A Simulation framework for a Multi-Purpose Autonomous Drone Navigation

Devender Soni
Dept. of Electronic Science
Kurukshetra University,
Kurukshetra, India

Abstract

In inventory applications, drones play a major role to detect the types of objects by using the various applications of computer vision technology. In this paper, we have simulated the smart autonomous drone for decoding of QR Codes along with object detection. Moreover, this drone can detect the obstacle and avoid it when doing a preplanned mission. Gazebo and ROS are the major open-source tools that helps to demonstrate the simulation. First, we will setup the necessary environment by integration and communication between all the individual tools. To make the drone collision free, avoidance system will be initiated, then we will publish the image topic that will be subscribed by OpenCV and pyzbar libraries to decode the QR Code. Finally, we will setup the YOLO darknet for the object detection.

Keywords: Drone, ROS, Gazebo, YOLO, Object detection, QR Code

Introduction

The drones, also known as quadrotor, refers to the unmanned aerial vehicle (UAV) are meant to carry out the different types of tasks in different work field such as industry, defense, medical etc. Drones ranges in different sizes and weight from some grams to some kilograms depending upon the tasks for they are being used. The smaller the size, more the convenient to handle. To make drones more advance and intelligent, are integrating with multiple sensors. Different drones can travel varying heights and distances. Very close-range drones usually can travel up to three miles, close-range 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 can go beyond the 400-mile range and up to 3,000 feet in the air.

In this paper, we are going to develop a multi-purpose autonomous drone that will be suitable for an industrial purpose. The whole process will be simulated with the help of free & open-source tools Gazebo and ROS. The software tools can implement the algorithms and concepts that are required in a real environment.

Methodology

ROS

The Robot Operating System (ROS) is a set of software libraries and tools that helps to build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. It is an open-source, meta-operating system for the robot. It provides the services we would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS implements several different styles of communication, including synchronous RPC-style communication over services asynchronous streaming of data over topics and storage of data on parameter server. ROS is not a realtime framework, though it is possible to integrate ROS with realtime code. ROS comes with different distributions such as Indigo, Kinetic, Noetic depending upon the different versions of operating system (OS). For e.g., Ubuntu 20.04 supports ROS Noetic and for this project same combination have been used.

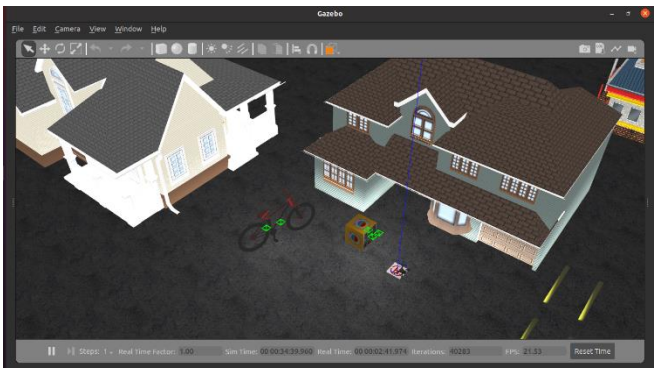
ROS supports multiple tools like RVIZ and RQT which will be used in our simulation. RVIZ is a 3d visualization tool for ROS applications. It provides a view of robot model, capture sensor information from robot sensors, and replay

captured data. It can display data from camera, lasers, from 3D and 2D devices including pictures and point clouds.

RQT is a software framework of ROS that implements the various GUI tools in the form of plugins. The tools can still run in a traditional standalone method, but rqt makes it easier to manage all the various windows on the screen at one moment.

Gazebo

Before testing the robotics operation on the real hardware its required to perform the simulation as it reduces the cost, risks, harms that can be caused by some fault in the robotic systems.



Gazebo Simulator with Iris Drone model placed on helipad

For the simulation of our drone, we are going to use the open-source 3D robotics simulator Gazebo. A well-designed simulator makes it possible to rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios. Gazebo offers the ability to simulate populations of robots accurately and efficiently in complex indoor and outdoor environments. It provides the various features such as sensors, plugins, prebuilt robotics models and it can access on cloud server such as Amazon AWS. We will use our own QR Code Box and Cycle models, created with the help of BLENDER – 3D modelling open-source tool, in the prebuilt world from Gazebo worlds repository.

PX4-Autopilot

If we look at the anatomy of an autonomous drone we will see camera, electronic speed controller (ESC), GPS module to pinpoint the Drone location, avoidance system to detect and avoid the obstacle, drone flight controller and other components.

