

ECE-592 PROJECT REPORT (A20-2)

SURVEILLANCE AND ROAD SIDE ASSISTANCE DRONE

Atharva Gupta
Angela Vivian D'Costa
Binoy Thomas
Srinivas Gopalakrishnan
Varun R Haritsa

April 2019

1 Abstract

Our project is focused on designing and developing a drone for autonomous surveillance and roadside assistance. This is aimed to complete the tasks of road following, arrow detection (to identify the start direction) and object detection. Our design can be made applicable to today's freeways as under-speeding and illicit stoppages are always concerns for a driver. This drone can be made extremely robust with a variety of functionalities like accident reporting, first-aid delivery in case of any emergencies, traffic law violation reporting, etc. For the application level design, we used image processing algorithms like, HSV masking, background subtraction, and midpoint computation for road following. The arrow sign detection includes a computation of Hough transform of the retrieved frame, and align the drone in the direction of the arrow. Object detection also focuses on contour extraction and area thresholding after appropriate noise removal, to detect boxes on the side of the road. The system level design encapsulates the use of ROS for the integration of distributed sensors (cameras and other on-board sensors) and controllers (NVIDIA TX2 and autopilot).

2 Introduction

The primary problem of interest is for the drone to autonomously follow a road. This road following can be extended to various applications. This is an important problem because continuous surveillance and monitoring on freeways and highways is crucial. Our drone presents a solution to this problem, because 24-hour manned surveillance everywhere is nearly impossible. The possible applications of this drone can be accident response and first aid delivery, monitoring for any traffic law violations (mostly over speeding, under-speeding, pulling up cars by the highway, etc.), identification of accident prone zones, warning drivers, etc.

Our contribution was to implement simple image processing algorithms instead of deep learning networks so that we could increase the computation speed. We were able to detect arrows which has not been implemented in any recent projects.

Presently, people have been using drones for getting crash data faster for accident sites [1], basic aerial police drones [2], Drones Raising Roadside Pavement inspection [3], Drones to catch dangerous drivers [4], Drones for road safety [5], finding lanes on roads [6]. After reading all these websites and what people have been doing till date we decided our course of work and proceeded in the project.

3 Approach, Results and Analysis

3.1 Arrow Detection

Arrow detection was done to decide whether the drone should travel forward in the direction it is facing or change its orientation in the direction of the arrow and then proceed with the mission. To begin with, the task we considered numerous approaches.

- We initially considered using Neural networks and Visual recognition to detect arrows and identify its direction. We used colored and black and white arrow images in different orientations to train linear classification models using Random Forest, Logistic Regression and IBM Watson Studio's Visual Recognition model. However, we found that this was not a very efficient way of working since we could not train the models with a very large dataset. Also the model would not be invariant to illumination
- We finally decided to come up with our own algorithm to solve this problem, considering that the arrow is in the visible range for the drone. The frame in which the arrow (red colour) is present is manipulated such that the arrow is made white and everything else in the background is black. To ensure the grass in the surrounding does not cause trouble if Green channel pixels < 150 and Red channel pixels > 250 make them white(255) and the rest black(0) and copy this to all the color channels. The image is then converted to Grayscale
- Then the portion of the image containing the arrow is cropped using a fixed box size around the arrow. Canny edge detection is done on this cropped image and it is used to generate the Hough Transform which is used to detect the orientation of the arrow

- The Hough transform of the image gives us the angle and perpendicular distance from the upper Horizontal axis of the image. These details are used in the following formulas to draw lines along the edges of the arrow

$$X_0 = \cos(\theta) * PerpendicularDistance$$

$$Y_0 = \sin(\theta) * PerpendicularDistance$$

$$X_1 = X_0 + 1000(-Y_0)$$

$$Y_1 = Y_0 + 1000(X_0)$$

$$X_2 = X_0 - 1000(-Y_0)$$

$$Y_2 = Y_0 - 1000(X_0)$$

- Since the height of the drone is going to be fixed, we consider a fixed distance between the parallel lines of the arrow. We further find the angle at which the arrow is aligned by ensuring the lines that intersect (arrow head) have the difference of the angles from the horizontal almost equal

