



MANUAL SWARM DRONE FIGHT CONTROL AND OBJECT DETECTION USING YOLO

PROJECT REPORT

Submitted by :

AADARSH AADITYA A : CB.EN.U4AIE20001

B.E.PRANAV KUMAAR : CB.EN.U4AIE20052

of

2nd SEM B.Tech CSE-AI

For the Completion of

19AIE114 - PRINCIPLE OF MEASUREMENTS AND SENSORS

CSE - AI

AMRITA VISHWA VIDYAPEETHAM, ETTIMADAI

Submitted on:

13th July 2022

ACKNOWLEDGEMENT

Nammah Shivaya, firstly of all of us express our gratitude to Amma as finally we were able to finish our assignment that has been given by our Robotic Operating Systems teacher to us.

Next, we would like to express our special thanks to our project guide and teacher Mr.Sajith Variyar V.V who gave us the golden opportunity to do this wonderful project on the topic “Manual Swarm Drone Fight Control And Object Detection Using Yolo”, which in turn helped us in doing a lot of research in fields we were less familiar with, via which we learned so many new things. We are really thankful to them.

Lastly, we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame and keeping us motivated throughout the process.

ABSTRACT

Drones are Unmanned Aerial Vehicles(UAV) that are remotely controlled either by humans or by computer programs. They range in size from under one pound to several hundred pounds. Drones were initially conceptualized and developed with militant motivation to reduce human participation. Primitive versions of Drones were in use since the Civil War (1861-1865). The U.S. military began using modern drones in 1995, the first used drones were deployed with intelligence gathering applications an weaponized drones in strike tasks.

Later Drones started finding more peaceful and productive applications such as civilian roles ranging from search and rescue, surveillance, traffic monitoring, farmers use them to check on fields and crops, firefighters fly drones over forests to check for wildfires, filmmakers may use them to record scenes in movies, scientists may fly drones into big storms to gather information such as temperature and wind speed. Currently, companies like Amazon and Walmart are interested in using drones to deliver goods to people's homes. Today there exists a combination of both commercial and personal drones. With these aspects in mind we look to explore a simulation of 'Manual Swarm Drone Fight Control And Object Detection Using Yolo'.

1. INTRODUCTION

1.1 Drones

A drone, in a technological context, is an unmanned aircraft. Drones are more formally known as unmanned aerial vehicles (UAVs) or unmanned aircraft systems (UASes). Essentially, a drone is a flying robot.

The aircraft may be remotely controlled or can fly autonomously through software-controlled flight plans in their embedded systems working in conjunction with onboard sensors and GPS.

Different drones are capable of traveling varying heights and distances. Very close-range drones usually have the ability to travel up to three miles and are mostly used by hobbyists. Close-range UAVs have a range of around 30 miles. Short-range drones travel up to 90 miles and are used primarily for espionage and intelligence gathering. Mid-range UAVs have a 400-mile distance range and could be used for intelligence gathering, scientific studies and meteorological research. The longest-range drones are called “endurance” UAVs and have the ability to go beyond the 400-mile range and up to 3,000 feet in the air.

1.2 Types of Drones:

1. **Single Rotor** - Single rotor helicopters look exactly like tiny helicopters and can be gas or electric-powered. The single blade and ability to run on gas help its stability and fly for longer distances. These UAVs are usually used to transport heavier objects, including LIDAR systems.
2. **Multi-Rotor Drones** - Multi-rotor drones are usually some of the smallest and lightest drones on the market. They have limited distance, speed and height, but make the perfect flying vehicle prototypes to understand flight concepts. These drones can usually spend 20-30 minutes in the air carrying a lightweight payload, such as a camera.
3. **Fixed Wing Drones** - Fixed-wing drones look like normal airplanes, where the wings provide the lift instead of rotors- making them very efficient. These drones usually use fuel instead of electricity, allowing them to glide in the air for more than 16 hours. These are applied in durability priority tasks.

We are going to be simulating quadcopters which falls under the Multi-Rotor Drone Category.

2. QUADCOPTER FLIGHT DYNAMICS

2.1 Theory

A quadcopter is a drone with 4 propellers. When formulating the dynamics we assume that the propellers are symmetrical positioned and oriented about the center of the drone and the center of mass of the drone lies within the drones body. Before exploring the motion of the drone we should understand the Forces that act on the drone,

Thrust – The force acting on an object perpendicular to the surface. It is a vector quantity, unit – N

Lift – The force that directly opposes the weight of the aircraft. It is a vector quantity, unit – N

Drag – The force acting opposite to the relative motion of any object moving. It is a vector quantity, unit – N

Dimension of Newton - $M^1 L^1 T^{-2}$

Torque – The twisting force that tends to cause rotation. It is a vector quantity, unit – Nm

