OBSTACLE DETECTION AND AVOIDANCE ON DJI TELLO DRONE

Guruvaraprasad Borra[1]

***Abstract*— The rapid development of autonomous navigation technologies worldwide has led to the quick development of technologies like object detection, and object tracking has become the priority. The main focus of this project is detecting and tracking the objects using YOLOv8 and DeepOCSORT algorithms. These technology applications are broadly utilized in robotic industries, unmanned aerial vehicles (UAVs), agriculture, etc. These advanced technologies are implemented on the DJI Tello Drone. Real-time detection of objects and tracking and dynamic planning of the direction of the drone to prevent a collision is effective and guaranteed by the methods incorporated in this research. We are going to find the results of the drone automatically avoiding the object to prevent collision with high precision. This research analysis presents how the low-cost and low-processing capability drone incorporates advanced technologies for providing an effective avoidance mechanism becoming the highlight of the project for environmental purposes including autonomous navigation, surveillance, and search and rescue operations. This project becomes the primary technique for future evolution techniques in UAV autonomous navigation and surveillance by improving its parameters and security. This is going to be the primary useful technique in real-world operations.** **KEYWORDS: Autonomous Navigation, Object Detection, Unmanned Aerial Vehicles,Computer Vision,YOLO, DeepOCSORT,Time-To-Collision(TTC), Real-Time Tracking.**

I. INTRODUCTION

With the success of advanced detection of objects and multi-object tracking detection algorithms, we developed a new method to operate drones with minimal processing capacity. In this work, we started utilizing advanced technologies such as the YOLO model and the Deep-OC-SORT algorithm to perform avoidance mechanisms for the drone. The history of Unmanned Aerial Vehicles (UAVs) is extensive and dates back two centuries. The word refers to aircraft and unmanned aerial vehicles, like drones, that have seen enormous development throughout the years, evolving from simple flying aircraft to their current uses in a variety of industries as well as in warfare. Drones were first introduced In 1917 as radio-controlled aircraft. After that, it was employed militarily throughout World War I and II. Drones became commercially available in 2006, but flying drones was not considered a hobby until 2013 when Amazon announced that drones would be utilized for deliveries. Later, in the evolution of drones, we find two types: one is operatable and the other one is autonomous. Drone technology's rapid uses is now seen in commercial applications such as pipeline inspection, agricultural appraisal, and security, as well as disaster relief management and border monitoring. It is now accessible as an application on smartphones and has become user-friendly. In our study, we are utilizing the DJI Tello drone. Tello firm has an application that allows us to play with the drone. These drones' characteristics include the capacity to capture 5MP images and HD 720p in MP4 format. When fully charged, it can fly for around 13 minutes, reaching a height of 30 meters at a speed of 8 meters per second and covering a distance of 100 meters. It weighs only about 80 grams when the propellers and batteries are included. The drone is a portable device due to its small size, and its propeller is three inches long. This drone was used as our autonomous method for the open-source programming framework. Object detection and avoidance are the primary aspects of our project using the UAV for enhancing safety and providing security in its environment. For efficient object detection, we used the YOLOv8 model, which is a state-of-the-art method to detect objects. DeepOCSORT algorithm was used for the tracking of objects that build on the strengths of traditional SORT by incorporating deep feature embeddings. The avoidance system was developed with the information provided by the tracking system. My contribution to this project was to use the advanced technologies implemented on the drone to avoid collision in a dynamic environment.

Fig. 1. Drone

A. Objectives

The key objective of this project was to capture the live video feed continuously from the drone camera to the laptop computer through Wi-Fi and process the frames using YOLOv8 to identify the objects in the frame. We also wanted to track the objects using the DeepOCSORT tracking algorithm using the detected objects and find the distance and speed. To inculcate the trajectory analysis we used the tracking information for finding the direction of the object. Another goal was to calculate the "time to collision" using the tracking information and making the avoidance mechanism by annotating information. Finally, we wanted to ensure efficient path planning and the safety of the drone.

B. Hardware

A laptop was used throughout the whole project process. This project used a Lenovo IdeaPad Slim 5 with an 11th gen Intel Core i5 CPU and Windows 11 (64-bit), and it had 16 gigabytes of RAM. Both the laptop and the drone were linked to the drone's Wi-Fi to control and command the drone from the laptop and to provide a seamless transition of the real-time video feed. We used these advanced technologies to achieve the estimated results for this research project. Thorough testing of the drone's detection, tracking, and avoidance capabilities was made possible by this gear. The DJI Tello Drone and laptop hardware combine to create a strong platform for creating and testing autonomous navigation systems, showing how affordable and easily available solutions may integrate cutting-edge technology to improve UAV capabilities.