Algorithm Summary for Arrow Detection

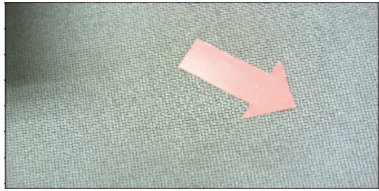
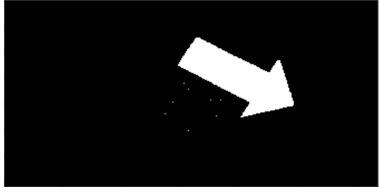
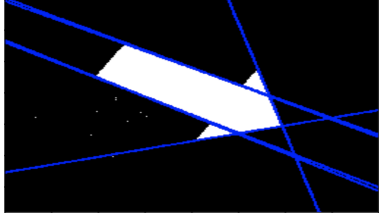
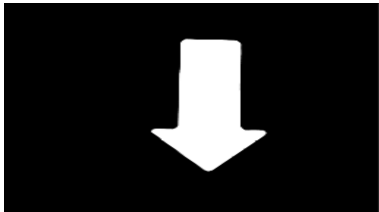
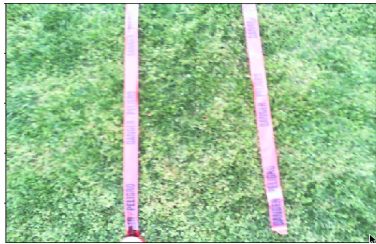
Step No.	Description	Images
1	Get the frame in which the arrow is present	
2	Convert the pixels belonging to the arrow (Red) to white and the rest of the image to black	
3	Crop out the portion of the image that contains the arrow and perform Canny Edge detection and Hough transform to generate the lines along the arrow's edges and get the angle of the arrow	
4	Rotate the arrow by the angle obtained above and determine if it's facing up or down to determine the final angle by which the drone has to be rotated	

Table 1: Arrow Detection

3.2 Road Following

The main goal of our project is Road Following. The drone must make an autonomous decision of where the road is and travel in between them. We had multiple approaches for proceeding with an algorithm for achieving this.

- **Neural Network:** For this we considered training a neural network model for the lane detection but since we were working on a prototype, this approach was not be feasible because the number of images required for training was very large
- **Edge Detection and Hough Transform:** We tried to approach the problem using Canny edge detector for the roads. Using Hough Lines to get the lanes replicated as a grayscale image which we can process further to get the results. This algorithm worked well when we were testing it on custom images and normal roads. But when we tried testing on the grass which will be our field for the demo, we got the details of the grass as well in the edge detection process which created a very noisy image. Even after removing the noisy areas, we had to make sure to undistort our image, such that curved roads appear as straight roads, to get Hough lines. Although this method was feasible for the drone to take decisions much faster on the flight, we decided to discard this algorithm, as this needs a lot of image transformations to achieve our goal
- **RGB Color Masking:** This algorithm is the simplest of all the algorithms we have used for Lane detection. Since we get a lot of background noise with the grass we decided to mask off the grass using an RGB mask and use the remaining part of the image to get the lanes. We approached the problem by creating a mask which allows only the intense red colored pixels. This solved our problem when we tested the code in normal conditions i.e. indoors and in normal illumination. We had a tough time testing the algorithm outdoors, especially on the grass. The effect of the surrounding background and the illumination whether it was dark or bright brought many changes to the image when captured by the camera. Since illumination was the main source of trouble we decided to get a mask in a different color space where illumination effects are tolerable
- **HSV Color Masking:** We found good results when we decided to use HSV mask to mask off all colors except red. This mask proved to be useful on different illuminations, as we could control the range of Hue of the image between 0 – 30 and 150 – 179. There was a small hitch in the algorithm when the sunlight directly hit on the tape with which we construct our lanes. Since the tapes are shiny they reflect a lot of sunlight which made red color appear close to the 100% saturation level. So, we modified our mask to allow all values of Saturation except the extreme bright lights. This proved successful as we were able to get a masked image of only the red dominant pixels. With this data we were able to calculate the midpoint of the roads and we were able to determine by what angle the drone has to change its orientation to follow the midpoint of the roads
- **Algorithm Flow for one image frame**

