Real-Time Face Detection and Tracking Using Tello Drone

Arijit Bhattacharjee
Department of Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
abh89@uclive.ac.nz

Prof. Richard Green
Department of Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
richard.green@canterbury.ac.nz

***Abstract*—The challenge of detecting and tracking people using unmanned aerial vehicles or drones has been extensively studied in the research communities. Various approaches have been proposed and tested to tackle this challenge using different algorithms and methods. This paper proposes a method to detect and track people using Tello drone in an indoor setting. Four different approaches including Haar cascade, object detection using convolutional neural network, and keypoint and pose estimation using neural networks are used to detect faces in this study. The different face detection algorithms or models have been tested and compared, and the experimental results demonstrated their effectiveness. The model leveraging the MediaPipe framework, having a frame rate of 38.5 frames per second, a 35.8% decrease in face area per meter away from the drone, and a visibility of 96% is found to be most suitable for real time face detection. A simple feedback control system is implemented for face-tracking and to maintain a stable distance from the person. As a result the drone autonomously detects and tracks people in a controlled environment.**

Keywords—tracking, Haar cascade, convolutional neural network, neural network, keypoint and pose estimation, feedback control system

I. INTRODUCTION

Face detection as a research topic is one of the most frequently studied topics in the last few years. Face detection is used to determine the presence or absence of a face in an image. Detecting faces is very easy for humans; however, this task is very complicated for the computer because there is some complexity associated with location, viewing angle, light, and occlusion [7]. There has been a lot of research going on this field and significant advances in face detection or recognition have been made. The range of approaches varies from using the basic approach of Haar cascade algorithm [10] to using latest state of the art neural networks [8, 9].

Drone usage is also growing rapidly. It includes applications ranging from surveillance to search and rescue. Moreover, the production as well as adoption of drones is expected to increase in the coming years [21]. Drones are flying machines that possess the ability to reach almost any location, thanks to their agility and aerodynamic features. Most drones, irrespective of their sophistication level, are equipped with a camera that permits them to capture video footage or images of the subjects they are observing or recording, in real-time. As a result, they can be leveraged for identifying and monitoring individuals.

Researchers have worked on real-time face recognition, object detection, motion tracking, and object tracking [11, 12, 13]. Different machine learning algorithms have been explored by researchers [1, 6] to design and implement people detection and tracking using drones. Furthermore, humans are interacting with drones more frequently and the need for safety and flight consistency is now more important than ever. In almost all cases, it is crucial that drones can immediately recognise objects and respond accordingly in real-time, more so if the object is a person [22].

This paper proposes a method to compare some of the techniques of face detection e.g. Haar cascade, object detection using neural network, keypoint and pose estimation. The comparison is done on the baseline of receiving video from the webcam and outputting the movement commands to the terminal. Thereafter, face detection and tracking is done using a Tello drone on a computer with limited specifications. Drone motion control uses a PID control algorithm [7] that can work well on a comprehensive system with good results.

II. BACKGROUND

Research into how machine learning algorithms can be incorporated to detect and track people using drones has been carried out by others in the field of computer vision [1, 2, 3, 4, 5, 7]. This section will evaluate the work of a few important research that contributed to the development of this field of real-time object detection and tracking using drones.

O. T. Cetinkaya et al. [1] provide a good introduction to the problem of human tracking using drones. Haar Cascade and Histograms of Oriented Gradients (HOG) were used to detect the target body. In order to improve the tracking accuracy in outdoor environments a Kalman Filter was used. After that, a Fuzzy Logic algorithm was used to maneuver the drone toward the human in real-time [1]. However, the evaluation metrics used are not well explained, and it is unclear how they were chosen.

Y. Pu et al. [2] present a system that utilizes a deep learning-based face recognition model called OpenPose to detect and recognize faces and estimate distance from aerial images captured by a drone. OpenPose can analyze the joint position of the human body quickly and accurately and locate human forms through the Bottom-to-Up connection [2]. It achieves high accuracy in recognizing faces from aerial images. However, the system's implementation may require a significant amount of computational resources, which may limit its practicality in some applications.

