Project-2 Report

Robotics-2 Aerial Robotics

ASU ID: 1222617175

Let's start with a basic understanding of the physics of quadcopter motion. The rotors on a quadcopter act like wings. They generate lift by rapidly rotating, drawing the air downward, and propelling the quadcopter into the air. The quad hovers in mid-air if the lift cancels out against gravity's acting force. A directed force is used to move the quadcopter in a specific direction. As the lift falls, so does the Drone's height.

Quadcopter Motor Setup: - The two neighboring motors rotate in opposing directions, while the two opposite motors rotate in the same way, resulting in a zero net force acting on the drone's body. When all of the motors rotate in the same direction, the torque generated causes the drone to rotate.

Hovering: The drone's hovering is fairly simple to understand. To begin, the motors generate a lift that is equal to the force of gravity acting on the body. As a result, both lift and gravity cancel out, causing the quadcopter to hover in mid-air.

Motion of the Drone: Drone movement is described using a nomenclature similar to that used to describe naval maneuver. I'll use the same vocabulary to describe the movements of our drone.

Roll: Moving side to side in relation to the front of the drone is performed by lowering the lift of the motors in the direction of movement.

To put it another way, if we need to roll to the left, we should lower the lift on the left side of the motors while increasing the lift on the right side. This causes the drone to move to the left, and vice versa makes a movement to the right side of the drone.

Pitch: Movements that force the drone to pitch forward (approach) you. The power delivered to the rear motors is increased. As a result of the forward net pressure, the Drone's nose tilts downward. You must also limit the power given to the two front motors to maintain angular momentum. The drone will move (pitch) rearward if you do the exact opposite.

Yaw: Turning the drone in a clockwise manner by rotating clockwise/anticlockwise around the center. The lift on anti-clockwise revolving motors must be raised, while the lift on clockwise rotating motors must be reduced. This is done to maintain the net forces of upward and downward motion at zero. As a result, there is an anti-clockwise torque. The drone spins clockwise to conserve Angular Momentum.

Take Off: To lift the drone off the ground, you'll require a net upward force. The motors generate a lift greater than the force of gravity, allowing the drone to take flight. For a safe drone landing, the opposite is true.

Code: https://github.com/dvenkatsai/aerialrobotics/tree/main/DroneControlUsingPython

video: https://www.youtube.com/watch?v=QKPY59LCKO0

LAB-1: Face Tracking Drone.

In this project, we'll use TELLO EDU to track human faces with the help of OpenCV and Python built-in modules.

The Viola-Jones algorithm is used to detect faces in this example; it is an object-recognition framework that permits the detection of picture features in real time. It's highly powerful, and its application in real-time face identification has proven to be particularly noteworthy. Viola-Jones is stronger at recognizing frontal faces than faces that are looking sideways, upwards, or downwards because it was designed for them. Before detecting a face, the image is converted to grayscale since it is easier to work with and there is fewer data to process. After detecting the face on the grayscale image, the Viola-Jones method determines the position on the colored image.

Viola-Jones draws a box (as seen on the right) and looks for a face within it. It's essentially seeking for haar-like traits, which will be explained later. The box travels a step to the right after cycling through all of the tiles in the shot. I chose a large box and large steps for example purposes, but you can change the box size and step size to fit your needs.

In smaller phases, a number of boxes identify face-like traits (Haar-like features), and the data from all of those boxes combined aids the computer in establishing where the face is. In our experiment, the entire approach was followed by three-steps:

1. Getting Frame Using Tello: Here, I've used built-in capabilities to get the real-time camera feed from the Drone and read the video frame by frame. Also, using OpenCV, resizing the image for speedier telecasting.

2. Detecting the Face in Live Video: As previously stated, the Viola-Jones algorithm is used to detect the face in live video. This technique catches frontal faces reliably and quickly. Haar Wavelets, a mathematical approach to identifying edges, lines, and four-rectangle features, are used to identify the frontal faces.

If the frame contains numerous images, we will focus on only the image closest to the camera, which will be updated using the face-detected rectangle's maximum area. The detected region of the face in the frame will have a higher area because the face is close to the camera.

3. Face Tracking: Face tracking is accomplished using a PID controller, which is used in a variety of industrial applications to regulate physical variables such as temperature, pressure, and speed. The PID controller uses a control loop feedback mechanism to control process variables, and it is the most accurate and stable controller available. The speed value will be adjusted using the PID equation while taking into account the faults, and the new speed values will be sent to the drone control to keep the yaw velocity for tracking the human face.