Step No.	Description	Images
1	Get an image from the onboard camera	
2	Convert the given image frame to the HSV color space	BGR to HSV Color space conversion: <code>cv2.cvtColor(image,cv2.COLOR_BGR2HSV)</code>

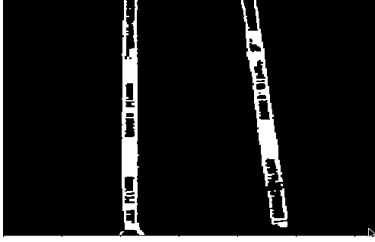
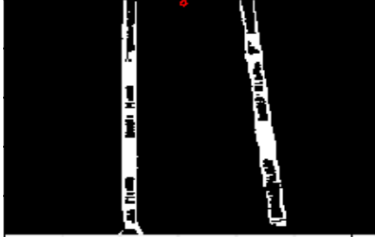
3	<p>Process the image with a mask which allows only red to pass through</p> <p>Hue: 0–15 and 155–179</p> <p>Saturation: 5–255</p> <p>Value: 0–255</p> <p>After masking we perform morphological operations</p>	
4	<p>From the obtained grayscale image find the midpoint of all the white pixels along the top row</p>	
5	<p>Get the angle that the drone must change from its current orientation</p>	<p>Top Midpoint = (pos_x, pos_y)</p> <p>Relative Drone Position in the image: $(H, W/2)$, where H is the height of the image and W is the width of the image.</p> $Angle(deg) = \tan^{-1}\left(\frac{(pos_y - W/2)}{(H - pos_x)}\right)$

Table 2: Road Following

- This function is invoked every time a frame is taken to calculate the angle by which the drone should orient itself. This makes the drone follow the lane and complete its mission

3.3 Object Detection

Object detection tries to emulate a real-world scenario where the drone has to detect vehicular stoppages on freeways. The approach, to detect the object was to mask the background. We applied a HSV mask to the retrieved frame to mask out the background. This then leaves a vivid foreground to continue with the noise removal and contour extraction. On obtaining a masked image, it is subjected to morphological image processing techniques. This process removed the salt and pepper noise from the frame making it clean for further processing. The next step was contour extraction. The contour extraction in OpenCV works best for purely binary or grayscale images (single channel images) and hence extracting contours from multi-channel images (like RGB, HSV) always becomes noisy. Therefore we used area thresholding where the contour is only considered if it has an area higher than a certain threshold.