Dimension of Newton meter - $M L^2 T^{-2}$

Torque is given by the formula,

$$\tau = rF \sin \theta$$

Here,

r is the radius

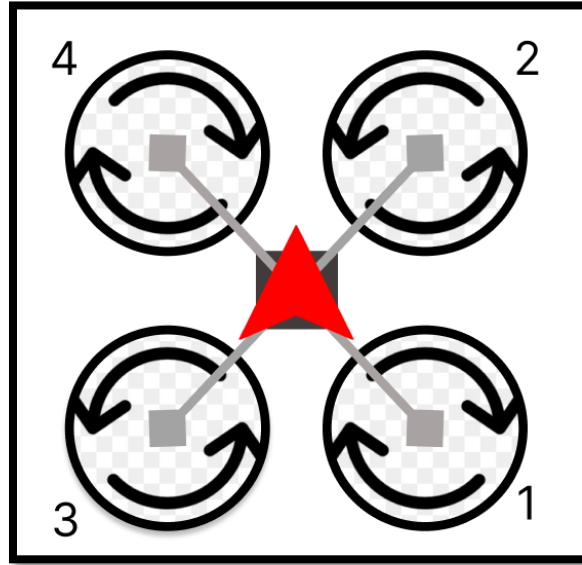
F is the force

θ is the angle between F and lever arm

Most Multi-Rotor drones use even number of propellers, this is an easy way to balance the net Torque created by the propellers. To explain, Newtons 3rd Law states that for every action, there is an equal and opposite reaction. In accordance to Newtons 3rd Law the propeller rotation generates a torque opposite of its own onto the body of the drone, if this torque is not dealt with it will lead to instability. An easy way to deal with this torque is to use an additional propeller that rotates in the opposite direction equivalent to the first one, similarly this imbues an equivalent torque in the drone that cancels the prior propeller's caused torque. Hence, the pair of propeller's torque balance out and the net torque on the drones body is zero.

Consider the following Fig 1, let the propellers of our quadcopter be labeled as shown,

It is important to note that the propellers are designed to push air in the same direction (downwards) regardless of the direction of rotation of the propeller. For our equations we are going to neglect drag.



The individual thrusts generated by our drones will be obtained using,

$$T_i = \rho A v_i^2$$

Here,

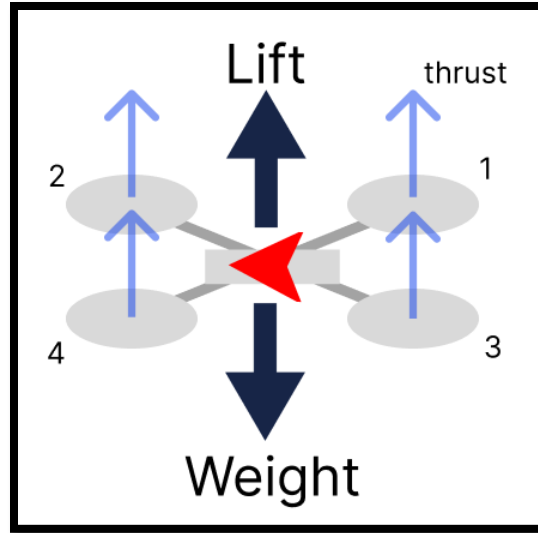
‘i’ refers to the ith propeller

A is the cross-sectional area of the propeller

ρ is the density of air

v is the velocity

The net Lift of the drone can be found by the summation of individual thrusts of each of the propellers,



$$L = T_{net} = \rho A \sum_{i=1}^4 v_i^2$$

The Weight is given by,

$$W = mg$$

2.2 Position and Orientation Transformations

The following motions 1 and 2 require only a change in position but not in orientation. Hence the propellers are expected to run at the same velocity.

1. Hovering

Hovering is the configuration of the drone in which it remains at the same point without any change in position or orientation. When the weight and the Lift are equal the drones vertical position remains the same.

$$\text{upward force} = \text{downward force}$$

$$L = W$$

2. Ascent and Descent

During Ascent the drone's altitude/height with respect to ground in consideration is expected to increase, this can be achieved when,

$$\text{upward force} > \text{downward force}$$

$$L > W$$

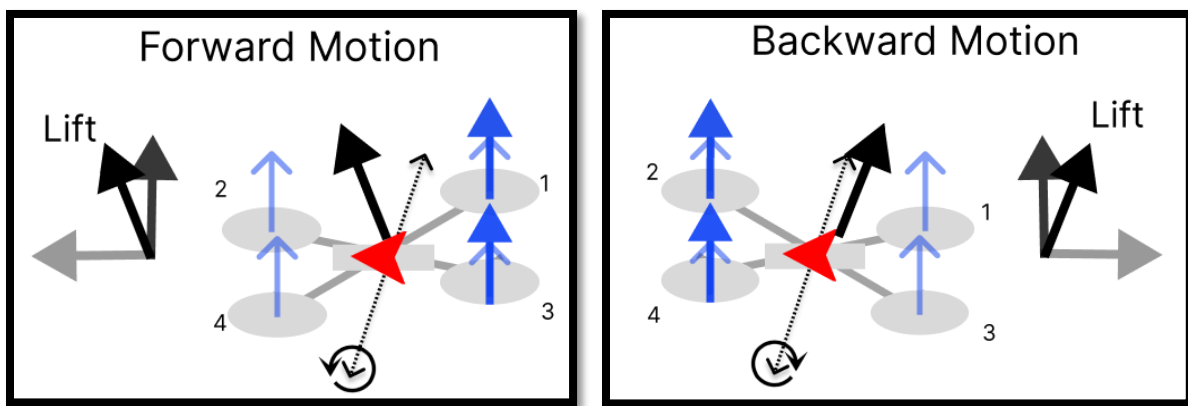
During Descent the drone's altitude/height with respect to ground in consideration is expected to decrease, this can be achieved when,

