Dynamic obstacle avoidance using Stereo Depth cameras onboard a Drone

Darshit Miteshkumar Desai
University of Maryland
College Park, Maryland
darshit@umd.edu

Shivam Sehgal
University of Maryland
College Park, Maryland
ssehgal7@umd.edu

Abstract

The paper describes the work done for the Final Project for the course of ENPM673-Perception for Autonomous Robots and ENAE788M-Hands On Autonomous Aerial Robotics on Dynamic obstacle avoidance and Hardware integration using Realsense Sensors like Lidar Camera, Stereo Camera on-board a MODAL AI M-500 drone. The paper describes the experimental testing, hardware integration process and methodology used on-board the drone to perform obstacle avoidance using the depth frames obtained from the stereo depth sensors. The paper at the end presents the controller architecture used to perform reactive avoidance and following of the obstacle based on the position of the obstacle in space. Many improvements on the existing code are needed to account for multiple obstacles in the frame and for tracking the obstacles.

1. Introduction

Drones and micro aerial vehicles are increasingly being used in environments that are highly cluttered and contain dynamic obstacles like people, machines, or other aerial vehicles in both indoor and outdoor environments. One of the main parameters affecting the optimality of an aerial robot avoiding an obstacle is Depth. Depth estimation onboard an autonomous vehicle can be used for various purposes like obstacle avoidance, 3d mapping, SLAM, etc. Heavy and expensive sensors like Lidar cannot be integrated with small form factor drones for the applications mentioned above. Hence, for this application stereo cameras can be used to accomplish this task. The importance of this task is evident in cluttered indoor or outdoor environments like construction sites, forests, etc where the drone has to rapidly plan and avoid dynamic and static obstacles.

This project initially aimed to develop a real-time stereo depth estimation system using two sets of cameras mounted on a drone. But after evaluation and the testing of sensors described in the upcoming sections, it was finally decided to move on to better sensors. The overall system architecture remains the same, in which a depth image is evaluated based on the U-V depth disparity calculation method mentioned in the literature here [1].

The paper is structured in the following manner, the next section called Hardware Integration Process, describes the hardware integration process currently used on the drone to achieve dynamic obstacle avoidance. In the subsections, the review of other sensors used like the Intel Realsense L515 and the Onboard Stereo Cameras are reviewed and discussed whether why they were not chosen for the task of obstacle avoidance. In the other subsections, the various configurations of connecting an external image sensor with the drone are described and the problems encountered with it.

The following section after that describes the structure of the algorithm used to segment an obstacle from the depth image, the ROS-node structure used to control the drone and the problems encountered during the testing of the said algorithm onboard the drone and on the ros bagged data. In the same section, the solutions to the problems are also described and what are the possible future improvements which could be done to improve this algorithm.

2. Hardware Integration of the Depth Sensor with the Modal AI M500 Drone:

The hardware architecture of the Drone and the integration of the Depth sensor with it were one of the difficult challenges of this project. Since MODAL AI and Realsense are one of the top trending hardware used in obstacle avoidance and flight autonomy and no one has yet integrated both platforms successfully yet, the team has decided to detail it's experience and the method used to integrate the hardware with the flight computer in this section.

2.1. Hardware Overview:

An overview of the current setup is provided in the images here 1 and 2

The Realsense D435i camera is used in the final setup described above. This is one of the cutting-edge stereo cam-

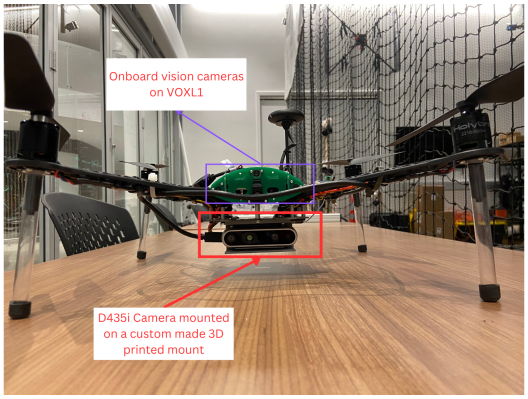

Figure 1. Front view of the drone, the Realsense D435i camera mounted on a 3D printed mount can be seen in the view along with the VOXL 1 stereo cameras