C. Software tools

The software tools used in this research form the foundation for real-time object recognition, tracking, and avoidance. Python is the primary programming language because of its extensive library ecosystem, ease of usage, and high flexibility. We used many libraries for this project, to investigate what each source would exhibit in terms of their capability for identifying the object, tracking it, and working the avoidance mechanism effectively. Libraries like opevCV for real-time computer vision applications, which include opening the video streaming window for reading the video and saving it in the local space mentioned. NumPy is a Python library that is necessary for managing numerical tasks, such as processing image and video data. The use of this highly useful library was essential to our project's success in accomplishing its objectives, such as object identification, tracking, and drone avoidance. Another Python SDK module for interacting with the Tello drone is called Djitellopy. This module was used to perform activities like object detection, tracking, and collision avoidance with objects inside the collision range. The next module we utilized for this project was Ultralytics, where the YOLO was imported to carry out tasks like object recognition in the frame. The Boxmot module was used to import DeepOCSORT. We used this to carry out tasks like object tracking, object re-identification, and object trajectory analysis. Other modules used for this project in different applications include Os, math, time, datetime, argparse, sys, pathlib, and pynput. We completed all tasks with a high rate of software utilization by utilizing this library. These Python ecosystem features allow drones to recognize, track, and avoid objects for safe drone navigation. We utilized Visual Code (VS Code) for code editing and Anaconda to manage all of the Python packages and modules in the project's development environment. We used these applications for our easy programming language and to improve the productivity of the project. With these software tools and technologies, we could handle the drone and accomplish our objectives with these libraries as we expected for the completion of our project. These are necessary for achieving the goals and showcasing the potentiality of the project using the advanced algorithms practically with the drones.

II. METHODS

In this section, we will describe the five main methods we used to ensure the project's success. Every stage of the project method ensures the project advancements of the technologies for detection, tracking, and avoidance. The five main key steps are:

- Initializing the Drone
- Capturing Frames
- Object Detection
- Object Tracking
- Making Avoidance Decision

The flow chart of the techniques used in this project is described in this UML diagram. Initially, we set up the drone to record the camera's video feed, transfer it to the laptop over the internet via Wi-Fi, and use computer vision algorithms to collect the frames. After that, we had to read

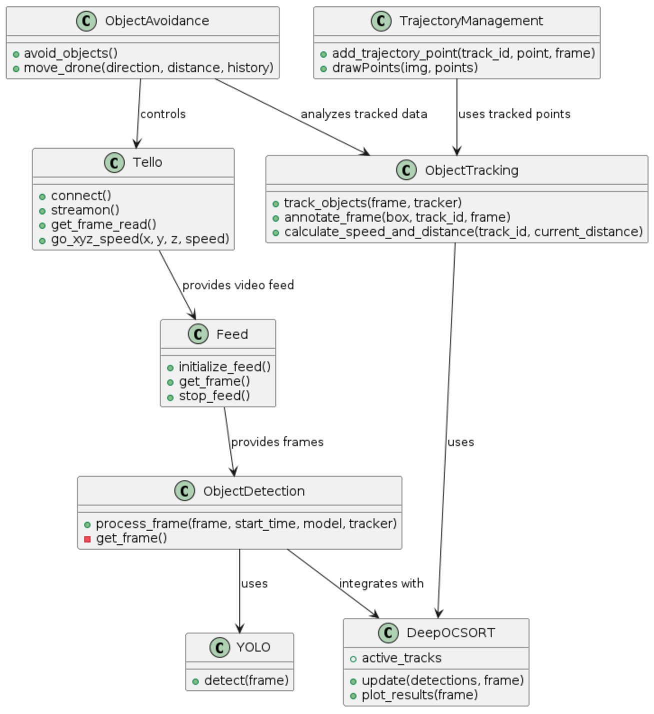

Fig. 2. Flow Chart of the Project

the frames,store them, run them through the YOLO model to identify any objects, and then send the information about the objects to DeepOCSORT so that it can monitor them using the distinct IDs that the YOLO model has supplied. Then, we had to make a judgment based on the analysis of the trajectory and the Kalman Filter's forecast on the object's likelihood of colliding in the future. Next, we will focus on the main steps of the project, discussing how these steps are working to succeed in the project goals.