Link to code: https://github.com/dvenkatsai/aerial-robotics/tree/main/FaceTrackingDrone

Link to video: https://www.youtube.com/watch?v=828IB02d3U4

LAB-2: Object Tracking Drone.

In this lab, we are going to track objects using TELLO EDU with the help of OpenCV, Python

built-in libraries.

Object detection is carried out here via color detection, which is based on the fine-tuning of the Hue, Saturation, and Value of the object that needs to be tracked by the drone. These values are recorded using a custom script that converts the color image to black and white, and then uses the script to set the Hue, Saturation, and Value so that only the object is highlighted.

Then, to track the displacement changes of the item being tracked, we'll use a PID controller. The drone's yaw angle will be changed based on the displacement change, and the item will be tracked this way.

The full strategy was followed by three steps in our trial.

1. Getting Frame Using Tello: Here, I've used built-in capabilities to get the real-time camera feed from the Drone and read the video frame by frame. Also, using OpenCV, resizing the image for speedier telecasting.

2. Detecting the Object in Live Video: As previously said, the detection is done using the lean technique, which captures the object and uses color contours to identify the colored object.

3. Object Tracking: Object tracking is accomplished with a PID controller, which is used in a variety of industrial applications to regulate physical variables such as temperature, pressure, speed, and so on. The PID controller uses a control loop feedback mechanism to control process variables, and it is the most accurate and stable controller available.

The speed value will be adjusted using the PID equation by considering errors, and the new speed values will be sent to the drone control to keep the yaw velocity for tracking the human face.

Link to code: https://github.com/dvenkatsai/aerial-robotics/tree/main/ObjectTrackingDrone

Link to video: https://www.youtube.com/watch?v=Ug8WsEbNCQM

LAB-3: Controlling Drone with Body Postures.

We'll be using human body postures to operate the TELLO EDU drone. This will be accomplished by feeding the drone the body poses. The MediaPipe framework is used to estimate the body positions. Mediapipe is a framework for creating multimodal audio, video, or any type of time series data. The MediaPipe framework may be used to create an outstanding ML pipeline for inference models such as TensorFlow and TFLite, as well as media processing routines.

Pose a Problem of Estimation: - In applications like measuring physical activities, sign language recognition, and full-body gesture control, human position estimation from video is crucial. It can, for example, be used as the foundation for yoga, dance, and fitness applications. In augmented reality, it can also enable the overlay of digital content and information on top of the physical world. Using our BlazePose research, which also drives the ML Kit Pose Detection API, MediaPipe Pose is an ML solution for high-fidelity body pose tracking, inferring 33 3D landmarks on the full body from RGB video frames.

In this project, we will create a KNN classifier and train the model using our data set, and will test the classifier in real-time. In our experiment, the entire approach was followed by three-steps

1. Getting Frame Using Tello: Here, I've used built-in capabilities to get the real-time camera feed from the Drone and read the video frame by frame. Also, using OpenCV, resizing the image for speedier telecasting.

2. Detecting Poses in Live Video: As mentioned, earlier the detection is happened using the MediaPipe algorithm, this algorithm captures the body poses accurately and fast. In applications like measuring physical activities, sign language recognition, and full-body gesture control, human position estimation from the video is crucial. It can be used as the foundation for yoga, dance, and fitness applications, for example. In augmented reality, it can also enable the overlay of digital content and information on top of the physical world. Using our Blaze Pose research, which also drives the ML Kit Pose Detection API, MediaPipe Pose is an ML solution for high fidelity body pose tracking, inferring 33 3D landmarks on the full body from RGB video frames.

3. Control the Drone using poses: The drone's maneuver parameters are changed based on the gesture ID.

Link to code: https://github.com/dvenkatsai/aerial-robotics/tree/main/PoseControlDrone

Link to video: https://www.youtube.com/watch?v=sr_uC8BlPq0

LAB-4: Controlling Drone with Hand Gestures.

We'll be using human hand gestures to operate the TELLO EDU drone here. This will be accomplished by using the drone's hand motions as input. The MediaPipe framework is used to estimate the hand motions. Mediapipe is a framework for creating multimodal audio, video, or any type of time series data. The MediaPipe framework may be used to create an outstanding ML pipeline for inference models such as TensorFlow and TFLite, as well as media processing routines.