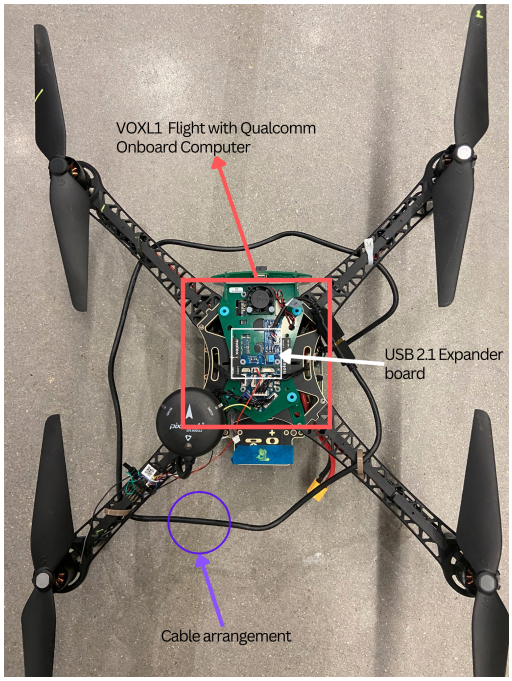

Figure 2. Top View of the drone. USB 2.1 V2 Expansion Board connected with the D435i, VOXL1 flight controller; the cable arrangement is done in a diamond shape to reduce magnetic interference during image transport

eras available in the market and has wide-ranging capabilities. The camera consists of a rgb module, 2 vision cameras, an IR projector, and an onboard Inertial Measurement Unit(IMU) to measure the motion of the camera. The camera has an onboard vision processor which processes the image frames captured by the cameras and gives outputs like Depth, aligned depth etc which can be accessed through an SDK coupled with a ROS wrapper in the form of rostopics. The unique feature which differentiates D435i from other stereo camera pairs is the availability of an IR projector which enables depth detection on featureless backgrounds and hence enhances the capability to differentiate between obstacles and plain backgrounds.

The VOXL1 flight computer provides a facility to integrate high-load cameras or sensors to be integrated with the flight controller such that the topics and the data captured by the external vision sensors can be integrated for flight autonomy tasks. There are two particular expansion boards that the team used to integrate and test the D435i and other Realsense Sensors. The Expansion board V1 and the Expansion board V2. The Expansion board V1 provides a 500 mA load output with a USB 2.1 configuration to support devices over a USB Female adapter. The team tested initially and was able to get the depth data. But since it only has a 500 mA output the sensor is not able to give the best performance all the time. The next expansion board V2 gave 2 Amperes of unrestricted output which could easily support the power-heavy requirements of a D435i Vision Processor and Camera components.

There were various other configurations that were tried for connecting the external vision sensors with the VOXL 1 computer. They are documented in the upcoming subsections. The next section provides a brief tutorial on installing the SDK of a Realsense Device on an ARM64 Edge computer.

2.2. Installation Procedure of the Realsense SDK:

ARM64-based chips are ubiquitous in edge computers and companion computers because they provide an advantage in the amount of power consumed and the computational speed over conventional computers. The VOXL 1 Flight computer is a Qualcomm SnapDragon-based computer with ARM64 architecture. The VOXL software architecture provides the user the capability to load and install Linux Ubuntu Docker containers. Since the VOXL has limited memory the original docker image with the Realsense SDK library and ROS Wrapper was built on a Macbook Air M1 computer which also has a ARM64 chip.

Ideally, to make the installation procedure straightforward, it is better to move the docker containers into an SD card on the VOXL computer which would provide sufficient memory to install the SDK. The procedure to move the Docker container to the SD card onboard a VOXL 1 computer is detailed here. Note before starting the installation process for the SDK, ensure that ROS-NOETIC is installed on board the docker container.