A. *Initializing the Drone*

The first step in the method is Initializing the drone, which processes the drone to prepare for steady flight and precise results. For this, we need to be adept enough to calibrate the drone. After calibrating the drone, it should be used for the project. We did IMU status calibration, which allows us to calibrate in all six axes. We then updated the drone to the new version. To become comfortable with the drone, we utilized the Tello app. We played with the drone by experimenting with take-off, moving it left and right, up and down, and clockwise and counterclockwise. After becoming comfortable with these neutral commands, we experimented with the flight modes, discovering Throw Go, 8D Flips, Up Away, and 360, which let us record videos while spinning and circulating. The last and most intriguing mode is the bounce mode, which lets the drone fly up and down within its range, such as 0.5 height and 1.2 meters of flat surface. Next, we looked at the Tello SDK file to see how we might leverage the Tello commands more effectively for our project. Using the Djitellopy library, we discovered the necessary components for the project and set up the drone to transmit control commands for the flight. To start the drone via Wi-Fi, we used the Tello Connect command. The following step is made easy by utilizing the drone camera to take videos while we stream on command to begin the streaming. After the YOLO and DeepOCSORT modules had been initialized, we had to execute the takeoff instruction later. To prevent the drone's safety feature from landing after 15 seconds if no instruction is received, we were primarily giving it the keep-alive command. This is a built-in feature to ensure the drone lands safely; however, because of the laptop processor's capabilities, initializing the advanced modules took some time.

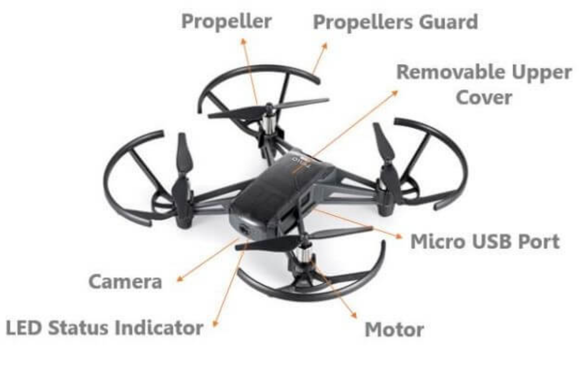

Fig. 3. Components of the Drone

This figure describes the parts of the drone. The components consist of the following: motor, micro USB port, camera, propeller, propeller guard, detachable upper cover, and LED status indicator.

B. *Capturing Frames*

Using the Tello SDK, we streamed the drone camera to obtain the live video feed while the drone was initializing. Afterward, the laptop receives the video feed over Wi-Fi. The real-time video is shown in a new window once the video feeds are analyzed frame by frame using the Open Source Computer Vision Library. A cv2 video writer creates a video taking 20 frames per second while keeping the spatial dimension (resolution) at 960*720 pixels. The output video frame should be recorded in color using the data format given by FourCC. The YOLO module then receives the frame for object detection.

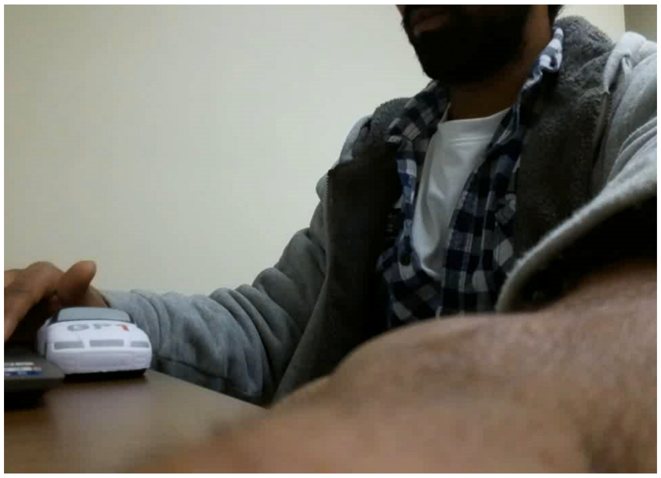

Fig. 4. Frame

C. *Object Detection*

The YOLO model will make use of the frames that the openCV has collected. The Convolutional Neural Network of the YOLO model processes the frames in order to identify the item using the class labels, boundary boxes, and confidence score. The object recognition model we're utilizing here, called YOLOv8 (You Only Look Once), is well-known for its accuracy and quickness. The YOLO operates by resizing the input image to 448*448 pixels and then passing it through a single CNN. Each grid cell in this image, which is 7 by 7, is in charge of identifying two bounding boxes (x,y) that have the width and height of the bounding box as well as the center of the bounding box. The confidence score is measured by

$$\text{Confidence} = P(\text{Object}) \times \text{IOU}(\text{Intersection Over Union})$$

