# Algorithmic Approaches for Object Tracking and Facial Detection Using Drones

Kareem Shahatta, Peter Savarese, Gina Egitto, Jongwook Kim
West Chester University
{ks1000662, ps980816, ge936577, jkim2}@wcupa.edu

**ABSTRACT**

Drones are unmanned aerial vehicles that have a variety of uses in many fields such as package delivery and search operations. Tello is a small, programmable drone designed for educational purposes. We developed algorithms using DJI Tello Py, an open-source Application Programming Interface, to command the movements of Tello for tracking a target object (i.e., human). Our algorithms utilize digital image processing techniques on Tello's live video stream to optimize the number of movements Tello needs to reach its target. This paper explains our approaches to implement object-tracking and facial detection for Tello, discusses lessons we learned, and highlights improvements for future work.

## 1 Introduction

From military use to video recording, drones are a versatile asset for everyone. Drones can be controlled either directly with a remote/app or programmed for a specific purpose. As drones have become smaller, less expensive in price, and easily programmable with computer programming languages, it has opened the door to rethink modes for object detection. Traditionally, detecting and tracking objects (e.g., human faces) have relied on static and stationary security cameras such as doorbells and Closed-Circuit Television (CCTV). This limitation requires multiple cameras for complete coverage of a premises. Drones easily address this issue since they are not stationary, being capable of moving on all three axes and performing yaw rotations.

We developed algorithms to detect human faces and control Tello drones [1] to follow a target object (i.e., human). Our algorithms for Tello allow us to use the included camera and its live camera feed for non-stationary object tracking so that we can move Tello toward a human by detecting a subject's face in the camera feed, calculating the required X, Y, and Z-axis movements, and sending the required movements back to Tello. Our algorithms prioritize how Tello moves and help detect its own movement in the air by stimulating a velocity for it. In return, Tello can reach its target using the minimum number of movements. Our experiments show that with the algorithms we developed, it is possible to have a drone follow a subject with only a single camera feed under real-world environments.

## 2 Approaches

### 2.1 Tello Drones

Tello is a drone made by Ryze Technology [1]. It is designed for educational and testing purposes and can be controlled by using the provided Application Programming Interface (API) with any programming language (e.g., Python, Java, C/C++, etc.) over a Wi-Fi network. This enables us to tailor Tello to accept specific commands and control its movements and flight patterns. Tello comes equipped with a 720p, 2592 x 1936 camera with an 82.6° Field of View [2]. Tello can fly for up to 13 minutes of total continuous flight time. Its compact size, measuring 98 x 92.5 x 41 mm and weight of 80 g, allows for a small and nimble testing platform. Tello also has an easy-to-remove 1.1 Ah battery that slides in and out for minimal downtime. Figure 1 shows a Tello drone equipped with propeller guards.


Figure 1: Tello Drone with Computer Mouse for Scale.

### 2.2 Drone API

We used DJI Tello Py [3], a Python API library, provided by Escoté et al. [4] to control Tello. This library serves as a way to utilize Ryze Technology's Software Development Kit (SDK) [5] for interacting with Tello drones. The DJI Tello Py library provides interfaces to communicate with Tello via Wi-Fi, using the User Datagram Protocol (UDP), a transport layer protocol of the Internet protocol suite (i.e., TCP/IP), for wireless communication. The SDK comes with multiple built-in commands for controlling Tello.

Listed below are commands provided by DJI Tello Py that we utilized.

- **Take off** instructs Tello to start its propellers, take off and maintain its position.
- **Land** instructs Tello to land and turn its propellers off.
- **Up/Down/Left/Right (X)** instructs Tello to move in a specific direction relative to its current position, where **X** denotes distance in centimeters.
- **Forward/Back (X)** instructs Tello to move forward or back relative to its current position, where **X** denotes distance in centimeters.
- **Streamon** instructs Tello to turn its camera on and begin a live stream of its camera feed over the Wi-Fi network.