Out of these, flight controller is the brain of the drone. The flight controller takes in inputs from the GPS module, compass, obstacle avoidance sensors, and the remote controller and processes it into information that is given out to the ESCs to control the motors. An example of this is seen when a drone is hovering during windy conditions. If wind is blowing drone will stay in its exact place this is because the flight controller sends the proper instructions to the ESCs and intern the motors to compensate for the wind factor.

For the simulation, we are going to use the PX4 as flight control software. PX4 is an open-source flight control software for drones and other unmanned vehicles. It provides a flexible set of tools for drone developers to share technologies to create tailored solutions for drone applications. PX4 provides a standard to deliver drone hardware support and software stack, allowing an ecosystem to build and maintain hardware and software in a scalable way. PX4 offers optimised APIs and SDKs for developers working with integrations.

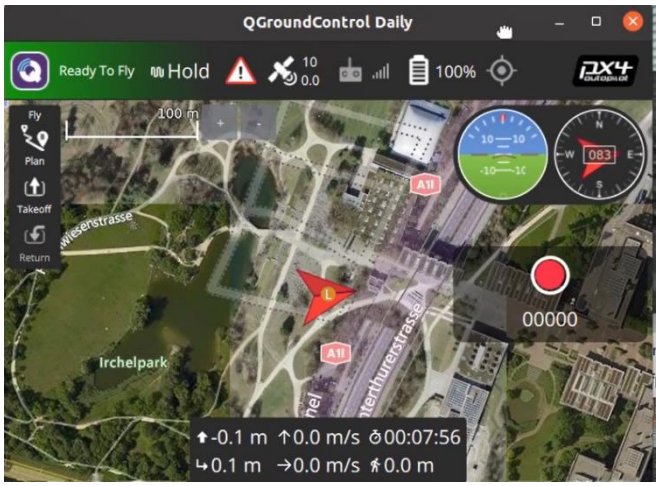
All the modules are self-contained and can be easily exchanged against a different module without modifying the core. PX4 supports both Software In The Loop (SITL) simulation, where the flight stack runs on computer (either the same computer or another computer on the same network) and Hardware In the Loop (HITL) simulation using a simulation firmware on a real flight controller board.

Obstacle Avoidance

To navigate a drone with obstacle avoidance system when following a predefined path, it must have sensors for that purpose. In simulation we use the PX4-Avoidance based Global path planner implementation with ROS which uses PX4 computer vision algorithms packaged as ROS nodes for depth sensor fusion and obstacle avoidance. Global Planner is a global, graph-based planner that plans in a traditional octomap occupancy grid. It builds a map of the environment and for the map to be good enough for navigation, accurate global position and heading are required.

GCS

To control the UAVs or drones from ground or sea, generally ground control stations (GCS) are required that provides the facilities for human control. QGroundControl is a software tool that provides full flight control and vehicle setup with video streaming for PX4 powered vehicles. It is mission planner for the UAVs or drones that

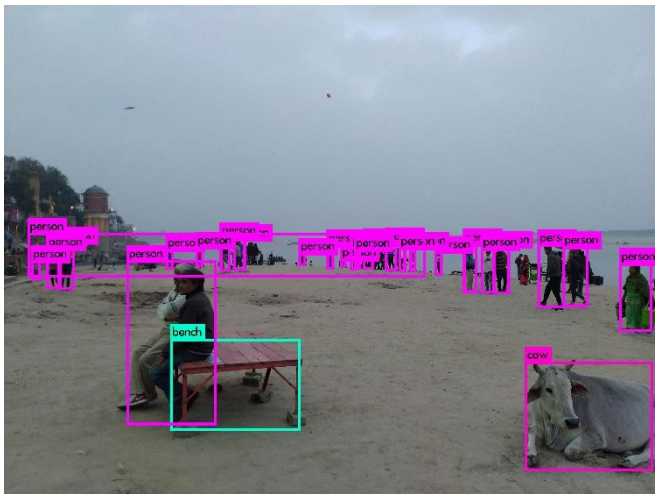


QGroundControl GCS

provides interface for mission planning just by adding some waypoints. The waypoints can be then converted into the geolocation points.

Object Detection

Object detection is a computer vision technique that allows to identify and locate objects in an image or video. With this kind of identification and localization, object detection can be used to count objects in a scene and determine and track their precise locations, all while accurately labeling them. Bounding boxes are created around these detected objects, which allows to locate where said objects are in each scene.