The procedure takes this high confidence score bounding box into account. The class probabilities for every object class that appears in the grid cell are then determined.

$$\text{Output} = 7 \times 7 \times (B \times 5 + C)$$

Where:

- 7×7: Grid Size
- B: Number of Bounding Boxes

- 5: Five attributes that YOLO Predicted values(x,y,w,h& Confidence Score)
- C: Number of Classes

After that, YOLO employs CNN to extract spatial information and predicts the necessary confidence score, bounding boxes, and class probabilities. These classes are identified as unique IDs for each object.

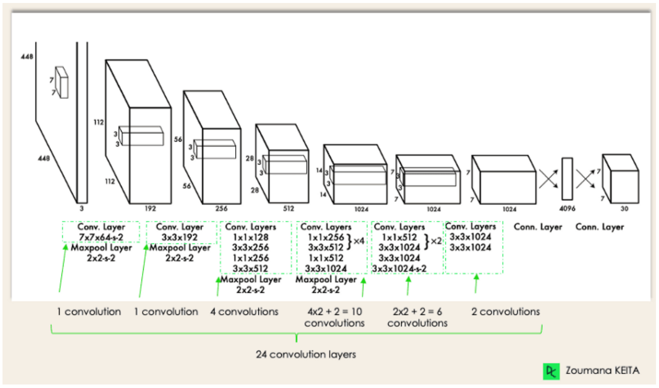

Fig. 5. Architecture

Figure 5 represents the architecture of the YOLO model, which includes 24 convolutional layers.

D. Object Tracking

The YOLO model was utilized in earlier research to identify objects based on their unique IDs. We can accomplish object tracking and continually forecast the objects' future positions and trajectories. To accomplish all of the aforementioned tasks, this research made use of a cutting-edge technique known as DeepOCSORT. Based on the classic MOT (Multiple Object Tracking) of SORT, Deep Learning Observation Centric Simple Online and Realtime Tracker is created using deep feature embeddings. That provides a cutting-edge solution for tracking objects by improving SORT and OCSORT to achieve deep learning feature identification and object reidentification of these changes makes the approach more useful for this project. To anticipate the future position of objects by ensuring speed, distance, and velocity, this advanced algorithm builds on the object recognition that took place in the previous technique. It then employs the Kalman Filter and optical flow technologies based on the unique IDs. The optical flow ensures that the pixels move inside the frame, which helps the Kalman Filter generate predictions. Bounding boxes aid in the feature extraction process when using deep neural networks. The Hungarian Algorithm, which links the objects to their unique IDs, aids in the data association process. The monitoring of the items in the frame then continues, and tracking management is carried out over time with the activities of the objects that are lost from the marked frames. When the procedure is finished, the objects have unique IDs and the modified position is recorded. The findings of this approach, which tracks objects and predicts their speed and time to collision—both of which are used to inform decisions in subsequent steps and the trajectory path of the toy—are displayed in Figure 7.