A. Boonsongsrikul and J. Eamsaard [3] propose a real-time human motion tracking system using the Tello drone and the MediaPipe library. The system is designed to track a person's movements and gestures in real time. They then describe the algorithm they developed for human motion tracking, which involves detecting the person's body using MediaPipe and then using optical flow to track their movements. In comparison with the previous schemes, the proposed scheme shows comprehensive performance in various experiments, considering light conditions, target movement speeds, and target distances, covering both indoor

and outdoor scenarios [3]. The drone used in the study has certain limitations. Thermal noise occurring when the drone has a continuous flight over a long time can cause the drone to become unstable [3].

C. Cifuentes-García, D. González-Medina and I. García-Varea [4] propose a people detection and tracking system using an onboard drone camera. They utilize semantic segmentation algorithms such as SVMHOG, MobileNet-SSD, or YOLOv2Tiny to detect and track people in real time. Once a person is detected, the drone's movements are adjusted to center the person in the image, thus enabling tracking of the person as they move. The results were compared for the three semantic segmentation algorithms. However, the paper does not provide information on the computational resources required to implement the system.

D. Lee [5] presents a novel approach for detecting and tracking objects in videos using convolutional neural networks (CNNs). The proposed approach involved using a pre-trained YOLOv3 model for object detection and then applying three detection and tracking algorithms to combine the object detector and visual tracker [5]. However, whether the approach can be used for object tracking using drones is to be ascertained.

A. S. Priambodo et al. [7] present a method for face tracking using a Haar cascade classifier and a PID controller for a quadcopter drone. The system is designed to detect and track a person's face in real time using a camera mounted on the drone and then control the drone's movement to keep the face in the center of the frame. However, sometimes, the face is not detected due to poor lighting, or the algorithm does not get the feature it is looking for at that position [7].

III. PROPOSED METHOD

The proposed method is divided into three components namely face detection, face tracking or position control system and comparing the detection models. Each of these steps is treated as an individual component of the method.

The versions of software and specifications of the system used during the proposed method are shown in Table I.

TABLE I. SOFTWARE AND SYSTEM SPECIFICATIONS

Code production	Python 3.7
IDE	PyCharm Community Edition 2020.2.3
Webcam	720p
RAM	8GB
GPU	4GB NVIDIA GeForce GTX 1050
Processor	Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz
Operating system	Windows 10
Drone	Tello drone

A. Tello Drone

The Tello drone is a small drone manufactured by Ryze Tech in cooperation with Scratch and Swift, as illustrated in Fig. 1. This mini drone was chosen because it offers the best solution when flying around people without giving danger to the people around them. The small size of the Tello drone with dimensions of 98×92.5×41 mm and a weight of 80 grams makes this drone also practical and easy to fly indoors [7]. In addition, Ryze Tech has enabled Scratch and Swift to develop the program through the Python SDK, making it relatively easy to build functions and enhance the capabilities of the Tello drone. The drone can be controlled by a smartphone or a laptop computer using 2.4 GHz communication [3].

Fig. 1. Example of a Tello drone.

B. Proposed System

The complete proposed system is shown in Fig. 2. A Tello drone that can be communicated via a Wi-Fi network is connected to a laptop. The camera from Tello drone captures an image transmitted to a laptop for further processing using a computer vision algorithm whose details are discussed further below.

To perform face detection and tracking, an onboard RGB camera is used. A total of four object detection and pose estimation models are used. Pre-trained weights and models are used, so that we do not require any previous learning stage, and therefore don't need to capture and store a large training dataset. This help us to save time and computation resource. The cross-platform computer vision library, Open Source Computer Vision Library (OpenCV) version 4.5.5 was used to capture images and perform pre-processing and post-processing of the images.

