INTER IIT TECH MEET
IIT KHARAGPUR | 25-27 MAR 2022

Inter IIT Tech Meet 10.0

DRDO's UAV-Guided UGV Navigation Challenge

Documentation

INTER IIT TECH MEET
IIT KHARAGPUR | 25-27 MAR 2022

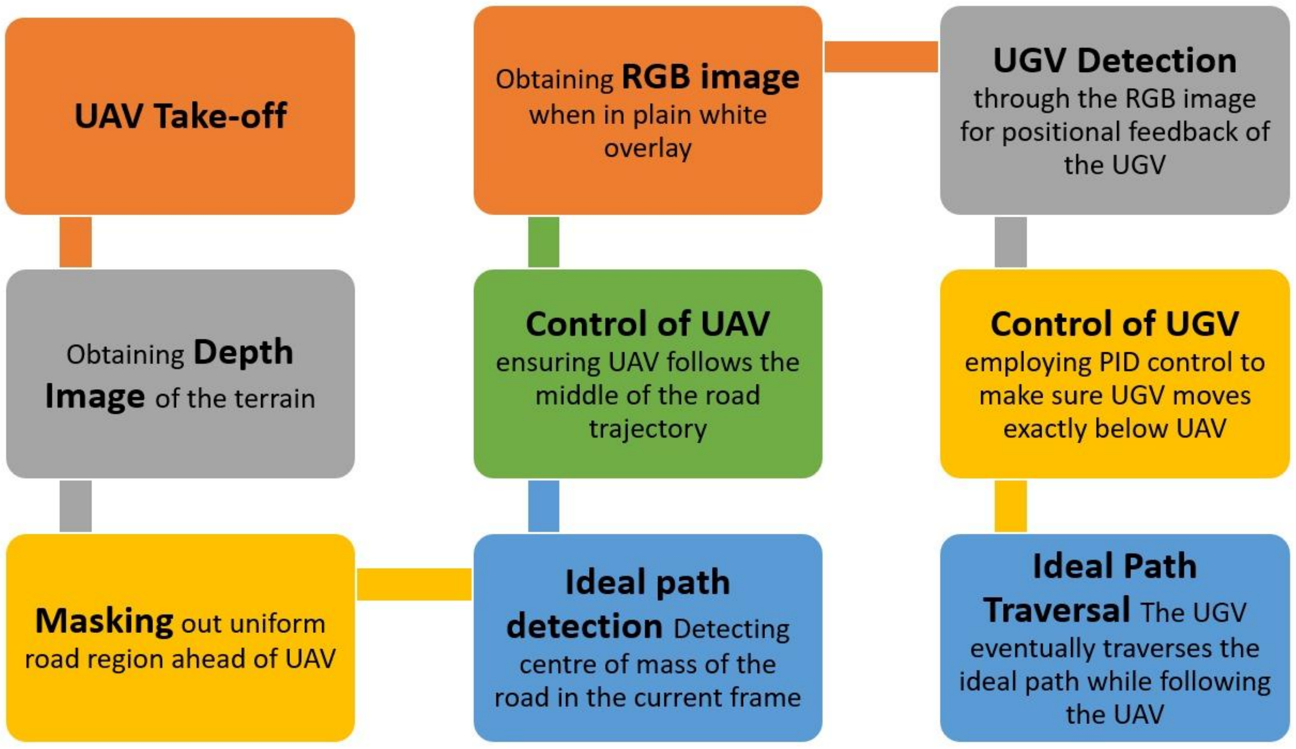
Index

1. Abstract

The challenge of driving a ground vehicle on roads in snow-covered mountains is a pressing one. We have proposed a novel solution for navigating an unmanned ground vehicle with the help of an unmanned aerial vehicle. Some key points used in our approach are:

- ➔ We have used depth image feed for road detection instead of RGB image because of low visual features in the RGB image. This makes our algorithm more robust and immune to differences in the terrain's visual features (colours, textures, etc.)
- ➔ Since the UAV's navigation is dependent on the depth image, and the UGV's navigation is dependent on the RGB image, both can be done simultaneously without the explicit need for a pre-generated map.
- ➔ We've not used any 3rd party package for mapping or navigation, thus making the algorithm superior in terms of originality and computational cost.
- ➔ Our algorithm is computationally quite efficient. World 1 is completely covered by UGV in 3 minutes.