$$\text{upward force} < \text{downward force}$$

$$L < W$$

3. Forward and Backward Motion

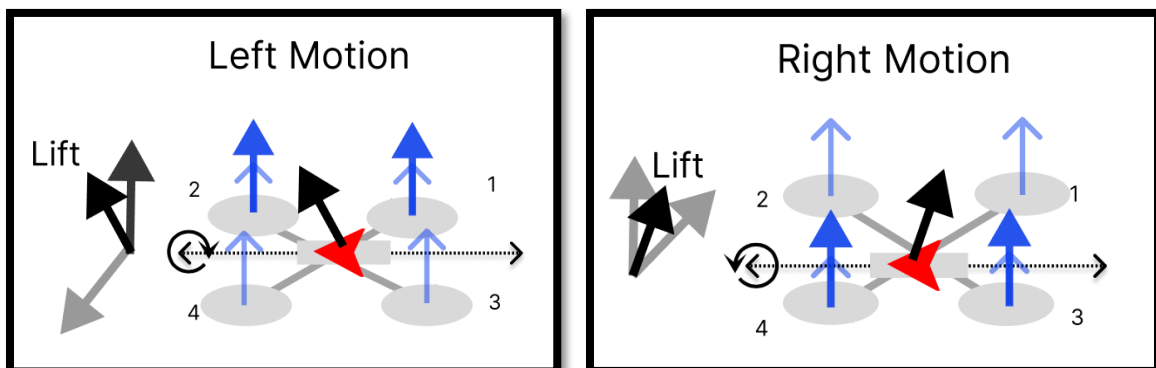
The forward and backward motions are impossible to achieve in the quadcopters because there are not actuators that control force about the axis. To creatively overcome this issue drones, manipulate their pitch in order to direct a portion/component of their Lift to act in the forward or backward direction.



For our drone, Forward motion can be achieved by increasing the thrusts of the 1st and 3rd propellers and Backward motion can be achieved by increasing the thrusts of the 2nd and 4th propellers.

4. Left and Right Motion

Similar to Forward and Backward the Left and Right motions are possible via manipulation roll.



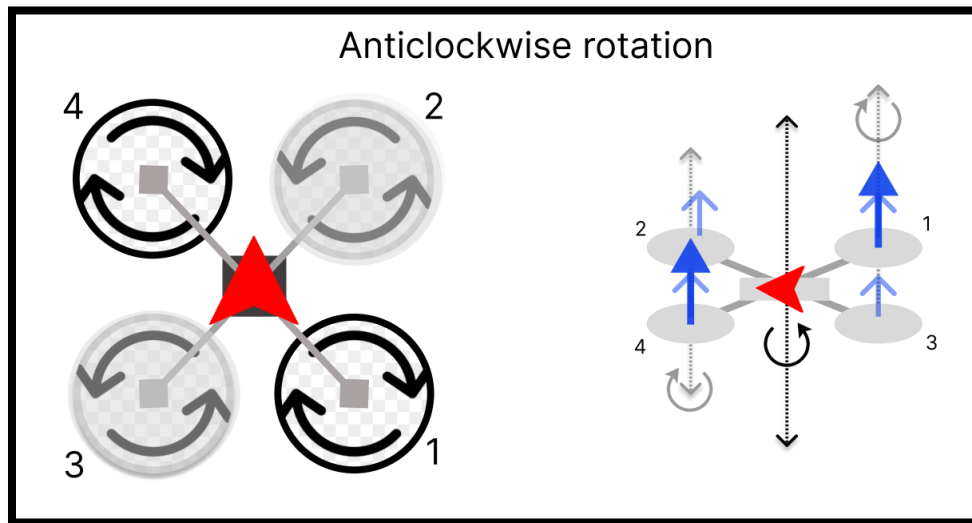
For our drone, left motion can be achieved by increasing the thrusts of the 1st and 2nd propellers and Right motion can be achieved by increasing the thrusts of the 3rd and 4th propellers.

5. Left (Anticlockwise) and Right (Clockwise) Rotation

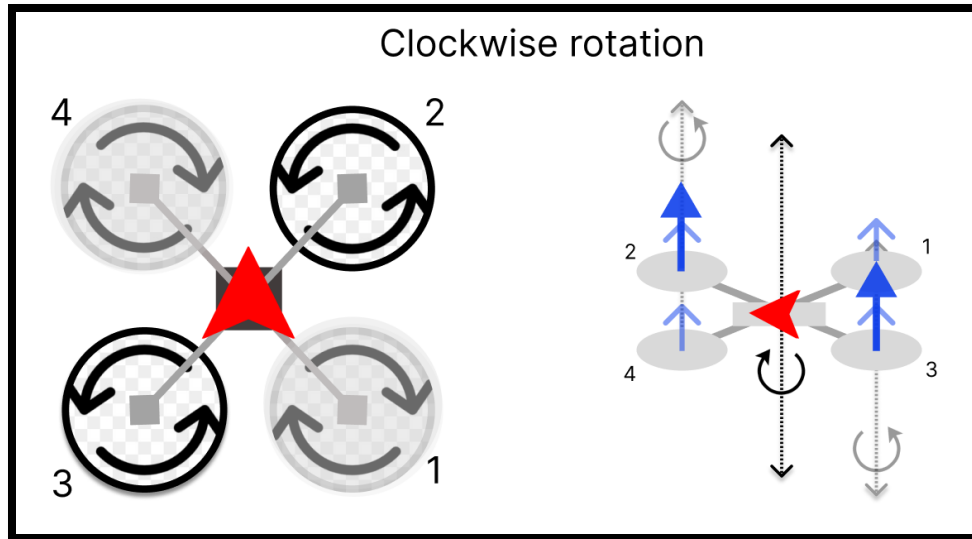
The controlled torque imbalance will allow us to rotate the drone in both clockwise and anticlockwise directions.