MediaPipe Hands for Gesture Recognition: Using MediaPipe Hands is a winning method in terms of both speed and flexibility. A rudimentary gesture recognition calculator is already included in MediaPipe, and it can be used in the pipeline. We required a more powerful solution that could update the recognizer's structure and behavior fast. A unique neural network with four Fully Connected layers and one Softmax layer for classification was constructed to accomplish this and to identify motions.

This simple structure takes a vector of 2D coordinates as input and outputs the gesture's ID. A basic neural network may readily handle such tasks instead of utilizing complex segmentation models with a more algorithmic recognition approach. It takes substantially less data and time to recognize gestures using key points, which is a simple vector containing 21 points' coordinates. What's more, because model retraining activities require substantially less time than the algorithmic technique, additional gestures can be simply incorporated.

The following motions will serve as the UAV's control conditions.

Link to code: https://github.com/dvenkatsai/aerial-robotics/tree/main/GuestureControlTello

Link to video: https://www.youtube.com/watch?v=LSFoWpoSpAk

LAB-5: YELLO.

We use the TELLO EDU drone to detect items using the well-known YOLO algorithm and then conduct a variety of actions based on the objects found. The YOLOv4 model, which is an extension of the YOLOv3 model, is used for object detection. It is built on a single Convolutional Neural Network (CNN) that divides the image into regions before predicting the border boxes and object probabilities for each region.

YOLO v4: YOLO v4 is well-known for its AP and FPS improvements. YOLOv4 prioritizes real-time object detection and training on a single CPU. On a Tesla V100, YOLOv4 achieved state-of-the-art performance on the COCO dataset, with 43.5 percent speed (AP) at 65 frames per second (FPS).

DropBlock Regularization, Data Augmentation, Mish-Activation, CrossStage-Partial-Connections (CSP), Self-adversarial-training (SAT), Weighted-Residual-Connections (WRC), and other elements all had a role in this success. The two types of models are one-staged object detectors and two-staged object detectors. Two-stage detectors operate in two stages: first, essential regions are determined, and then regions are categorized to determine whether the object is detected in that region.

Because it is a single staged object detector, YOLOv4 is more accurate and efficient than R-CNN, F RCNN.

Anchor boxes are used to hold a variety of objects in a single frame with the center in the same cell. The grid was utilized to detect a single object in a frame, on the other hand.

The length of the ground truth and prediction arrays will alter if multiple anchor boxes are changed. Consider a cell with 80 prediction classes, i.e. [Pc, P1, P2... P80, X1, Y1, X2, Y2], totaling 85. For these 9 anchor boxes, the array length will be 85*9 = 765 predictions. Anchor box with different scales plated around (0,0).

Link to code: https://github.com/dvenkatsai/aerial-robotics/tree/main/YELLO

Link to video: https://www.youtube.com/watch?v=_1vhkmMgE7U

LAB-6: Collision Avoidance Drone.

In this Lab experiment, we have considered human face for the detection, and the drone will be able to identify different items in the acquired footage in real-time, following the object and making sure the drone doesn't get too close to it. We maintain track of the area of the object, and if the area is bigger than the threshold area, we raise an alert and drive the drone backward, just like we did in the previous lab. We utilize a PID controller to determine the drone's speed and motions for this continuous track of objects.

1. Getting Frame Using Tello: Here, I've used built-in capabilities to get the real-time camera feed from the Drone and read the video frame by frame. Also, using OpenCV, resizing the image for speedier telecasting.

2. Detecting the Human Face in Live Video: The YOLO algorithm is used to detect the human face in live video.

If the frame contains numerous images, we will focus on only the image closest to the camera, which will be updated using the face-detected rectangle's maximum area.

The detected region of the face in the frame will have a higher area because the face is close to the camera.

3. Moving Away from the Face: Face tracking is accomplished using a PID controller, which is used in a variety of industrial applications to regulate physical variables such as temperature, pressure, and speed. The PID controller uses a control loop feedback mechanism to control the process variable, and it is the most accurate and stable controller available.

The speed value will be adjusted using the PID equation while taking into account the faults, and the new speed values will be sent to the drone control to keep the yaw velocity for tracking the human face. If the object's area is larger than the average, the PID controller slows down the drone's movement and pushes it back.

Note: The above-mentioned drone applications have been implemented using python built-in libraries which are djitellopy, cv2, NumPy, Haarcascade.

Link to code: https://github.com/dvenkatsai/aerial-robotics/blob/main/YELLO/collision_avoidance.py

Link to video: https://drive.google.com/file/d/1wZbHZgngrudO6G3PtYMV-VPYJezN1vXc/view?usp=sharing