Listed below are commands we created for controlling Tello.

- **Camera Feed** uses Tello's $Streamon$ command and starts streaming Tello's camera feed, opens a window with a live camera stream preview, and begins detecting by using our **Detect** command.
- **Detect** turns on face detection to detect faces from Tello's camera feed. Once a face is detected, our program computes X, Y, and Z-axis movements and commands Tello to move, resulting in the face being in the center of the camera feed.

### 2.3 Object Recognition

We used the Open Computer Vision Library (OpenCV) [6] and its digital image processing functions for object tracking and facial detection on Tello's live video stream. We used digital image processing techniques to highlight and extract data from Tello's video stream. This data includes the number of humans and type of objects found. We analyzed Tello's livestream video by breaking it down into image frames, which are individual images extracted from a video, and processed one at a time. OpenCV provides real-time image processing functionality, allowing us to instantly analyze each image frame on Tello's live stream video as we capture it. We used an algorithm called Haar Cascade classifier [7], which is included in OpenCV, to perform face detection.

We chose the Haar Cascade algorithm because it is already trained with a massive dataset of human faces and it allows us to control its accuracy at detecting human faces. The algorithm detects human faces by scanning and analyzing a single image frame at a time. If the classifier successfully detects a human face in an image frame, it draws a green square around that face, starting from the top left corner of the face (see Figure 2). From the green square we extract its dimensions (width and height) to calculate how far the target is from Tello, and its coordinates on X and Y-axes to determine where the target is standing. The extracted data is then used to provide movement instructions for Tello.

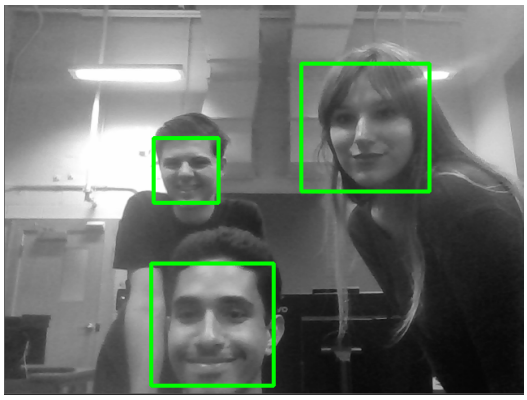

Figure 2: Human Faces Detected Using Haar Cascade Algorithm.

## 3 Implementation

### 3.1 Multi Threaded Programming

We used multi-threaded programming techniques in Python to communicate with Tello. A multi-threaded program can perform multiple tasks in a program concurrently (i.e., one task per thread). Controlling Tello while receiving image frames from it requires a minimum of two threads; one for the camera feedback and another for the command prompt. The camera feedback thread handles the digital image processing and facial detection tasks. This thread runs when Tello turns on, and it receives and displays Tello's live stream video in real time. We used the camera feedback thread to monitor what Tello sees and analyze how fast or accurately it can detect human faces. The command prompt thread is used when we (users) send a command to Tello. This thread takes the user's command, converts it into Tello instructions, and sends it to Tello over Wi-Fi. Tello responds to the command prompt instruction with either a confirmation or an error if it fails to execute.

Major tasks of the command line thread includes:

- Checking the battery status
- Starting/Stopping the propeller
- Turning the object recognition on/off
- Sending manual movement commands

### 3.2 Initial Movement

There are two scenarios in which Tello has to juggle between moving and handling facial detection. The first scenario is when Tello turns on, its facial detection ability is turned off until Tello becomes stable while hovering in the air. Facial detection is then automatically turned on after ten seconds of taking off. We did this to prevent Tello's facial detection ability from crashing when Tello slowly tries to balance itself in the air. We found from our tests that ten seconds is the ideal time for Tello to become steady in the air after taking off.