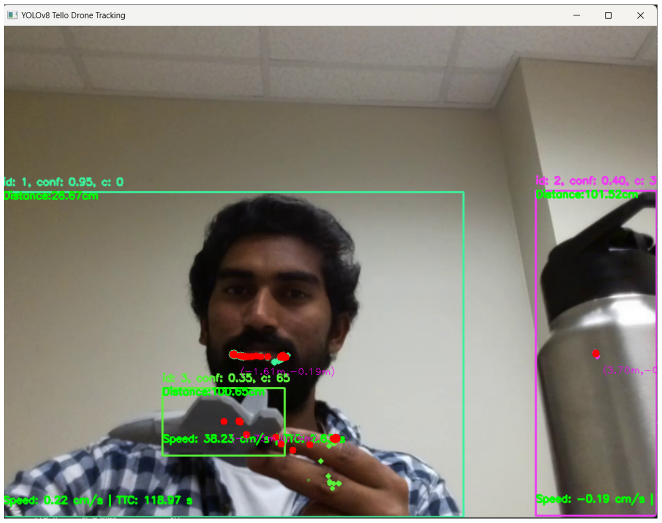

Fig. 6. Object Tracking

E. Making Avoidance Decision

One of the project's primary goals and objectives is obstacle avoidance. Here, the avoidance mechanism uses the data from the cutting-edge technologies covered in the earlier sections. The YOLO, known for its precision, handles obstacle detection, providing the tracking algorithm with accurate bounding box classes and unique IDs. Using the following equation, the tracking algorithm utilizes the distances of the tracked obstacles.

$$\text{Distance} = \frac{(\text{Object Width} \times \text{Focal Length})}{\text{Width in Pixels}}.$$

After calculating the distance, the object tracker keeps adding further object-tracking data. In this instance, the most recent 25 frames are retained, while the remaining data is withdrawn. Using distance, we estimate the speed and time to collision (TTC) from the equation below.

$$\text{TTC} = \frac{\text{Current distance}}{\text{speed of the object}}.$$

Here, we used the computed data to decide on the best action to prevent collisions in real-time. We utilized a threshold of a minimum of 18 frames of tracking information to ensure that there is sufficient evidence to make an informed decision. This approach indicates that after 18 frames, the object will be considered to determine whether it falls within the TTC range. We designed our proposed mechanism to instruct Tello to decide once we received all the data from the object with a minimum of 18 historic timestamps in object tracking data. The avoidance mechanism checks if the TTC of the object is in less than 3 seconds, then we pass the mechanism to make an obstacle avoidance decision. With the TTC data and the relative frame, the center of the object position is utilized to make decisions. Here, the decision-making is in two situations, with the appropriate decision-making of the four cases. An obstacle is a severe threat when the object is in the range of a threshold percentage of 30% of the frame width from the center, i.e., in the central 60 percent of the frame. If this is the case, when the object is located on the left side of the frame center, the algorithm decides to move the drone to the right to avoid collision by moving 80 cm right. If the object is located on the frame center's right side, it will move to the left for 80 cm in the second case if it is not in the immediate threat range, where the object is in the drone path but a little away from the frame center. Then, the object moves slightly to the left if it is on the right for 40 cm. If the object is on the right, the drone moves to the left by 40 cm.

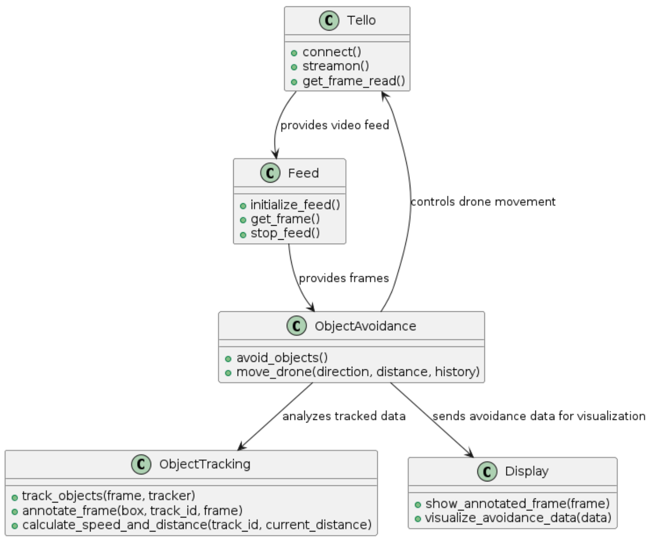
Tello
connect()
streamon()
get_frame_read()
provides video feed
controls drone movement
Feed
initialize_feed()
get_frame()
stop_feed()
provides frames
ObjectAvoidance
avoid_objects()
move_drone(direction, distance, history)
analyzes tracked data
sends avoidance data for visualization
ObjectTracking
track_objects(frame, tracker)
annotate_frame(box, track_id, frame)
calculate_speed_and_distance(track_id, current_distance)
Display
show_annotated_frame(frame)
visualize_avoidance_data(data)

Fig. 7. Making Avoidance-Decision

Figure 7 provides the UML diagram of Making Avoidance Decision.