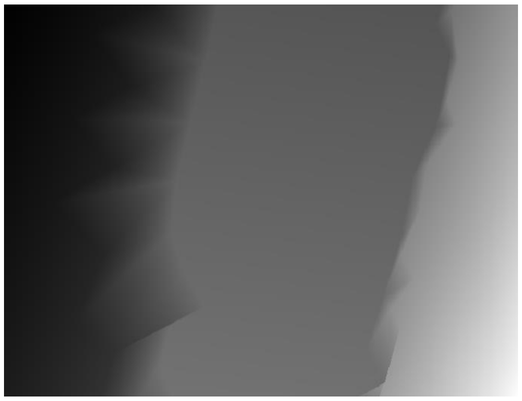
The approach developed is depicted stepwise as follows:

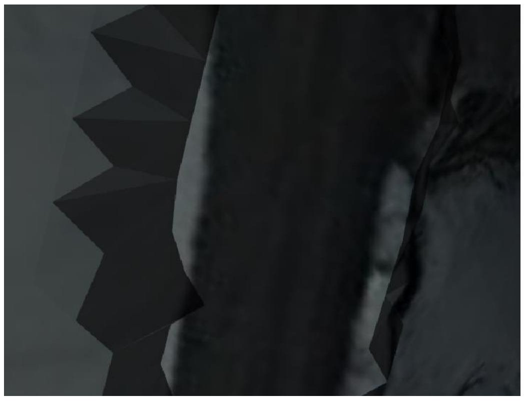

UAV Take-off
Obtaining Depth Image of the terrain
Masking out uniform road region ahead of UAV
Ideal path detection Detecting centre of mass of the road in the current frame
Control of UAV ensuring UAV follows the middle of the road trajectory
Obtaining RGB image when in plain white overlay
UGV Detection through the RGB image for positional feedback of the UGV
Control of UGV employing PID control to make sure UGV moves exactly below UAV
Ideal Path Traversal The UGV eventually traverses the ideal path while following the UAV

2. Road Detection and Mapping

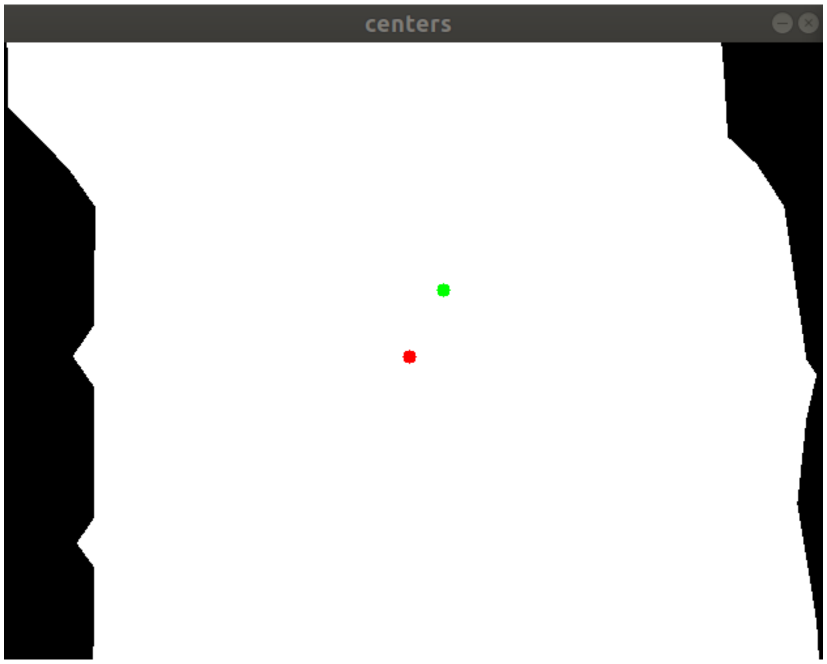
As is clear from the worlds provided, the road is a relatively flat region between a combination of valley and a cliff. However, the visual features of the road in the RGB image are not distinct enough to be able to be differentiated from the surrounding terrain. Thus, for detecting the road below the UAV, we've worked with the depth image of the drone published on the /depth_camera/depth/image_raw topic.