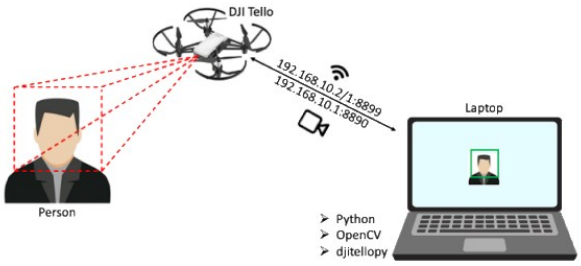

Fig. 2. Proposed system overview

The drone was first connected to the computer via Wi-Fi receptor on the drone. Using one thread, the video was streamed to the standard output of the terminal using the djitellopy library version 2.4.0, via a RealTime Streaming Protocol (RTSP) port. The RGB video was retrieved with a resolution of 360 x 240 pixels and colour channels were rearranged to match the input requirements of OpenCV and passed through four different models, namely Haar Cascade Classifier, YOLOV3, MediaPipe, and OpenPose.

C. *Haar Cascade Classifier*

The Haar Cascade classifier uses a machine learning object detection algorithm to identify objects in images or videos. It is trained on a lot of positive and negative images to detect objects in other images. It is particularly used to detect frontal faces in images by converting them to grayscale and applying the classifier with a certain scale factor and a minimum number of neighbors.

The Viola-Jones algorithm for face detection uses a cascade of simple Haar-like features to classify sub-windows of an image as containing a face or not. These features are efficiently computed using integral images and are selected using AdaBoost, which iteratively combines weak classifiers to form a strong classifier. The cascade structure enables efficient computation by rejecting non-face sub-windows at the early stages of the classification process. The algorithm also includes various optimization techniques, such as feature sharing and scaling, to improve performance and reduce computational costs. The resulting face detector is robust to changes in illumination, pose, and occlusion and can run in real time on a wide range of devices [10].

The next step involves detecting and tracking the largest face in an image. The algorithm iterates over detected faces, calculates the center and area, and stores them in lists. It finds the index of the largest face and accesses the center and area using the index.

D. *You Only Look Once Version 3*

The You Only Look Once version 3 (YOLOv3) is an object detection model that can detect and classify objects in real time with high accuracy. It uses a single neural network that predicts bounding boxes and class probabilities directly from full images in a single evaluation. YOLOv3 uses a feature extractor based on Darknet-53, a variant of the Darknet architecture, and has three output scales with different grid sizes, which allows the model to detect objects at different scales. YOLOv3 also employs several novel techniques, such as multi-scale predictions, a novel loss function, and spatially constrained anchor boxes, to improve object detection performance [11].

This model is used for face detection and recognition. The next step involves looping through the outputs of a YOLOv3 model and checking for detections with a confidence score greater than 0.5 and class ID 0 (person). For each detection, the bounding box coordinates were extracted and the area of the bounding box was calculated. The center coordinates of each bounding box were also tracked and appended to a list. The algorithm then uses non-maximum suppression to remove overlapping bounding boxes and draws circles and rectangles around the remaining ones. The class name, confidence score, and area of the bounding box for each detection can also be extracted. Finally, the bounding box with the maximum area is chosen and the list of face centers and areas that can be accessed are updated.

E. *Mediapipe*

The MediaPipe framework offers a modular pipeline to develop and deploy computer vision and machine learning models, including BlazePose, on various platforms and devices. It includes face detection and landmark tracking, hand detection and landmark tracking, pose estimation, and holistic tracking (face, hand, and pose tracking). BlazePose is a body pose tracking model developed by Google Research that detects key points on the human body with high accuracy and speed. It uses a lightweight neural network architecture based on the BlazeFace model and a single-shot detection algorithm. The model is trained on large-scale datasets and can track multiple people in real time on mobile and embedded devices. MediaPipe Pose is a machine learning solution that infers all landmarks and a background segmentation mask for the whole body from RGB video frames, with a total of 33 human poses as displayed in Fig. 3 [12]. The MediaPipe holistic framework simultaneously tracks human pose, face landmarks, and hands in real-time, using four steps: estimation of the human pose and landmark model with MediaPipe Pose, selection of inferred pose landmarks in three regions of interest (face, both hands), re-cropping of these regions with a full-resolution input frame for improvement, and application of task-specific models to estimate corresponding landmarks for each hand and face [3].

