TEAM A3: Search and Rescue Drone

Goushika Janani Janagarajan*, Bharath Renjith, Mohnish Ramani, Lokeshwar Deenadayalan

gjanaga@ncsu.edu, bchenna@ncsu.edu, mramani@ncsu.edu, ldeenad@ncsu.edu

North Carolina State University

ABSTRACT

This project report provides a detailed narrative of an autonomous search and rescue UAV system designed for the purpose of emergency situations. This project was chosen with the idea to tap the UAV's unique skillset like manoeuvrability, vantage point view, ability to traverse deserts, collapsed structures, etc. Hence, it can be used in rescue missions where it can locate a person in distress and provide them with essential care. The UAV equipped with a companion computer in addition to the flight controller, sensors (IMU, GPS, compass, camera), package drop mechanism, and a ground base station make up the list. The software used to virtually simulate and control the drone are QGroundControl and Gazebo. The detection of humans is achieved with the trained neural network (YOLO v3). All these components combine to form the whole autonomous system. The UAV scans a predefined area taking pictures at regular intervals and upon detecting a person it updates the person's coordinates as the new waypoint, so the drone goes nearby by and hold its position giving out an alarm. Once that's done it returns to the home location after which further rescue operations can be executed.

INTRODUCTION

This project majorly focuses in the field of the search and rescue. In case of a disaster, or any accident, it is difficult to track people who are injured or stuck in a large area. Most of the rescue operations pertain to hikers who have lost their way during their hike. Rescue teams do not have the privilege to scout the area before executing their operation. Without prior scouting, the rescue teams are left in a maze with no map. There have also been cases where people have been found missing and it is very unfortunate for the rescue teams. In such cases, this type of UAV system is used to scout the affected area and identify the targets. The target's location is acquired thus benefitting the team with time to plan their rescue operation. The aerial view is quite suitable in detecting people rather than from ground. In addition to that, UAV can travel to inaccessible areas under extreme weather conditions as well. Thus, there is an increasing need for such type of drones in market due to its advantages.

In recognition of the potential application of UAS in Search and Rescue(SAR), several U.S government agencies including the Department of Homeland Security, Federal Aviation Authority (FAA), National Aeronautics and Space Administration (NASA), Department of Transportation (DOT) and Department of Defence (DoD) sent an acknowledgment to the Unmanned Aerial System (UAS) Search and Rescue (SAR) committee identifying the potential of UAS for Search and Rescue (SAR) mission and providing guidance on how to implement this application [1]. As a result this has translated to even local police and fire departments adopting drones, and usually collaborate with local search and rescue teams in using them for time-sensitive rescue operations. There are several drones in the market for SAR operations which are highly expensive. Not everyone would be able to afford such drones. Hence, our project focused on making it financially feasible without compromising its performance. This would allow students like us to contribute to the society by bringing in affordable quality drones for SAR operations. Large portions of our initial testing were on simulation environments, this gave us the opportunity to test possible scenarios and tackle them without having to flying a drone. This reduced the development cost. The experimental results of our project serve as a solid proof to our claim.

RELATED WORK

The primary foundation for this project is a similar one performed at NC State during 2018[2]. It involved the same objective of detecting a human autonomously in a predefined search area and dropped the package. The task was segregated into two segments wherein, the search area is navigated using a predefined route setup using the Ground Control Station (GCS). During the navigation the companion computer keeps getting the camera feed from a USB camera and processes it using a pretrained network to detect for humans in the image. If the companion computer detects a human in the processed image it initiates the package drop. The team had gone for this approach to reduce the processing time. The pretrained network used is darknet (YOLO). This was backed with the reason that it gives a quick prediction and good accuracy for the detection rate. This project is taken as the baseline for our project.

A research team at Dalle Molle Institute for Artificial Intelligence, the University of Zurich and NCCR Robotics have come up with a drone that autonomously navigates a forest trail using a deep learning neural network [3].