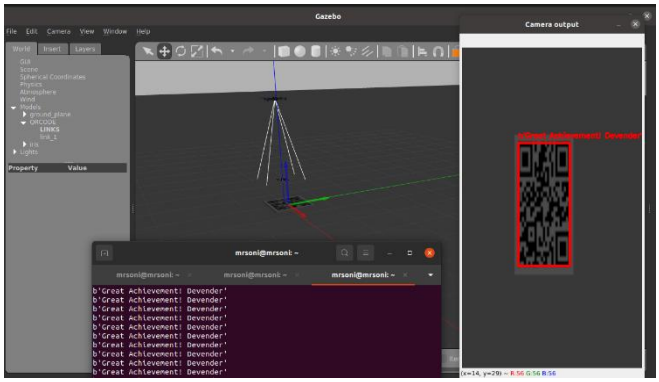
Sample Image Detection using Darknet

Implementation of the real time object detection in our drone simulation by using YOLO (You Only Look Once) that works on trained weights and configuration files which includes the set of classes of objects to be detected. The pretrained model of the convolutional neural network can detect pre-trained classes including the dataset from VOC and COCO, or we can also train a network with our own detection objects.

YOLO uses the DARKNET as the detector, an open-source neural network framework written in C and CUDA. Darknet prints out the objects it detected, its confidence, and how long it took to find them. We didn't compile Darknet with OpenCV so it can't display the detections directly. Instead, it saves them in predictions.png. We are using the object detection with ROS so we will use the ROS package developed for YOLO(V3) that depends upon the OpenCV and Boost.

QR Code Decoding

A QR (Quick response) code is a type of barcode that can be read easily by a digital device, and which stores information as a series of pixels in a square-shaped grid and used to track information about products in a supply chain, marketing, and advertising campaigns. More recently, they have played a key role in helping to trace coronavirus exposure and slow the spread of the virus.



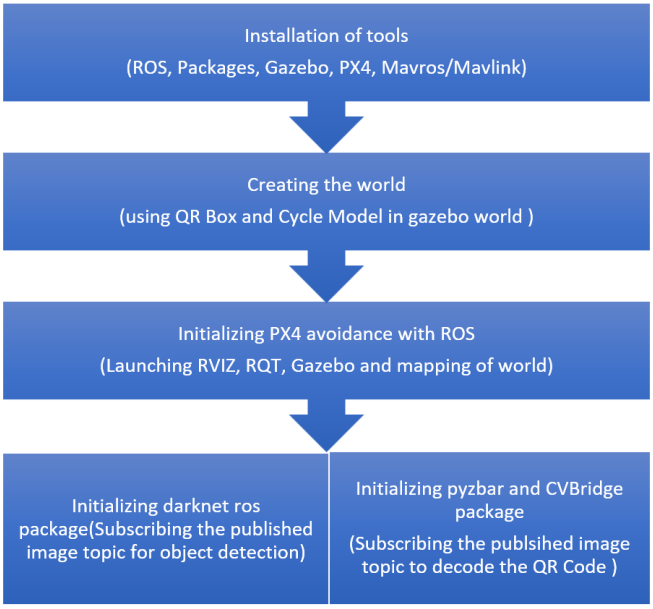
Sample output of decoded QR Code in terminal & Image

There are several ways to detect and decode the QR codes within ROS environment, but we are using a very simple approach. Image processing with the help of rospy along with OpenCV and Pyzbar libraries using Python script. Published ROS images by drone need to be convert to the OpenCV images which can be implement by CvBridge, so that pyzbar can then detect and decode the QR Code from its libraries.

Implementation

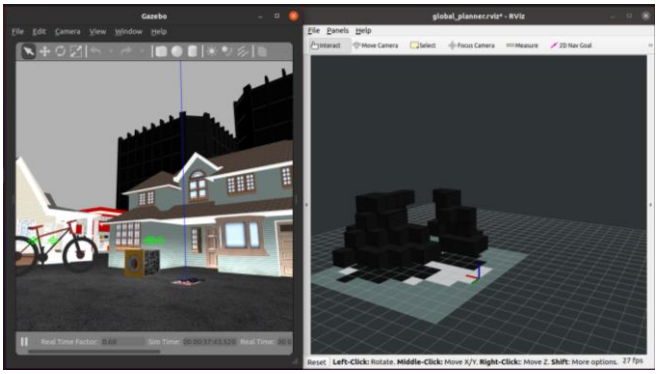
Tools	*Version*
OS	Ubuntu 20.04
Python	3
ROS	Noetic
Gazebo	11
QGroundControl	4.0
PX4	1.12.3
YOLO	Darknet_ros

We will start with launching PX4 with Gazebo simulator to load the iris drone in our test_city world with some custom 3D models like QR Code Box and Cycle modeled with Blender. Now we will integrate them with ROS which is a general-purpose robotics library that can be used with PX4-Avoidance, as discussed earlier, for drone application development to accomplish the path planner.