The second scenario is when Tello is about to begin moving. We turn facial detection off when Tello moves and turn it back on when it hovers in the air. We did this to not cause any calculation conflict around when we move Tello or the target (i.e., human) moves. Therefore, when Tello is moving, it does not detect if the target changes its position, so it does not cause any interruption to its movement. This behavior causes Tello to move like a chess piece on a grid, making one movement at a time, and does not sense its environment until it stops moving. When Tello's facial detection is on, the camera feedback thread takes each image frame from Tello's live video and analyzes it using the Haar Cascade algorithm. As shown in Figure 2, a green square is drawn around every face detected in the image frame to indicate a match. Once the image frame is scanned and analyzed, we pass it to our Priority Moving Algorithm.

### 3.3 Priority Moving Algorithm

Our Priority Moving (PM) algorithm determines when and how Tello should move. We created this algorithm to overcome a limitation from Tello's Software Development Kit (SDK) [3] – Tello can move only on one axis at a time (see Figure 3). The PM algorithm determines which axis it should move along first. The axes are prioritized in order from highest to lowest as the Z-axis (forward and backward), the X-axis (left and right), and the Y-axis (up and down). The movement algorithm processes the axes and coordinates from the extracted image frames to determine which axis to move along first. In return, this helps Tello reach its target using the minimum possible number of movements.

Tello's first movement occurs on the Z-axis. We move Tello along the Z-axis by the difference between the target's Z-coordinate and our maximum cap limit (210 pixels). The maximum cap limit is a value that Tello can not move any closer beyond. Without the maximum limit, nothing prevents Tello from colliding with the target or getting too close that it will not accurately detect movement on the X or Y-axis. The maximum cap limit for the Z-coordinate does not reflect the distance from Tello to the target, but instead, it reflects the diagonal length of the square drawn on the target's face. The maximum cap limit also serves as an ideal distance between Tello and the target so that Tello can begin moving on another axis like the X and Y-axes. Once Tello is close enough to the target, we focus on the X-axis.

For the movement on the X-axis, we first calculate the average movement of the target on the X-axis by storing ten consecutive X-coordinates. If the target's X-axis movement exceeds our X-axis move limit value (5 cm), we move Tello on the X-axis by the average value we calculated. Otherwise, we ignore the target's movement on the X-axis. Here is the reason: when Tello hovers in the air, it moves a little, causing a slight change in the target's X-coordinate. Hence, we assume that any X-axis movement below 5 cm will be considered insignificant and ignored. Note: Tello cannot perform facial detection while moving on the X-axis simply because it is not able to differentiate between its movement and the target's movement, causing Tello to keep moving indefinitely.

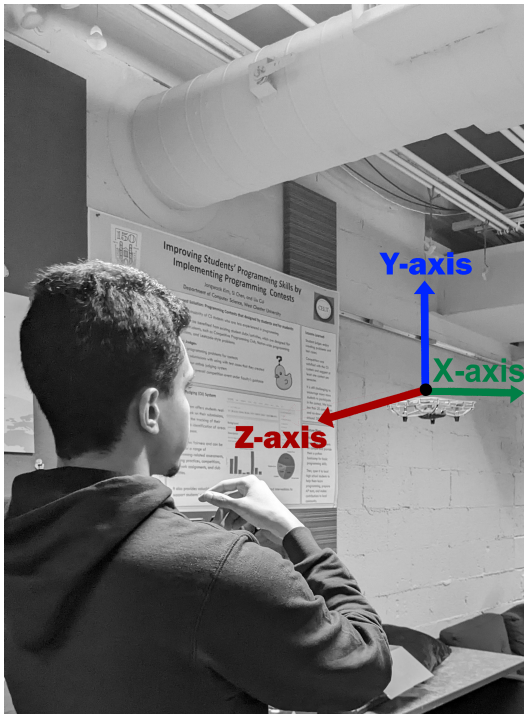


Figure 3: Direction of Tello's X, Y, and Z-axes.