The next step involves processing an input image to detect faces using the face detection model in MediaPipe. For each detected face, a bounding box is drawn around it and its center and size are calculated. The center and size of each detected face are stored in separate lists. If at least one face is detected, the input image with the largest detected face's center and size is returned.

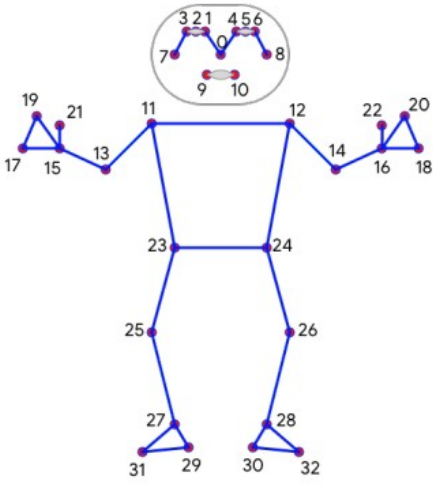

Fig. 3. MediaPipe keypoints

F. *Openpose*

OpenPose is a real-time multi-person 2D pose estimation system that estimates the 2D locations of body joints of multiple people in a single image or video frame. The algorithm starts by detecting the person's body using a single-person body part detector and then proceeds to

estimate the 2D keypoint locations of the detected body parts. It uses a part affinity field representation to capture the spatial relationships between body parts and then refines the keypoint locations using a graphical model based on the estimated part affinity fields. Finally, the algorithm performs non-maximum suppression to remove duplicate detections and outputs the final set of 2D pose estimations for all detected persons in the frame. OpenPose can handle both the single-person and multi-person scenarios, and it can estimate the poses of people even when they are occluded or in complex situations [13].

An input image was passed and pose keypoints were extracted. The BODY_25 model was used in this study. It checks if any pose keypoints are detected, and if so, it loops through each keypoint to obtain its x and y coordinates, which are then appended to separate lists. The nose, eyes, and ears keypoints are also extracted to calculate the center point of the face. As the shoulders and chest are joint points that are less likely to be deformed by the human body, these two nodes could be calculated and extracted. The positions of the left and right shoulders are obtained to calculate the distance between them, which is used to determine the coordinates of the top-left and bottom-right corners of the bounding box. A bounding box is then drawn around the face using these coordinates, and the area of the bounding box is calculated. The center of x and y coordinates are converted to a NumPy array, and the function returns the center point and area of the bounding box.

G. Tracking Algorithm

In order to be able to do tracking, a parameter is needed that becomes a reference for the drone in making movements. The method proposed in this study is to create a set point in the form of a bounding box with a certain size located in the center of the image, as shown in Fig. 4. The object detection bounding box results are compared with the middle bounding box so that the error rate of the drone's position against the object's face is known [7]. There are three parameters, namely x, y, and a, representing the distance on the x and y axes, and a or area, the area of the face. The area of the face can be used as a measure for the distance between the drone and the face. This error is then used as a reference in calculating the proportional-integral-derivative (PID) control system to generate a control signal.[7]

Algorithm for simple implementation of a feedback control system for a face-tracking system is discussed below.

The face tracking algorithm takes in information about the position and area of the face in an image, as well as the width of the frame and a previous error term. The algorithm calculates an error term based on the position of the face in the image and the desired position and uses a PID controller to adjust the speed of the system.

