A Technical Report

On

D.R.D.O. SASE's UAV Fleet Challenge

Indian Institute of Technology

Gandhinagar

Team Members:-

Ajay Kumar Ucheniya

Deepanshu Singh

Aditya Pandrangi

Devesh Kumar

Purna Kukadiya

Satish Singh

Shivani Singhal

Henil Shah

Ailneni Rakshita Rao

Pravesh Srivastava

Submitted by-

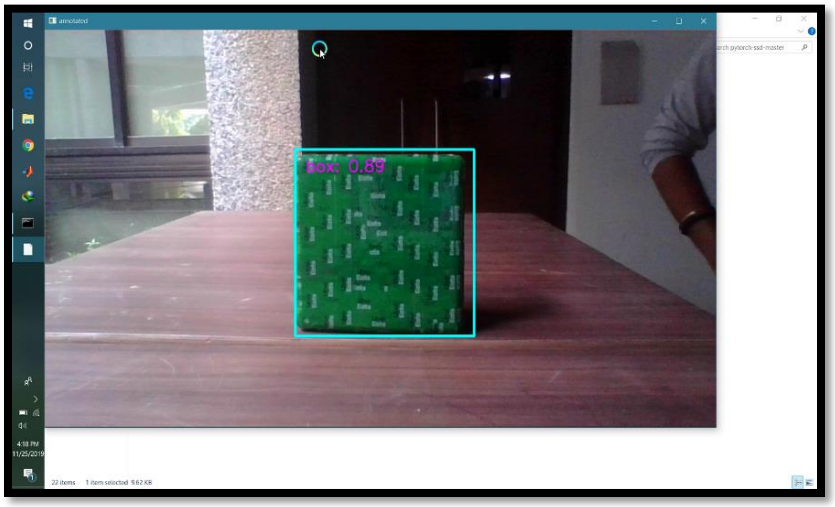
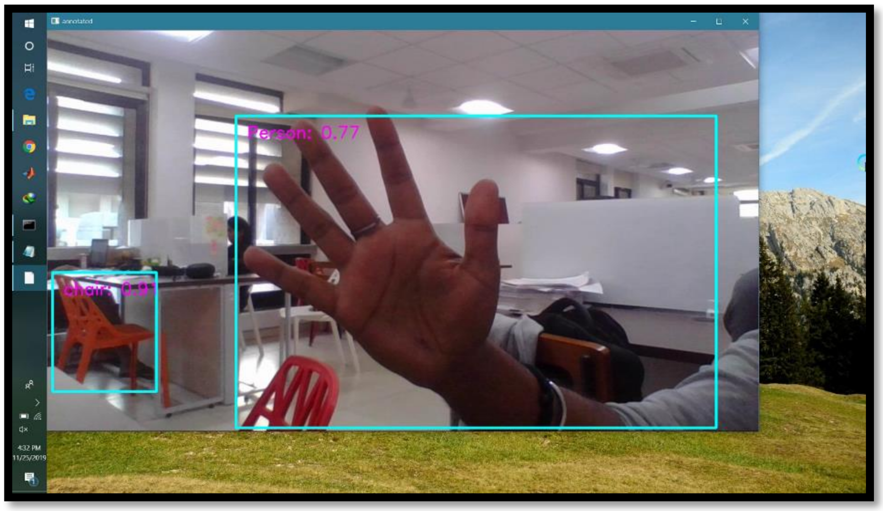
Team Name: - Dare2Dream

OBJECT DETECTION

The object detection is achieved by using deep learning approach. **YOLO** does the job efficiently. We trained the model on **ObjectNet3D: A Large-Scale Database for 3D Object Recognition.** To reduce onboard computing time we applied transfer learning by creating our own dataset, (By creating our own cuboid).

Work Done

We have trained YOLO object detection system based on our desired object class. In initial stage for confirmation of working of the trained model, we have implemented it on webcam for detection purpose to observe the characteristic in real-time situation. With limited amount of training set images results seems to be quiet promising proceed with the same.

Detecting the target: A sample box which was detected with probability score of 0.89.	
Detecting random objects: As can be seen It detects other 3d objects such as chair with probability score of 0.91 and human hand as a person with probability score of 0.77.	

Our Approach (Optimization and Implementation):

1. We will use NNPACK to optimize Darknet (open-source neural network framework), generally used for an embedded device using ARM CPU.
2. NNPACK implementation in Darknet can be used to improve transform-based convolution computation, allowing for up to 40%+ faster inference performance on non-initial frames.
3. Darknet-nnpack is available to be used on Raspberry Pi for the implementation of Yolo-versions.
4. Even with all the optimization, the full Yolo version is not possible to run on Raspberry pi. So, we will be using **Tiny-Yolo** which can be run at about 1 frame per second rate.