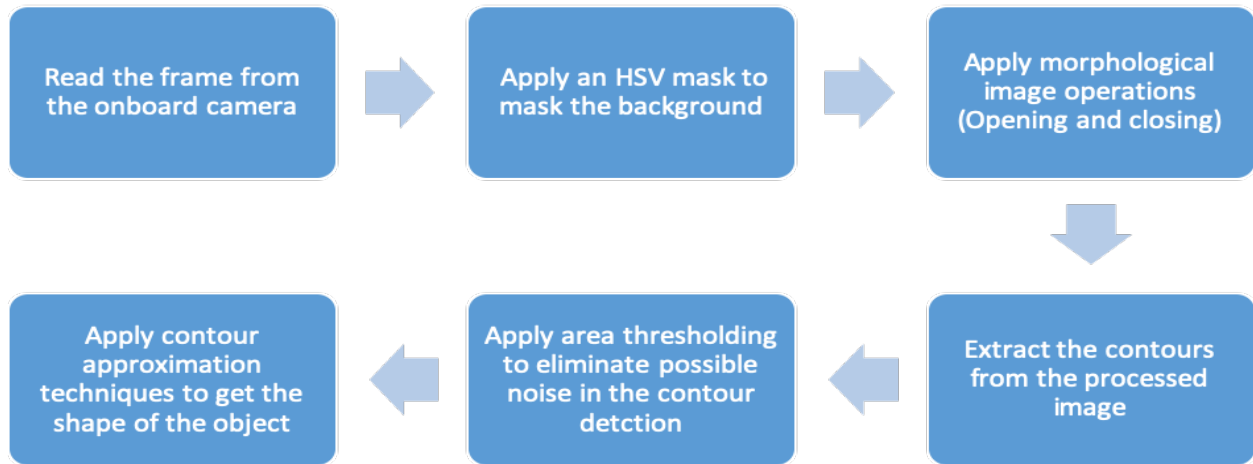

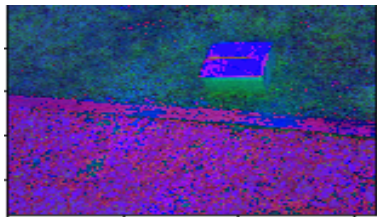


Figure 1: Object Detection Flowchart

Algorithm Flow for one image frame

Step No.	Description	Images
1	Get the image from the camera	
2	Convert the image into the HSV colour space	
3	Define the HSV mask and apply it on the image	$\text{light} = (0, 100, 20)$ $\text{dark} = (60, 240, 255)$ $\text{mask} = \text{cv2.inRange}(\text{img_hsv}, \text{light}, \text{dark})$ $\text{result} = \text{cv2.bitwise_and}(\text{frame}, \text{frame}, \text{mask} = \text{mask})$
4	Perform morphological operations on the image (Opening	$\text{opening} = \text{cv2.morphologyEx}(\text{mask}, \text{cv2.MORPH_OPEN}, \text{kernel})$


	and closing)	closing=cv2.morphologyEx(opening, cv2.MORPH_CLOSE, kernel)
5	Identify and extract the contours	2990.5, 1 Area of contour, Number of contours
6	Identify the object based on the extracted contours	

Table 3: Object Following

3.4 Software

We used the Nvidia TX2 Orbbity carrier board as the companion computer on the UAV. All logics to fly the drone were developed on ROS and python. The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour. ROS works on the publisher- subscriber mechanism. ROS nodes communicate with each other and exchange messages over ROS topics. Nodes are not aware of who they are communicating with. Request / reply is done via a Service, which is defined by a pair of messages. ROS code is written in packages which are built using catkin-tools.

The PX4 communicates with the simulator (e.g. Gazebo) to receive sensor data from the simulated world and send motor and actuator values. It communicates with the GCS and an Offboard API to send telemetry from the simulated environment and receive commands. Gazebo7 was used for SITL. Our drone emulated the simulation perfectly. This made it very easy to program our drone and ensure that it works safely where it is flying. MAVROS is a MAVLink extendable communication node for ROS with proxy for Ground Control Station. This package provides communication drivers for various autopilots with MAVLink communication protocol. Additionally it provides UDP MAVLink bridge for ground control stations (QGroundControl). MAVROS consists of messages and services which can be used to change the parameters of the drone or control its position and orientation. QGroundControl provides full flight control and vehicle setup for the PX4. It can be used to fly, plan and complete various tasks on the drone.

For our project, we used the OFFBOARD mode available on QGroundControl to send commands to the onboard NVIDIA TX2 companion computer. ROS was found to be very useful as it provided an intuitive mechanism to communicate between QGroundControl and the drone.

We had two ways of following the road and sending commands

- PID control: By sending a different velocity to each of the motors we could make the drone to follow the road. This method was inefficient because of the effects of wind on the drone. Another reason was that the camera would have to take images way more often and this would increase the computations.
- Waypoint navigation: We chose this method to complete our task because it ensured that our drone reached the target location because of the feedback sent to the ground station. The onboard computer would have to process fewer images because they would be processed only at the waypoints. We increment the waypoint depending on the angle the road makes with the present position of the drone. The orientation of the drone was also made to face the direction of motion of the drone.