Once the docker image is ready, follow the instructions as given on the github readme file here, ensure that a *sudo aptget update* is performed before running the bash script. The bash script basically modifies the kernel to install the USB drivers required for connecting with the Realsense hardware, it also downloads the SDK which is re-

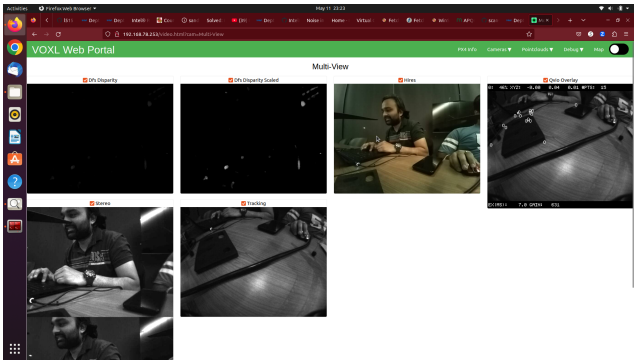

Figure 3. Disparity images outputted from the VOXL MPAs

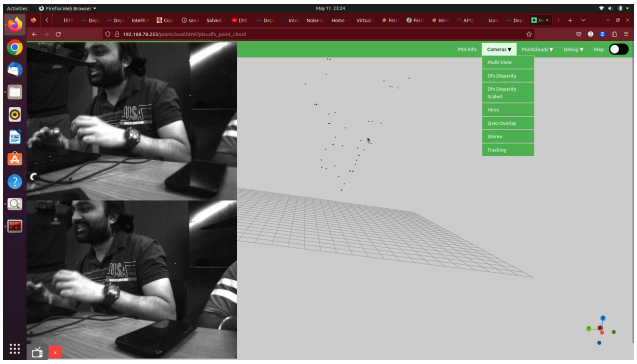

Figure 4. Point cloud data outputted by VOXL onboard stereo

sponsible for communicating with the vision processor and publishing image pipelines. At the end of the bash script, it makes the complete SDK and saves it in Ubuntu OS.

If by any chance there is no hardware to test the SDK installation use the Issue link mentioned here to test out the SDK and ROS-Wrapper installation using a bag file.

2.3. Testing out other sensors and hardware configurations:

During the course of the project, the team tested out various sensors available onboard the VOXL computer as well.

2.3.1 VOXL 1 Stereo Cameras:

The camera calibration was a big issue with the VOXL 1 stereo cameras. There were two issues faced, one was the failure of the intrinsic parameter calibration other was the failure of calibrating the extrinsics. The failure of the stereo cameras is well documented in the MODAL AI forum link posted by one of the team members here. The below image shows one of the examples where the OEM calibration failed for both intrinsic and extrinsic parameters of the camera. Even after calibrating the stereo cameras with the Open CV calibration method the accuracy of the generation of the disparity maps and the point clouds was not up to the mark. The specimen images are displayed here 3

2.3.2 Scanse SweepScan Lidar

he team interfaced a SCANSE Sweepscan LIDAR to get the point cloud of the points scanned by the LIDAR. The repository of the SDK used for doing this is on this link. The team tested the LIDAR and found that this might be unreliable considering the drone has all 3 DOF available to avoid obstacles. The Sample data image from the 2D lidar is given here 5

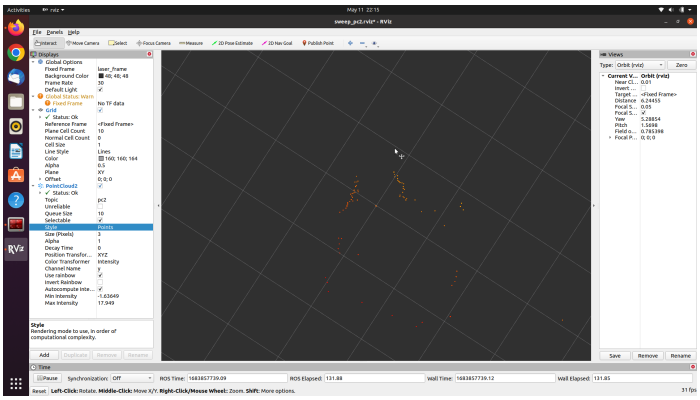

Figure 5. Point cloud data outputted by VOXL onboard stereo

2.3.3 L515 Lidar Camera

The L515 Camera is a Realsense Device which is a solid-state MEMS-based IR laser Lidar that projects beams in a projected area and retrieves a 3d point cloud. It has an onboard RGB sensor with IMU and it also gives depth data in the form of depth images. The only disadvantage of this Lidar is the incompatibility with the USB 2.0 architecture and publishing topics on the Realsense ROS-Wrapper as it fails to establish a pipeline on that protocol. The team did successfully connect a USB3.2 TX-RX swapped cable on the VOXL's ADB port but since the cable length combined with the Intel Realsense cable was greater than 2 meters it was ruled out at the end.(Also the Realsense cable cannot be looped because magnetic interference caused the device to get disconnected). The image of the setup is given below in Figure 7 and the image output is shown in the realsense-viewer in Figure 6