The team collected 20,000 images from a forest trial using three cameras to get the front, left and right of the trail. These images were used to train a network (consisting of 10 layers, 150,000 weights and 0.5 million neurons) which managed to classify if the seen image is left, right or forward. This helped to navigate the trail using a single camera with an 85 percent accuracy. The objective of this research is to depend on the visual imagery and not depend on GPS which might fail in a forest and is not very accurate. This drone could be used for search missions in forest trails.

APPROACH

DESIGN

The design of the UAV is the most important aspect of the whole project. The below figure gives an overall summary of the system.

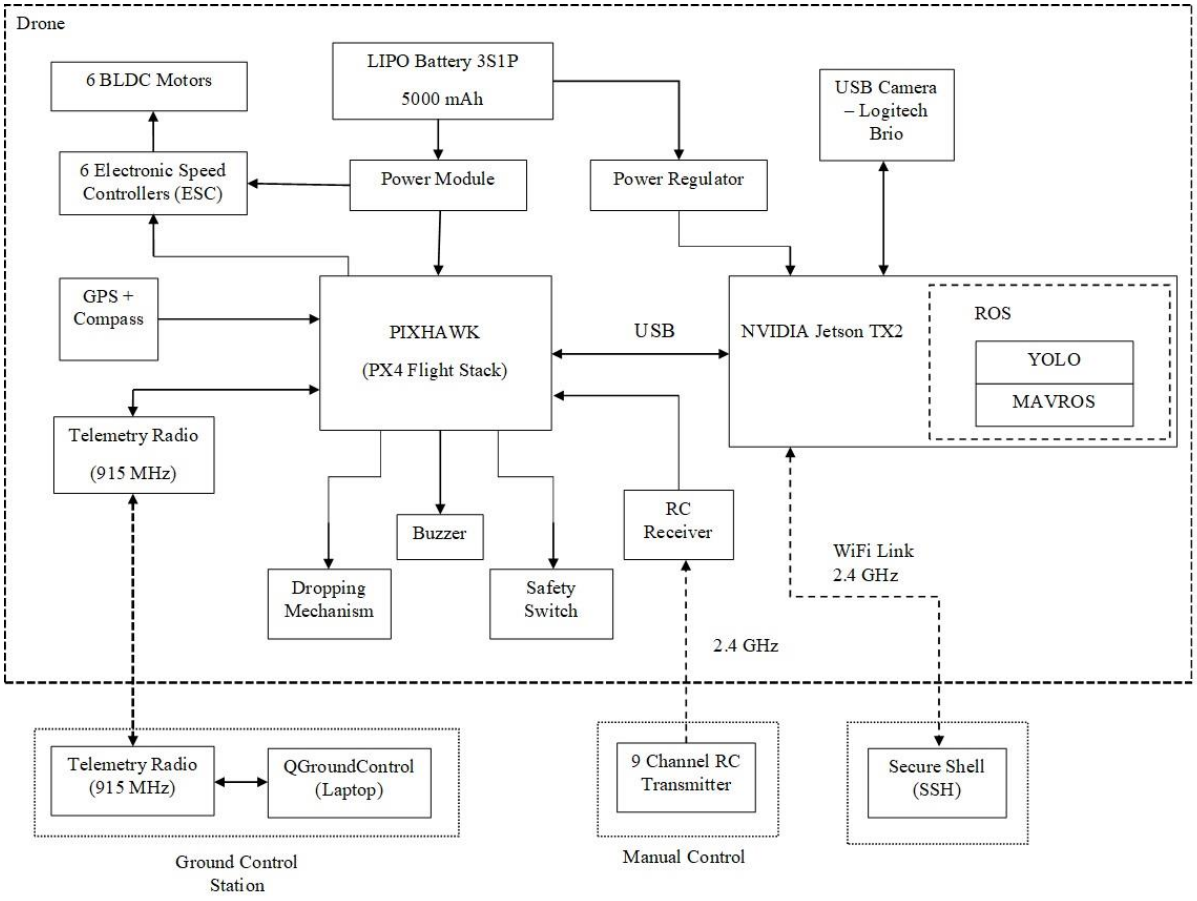


Figure 1: System Block diagram

All hardware and software components and their functions have been listed out in **Table 1, 2, 3** and **4.**