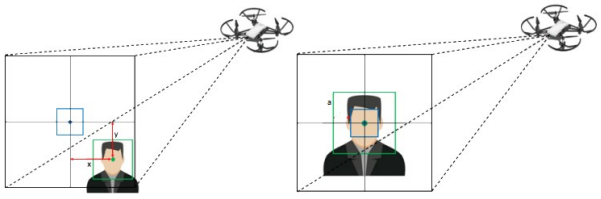

Fig. 4. Error calculation method. a) Error calculation in x and y axis. b) Error calculation in distance between drone and face

In a PID control system, a variable e represents the error obtained from the difference between the actual value of the position and the desired set point. The core of PID techniques is to tune the output on the system based on the behaviour of the error function. The general form of the equation of the PID controller [7] is written as follows:

$$u(t) = K_p\, e(t) + K_i \int e\, dt + K_d \frac{de}{dt} \qquad (1)$$

where t is the time and e is the error function. e(t) is the error in the system at time instant t. K_p , K_i and K_d are the coefficients for each component (proportional, integral and derivative) and the parameters to be changed to tune the effectiveness of the formula [23].

The most basic form of feedback control is called P Control because the control is proportional to the error: u = K_p * e. K_p is a constant of proportionality, so larger K_p will mean the controller will be more aggressive in the sense that it will apply large control efforts when it encounters large errors [15].

If we kept increasing K_p to reduce the lag, we would increase the overshoot. On the other hand, if we tried to reduce the overshoot by lowering K_p, we would cause the drone to move slower, increasing the lag. To solve this problem, we can add a derivative term to our controller to make a PD Controller. Our control becomes u = K_p * e + K_d * e' where e' is the error derivative and K_d is how much weight we assign to the derivative term. The derivative term keeps the drone response time the same while reducing its overshoot [15]. For easier implementation we have removed the integral component by setting K_i to 0, but it is still technically a PID control.

The algorithm used in this study is shown in Fig. 5. In general, it is an iterative process of face recognition, and calculation of control signals used to control forward or backward and rotational around the vertical axis or yaw movements.

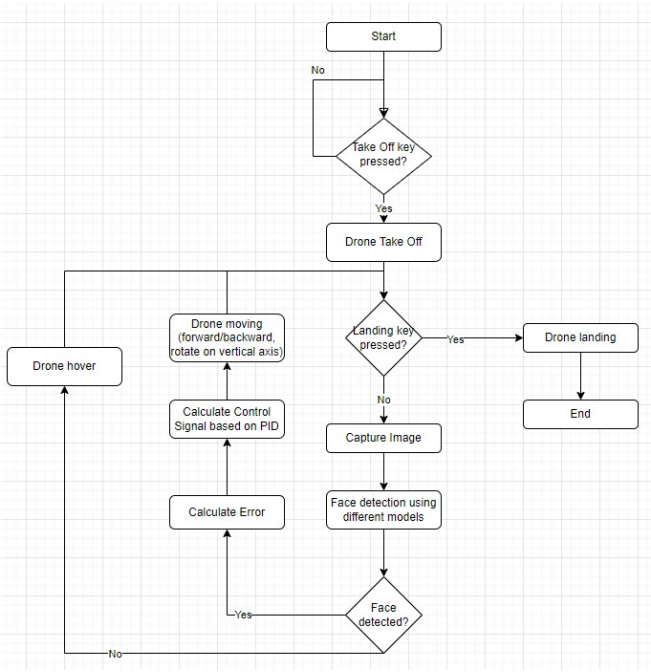

Fig. 5. Flowchart of the face tracking algorithm

H. Comparing models

In order to compare the different models the models' relative performance for execution time were considered. Execution time here refers to the time take by the model to detect a face. The execution time from the webcam stream and the drone stream were noted. For simplicity, the output from the drone stream with flight commands was not considered for comparison. There would be a slight increase in the time taken to read and detect a person in a given frame from the webcam stream to the drone stream. The execution time is expected to follow a similar trend as observed in the webcam stream for the drone stream, considering we are using the same drone and tracking algorithm [15]. Moreover, a distance of 1 metre and 2 metres were marked from the camera and area of the face detected from these distances were also noted. The reason for this was to determine the feasibility of face detection from larger distances. In order to determine the extent of the face visible on the frame, the visibility score was calculated by dividing the number of relevant pixels by the total number of pixels in the image or region for Haar Cascade and YOLOv3 model. Otsu's method [18] was used to automatically determine the threshold value that separates the face region from the background in a grayscale image. In the MediaPipe library, the visibility score can be directly extracted from the output of the face detection model. In the OpenPose library, visibility was calculated from the confidence score for the face keypoints.