By convention anticlockwise torque is positive

When the thrust/force generated by clockwise rotating propellers is increased the torque generated by them in turn is increased. Whereas the thrust generated by the anticlockwise rotating propellers remains the same. Naturally there will be a net negative torque and equivalent positive torque in the drone's body. This torque allows the drone to rotate left,



The same can be inferred for clockwise rotation, allowing the drone to rotate right,



For our drone, Anticlockwise rotation can be achieved by increasing the thrusts of the 1st and 4th propellers and Clockwise rotation can be achieved by increasing the thrusts of the 2nd and 3rd propellers.

3. IMPLEMENTAION

3.1 Tools Used

ROS - Robot Operating System is an open-source robotics middleware suite. Although ROS is not an operating system but a set of software frameworks for robot software development

Mavros - Package provides communication driver for various autopilots with MAVLink communication protocol. Additional it provides UDP MAVLink bridge for ground control stations.

Ardupilot - ArduPilot is an open source, unmanned vehicle Autopilot Software Suite, capable of controlling autonomous vehicles such as Multirotor drones Fixed-wing and VTOL aircraft.

Gazebo – Gazebo is a toolbox of development libraries and cloud services to make simulation easy. It allows us to Iterate fast on your new physical designs in realistic environments with high fidelity sensors streams.

We will be using the above tools to allow us to implement a simulation of,

3.2 Drone Flight

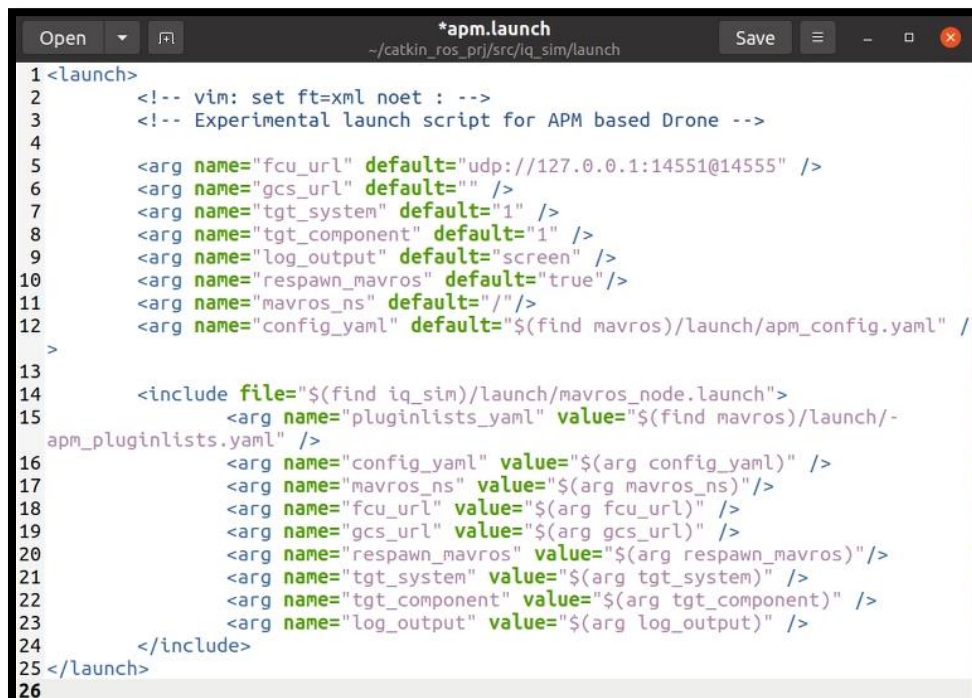
3.3 Object Recognition

3.4 Manual Swarm Drone

Ardupilot and ROS

Ardupilot is a generic autopilot suite used for various practical applications. In this project, ardupilot will be used alongside ROS. Typically, onboard data is fed into ardupilot through serial communication. However, we will simulate this using UDP/TCP packets. ROS and Ardupilot will communicate through a common udp/tcp socket.

Ardupilot uses MAVLink Protocol. Hence, we need two packages, MAVLink and MAVROS to facilitate communication between ROS and Ardupilot. MAVROS acts as a middleman to talk between ROS and Ardupilot. Once the packages for ROS (MAVLink and MAVROS) are installed, communication can be configured in the apm.launch file.