The depth camera being blind to the image overlay outputs an image that only has different pixel intensities corresponding to the depth of the terrain points below. The road thus can be distinguished, being a very slight gradient, from either side (valleys and cliffs), having higher values of intensity gradients.

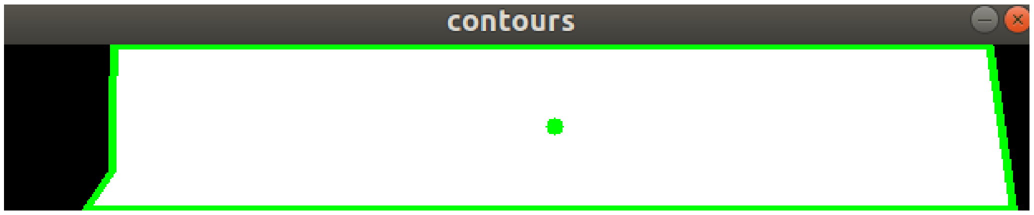

Depth Image


RGB Image

We've used OpenCV to detect the centre of the road. The road's slope in the forward direction is calculated using the intensity variation of pixels. Based on this slope, binary thresholding is done to mask out the region consisting of only the road area on which the drone has to travel. We then find the centre of mass of a band of the road directly ahead of the UAV (100 pixels long in the image frame) which coincides with the centre point of the road (green dot in fig).



Masked Image of the road depicting the road's centre (the desired position of the UAV) and the UAV's current position.



The 100-pixel long band ahead of the UAV, and its centre of mass.

Since our algorithm aims to ensure that the UAV always travels along the centre of the drone, the coordinates of the UAV are essentially the coordinates of the centre of the road. Thus, we are storing these coordinates in a rosbag file and as a plot using Matplotlib.