IV. RESULTS

The test is carried out indoors with the different pre-trained face detection models. The results of the proposed method were gathered by measuring the time taken in seconds by the system to detect and track face in each frame. The results of execution time from webcam stream and drone stream are shown in Table II. It is established from Table II that the drone stream has proportionally higher execution time than the webcam stream but the trend across the models are similar. The average frame rate expressed in frames per second was calculated in a given trial. Furthermore, the area of the bounding box for the face was calculated at a distance of 1 metre and 2 metres respectively from the camera, and the percentage decrease in the area was noted. Visibilty was also calculated. The results comparing the models from the webcam stream are shown in Table III. Some examples of the output frames for different models are also shown in Fig. 6, Fig. 7 , Fig. 8 and Fig. 9.

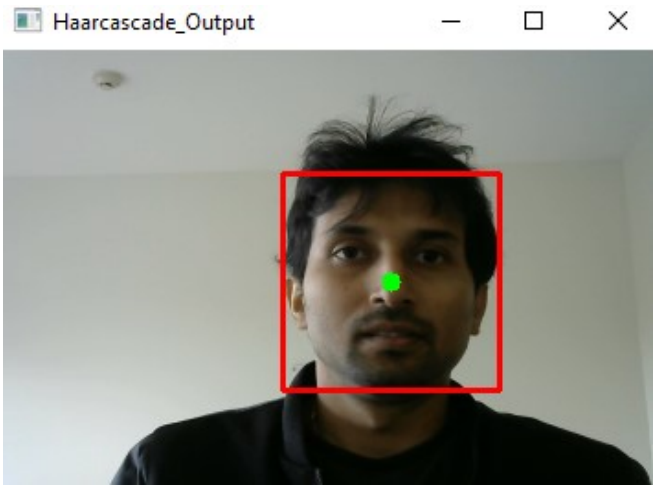

Fig. 6. Output from Haar cascade classifier.

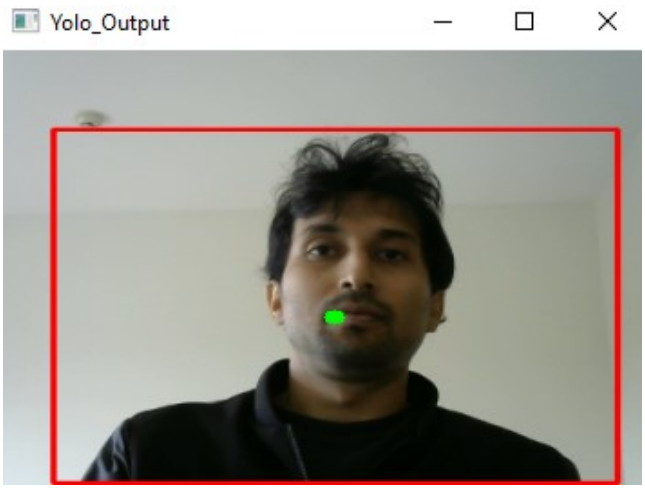

Fig. 7. Output from YOLOv3 model.

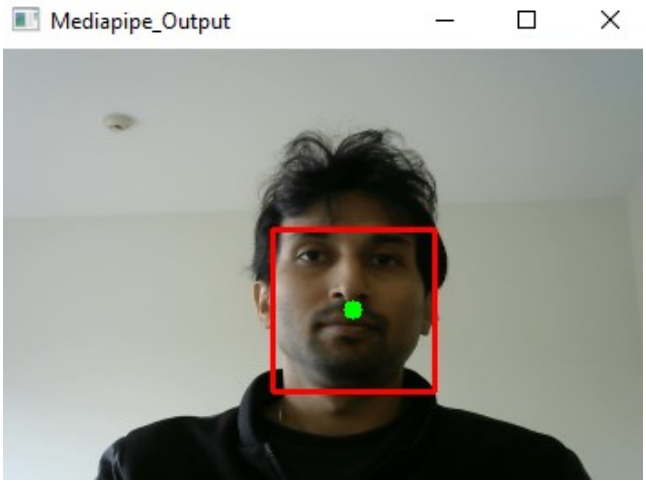

Fig. 8. Output from MediaPipe model.