III. EXPERIMENTS

In this section, we provide experimental results for this research to demonstrate the achievements of advanced technologies. We conducted experiments on the drone multiple times in an indoor environment. The experiment was conducted to determine the drone's ability in three cases.

- Immediate Threats
- Moderate Threats
- No Threats

1) Immediate Threats: 1) Immediate Threats: The drone will locate the object in the first scenario if it is within 30% of the horizontal width of the frame. There is a significant risk of collision because the object is on a direct flight path. If the object is inside the collision range, which is TTC<3 seconds, the drone will be informed to move 80 cm to the left if the object is on the right or move 80 cm to the right if the object is on the left.

2) Moderate Threats: 2) Moderate Threats: In this scenario, the object will be on the drone's path through the obstacle collision but slightly away from the center. These objects are considered while minor changes to the flight path are made. When an object is inside the collision range, defined as TTC<3 seconds, the drone will be alerted to move 40 cm to the left if it is on the right or 40 cm to the right if it is on the left. This makes it easier to separate the hazards to the drone's immediate flight path from moderate ones.

3) No Threats: In this scenario, the drone will be in its path because the detected objects are not in the TTC range. This is because to find the drone's movement when there is no threat, it should be in its way.

A. Results

We conducted many tests using the drone to execute autonomous object recognition, tracking, and avoidance mechanisms to demonstrate the effectiveness of the suggested approach. The YOLO model's CNN layer was applied to the drone's frame capture results to identify the objects in the frame. The advanced algorithm DeepOCSORT was used to track the discovered items. Here, we could determine the object's distance and use that distance to get the object's speed and TTC. By demonstrating the drone's capacity to dynamically modify its flight path, the avoidance system worked to prevent collisions. This study introduces the YOLOv8 for real-time object identification, DeepOCSORT for tracking, and trajectory planning, and it helps to avoid collisions of objects. The evaluation of the research was calculated using four main metrics.

- Detection Accuracy
- Tracking Stability
- Avoidance Success
- Processing Speed

1) Detection Accuracy: Here, the detection accuracy of the YOLOv8 model is ascertained by identifying the bounding box of the object position and object categorization with an average of 75% to 98%, we could confirm that the object was accurately spotted in the frame.

2) Tracking Stability: We were determining the DeepOCSORT algorithm's tracking stability in this metric and verifying the function of the algorithm that keeps the objects' assigned IDs consistent throughout the frame until they leave and trying to track for a few moments, as well as the objects' trajectory and distance to anticipate the TTc to minimize mistakes.

3) Avoidance Success: In this metric, we determine the performance of the avoidance system. We monitor the system's effectiveness in responding to threats and making decisions to acquire movement to avoid a collision. The system is effective if it keeps the drone within a safe distance of the objects in its frame by executing the imminent, moderate, and no threats.

4) Processing Speed: In this section, we looked at our system's performance speed. When the video feed is sent from the drone to the processing laptop over Wi-Fi, we see a slight delay in real-time frame processing. The processing time, which involves analyzing the frame, identifying the object using YOLO, tracking the object using DeepOCSORT, and making avoidance decisions, was measured by looking at the time before the frame was analyzed. We found the time after completing the avoidance analysis once more. The processing time is calculated by subtracting the beginning and end times.

B. Challenges and Improvents

This section will outline our difficulties and provide ways to improve this research. After calibrating the drone's IMU, we discovered the most challenging issue with the project: we missed the built-in feature that causes the drone to automatically land if it doesn't get a command within 15 seconds of taking off. We discovered this issue later and started sending stay-alive commands every ten seconds. Then, we found that the drone battery's charging capabilities were a physical constraint; we had problems with the battery during testing. To make the drone safe, we also discovered how complicated the algorithms were and combined them to obtain the tracking data. Next, we had to launch a new window to write and show a video on a laptop using the OpenCV library. The complexity of the cutting-edge technology, which requires time to implement, was the cause of the issue. After implementing state-of-the-art technology, we started the drone's takeoff and streaming. Next, we discovered a two to five-second delay when the live video streaming was sent to the laptop for frame processing. The average time is about 3.5 seconds, with a maximum of 5. We further improved the algorithm by fine-tuning the parameters to achieve faster results. We upgraded the tiny drone to use advanced drones. We tried advanced drones. We did the experimentation in an external environment and tested the drone external environment and test drone in different conditions, like light, dark, and inclinations. Improving the trajectory and path planning will enhance the system's performance, and improve the research results.