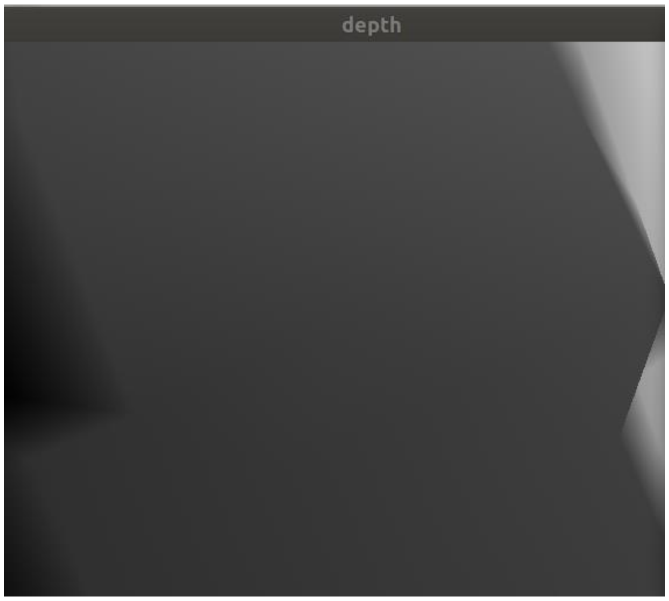
3. UAV Traversal

Take-off

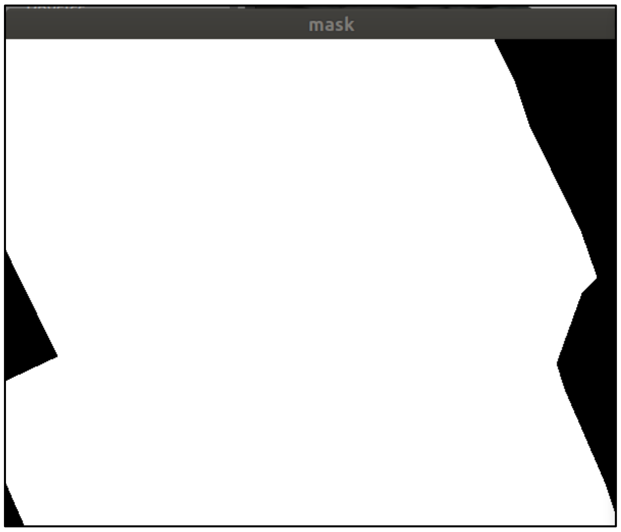
The UAV navigation begins from it taking off from the spawn location (within 5 m of the UGV). The take-off has been achieved through the Ardupilot framework. Sensors like IMU, GPS and gyroscope are calibrated and once this is complete, the UAV gets into “guided’ mode (using rosservice /mavros/set_mode), then arming of the UAV starts (using rosservice /mavros/cmd/arming). To proceed with the take-off (through /mavros/cmd/takeoff) we then publish a take-off height of 15 m (to ensure that the UAV satisfies the height constraint of 20 m).

Path-planning

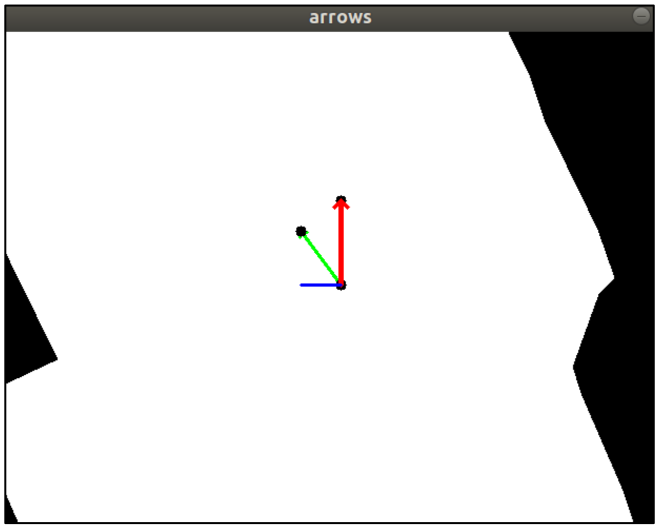
Once we get the coordinates for the centre of the road through image processing, the next job is to calculate the relevant commands which need to be published to the UAV. Since the RGB-D camera of the UAV is facing directly downwards, the centre of the image published by the depth camera will be the UAV’s x and y coordinates and the image’s vertical axis will be the UAV’s current heading trajectory. The transversal distance between the UAV’s current position and the road’s centre (dy) gives the error signal for calculating the UAV’s velocity perpendicular to the road (blue line in fig). The angle (θ) between the UAV’s heading direction (red arrow in fig) and the road’s trajectory (green arrow in fig) acts as the error signal to calculate the yaw which needs to be published for the UAV to align itself along the central trajectory of the road.



Depth Image



Masked Depth Image



Masked Image depicting drone's trajectory, road's trajectory and distance between UAV and road centre

Navigation

We are working with the local coordinate frame of the UAV (coordinate frame 8 in Mavros, with type mask 3008) so that the UAV can easily change its yaw and other important parameters in real-time without the need for any tedious transformations. The commands are published through the /mavros/setpoint_raw/local topic. 4 commands are published to the UAV which have been explained below:

- **Forward velocity in the x-direction (velocity.x)**: A continuous forward velocity is published to the UAV so that it traverses the road. This velocity depends on the yaw angle published to the UAV. If the UAV is negotiating a sharp turn, the forward velocity is reduced (to 0.7 m/s), while on relatively straight trajectories, the forward velocity is 1 m/s.
- **Transversal velocity in the y-direction (velocity.y)**: Using a P controller, a velocity is published in the horizontal direction (w.r.t. the image frame) which drives the distance error dy to 0. It provides an additional element of control in the horizontal direction, keeping the UAV in the centre of the road.
- **Vertical position in the z-direction (position.z):** Using a P controller, a position in the z-direction is published so that the UAV's elevation from the road is always fixed

and equal to the take-off height of 15 m. This also ensures that the UAV doesn't exceed the height constraint of 20m. Another advantage of controlling the height of the UAV is that the depth image thus obtained is always relatively uniform in terms of the width of the road, as well as both of its edges are inside the frame.