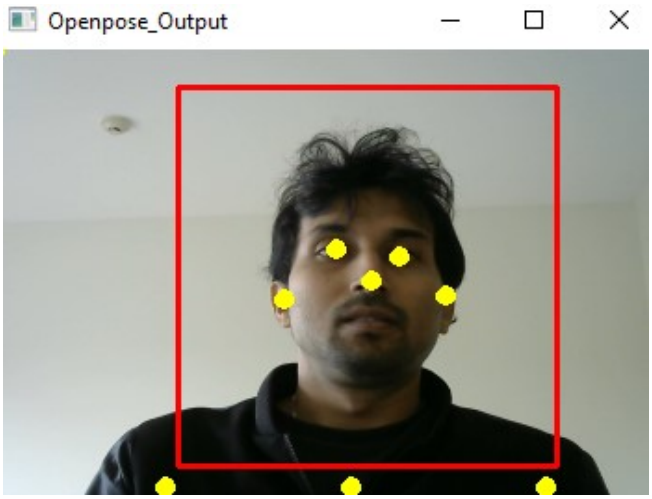

Fig. 9. Output from OpenPose model.

TABLE II. EXECUTION TIME

Stream	Haar Cascade	YOLOv3	MediaPipe	OpenPose
Webcam stream	0.09 sec	0.74 sec	0.026 sec	0.12 sec
Drone stream	0.14 sec	1.15 sec	0.035 sec	0.16 sec

TABLE III. COMPARISION BETWEEN MODELS ON WEBCAM STREAM

Metric	Haar Cascade	YOLOv3	MediaPipe	OpenPose
FPS	11.1	1.4	38.5	8.3
% Change of area per metre	71.7 %	11.9 %	35.8 %	72.5 %
Visibility	68	62	96	100

As Table II and Table III exemplify, the MediaPipe model has the fastest execution time and the highest frame rate. The percentage change of area per metre is lowest for YOLOv3. However, it has the slowest execution time. OpenPose has excellent visibility, but the decrease in the area with distance is the highest. Moreover, it was observed that the face is not detected sometimes due to poor lighting conditions for the Haar Cascade classifier and YOLOv3 model. No such problems with face detection were observed for models using keypoint and pose estimation i.e. MediaPipe and OpenPose.

Considering all the factors, MediaPipe seems to be the best performing model that can be used for face detection in drones. The results show that the proposed approach can be used to evaluate different face detection and tracking algorithms for drones.

A. Limitations

The Tello drone used in the study has certain limitations such as short battery life and limited flight time, which may impact the ability of the system to track human motion accurately in certain scenarios. Furthermore, the paper only presents the results of testing in a single indoor environment.

V. CONCLUSION

This paper proposed a method to integrate and compare object detection models with a Tello drone. The model leveraging the MediaPipe framework with a frame rate of 38.5 frames per second, a 35.8% decrease in face area per meter away from the drone, and a visibility of 96% is found to be most suitable for real time face detection. The method also demonstrates how a drone can be made to successfully detect and track a face in real time and establish and maintain a safe and stable distance from the person.

A. Future Research

The proposed approach could be improved by carrying out the test in other environments with different lighting conditions, backgrounds, and different types of human motion. More degrees of freedom for face detection and tracking could be tried out. Moreover implementing gesture commands to the proposed approach could make the person-drone interaction more advanced.

ACKNOWLEDGMENT

I would like to thank Professor Richard Green for his supervision and guidance. This research was supported by the Department of Computer Science and Software Engineering at the University of Canterbury.

REFERENCES

[1] O. T. Cetinkaya et al, "A fuzzy rule based visual human tracking system for drones," in 2019, . DOI: 10.1109/UBMK.2019.8907104.