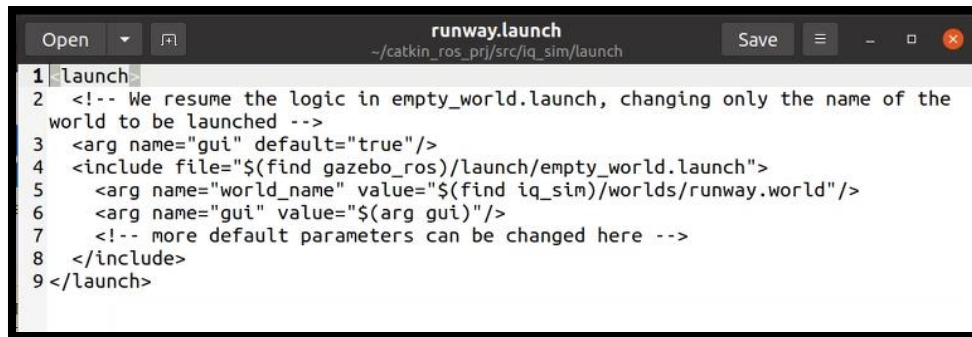


```
1 <launch>
2   <!-- vim: set ft=xml noet : -->
3   <!-- Experimental launch script for APM based Drone -->
4
5   <arg name="fcu_url" default="udp://127.0.0.1:14551@14555" />
6   <arg name="gcs_url" default="" />
7   <arg name="tgt_system" default="1" />
8   <arg name="tgt_component" default="1" />
9   <arg name="log_output" default="screen" />
10  <arg name="respawn_mavros" default="true"/>
11  <arg name="mavros_ns" default="/" />
12  <arg name="config_yaml" default="$(find mavros)/launch/apm_config.yaml" />
13
14  <include file="$(find iq_sim)/launch/mavros_node.launch">
15    <arg name="pluginlists_yaml" value="$(find mavros)/launch/-
16    apm_pluginlists.yaml" />
17    <arg name="config_yaml" value="$(arg config_yaml)" />
18    <arg name="mavros_ns" value="$(arg mavros_ns)" />
19    <arg name="fcu_url" value="$(arg fcu_url)" />
20    <arg name="gcs_url" value="$(arg gcs_url)" />
21    <arg name="respawn_mavros" value="$(arg respawn_mavros)" />
22    <arg name="tgt_system" value="$(arg tgt_system)" />
23    <arg name="tgt_component" value="$(arg tgt_component)" />
24    <arg name="log_output" value="$(arg log_output)" />
25  </include>
26 </launch>
```

Further, parameters can be set in mavros/launch/apm_config.yaml file.

3.2 Drone Flight

With the availability of drone models, we launch runway.world and with the drone model preloaded in the world.

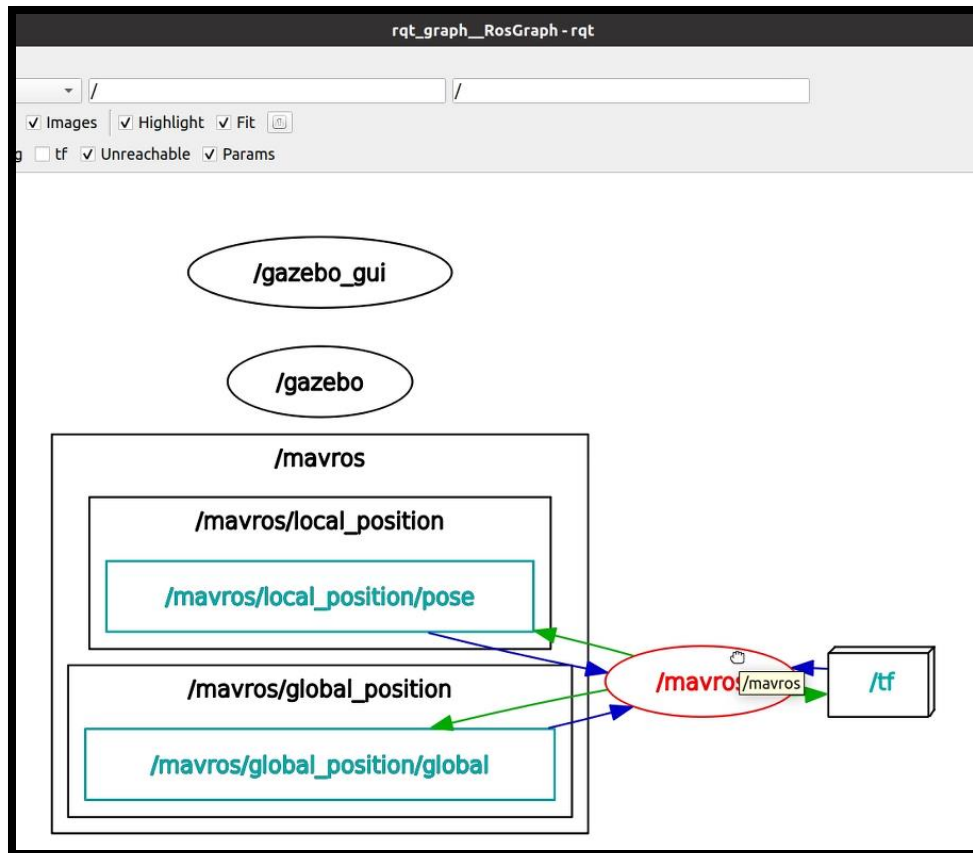


```
1 launch
2 <!-- We resume the logic in empty_world.launch, changing only the name of the
3 world to be launched -->
4 <arg name="gui" default="true"/>
5 <include file="$(find gazebo_ros)/launch/empty_world.launch">
6   <arg name="world_name" value="$(find iq_sim)/worlds/runway.world"/>
7   <arg name="gui" value="$(arg gui)"/>
8   <!-- more default parameters can be changed here -->
9 </include>
10 </launch>
```

For the drone model , a copy of the 3DR Iris model taken from https://github.com/PX4/sitl_gazebo/tree/master/models. The model is defined in /models/drone1/model.sdf along with the supporting files in /model/drone1/. In the SDF file, notice that the plugin for ardupilot is installed. This allows ardupilot to directly control the motors through udp/tcp prots itself. The choice of in and out ports can be specified in model.sdf file.