The algorithm along with their ROS commands are given in figure

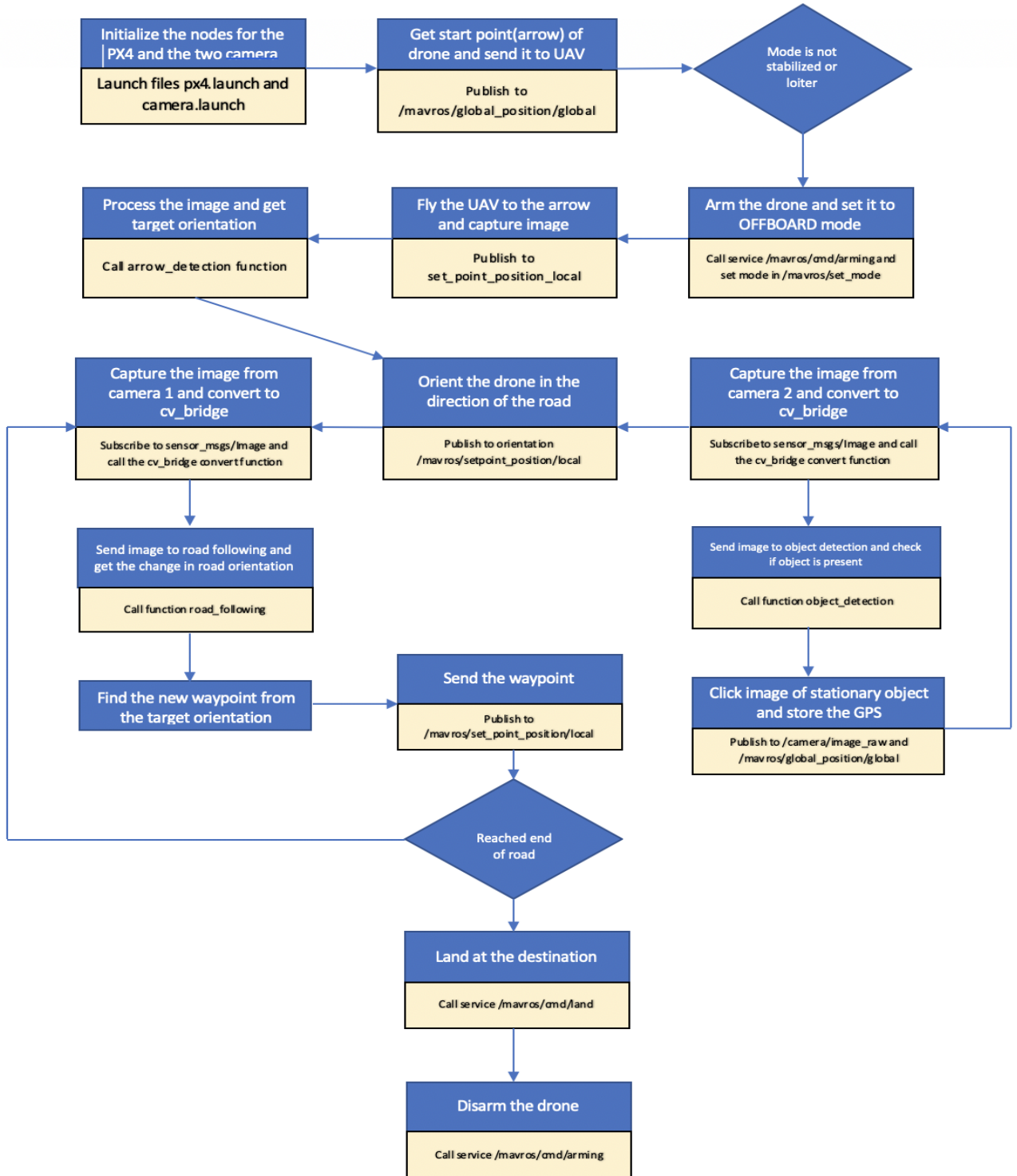


Figure 2: Software Flowchart

4 Conclusion

We were able to implement the task of building a ‘Surveillance and Road Side Assistance Drone’. We used ROS and QGroundControl to simulate the drone operations. This was then tested on the drone. Since our algorithms had to consider illumination as an important factor we fixed the color of the arrow and the makeshift road lines to be red. Once the cameras were interfaced, the obtained images were used to run our arrow detection, road following and object detection algorithms. If a stationary object was detected its GPS coordinates were sent to the Ground Control Station.

References

- [1] <https://www.futurity.org/drones-car-crash-data-1966362/>
- [2] <https://www.pogo.org/analysis/2018/09/these-police-drones-are-watching-you/>
- [3] <https://blog.dronedeploy.com/drones-raise-the-bar-for-roadway-pavement-inspection-9c0079465772>
- [4] <https://www.marketplace.org/2017/11/13/world/france-drones>
- [5] <https://www.arrivealive.mobi/assistance-of-the-drone-in-accident-investigation-and-road-safety>
- [6] <https://towardsdatascience.com/finding-lane-lines-on-the-road-30cf016a1165>