Considering users on elevated platforms like hills and cliffs, we implemented the Y-axis movements for Tello to follow the target going up and down staircases. For the Y-axis, we use the same algorithm as the X-axis but instead use the Y-coordinate and the Y-axis. Therefore, it runs into the same problem (as the X-axis) of not stopping its facial detection while moving along the Y-axis. To overcome this problem for both the X and Y-axes, we created our own Artificial Velocity algorithm to predict when Tello begins to move and when it stops moving.

### 3.4 Artificial Velocity Algorithm

We created the Artificial Velocity (AV) algorithm to predict when Tello will stop moving since the SDK does not provide this ability. This algorithm uses the current speed of Tello to calculate how much time Tello needs to reach the target from when it is instructed to move. We first figure out the distance that Tello will travel in centimeters using the PM algorithm on the X or Y-axis. Second, we use Tello's speed value to estimate how fast it will travel. Finally, we combine these two values to create an artificial velocity for Tello to estimate the time it will take to travel somewhere. Since we can approximately estimate the time Tello will take to fly on the X or Y-axis, we can use this to turn facial detection off when Tello is moving and then turn it back on when Tello stops. To achieve all of this we implemented a timer using multi-threaded programming skills (Section 3.1), and activated only when Tello

moves along the X or Y-axis.

## 4 Evaluation

We spent a total testing time of 20 hours to validate Tello's face detection and tracking functionality. We performed testing in a classroom with desks, monitors, whiteboards, etc. for real-world settings. During each test, a single subject stood against a background consisting of a white wall, the aforementioned classroom objects, and adequate lighting, simulating realistic conditions. To account for AI biases with different human demographics, three subjects participated in the experiments: A white female with long hair, a Middle Eastern male with short hair, and a white male with short hair. Tello was situated on a desk and once commanded to take off, Tello rose to be approximately at eye level with the subject.

For evaluating our PM algorithm, we assigned three roles to our team members in a rotating pattern during each test. The three roles were the evaluator, the test subject, and the observer. First, the test subject stood in front of Tello and became its target. Second, the evaluator evaluated the extracted data from each image frame. Third, the observer observed the tests from a bystander perspective. When the subject moved, the evaluator looked over the movement instructions and took notes of its accuracy, where then the observer confirmed if the movement instructions were properly executed. This helped us fine-tune the entire algorithm to create smoother movement.

For X-axis tracking and movement, we checked if Tello centered the subject following its left or right movement. Similarly, for Y-axis we observed that Tello maintained the subject's face centered as the target moved up and down. For Z-axis tracking, we checked if our PM algorithm could control Tello by moving toward (or away from) the subject when the length of the diagonal is less (or greater) than 210 pixels. With this, our evaluator confirmed that the maximum cap (210 pixels) was correctly preserved.

## 5 Related Work

Jintasuttisak et al. [8] created an animal detection and tracking system using DJI Mavic 2 Pro drones [9]. For object detection, they used a Deep Neural Network (DNN) algorithm (i.e., YOLO-V5I [10]) and found the best sub-version of YOLO that yielded the highest Mean Average Position for their tracking of Oryx animals. In our project, we tracked humans using a DJI drone as well, opting for OpenCV, a computer vision library, rather than using a DNN vision model.

Pawlicki et al. [11] measured Tello's image recognition performance in terms of speed and accuracy in detecting AprilTags [12], which are images similar to QR codes. They evaluated the drone's ability to read AprilTags from various distances and angles using snapshots, but did not use the video streaming capabilities of the Tello drone as we did. As we found during our experiments, they also concluded that effective lighting conditions are necessary to ensure the algorithms operate accurately.

Subash et al. [13] utilized Tello to detect objects and people, using Mask Region-based Convolutional Neural Network (Mask R-CNN) [14], a deep learning model used for object detection and pixel-by-pixel outlining, and by using OpenCV libraries as we did. They enabled the drone to recognize objects within its field of sight, including classrooms and objects within large photos. They trained the algorithm using annotated images allowing the drone to learn and improve its detection accuracy over time. While their project utilized detailed object outlines, we opted for traditional bounding boxes since our facial detection task only required approximate object locations.