- **Yaw angle (yaw):** Using a PD controller, an accurate yaw angle is published to the drone so that it aligns itself with the direction of the road and drive θ to 0. Upper and lower bounds are enforced on this angle to prevent oscillations and noise. As long as the yaw is aligned along the trajectory of the road, the depth image obtained is also such that the forward path is exactly vertical, in turn guiding the UGV on the forward path.

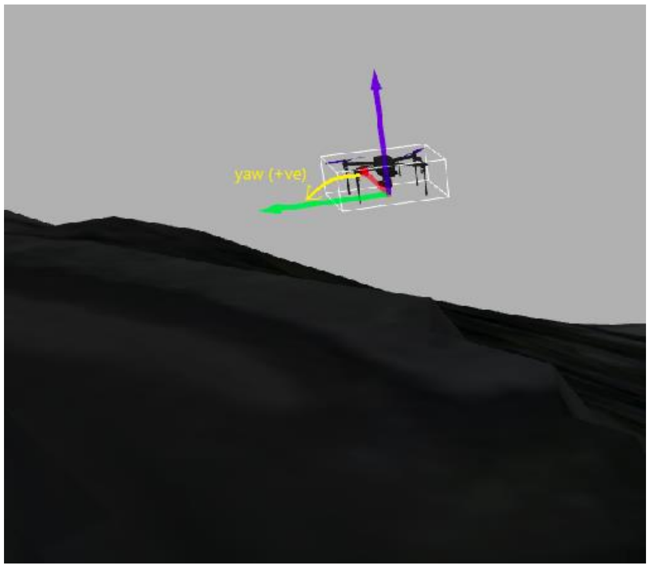


Image of the drone depicting its x, y and z and yaw angle nature in the local frame

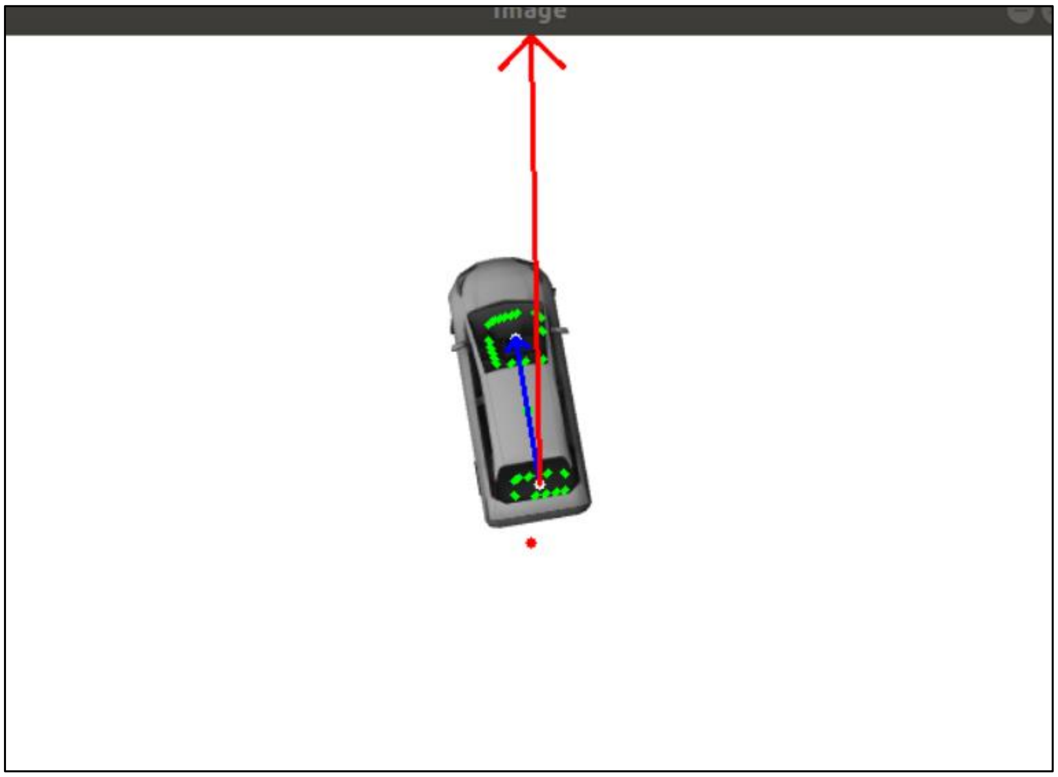
Through the aforementioned 4 controls in respective 4 degrees of freedom, the UAV traverses a path directly above the mid-point trajectory of the road and at a constant height