[2] Y. Pu et al, "Aerial face recognition and absolute distance estimation using drone and deep learning," The Journal of Supercomputing, vol. 78, (4), pp. 5285-5305, 2022.

[3] A. Boonsongsrikul and J. Eamsaard, "Real-Time Human Motion Tracking by Tello EDU Drone," Sensors (Basel, Switzerland), vol. 23, (2), pp. 897, 2023.

[4] C. Cifuentes-García, D. González-Medina and I. García-Varea, "People detection and tracking using an on-board drone camera," in Robot 2019: Fourth Iberian Robotics ConferenceAnonymous Cham: Springer International Publishing, 2019, pp. 668-680

[5] D. Lee, "CNN-based single object detection and tracking in videos and its application to drone detection," Multimedia Tools and Applications, vol. 80, (26-27), pp. 34237-34248, 2021.

[6] T. T. Do and H. Ahn, "Visual-GPS combined 'follow-me' tracking for selfie drones," Advanced Robotics, vol. 32, (19), pp. 1047-1060, 2018.

[7] A. S. Priambodo et al, "Face Tracking for Flying Robot Quadcopter based on Haar Cascade Classifier and PID Controller," Journal of Physics. Conference Series, vol. 2111, (1), pp. 12046, 2021.

[8] T. Tsai and P. Chi, "A single-stage face detection and face recognition deep neural network based on feature pyramid and triplet loss," IET Image Processing, vol. 16, (8), pp. 2148-2156, 2022.

[9] S. Hangaragi, T. Singh and N. N, "Face Detection and Recognition Using Face Mesh and Deep Neural Network," Procedia Computer Science, vol. 218, pp. 741-749, 2023.

[10] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," International Journal of Computer Vision, vol. 57, (2), pp. 137-154, 2004.

[11] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018.

[12] V. Bazarevsky et al, "BlazePose: On-device Real-time Body Pose tracking," 2020.

[13] Z. Cao et al, "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, (1), pp. 172-186, 2021.

[14] Blake List, "Real-Time Object Tracking and Following with MiniDrones", Computer Vision Lab, University of Canterbury, Tech. Rep., 2019

[15] A. Parande. "Flying a Drone with Python: PID Control." Internet: https://blog.devgenius.io/flying-a-drone-with-python-pid-control-7001a41f54ac, Oct. 2, 2022 [Mar. 29, 2023].

[16] "Setting the PID controller of a drone properly." Internet: https://www.technik-consulting.eu/en/optimizing/drone_PID-optimizing.html, n.d. [Mar. 29, 2023].

[17] P. Ryan. "Tello drone Face follower." Internet: https://medium.com/swlh/tello-drone-face-follower-4b8313b6c40e, Jun. 15, 2020 [Mar. 25, 2023].

[18] J. Xue and D. M. Titterington, "t -Tests, F -Tests and Otsu's Methods for Image Thresholding," IEEE Transactions on Image Processing, vol. 20, (8), pp. 2392-2396, 2011.

[19] M. Hassan. "Drone-Face-Tracking." Internet: https://github.com/murtazahassan/Drone-Face-Tracking, Jul. 25, 2020[Mar. 17, 2023].

[20] Z. Cao et al. "openpose." Internet: https://github.com/CMU-Perceptual-Computing-Lab/openpose, Jul. 11, 2022[Apr. 2, 2023].

[21] Anonymous "Unmanned aircraft systems overview: outlook for the domestic drone industry," The Congressional Digest, vol. 95, (6), pp. 2, 2016.

[22] N. J. Goodall, "Can you program ethics into a self-driving car?" IEEE Spectrum, vol. 53, (6), pp. 28-58, 2016. . DOI: 10.1109/MSPEC.2016.7473149

[23] M. Maldini, "PID Control explained" Internet: https://maldus512.medium.com/pid-control-explained-45b671f10bc7#:~:text=PID%20control%20is%20a%20mathematical%20approach%20to%20a%20broad%20range,point%20to%20implement%20a%20solution, Dec 18, 2018 [Mar. 29, 2023]