Pohudina et al. [15] implemented drone swarms using Tello. Testing up to four drones at a time, they developed an algorithm to coordinate group flights. They found that complex movements, such as flips, result in the strongest positioning error, and that movements along the X-axis (left and right) incur greater positioning error than movements along the Z-axis (forward and backward). Like us, they opted for Tello drones due to the DJI Tello Py, affordability, and safe design.

## 6 Conclusions

We implemented object tracking drones using Tello, which is merely 98 x 92.5 x 41 mm in size and is equipped only with a 720p camera and no advanced sensors. We used existing software libraries: DJI Tello Py to control Tello remotely, OpenCV to perform digital image processing, and geometry for providing movement instructions. DJI Tello Py also provided APIs to utilize a live video stream of Tello's camera, allowing us to break the video into multiple image frames and process each individually. We used the Haar Cascade algorithm in OpenCV for facial recognition and data extraction on each image frame. Additionally, we created our own algorithms – Priority Moving Algorithm and Artificial Velocity Algorithm – for determining which axis Tello should move along first and for estimating a velocity for Tello to predict when it started and stopped moving.

Our work paves the way for a more versatile and robust drone with face detection and tracking. For example, expanding detection capabilities to recognize multiple faces will allow for better target selection by specifying which target of follow. Also, incorporating yaw rotation can enable 360° face detection and tracking so that it eliminates the X and Y-axes computations, allowing Tello to have a wider range of movement. While our current implementation restricts Tello's movement to move like a rook on a chessboard pattern due to limitations in the DJI Tello Py library, it successfully demonstrates real-world object tracking using a single camera and opens the door for future advancements in face detection and tracking.

## References


[1] Tello Drone, https://www.ryzerobotics.com/tello.

[2] Tello Specifications, https://www.ryzerobotics.com/tello/specs.

[3] DJI Tello Py, https://djitellopy.readthedocs.io/en/latest/tello.

[4] DJI Tello Py GitHub, https://github.com/damiafuentes/DJITelloPy.

[5] Tello SDK 2.0 User Guide, https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%202.0%20User%20Guide.pdf.

[6] OpenCV, https://opencv.org.

[7] Cascade Classifier, https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.

[8] T. Jintasuttisak, A. Leonce, M. Sher Shah, T. Khafaga, G. Simkins, E. Edirisinghe, Deep Learning based Animal Detection and Tracking in Drone Video Footage, in *International Conference on Computing and Artificial Intelligence*, page 425–431 (2022).

[9] Mavic 2 Pro Specifications, https://www.dji.com/mavic-2/info.

[10] YOLO-V5, https://pytorch.org/hub/ultralytics_yolov5.

[11] M. Pawlicki, K. Hulek, A. Ostrowski, J. Mozaryn, Implementation and Analysis of Ryze Tello Drone Vision-based Positioning using AprilTags, in *International Conference on Methods and Models in Automation and Robotics*, pages 309–313 (2023).

[12] AprilTags Project, https://april.eecs.umich.edu/software/apriltag.

[13] K. V. V. Subash, M. V. Srinu, M. Siddhartha, N. S. Harsha, P. Akkala, Object Detection using Ryze Tello Drone with Help of Mask-RCNN, in *International Conference on Innovative Mechanisms for Industry Applications*, pages 484–490 (2020).

[14] K. He, G. Gkioxari, P. Dollár, R. B. Girshick, Mask R-CNN, *CoRR*, *abs/1703.06870*.

[15] O. Pohudina, M. Kovalevskyi, M. Pyvovar, Group Flight Automation Using Tello EDU Unmanned Aerial Vehicle, in *International Conference on Computer Sciences and Information Technologies (CSIT)*, pages 151–154 (2021).