4. UGV Traversal

Devoid of any sensors, the UGV's traversal of the terrain depends entirely on the UAV's path and the positional feedback provided by it.

Positional Feedback

Positioned just above the Ground Vehicle, the UAV provides an RGB image which is used to detect the position of the UGV in the image frame in units of pixels. This is done via OpenCV, first masking out the UGV from the rest of the background. Since the UGV has very distinct visual features from its surroundings, it can be detected easily. Then finding out the centre of mass of the contour thus obtained, resulting in the exact coordinates of the UGV relative to the UAV.
The orientation of the vehicle is found through image processing on the top-view of the vehicle. It is done by detecting the centre of masses of the two windscreens on the vehicle and making a line joining the two.



Detecting pose and orientation of UGV through OpenCV

Path Planning

We have the coordinates and orientation of the UGV in the image frame of the UAV. A local waypoint towards a position that is trailing the UAV's position (which will coincide with the road's centre) is given to the UGV so that the UGV always follows the UAV. This is achieved by driving the vertical distance (in the image frame) between the UAV's position and the waypoint (dy) and the difference between the UGV's current heading angle and the drone's trajectory (which will be a vertical line in the image frame, denoted by θ) to 0.

Navigation

We are working with the image frame of the UAV. 4 commands are published to the UGV through the /prius topic which are explained below:

- Initialising the gear (shift_gears): Since the UGV is initially stationary with a gear value of 1 (Neutral), we change the gear to value 2 (Forward) so that the UGV can start its traversal.
- Accelerating the UGV (throttle): Using a P controller, the vertical distance between the UGV and the local waypoint (50 pixels below the centre of the image), dy, is diminished. This value is positive (with a maximum value of 1) when the UGV is lagging behind the waypoint and is 0 if the UGV has reached or crossed the waypoint.
- Decelerating the UGV (brake): Using the same controller as above, if the UGV overshoots the target (dy is negative), the throttle command is immediately set to 0 and a brake command is published. The value of the brake is calculated based on the extent of overshoot (with a maximum value of 1). If the car lags behind the local waypoint, the brake value is set to 0.
- Steering the UGV (steer): Using a PD controller, the difference between the UGV's current heading angle and the desired heading angle obtained through image processing (θ) is driven to 0.

Through the above controls, the UGV is made to follow the UAV as it moves ahead on the mid-point trajectory of the road.

5. Alternate Approaches

Using a 3rd-party Mapping package

We tried the more obvious approach of mapping the terrain using Rtabmap, Octomap and similar packages in the 1st phase, storing the generated map as a file, and then using this map for the navigation of the UAV (along with the UGV) in the 2nd phase (with the snow overlay).
However, this seemingly efficient approach has many flaws:

- Since the UGV doesn't have any sensors and feedback to the UGV can be provided only through the UAV's camera feed, the UAV would need to anyways fly directly above the UGV along the centre of the road during the traversal of the UGV. This would mean that the UAV travels a path, makes a map, and then travels the same path again, thus making the map, to a large extent, redundant.
- Using the above-mentioned mapping algorithms requires many tedious transforms between different local frames and the global frame.

Using Rosbag

After successfully navigating the UAV along the centreline of the road (on the image overlay version of the world), we recorded the UAV's odometry data and the commands published to it in a rosbag file. However, when we tried replaying the rosbag file on the plain overlay version of the world, we observed significant deviations in the UAV's path. After researching about the issue, we found that replaying a rosbag file has its inherent errors and since the environment also has some noise, it is not practical to use rosbag for solving the problem statement