3. Obstacle detection and Avoidance

3.1. Process Flow

The figure 8 gives a high-level view of our approach for detecting and avoiding obstacles. In the sections to come

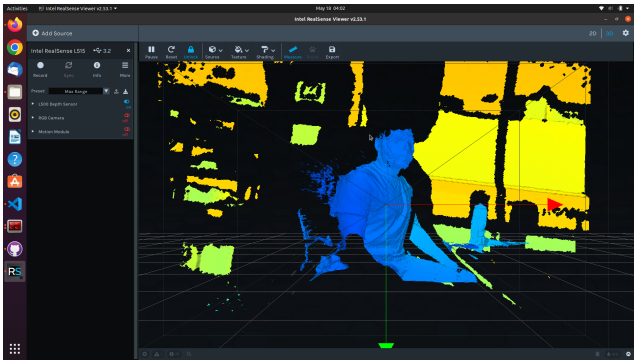

Figure 6. Point cloud data outputted by L515 Solid State Lidar

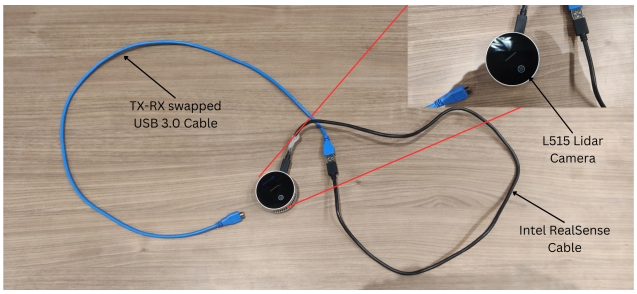

Figure 7. The setup required for connecting the L515 Lidar camera to a VOXL1 computer. The cable can be estimated to be of 2 mtrs

we will be explaining each part of our process.

3.2. U-Depth Map

To generate U-Depth map from a depth map we need to compute the column depth histogram of the depth image. Each row of the U-depth map is a depth bin for a range of depths. So let's say we have depth bins of size 10 then the $10^{(th)}$ row would have all the depth between range 100mm and 110mm. This is just an example,for different use cases different sizes of bins can be defined.

See figure 9 and figure 10, which shows the U-depth map for a chair. When an obstacle is in the front of the camera the size of the corresponding bin becomes larger. Using this characteristic of the U-depth map a bin of histogram can be considered as a point of interest if its value if greater than $T_{(}poi)$:

$$T_{poi} = \frac{f_{Tho}}{d_{bin}} \tag{1}$$

This can be used to detect multiple obstacles, in our code we used findcontours function from openCV to make a bounding box on a area in U-depth map with maximum intensities.

3.3. Obstacle Pose Detection

The obstacle is defined as 3-dimensional box and let $P^w{}_o = (x^w{}_o, y^w{}_o, z^w{}_o)^T$ be the position of the center of

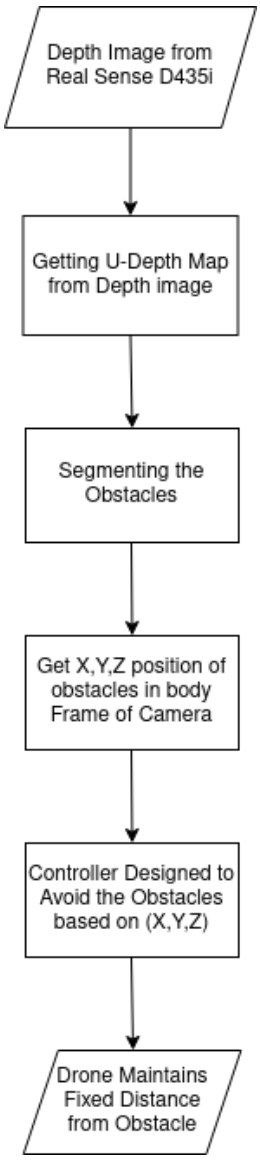

Figure 8. Process flow for obstacle detecting and avoidance

the box, W is the world frame and B is body frame. The dimensions of the box is define by $s^w{}_o = (l^w{}_o, w^w{}_o, h^w{}_o)^T$. Using the bounding box we formed in the U-depth map as shown in Figure 10. The upper left corner is defined by pixel values (u_l,d_1), and the bottom right corner (u_r,d_r) we can use this information to get the x,y position as well as the length/thickness of the object as follows:

$$x_o{}^B = d_b, y_o{}^B = \frac{(u_l + u_r)db}{2f} \tag{2}$$

$$l_o{}^B = 2(d_b - d_t), w_o{}^B = \frac{(u_r - u_l)db}{f} \tag{3}$$

Then we can further find the bounding box for the depth image. For this we need to group the depth image points

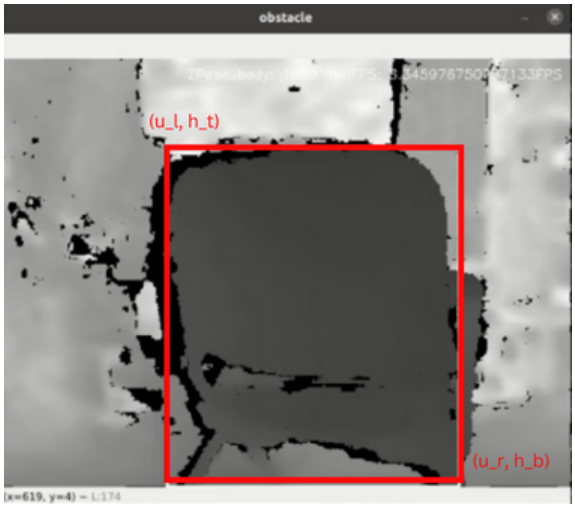

Figure 9. U-depth Map

Figure 10. U-depth Map

within the columns index [u_l,u_r] and depth values with the range [d_t,d_t+l_o]. Doing this we get the upper left corner of the bounding box (u_l,h_t) and (u_r,h_b) be the bottom right corner. Using this we can compute the vertical position of obstacle z_{oB} and height as:

$$z_{oB} = \frac{(h_t + h_b)d_b}{2f}, h_{oB} = \frac{(h_t - h_b)d_b}{2f} \quad (4)$$

3.4. Rotating the Frames

All these equations are considering all the pixel values extracted with respect to the center of the image. Now we have the position of the obstacle in the body frame(camera) as ${P_o}^B$=(${x_o}^B$,${y_o}^B$,${z_o}^B$). To get the position of the obstacle in the drone's body frame we had to rotate these positions by 180 degrees with the x-axis and then 90 degrees with the z-axis. Refer to figure 11 for a better understanding of the drone frames and camera frames.

3.5. Controller

For our project, we have designed 2 controllers one to stop at a fixed distance from the obstacle and another to servo a fixed relative distance from the obstacle. Refer figure12 and figure 13