Component	Type	Role
PixHawk	Hardware	To interface peripheral devices, sensors and software.
PX4 Flight stack	Software	To control 6 rotors, based on sensor and manual controller input.
RC Receiver	Hardware	To receive signals from RC Controller
GPS + Compass	Hardware	To obtain the co-ordinates and orientation of the drone
Telemetry Radio	Hardware	To transmit/receive data signals to Ground station
Safety Switch	Hardware	To prevent accidental arming
Buzzer	Hardware	To communicate the status of the Pixhawk via sound.
LiPo Battery	Hardware	To power the components of the drone.
Power Module	Hardware	To regulate the power to PixHawk and ESCs
BLDC motors	Hardware	To rotate the propellers.
USB Camera	Hardware	To provide 2-D view of the environment to the neural network
NVIDIA Jetson TX2	Hardware	To run the project mission

Electronic Speed Controllers (ESCs)	Hardware	To drive the rotors.
Dropping Mechanism	Hardware	To drop the package.
MAVROS	Software	To communicate between TX2 and Pixhawk
YOLO	Software	To detect humans in the ground.

Table 1: UAV Design Components and Primary Software

Component	Type	Role
Telemetry Radio	Hardware	To transmit/receive mode and channel information to PixHawk.
QGroundControl	Software	To monitor mission and fail safe.

Table 2: Components in Ground Control Station (GCS)

Component	Type	Role
RC Transmitter	Hardware	To set modes and for manual control of drone

Table 3: Components in Manual Control

Component	Type	Role
Secure Shell (SSH)	Software	To remotely run scripts on Nvidia Jetson TX2

Table 4: Additional Software Component

In our project, we have designed a hexacopter to perform the search and rescue operation. The companion computer – Nvidia Jetson TX2 with Orbitty Carrier board is used. The PID tuning of the drone was performed using QGroundControl with the help of Spektrum DX7 RC transmitter before flight to make sure it is stable to control over manual control. The PX4 flight stack was used for greater stability.

The Nvidia Jetson TX2 communicates with the PixHawk flight controller through USB. The Nvidia Jetson TX2 also has the Robot Operating System (ROS) running on it. The ROS framework has 2 processes running in parallel i.e. human detection algorithm through YOLO and navigation through MAVROS.

Figure 1: Drone setup

To test the performance of the drone during flight, we set up a sample mission using QGroundControl as shown in Fig.2. This was focused to check if the drone was able to fly high enough for our mission. Fig3. shows a snapshot of the drone during the trial test.

Figure 2: Navigation Testing using QGroundControl

Figure 3: Drone flight during testing

HUMAN DETECTION ALGORITHM

Objective of the project is to fly the drone as high as possible and detect people in the ground as quick as possible. In order to get a clear picture from 10m height, we used a Logitech Brio camera (1920x1080 resolution).

YOLO is a state-of-art, real time object detection system, usually preferred for its performance. Based on previous studies [4], we found that YOLO V2 struggled with small object detections. Hence, we experimented with YOLO V3 and Tiny YOLO to select the suitable one for our project. The accuracy of Tiny YOLO was very poor. Therefore, we preferred YOLOv3 as it proved to be more accurate without compensating the performance speed. We are using a COCO Dataset pre-trained model of YOLO V3. To reduce the processing load in TX2, we shifted to image processing rather than video processing.

YOLOv3 utilizes the CUDA cores on TX2. But YOLOv3 resizes every input image to 416 x 416 and hence fails to detect humans on HD images taken from 10m height. Hence as a wraparound we split the HD image into 5 patches and test for human on each of them.

On detecting a human, their image co-ordinates are transformed into world co-ordinates using camera matrix. From the current position of the drone, the distance to the human is estimated using Euclidean distance formula, and the new waypoint is generated.

