access 2021; 9: 63043–63058.

[13] Vemprala S, Saripalli S. Monocular vision based collaborative localization for

micro aerial vehicle swarms. In: IEEE. ; 2018: 315–323.

[14] Nazemzadeh P, Fontanelli D, Macii D, Palopoli L. Indoor localization of mobile robots through QR code detection and dead reckoning data fusion. IEEE/ASME Transactions On Mechatronics 2017; 22(6): 2588–2599.

[15] Faigl J, Krajník T, Chudoba J, Přeučil L, Saska M. Low-cost embedded system for relative localization in robotic swarms. In: IEEE. ; 2013: 993–998.

[16] Amer K, Samy M, ElHakim R, Shaker M, ElHelw M. Convolutional neural network-based deep urban signatures with application to drone localization. In: ; 2017: 2138–2145.

[17] Haameid RD, Al-Abudi BQ, Hassan RN. Automatic Object Detection, Labelling, and Localization by Camera's Drone System. Iraqi Journal of Science 2021: 5008–5023.

[18] Ahmed D, Qureshi WS, Aijaz SA, Imran BM, Naqvi SMA, Lin CY. Towards Selfie Drone: Spatial Localization and Navigation of drone Using Human Pose Estimation. In: IEEE. ; 2021: 1–7.

[19] Khattar F, Luthon F, Larroque B, Dornaika F. Visual localization and servoing for drone use in indoor remote laboratory environment. Machine Vision and Applications 2021; 32(1): 1–13.

[20] Husodo AY, Jati G, Alfiany N, JatmikoW. Intruder drone localization based on 2D image and area expansion principle for supporting military defence system. In: IEEE. ; 2019: 35–40.

[21] Pavliv M, Schiano F, Reardon C, Floreano D, Loianno G. Tracking and relative localization of drone swarms with a vision-based headset. IEEE Robotics and Automation Letters 2021; 6(2): 1455–1462.

[22] Yi J, Srigrarom S. Near-Parallel Binocular-Like Camera Pair for Multi-Drone Detection and 3D Localization. In: IEEE. ; 2020: 204–210.

[23] Nam SY, Joshi GP. Unmanned aerial vehicle localization using distributed sensors. International Journal of Distributed Sensor Networks 2017; 13(9): 1550147717732920.

[24] Floros G, Van Der Zander B, Leibe B. Openstreetslam: Global vehicle localization using openstreetmaps. In: IEEE. ; 2013: 1054–1059.

[25] Zamir AR, Shah M. Accurate image localization based on google maps street view. In: Springer. ; 2010: 255–268.

[26] Le Barz C, Thome N, Cord M, Herbin S, Sanfourche M. Global robot

egolocalization combining image retrieval and hmm-based filtering. In:; 2014:6-p.

[27] Bansal M, Sawhney HS, Cheng H, Daniilidis K. Geo-localization of street views with aerial image databases. In: ; 2011: 1125–1128.

[28] Zhang H, Wang G, Lei Z, Hwang JN. Eye in the sky: Drone-based object tracking and 3d localization. In: ; 2019: 899–907.

[29] Schönberger JL, Pollefeys M, Geiger A, Sattler T. Semantic visual localization. In: ; 2018: 6896–6906.

[30] Li J, LiuY,Wang J,Yan M,YaoY. 3D semantic mapping based on convolutional neural networks. In: IEEE. ; 2018: 9303–9308.

[31] Garcia A, Mittal SS, Kiewra E, Ghose K. A convolutional neural network feature detection approach to autonomous quadrotor indoor navigation. In: IEEE. ; 2019: 74–81.

[32] Ta DN, Ok K, Dellaert F. Vistas and parallel tracking and mapping with Wall–Floor Features: Enabling autonomous flight in man-made environments. Robotics and Autonomous Systems 2014; 62(11): 1657–1667.

[33] Lin Y, Gao F, Qin T, et al. Autonomous aerial navigation using monocular visual-inertial fusion. Journal of Field Robotics 2018; 35(1): 23–51.