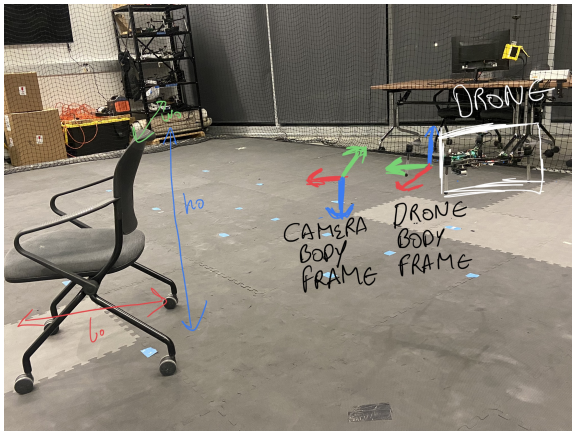

Figure 11. U-depth Map

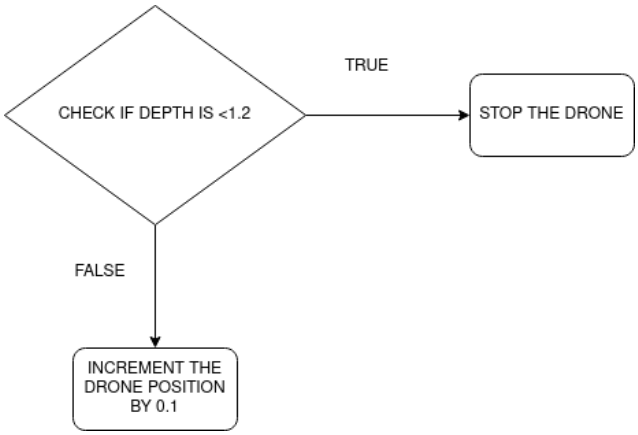

Figure 12. Controller to stop at a fixed distance from obstacle

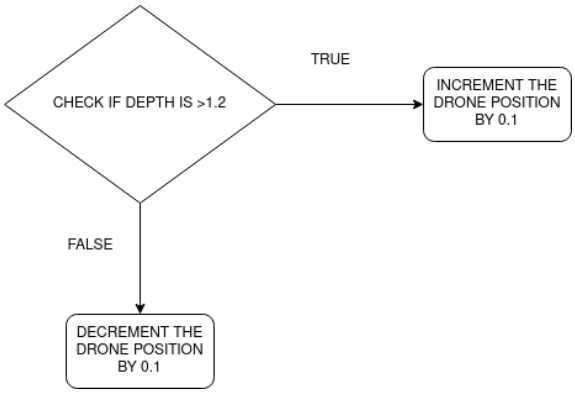

Figure 13. Controller to Keep a relative distance from Obstacle

4. Software

Ros noetic middle-ware is used to communicate between the camera, offboard computer and the drone's onboard controller. The drone uses a PX4 controller which uses mavlink protocols, mavros provides a bridge be-

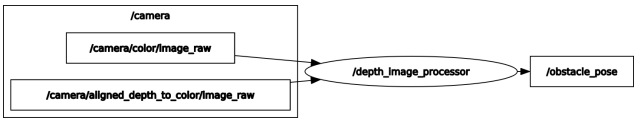

Figure 14. Image processing node run on the onboard computer

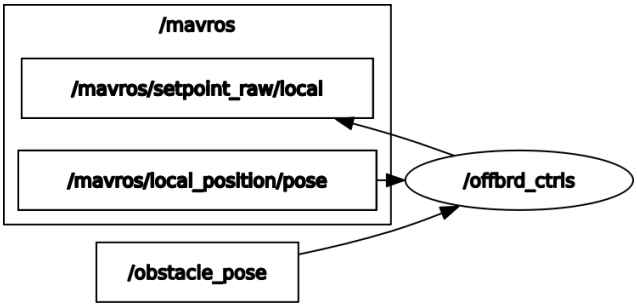

Figure 15. Offboard control node run on the drone's onboard system

tween mavlink and ROS. The camera is connected to the drone onboard system from which a is launched which publishes depth topics at 60Hz. From our offboard computer over the same ROS MASTER we launch the nodes($depth\_image\_proceesor$) that subscribe to the topics ($camera/aligned\_depth/image\_raw$). The image processing is done on the offboard system then publishes the obstacle position at 5Hz on the topic ($obstacle\_pose$). The offboard controls node is launched in the onboard system which subscribes to the obstacle position topics, these are processed and control commands are published to mavros which then communicates with PX4 to move the drone.
A combination of Python and C++ was used to program the codes.

5. Result and Conclusions

We were able to detect the obstacle, get it's center position in the drone body frame and avoid obstacle using the two methods as explained above. The figure 16 Our goal was to move the drone in dynamic environment but due to issues with the hardware and time constraints we were not able to finish it. In conclusion we found that the U-depth is robust method to detect obstacle, if couple with Kalman filter detection and avoidance of dynamic obstacles can be performed.Refer [1]. The final code on Github and the demonstration videos are linked in here. Videos Link, Github Link

6. Acknowledgement

We thank Professor Joseph Conroy and Ivan Penskiy for their continuous cooperation and support in achieving the results for this project and providing the necessary hardware required for testing out our algorithm onboard the drone. We also would like to thank the open-source community who provided us with the necessary help on the issues with the integration of the realsense hardware onboard the drone.

Figure 16. Point cloud data outputted by VOXL onboard stereo

References

[1] Jiahao Lin, Hai Zhu, and Javier Alonso-Mora. Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments, 2020. Face and Gesture submission ID 324. Supplied as supplemental material. 1, 6