CAMERA CALIBRATION

Once human is detected by YOLOv3 we should map the image coordinates of human (centre of bounding box) to the world coordinates, for which we need camera matrix. We performed a calibration using OpenCV to find the camera matrix. A 9x7 checkerboard image of size 2cm was used for the calibration. We captured images of checkerboard at several orientations and used calibrateCamera() from OpenCV toolkit to perform the calibration.

We tested the camera matrix generated, by taking the picture of checkerboard at a known distance from camera (parallel to image plane). We then measured the pixel coordinates of two adjacent corners of the checkerboard in image plane and used intrinsic matrix to find its actual length in world plane. The measured value was almost close to the actual value of 2 cm.

EXPERIMENTAL RESULTS & ANALYSIS

The waypoint generation and the detection were scripted in Python for ROS. In order to test our waypoint generation code, we preferred SITL approach for both safety measures and economical option. The figure below shows QGroundControl in sync with the Gazebo environment. This helped us to visualise our code and make changes accordingly before testing the actual code on the drone. The detection could not be verified using simulation due to limitations of gazebo (Camera Plug-in)

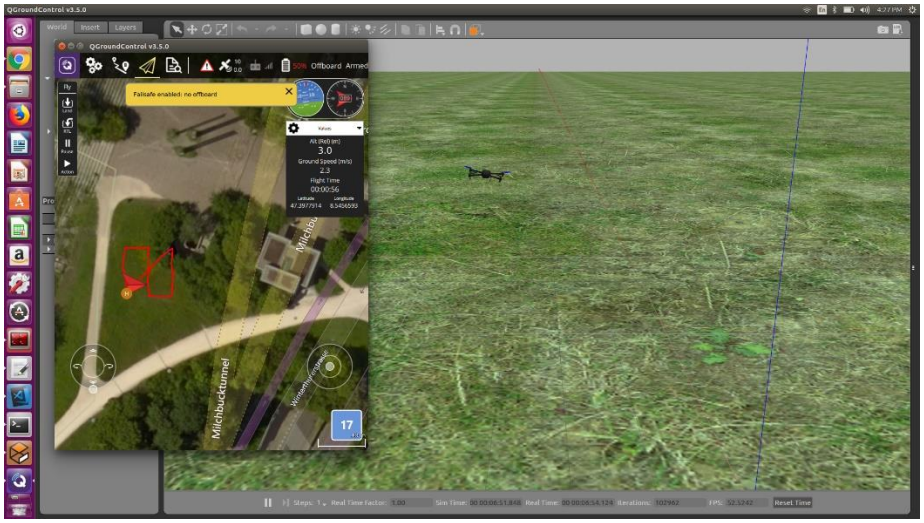


Figure 4: Gazebo Simulation

This resulted in opting static testing of the detection algorithm. From the rooftop of Hunt Library, we tested the detection portion of the code using the same camera and Tx2 meant for the experiment.

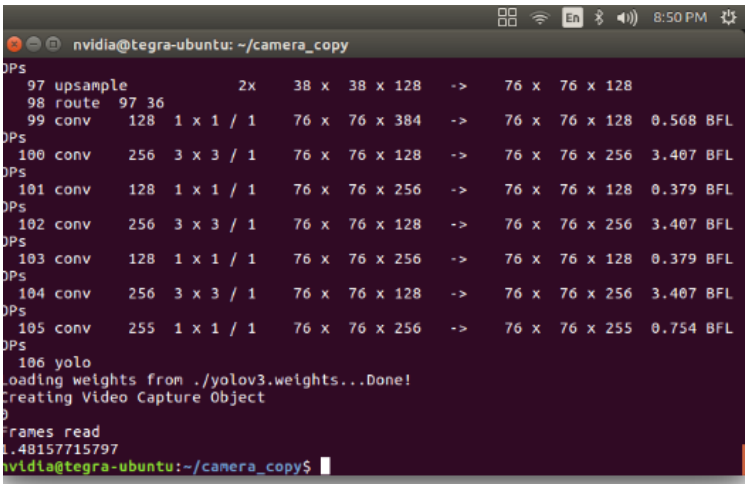


Figure 5: Detection Time

Figure 6: Static test for human detection

YOLOv3 takes about 0.7 seconds to evaluate each image of size 416x416. Each HD image from the camera is split into 5 patches as shown in Fig.6. In the above case, the human was detected in the second split and hence the detection time was about 1.48 seconds as shown in Fig 5. The best case would be 0.7 seconds to detect and the worst case would be approximately 3.5 seconds.