Possible Constraints and Our Solutions

- **Foggy condition in real terrain:** This might be a hurdle as it may lead to a blurry vision for our drone. However, we used **DEHAZE** to remove this hurdle.
- **Time:** Sometimes, the overall detection task may lead to high processing time. This delay is reduced by optimizing our algorithm through quantization of YOLO for implementing it on our microprocessor. Also we are sending target real coordinates from a distance which makes the detection task quite fast.
- **Motion Blur:** To deal with this, we will limit the speed of drones and have our own (IIT GANDHINAGAR) optimized algorithm to clear out the blur.

COMMUNICATION PROTOCOL

Obtaining coordinates (Phase 1 of communication protocol)

Region of localization of drones will be predefined by trisecting the given arena (40m×40m), where each drone will be assigned one-third of the trisected region, respectively. Proposed distribution model for the same using swarm technology is as follows:

Each drone will be mounted with a camera at a particular inclination with an accurate GPS. They will be allowed to free fly and detect as per their trisected region. Processing for object detection will be on board. Continuously accurate coordinates of the flying drone will be known to us. Utilizing the concept of triangular similarity, we can calculate the horizontal distance of the target from the drone camera. These coordinates can be later plotted on **Yahoo mapper.**

Communicating coordinates: GCS (Phase 2 of communication protocol)

Proposed Approach (Model: Triangle Similarity)

To determine the distance from our camera to a known object or marker, we are going to utilize triangle similarity. The triangle similarity goes something like this:
Let’s say we have a marker or object with a known width W. We then place this marker some distance D from our camera. We take a picture of our object using our camera and then measure the apparent width in pixels P. This allows us to derive the perceived focal length F of our camera:

$$\mathbf{F} = (\mathbf{P} * \mathbf{D}) / \mathbf{W}$$

As we continue to move the camera both closer and farther away from the object/marker, we can apply the triangle similarity to determine the distance of the object to the camera:

$$\mathbf{D'} = (\mathbf{W} * \mathbf{F}) / \mathbf{P}$$

- For communicating coordinates wirelessly among drones and ground control stations, we will be using Xbee device. All modules will be communicating wirelessly by configuring Xbee as the access point onboard Raspberry Pi processor (creating a likely RPI wireless hotspot) and using the same frequency.

Intra and Intercommunication (Phase 3 of communication protocol)

We are using Robotic Operating System for establishing a peer to peer network among drones. We will be working on PX4 firmware which is supported by Pixhawk hardware and provides MAVLink messaging protocol.

(Proposed Approach: Ground Control Station)

- Both on board computers and ground control stations will be having UNIX based operating system for installing our required packages such as ROSLIBJS.
- PX4 is controlled from the ground station computer via Xbee. The GCS computer will be communicating through MAVLink API such as the MAVSDK or MAVROS.

(Proposed Approach: Onboard Computer)

- After working on libraries an environment variable (bash setup) on each machine in the ROS network, will allow them to connect to the same instance of ROS master (granting the same capabilities as if the nodes were executing on the same computer).
- To send information between MAVLink systems we will consider using Commands, which are sent using command protocol.
- MAVROS will provide a mechanism for opening the OBC's serial port for sending and receiving MAVLink packets to and from the Flight Controller. On the OBC, this will be carried out through a Universal Asynchronous Receiver-Transmitter (UART) interface, and at the other end of the communication link, the connection will end at the Pixhawk's TELEM2 port.
- $echo$ROS MASTER_URI gives our computer's localhost IP addresses (Xbee). By setting up both the networks we will be creating a secure shell connection (SSH), which will allow remote login to the drone network from GCS.
- Thus, leading to a peer to peer design and communication among drones and GCS as well. Lastly, the processing of object detection as mentioned previously is done on board.

Hardware Components

1. Raspberry pi4
2. Pixhawk flight controller
3. XBee and radio-telemetry for communication.
4. GPS
5. BLDC Motors

Above stated hardware components are major requirement of our system.

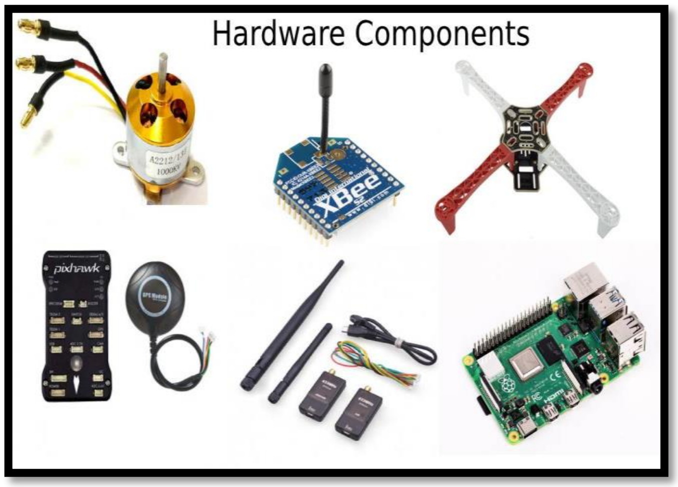


Figure 1: Proposed hardware required