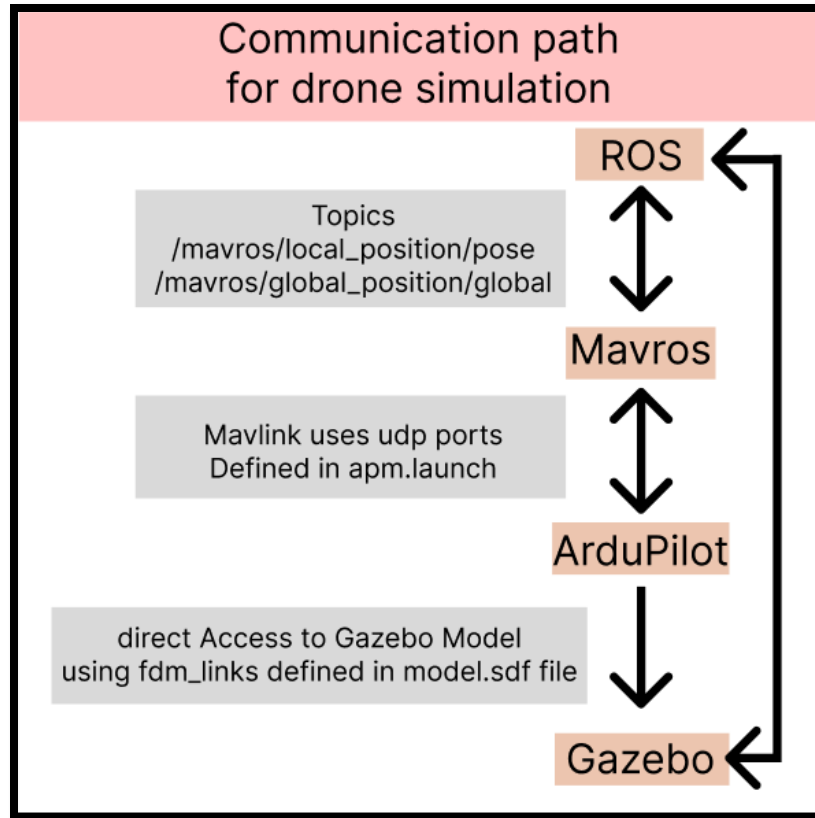


```
138 <link_name>iris::rotor_3</link_name>
139 </plugin>
140 <plugin name="arducopter_plugin" filename="libArduPilotPlugin.so">
141   <fdm_addr>127.0.0.1</fdm_addr>
142   <fdm_port_in>9002</fdm_port_in>
143   <fdm_port_out>9003</fdm_port_out>
```



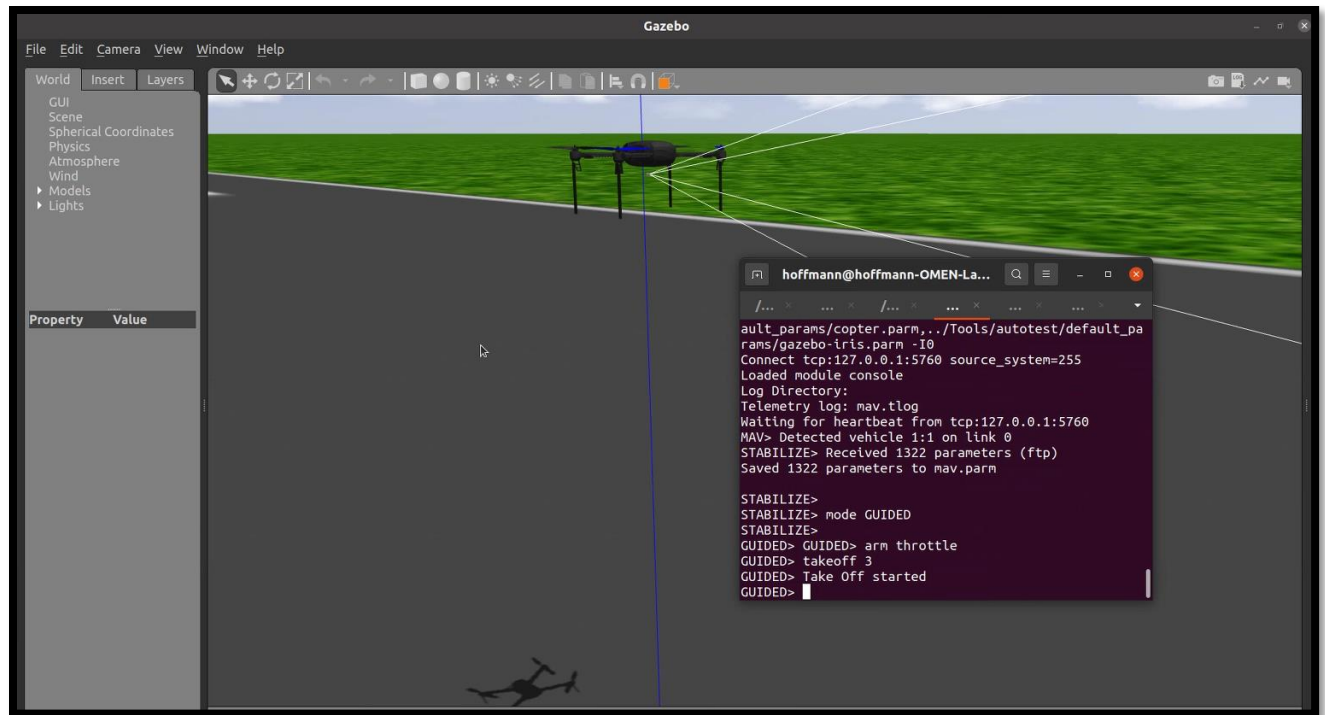
Now, we can directly control the drone using MAVLINK commands. These commands are typically high-level commands like takeoff, return to land etc. Ardupilot translates these commands and acts as a control system to instruct the drone to act appropriately. These instructions are given through udp sockets and are hence not visible in topics.