Having tested and verified, we ran the code on the drone for real time performance.

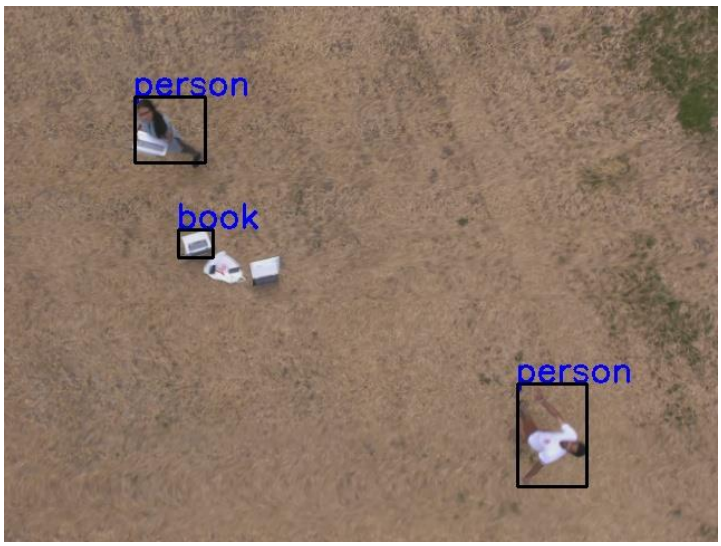


Figure 7: YOLO Testing on Drone

We tested the mission several times and we inferred from the results that on an average of 75% of the trials, our SAR drone was able to detect humans with 90 % confidence. Fig.7 shows the successful detection of the humans. However, the detection failed in the remaining instances either giving false detection or no detection at all as seen in Fig. 8. This could be because of the lighting issue, orientation of the person, color, occlusion and the way the network is trained itself.

Figure 8: YOLO Testing on Drone

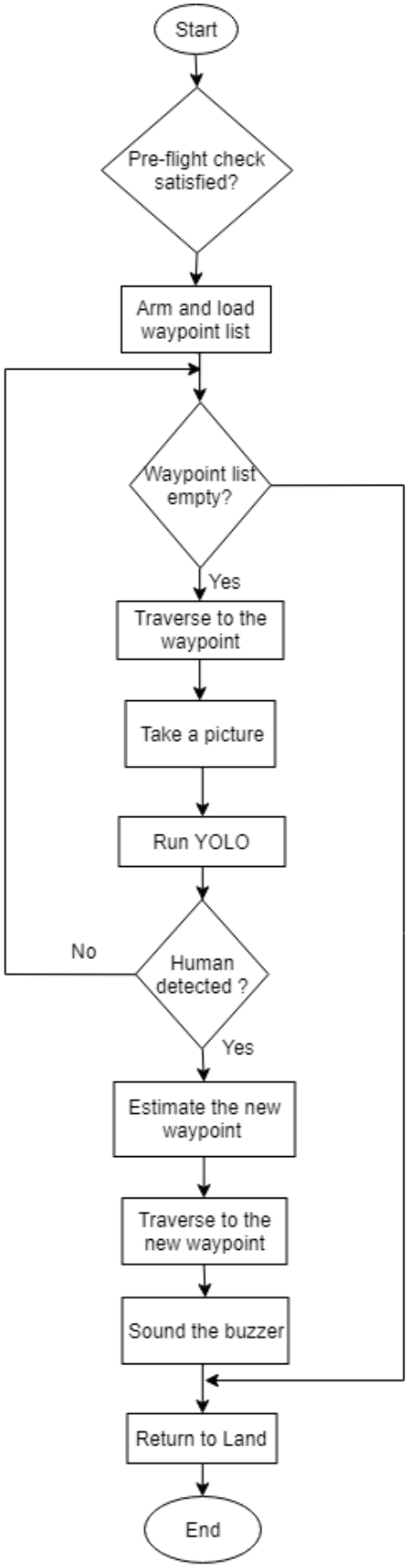
CONCLUSION

Our team was able to successfully scout a large area in a predefined manner and detect humans using the inhouse designed drone. This was performed at an altitude of 10m which, is an improvement on the previous work [3].