IV. CONCLUSIONS

This research accomplished object detection and avoidance using advanced technologies. Advanced technologies like the YOLO and the DeepOCSORT play a crucial role in identifying and tracking the objects over time for the project, which helps achieve all the objectives. The drone went through demanding testing conditions multiple times to make the drone find the person's speed, and when it is in the collision range, it should find the path to prevent the collision. This study highlights the use of cutting-edge technologies that integrate with small drones effectively to improve the use of UAVs in the autonomous navigation field. This research helps us pave the road to future advancements in the autonomous navigation and robotic industries by developing this type of system. This research helped me improve my skills in advanced technologies like object detection, tracking, and making an avoidance mechanism.

ACKNOWLEDGMENT

I would like to express my gratitude to Drs. Mahmut Reyhanoglu, Abiye Seifu, Qi Zou, and Mohammad Jafari for their unwavering support and motivation during this endeavor. The principles of this study benefited from the guidance and mentoring of Dr. Jafari. I would like to express my gratitude to him for inspiring me to tackle the challenges that arose during the project. I would like to express my gratitude to Columbus State University's Robotics Department for offering me an excellent opportunity to develop my abilities, as well as the necessary equipment and a collaborative research setting.

REFERENCES

[1] Tello Development - Getting Started, https://docs.google.com/presentation/d/1Sn4Fp9BBiagm-aFM6no5Fu8qTxBVWxiDcdvqiEoDqZ4/edit#slide=id.gac36f8f75c_0_0, Accessed 30-08-2024.

[2] DJI Tello Obstacle Avoidance with YOLOv4, https://www.youtube.com/watch?v=GY7GjPntrx4&list=PLMrmVZVtQTc2su8F2FB_I-lkrv83rUgcI&index=7, Accessed 30-08-2024.

[3] Tello SDK 2.0 User Guide, https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%202.0%20User%20Guide.pdf, Accessed 30-08-2024.

[4] DeepSORT — Deep Learning applied to Object Tracking, https://medium.com/augmented-startups/deepsort-deep-learning-applied-to-object-trac=king-924f59f99104, Accessed 30-08-2024.

[5] Simple Online and Realtime Tracking Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, Ben Upcroft Submitted on 2 Feb 2016

[6] Simple Online and Realtime Tracking with a Deep Association Metric Nicolai Wojke, Alex Bewley, Dietrich Paulus Submitted on 21 Mar 2017

[7] Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking Jinkun Cao, Jiangmiao Pang, Xinshuo Weng, Rawal Khirodkar, Kris Kitani Submitted on 27 Mar 2022

[8] Deep OC-SORT: Multi-Pedestrian Tracking by Adaptive Re-Identification Gerard Maggiolino, Adnan Ahmad, Jinkun Cao, Kris Kitani Submitted on 23 Feb 2023

[9] Git Hub of DEEPOCSORT https://github.com/GerardMaggiolino/Deep-OC-SORT Accessed 30-08-2024.

[10] Hungarian ALgorith https://hungarianalgorithm.com/hungarianalgorithm.php Accessed 30-08-2024.

[11] Ultralytics https://github.com/ultralytics/ultralytics Accessed 30-08-2024.

[12] You Only Look Once: Unified, Real-Time Object Detection Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi Submitted on 8 Jun 2015 (v1)

[13] YOLO9000: Better, Faster, Stronger Joseph Redmon, Ali Farhadi Submitted on 25 Dec 2016

[14] Real-Time Flying Object Detection with YOLOv8 Dillon Reis, Jordan Kupec, Jacqueline Hong, Ahmad Daoudi Submitted on 17 May 2023

[15] Distance was Measured using this reference https://github.com/Asadullah-Dal17/Distance_measurement_using_single_camera/tree/main/Speed Accessed 30-08-2024.

[16] Distance MEasuring https://www.1stvision.com/machine-vision-solutions/2015/09/imaging-basics-calculating-lens-focal.html Accessed 30-082024.

[17] Distance MEasuring https://pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-=python-opencv/ Accessed 30-082024.

[18] PlantUML Online https://plantuml.online/uml/ Accessed 30-08-2024.

[19] Drone Evolution https://percepto.co/the-evolution-of-drones-from-military-to-hobby-=commercial/ Accessed 30-08-2024.