In essence,



Controlling a Drone

With ardupilot up and running, set the mode to guided, and arm throttle and takeoff, you should see the drone flying in gazebo.



Further, the drone's position, yaw etc. can be set using MAVLINK commands.

3.2 Object Recognition Using Yolov3

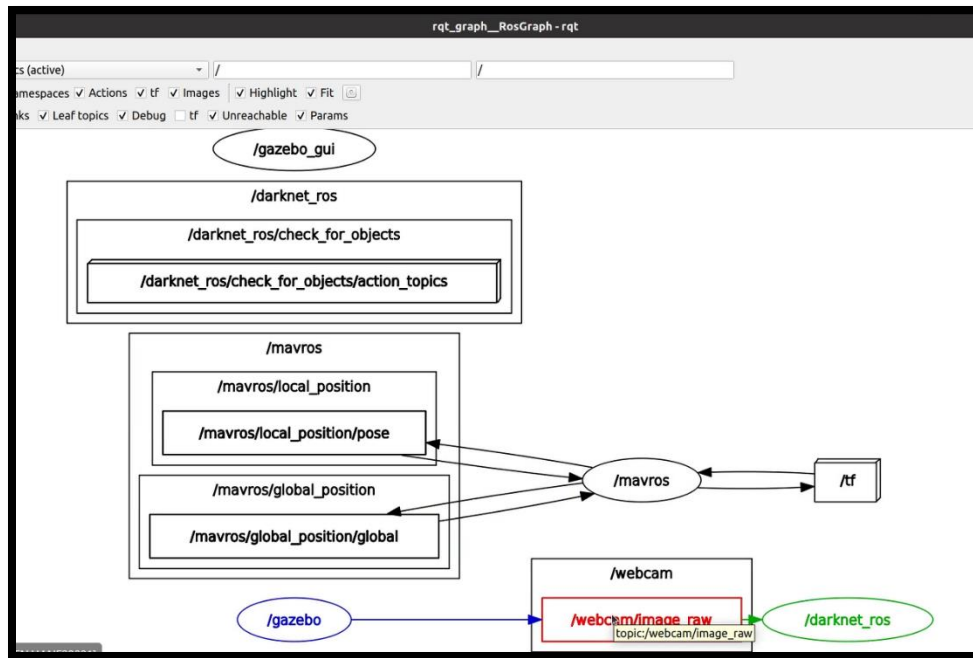
YOLO is the abbreviation for 'You Only Look Once', The YOLO model was first described by Joseph Redmon, et al. in the 2015 paper titled "You Only Look Once: Unified, Real-Time Object Detection." The approach involves a single neural network trained end to end that takes a photograph as input and predicts bounding boxes and class labels for each bounding box directly. The technique offers lower predictive accuracy (e.g. more localization errors), although it operates at 45 frames per second.

Further improvements to the model were proposed by Joseph Redmon and Ali Farhadi in their 2018 paper titled "YOLOv3: An Incremental Improvement."

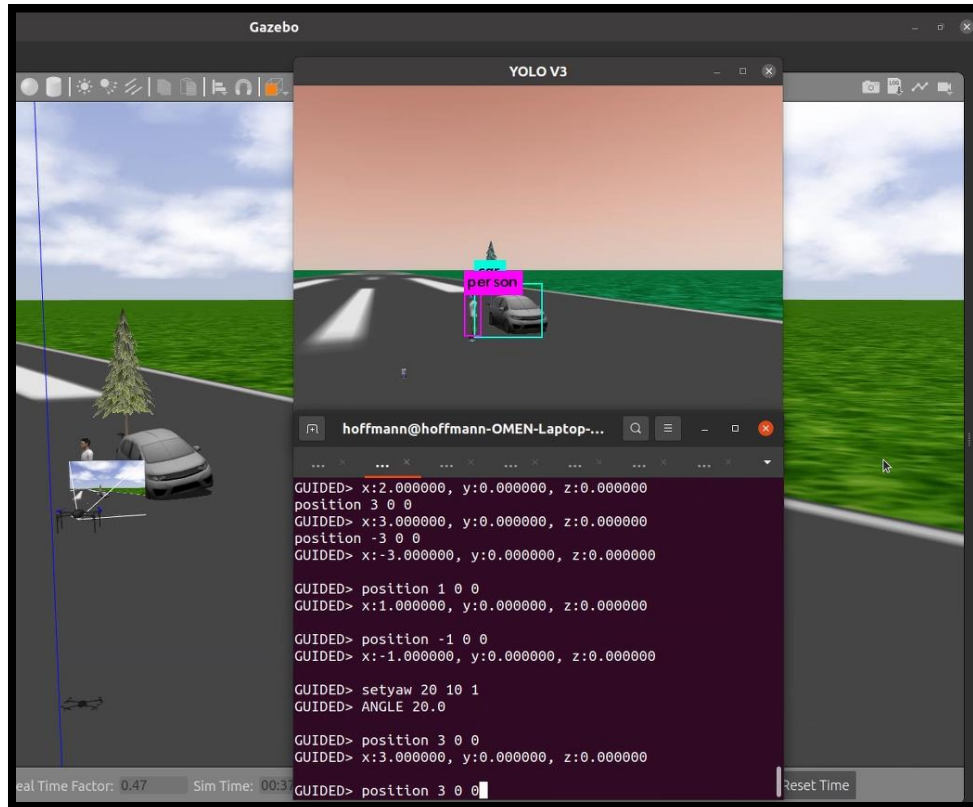
We use the DARKNET ROS package, which comes with a pretrained yolo net. It reads data from a specified topic in the config.yaml file. It reads every frame and published bounding boxes, prediction, etc as topics which will be available across all other nodes. To start the darknet, after cloning run