During the project we came across few situations which we felt could be addressed in the future.

- Gazebo for ROS Kinetic didn't have appropriate camera plugin to test the code in simulation. This is present in later versions of ROS.
- During testing, on a few occasions we came across the low battery warning issue from the QGroundControl though the battery had over 80% capacity. This could be because the peak current consumption is making the QGC to falsely alarm.
- Nvidia Tx2 is crashing when it is trying to write images to the system. This could be due to memory issue. Needs to be investigated in the future.

FLOWCHART FOR PROJECT MISSION



APPENDIX

Endurance Calculation:

$$
\begin{aligned}
\text{Acceleration due to gravity} &= 9.8 m/s^2 \\
\text{Air Density, } \rho &= 1.225 kg/m^3 \\
\text{Diameter} &= 0.254 m \\
\text{Number of propellers} &= 6 \\
\text{Total propeller area,A} &= n * \pi * (D/2)^2 \\
&= 0.30387 m^2 \\
\text{Total mass of UAV } m_{t0} &= 2.62 kg \\
\text{battery Mass} &= 0.403 kg \\
\text{Mass fraction of battery, } MF_{batt} &= 0.1538 \\
\text{At maximum power, Thrust} &= 51.38 N \\
P_{ideal} &= \sqrt{\frac{Thrust^3}{2 * \rho * A}} = 426.8391 W \\
\text{Reasonable assumption for efficiency} & \\
\text{efficiency of motor} &= 0.85 \\
\text{efficiency of propellers} &= 0.8 \\
\text{efficiency of ESC} &= 0.8 \\
\text{Propulsion Efficiency of Hexacoptor, } \eta_{eff} &= 0.544 \\
P_{batt} &= P_{ideal} * \eta_{eff} = 232.2005 W \\
\text{Endurance} &= MF_{batt} * E_{spec} * \eta_{eff} * \sqrt{\frac{2\rho A}{g^3 m_{to}}} \\
&= 12.025 mins
\end{aligned}
$$

Camera Footprint Calculation:

When the drone is facing vertically down

$$
\begin{aligned}
Height &= 10m \\
HorizontalFOV &= 82.1 degree \\
VerticalFOV &= 52.2 degree \\
Horizontal\ Range\ of\ drone &= 10 * \left(tan\left(\frac{HFOV}{2}\right) - tan\left(\frac{-HFOV}{2}\right)\right) \\
&= 10 * (0.8708 + 0.8708) \\
&= 17.416m \\
Vertical\ Range\ of\ drone &= 10 * \left(tan\left(\frac{VFOV}{2}\right) - tan\left(\frac{-VFOV}{2}\right)\right) \\
&= 10 * (0.4899 + 0.4899) \\
&= 9.7986m
\end{aligned}
$$

REFERENCES

[1] https://connect.ncdot.gov/resources/Aviation%20Resources%20Documents/SAR_UAS_Addendum(July2016_Vers_1-0)-Final.pdf

[2] https://people.engr.ncsu.edu/mlsichit/Teaching/492/index.html

[3] https://www.theverge.com/2016/2/11/10965414/autonomous-drones-deep-learning-navigation-mapping

[4] Embedded Real-Time Object Detection for a UAV Warning System,Nils Tijtgat-Wiebe Ranst-Bruno Volckaert-Toon Goedeme-Filip Turck - 2017 IEEE International Conference on Computer Vision Workshops (ICCVW) - 2017