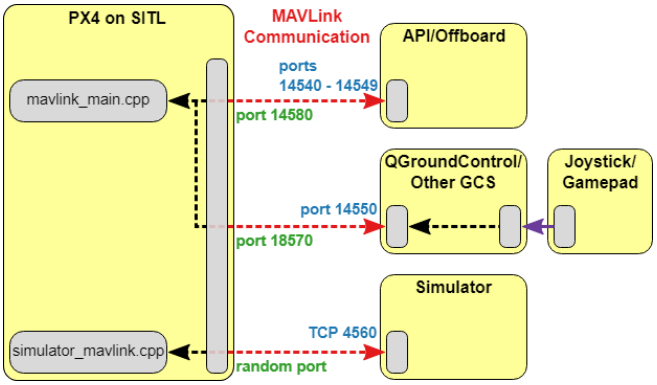


Sequence of implementation

To integrate, MAVROS need to be installed for respective ROS version. It is a ROS package that enables MAVLink extendable communication between computers running ROS for any MAVLink enabled autopilot, ground station, or peripheral. The different parts of the system connect via UDP and can be run on either the same computer or another computer on the same network.



Gazebo-ROS (Left to Right)



SITL simulation environment for any of the supported simulators (Gazebo)

So, we have setup our environment for the simulation, but we need to launch the simulation with Obstacle avoidance node therefore we will use the Global Planner package with PX4 and Gazebo simulator in our custom world. This process will launch the RVIZ that maps the environment and visualize the possible path based on the algorithms. To see the drone and world, a separate node is required to run in the terminal to launch the simulation in Gazebo simulator.

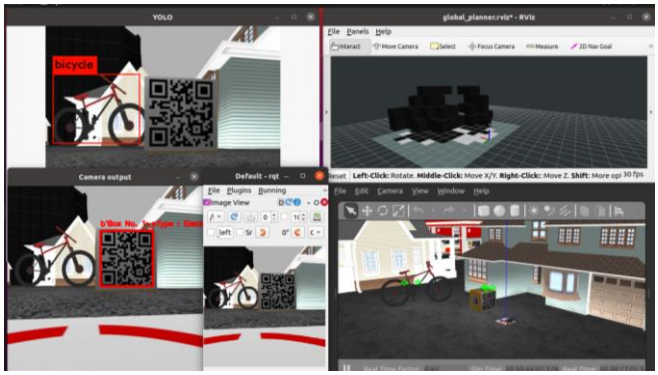
We can use the QGroundControl to plan our mission but to make the process more effective we will make a python script to command the drone and start the mission using MAVSDK library and execute the script in terminal.

Once the mission has been scripted, we will create another python script for image processing that will take the images published by rostopic of drone image, convert it into the OpenCV image format and finally decode the QR Code present in the image. The decoded information will be available in the terminal and as the OpenCV image output also.

Decoded QR code by using Python script in ROS-Gazebo

Now one more task need to be accomplished i.e., the object detection. After launching the mission and Image processing scripts, we will launch the darknet ROS package, but before we need to modify some configuration files present in the YOLO folder to subscribe the image that is being published by ROS. It can be done by changing the image topic in ros.yaml file to the topic that is being published by ROS.

Object Detected as Cycle in ROS - Gazebo



Final Output (From Top left to bottom right): YOLO-Object detected as Cycle, Camera Output-QR Code decoding, RQT-Real Time Image captured by Drone, Gazebo-World, RVIZ-Path Mapping

Conclusion

After setting up the environment with every tool we discussed, performed our projected simulation with autonomous iris drone enabled with stereo vision sensor to fly in the custom world. Python played a major role in our simulation as by running scripts, required to use GCS very less. Gazebo allows simulation to sense the dynamic behavior in a more realistic and effective world. ROS helped to run simulation with ROS packages and nodes by publishing and subscribing the topics with the simulator. Real time captured images from drone, sent to OpenCV which then targeted by pyzbar libraries and darknet to decode the QR Code and detect the type of object by drawing a bounding box around respectively. Finally, we achieved our projected goal to accomplish an autonomous navigation obstacle avoidance drone with QR Code decoding along with object detection.

Future Work

To implement the whole project on a real hardware drone in real environment for the commercialization use of multipurpose drone.

Acknowledgement

All the work purely done by open-source software tools and credits belongs to the original sources and references. Thanks to Cionlabs Pvt. Ltd. and CEO Mr. Sanjay Ahuja for mentoring during the project. Also, Thanks to the dept. of Electronics Science, Kurukshetra University, Kurukshetra, India.

Demonstration

Video Demo

References

1. https://www.ros.org
2. https://px4.io
3. https://wiki.ros.org/cv_bridge
4. https://gazebosim.org
5. https://github.com/leggedrobotics/darknet_ros
6. https://pypi.org/project//pyzbar
7. https://qgroundcontrol.com
8. https://blender.org