Roslaunch darknet daknet.launch

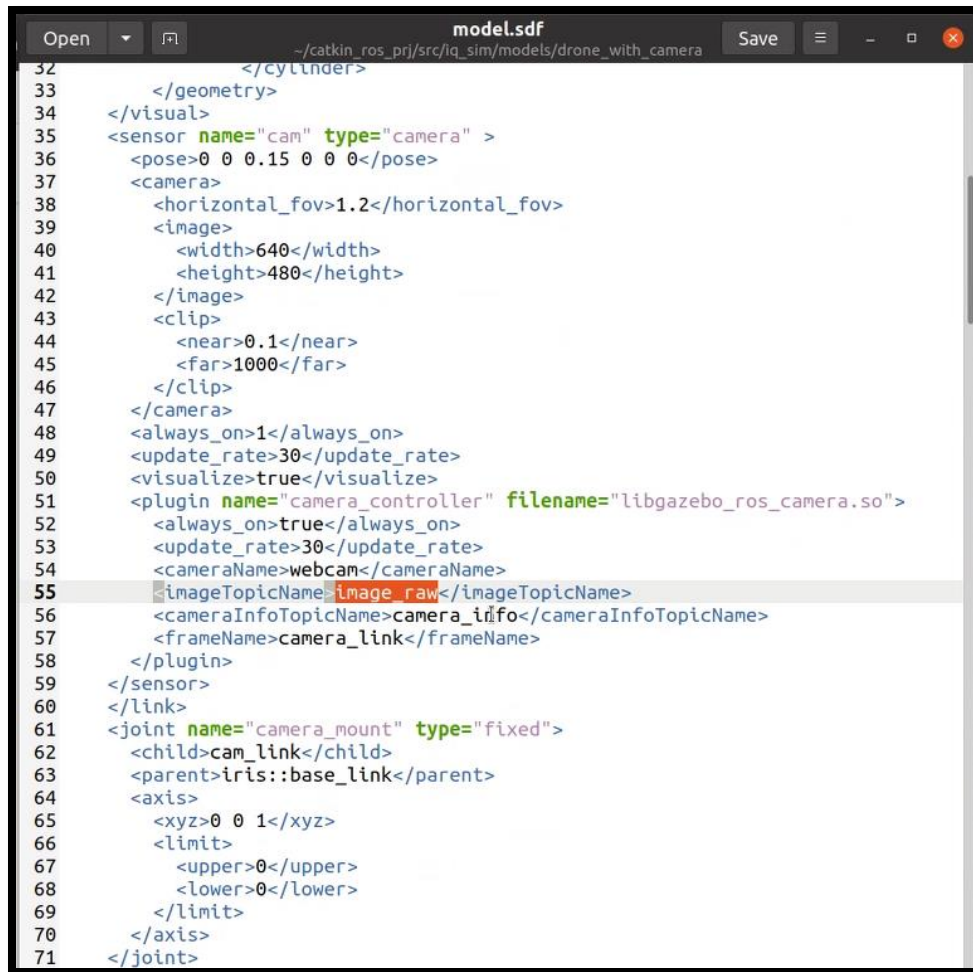
This should start up a xterm window, with camera feed of the drone. To set the topic too which darknet should subscribe, change the value in config.yaml to “/webcam/img_raw”, which is the feed coming from the drone.



Here is a screenshot of the same.



The camera on the drone is mounted on to the base link and is configured to publish frames to the topic “image_raw” withing the models/drone_with_camera .

A screenshot of a code editor window titled 'model.sdf'. The window shows XML code for a drone model. The code includes a cylinder geometry, a camera sensor, and a camera mount joint. The camera sensor is named 'cam' and is of type 'camera'. It has a horizontal field of view of 1.2, a width of 640, and a height of 480. The camera mount joint is named 'camera_mount' and is of type 'fixed'. It is attached to the 'iris::base_link' and has a child link named 'cam_link'. The code is as follows:

```
32     </cylinder>
33   </geometry>
34 </visual>
35 <sensor name="cam" type="camera" >
36   <pose>0 0 0.15 0 0 0</pose>
37   <camera>
38     <horizontal_fov>1.2</horizontal_fov>
39     <image>
40       <width>640</width>
41       <height>480</height>
42     </image>
43     <clip>
44       <near>0.1</near>
45       <far>1000</far>
46     </clip>
47   </camera>
48   <always_on>1</always_on>
49   <update_rate>30</update_rate>
50   <visualize>true</visualize>
51   <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
52     <always_on>true</always_on>
53     <update_rate>30</update_rate>
54     <cameraName>webcam</cameraName>
55     <imageTopicName>image_raw</imageTopicName>
56     <cameraInfoTopicName>camera_info</cameraInfoTopicName>
57     <frameName>camera_link</frameName>
58   </plugin>
59 </sensor>
60 </link>
61 <joint name="camera_mount" type="fixed">
62   <child>cam_link</child>
63   <parent>iris::base_link</parent>
64   <axis>
65     <xyz>0 0 1</xyz>
66     <limit>
67       <upper>0</upper>
68       <lower>0</lower>
69     </limit>
70   </axis>
71 </joint>
```

Conclusion

A working simulation of drone using ardupilot is done in this project. Further, we would like to extend this to a implementable, cheap drone in the near future.