[34] Zou D, Tan P. Coslam: Collaborative visual slam in dynamic environments. IEEE transactions on pattern analysis and machine intelligence 2012; 35(2): 354–366.

[35] Tiemann J,Wietfeld C. Scalable and precise multi-UAV indoor navigation using TDOA-based UWB localization. In: IEEE. ; 2017: 1–7.

[36] 36. Salado AM, Vandeportaele B, Lacroix S, Hattenberger G. Flight autonomy of micro-drone in indoor environments using lidar flash camera. In: Citeseer. ; 2010.

[37] Carrillo-Arce LC, Nerurkar ED, Gordillo JL, Roumeliotis SI. Decentralized multi-robot cooperative localization using covariance intersection. In: IEEE. ; 2013: 1412–1417.

[38] Nerurkar ED, Roumeliotis SI, Martinelli A. Distributed maximum a posteriori estimation for multi-robot cooperative localization. In: IEEE. ; 2009: 1402–1409.

[39] Knuth J, Barooah P. Distributed collaborative localization of multiple vehicles from relative pose measurements. In: IEEE. ; 2009: 314–321.

[40] Martinelli A, Pont F, Siegwart R. Multi-robot localization using relative observations. In: IEEE. ; 2005: 2797–2802.

[41] Indelman V, Nelson E, Michael N, Dellaert F. Multi-robot pose graph localization and data association from unknown initial relative poses via expectation

maximization. In: IEEE. ; 2014: 593–600.

[42] Sorbelli FB, Das SK, Pinotti CM, Silvestri S. Precise localization in sparse sensor networks using a drone with directional antennas. In: ; 2018: 1–10.

[43] 43. Sorbelli FB, Das SK, Pinotti CM, Silvestri S. Range based algorithms for precise localization of terrestrial objects using a drone. Pervasive and Mobile Computing 2018; 48: 20–42.

[44] Sorbelli FB, Pinotti CM. Ground localization with a drone and uwb antennas: Experiments on the field. In: IEEE. ; 2019: 1–7.

[45] Steup C, Beckhaus J, Mostaghim S. A Single-Copter UWB-Ranging-Based Localization System Extendable to a Swarm of Drones. Drones 2021; 5(3): 85.

[46] Magnago V, Palopoli L, Buffi A, et al. Ranging-free UHF-RFID robot positioning through phase measurements of passive tags. IEEE Transactions on Instrumentation and Measurement 2019; 69(5): 2408–2418.

[47] Grzonka S, Grisetti G, BurgardW. A fully autonomous indoor quadrotor. IEEE Transactions on Robotics 2011; 28(1): 90–100.

[48] Carrio A, Tordesillas J, Vemprala S, Saripalli S, Campoy P, How JP. Onboard detection and localization of drones using depth maps. IEEE Access 2020; 8: 30480–30490.

[49] Alarifi, A., Al-Salman, A., Alsaleh, M., Alnafessah, A., Al-Hadhrami, S., Al-Ammar, M. A., & Al-Khalifa, H. S. (2016). Ultra wideband indoor positioning technologies: Analysis and recent advances. Sensors, 16(5), 707.

[50] Bjelonic M. YOLO ROS: Real-Time Object Detection for ROS. https://github.com/leggedrobotics/darknet_ros; 2016–2018.

[51] Lin TY, Maire M, Belongie S, et al. Microsoft coco: Common objects in context.In: Springer. ; 2014: 740–755.

CURRICULUM VITAE

2013 – 2019	B.Sc., Electrical& Electronics Engineering, Abdullah Gul University, Kayseri, TURKEY
2019 – 2022	M.Sc., Electrical & Computer Engineering, Abdullah Gul University, Kayseri, TURKEY

SELECTED PUBLICATIONS AND PRESENTATIONS

Güler, S., Yıldırım, I., Alabay, H., "Mutual Relative Localization in Heterogeneous Air-Ground Robot Teams", 19th International Conference on Informatics in Control, Automation and Robotics, July 2022, Accepted