Final Report - Part III: System Performance

CSC Students
Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27607

ECE Students
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, North Carolina 27607

Civil Students
Department of Civil Engineering
North Carolina State University
Raleigh, North Carolina 27607

I. Introduction

In this report we present a summary of results as well as an analysis of failures observed during our demonstrations. An emphasis is placed on graphics to provide a quick visual understanding of the performance and highlight where systems failed.

II. Summary of Results

The images and videos captured on the day of demo can be found here.

A. Demo I

1) Husky: The demo I consisted of:
1. Husky navigating autonomously in an indoor environment. The Husky used the input from LIDAR alone to accomplish this task.

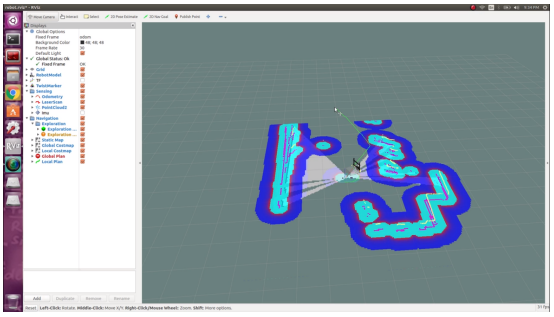


Fig. 1. Lidar map

2. There was also blimp tracking implemented on the husky. However, the markers used were color markers.

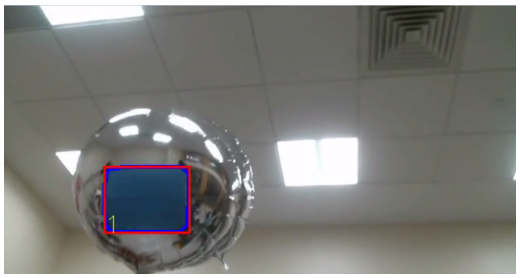


Fig. 2. Blimp tracking

2) Blimp Indoor: For the first demo the Indoor blimp team finalized on various aspects of the blimp design like the envelope, payload carrier and integration of all the sensors and actuators with the Raspberry pi computer. In the demo the goal was to navigate the blimp using simple motion commands like moving forward, backward and up-down motion. Along with this compatibility issues for integration of visual and IMU data with SLAM as well as CA team were successfully resolved.

3) Blimp Outdoor: For the first demo the out door blimp team decided to complete the envelope and payload carrier along with all required sensors and motors except pan and tilt mechanism. The main goal for this demo was to navigate the blimp using remote commands and gather sensors data.

4) SLAM A: In this demo, ORB SLAM2 algorithm was successfully implemented on Jetson for both Stereo (Zed) and Monocular (Raspberry Pi Camera) cameras. Both the calibrated cameras were used to stream the live video feed, which was subscribed by the Jetson. The output of the code was the point cloud and the localized co-ordinates of the robot(Husky/Blimp) which were published for use by the Context Awareness teams.

5) Context Awareness - A: In the first demo we decided to experiment with different segmentation algorithms. We had three Context awareness teams and each team decided to try a different segmentation algorithm. We chose SegNet for doing semantic segmentation. For the training of SegNet we used the NVIDIA Deep Learning GPU Training System (DIGITS) platform. The NVIDIA Deep Learning GPU Training System (DIGITS) puts the power of deep learning into the hands of engineers and data scientists. DIGITS can be used to rapidly train the highly accurate deep neural network (DNNs) for image classification, segmentation and object detection tasks. DIGITS simplify common deep learning tasks such as managing data, designing and training neural networks on multi-GPU systems, monitoring performance in real time with advanced visualizations, and selecting the best performing model from the results browser for deployment. DIGITS is completely interactive so that data scientists can focus on designing and training networks rather than programming and debugging. Using DIGITS we trained the SegNet model with the CityScapes open source dataset. We deployed the pretrained model of the SegNet on Jetson TX1. Inferencing was done on Jetson and we achieved 0.5 fps rate.

6) SLAM B: The primary aims of this demo were to demonstrate environmental mapping through the generation of

a point cloud and occupancy grid using real time video data from the ZED stereo camera using rtabmap. The output of this demo was the visualization of the occupancy grid and point cloud on the Jetson using rtab-map.

7) Context Awareness - B:

a) We got FCN alexnet working on our Jetson. This required the installation of caffe. We were also able to subscribe to topics published my blimps (though we did not use them in our first demonstration). We were able to show segmentation with preloaded images on Jetson.

8) Context Awareness - C: We were trying to implement transfer learning algorithms to precisely identify the objects present in CFL site. We used a pre-trained model, subscribed to images sent by outdoor blimp and tried drawing bounding boxes around the identified objects. Figure 3 shows an illustration of the same.

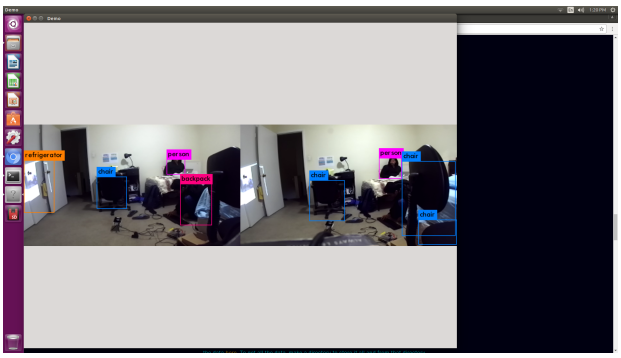


Fig. 3. Object detection and recognition with pre-trained model

B. Demo II

1) Husky: The demo II consisted of:

1. Husky navigating autonomously in the construction environment. The Husky used the input from LIDAR alone to accomplish this task.
2. The blimp tracking was implemented on the husky using whycon markers.

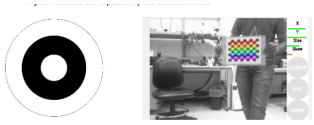


Fig. 4. Blimp tracking using whycon marker

2) Blimp Indoor: For the second demo the goal was to solve the issues that we faced during the first demo, optimize certain aspect to improve the performance and carry out some new implementation. One of the major issue that we faced was too many wires that added to the complexity in handling and untidy appearance of the blimp. Hence for the second demo the first goal was to design, test and successfully implement a PCB. With this we had a compact implementation of hardware within the gandola. The next task to optimise was the envelope. For the first demo multiple envelopes were stacked to carry

Fig. 5. VINS Mono output-Demo 2

the payload. In second demo we selected envelopes where the payload was carried by two envelopes only. The third goal achieved was to optimize the navigation control program in ROS. Finally, one of the new implementation that was incorporated was the PAN-TILT mechanism.

3) Blimp Outdoor: The goal in the second demo was to attach pan and tilt mechanism and camera to the payload carrier and publish the sensors data using ROS. In addition to this goal, Outdoor blimp team wanted to test the control system and fix the previous issues with the envelope.

4) SLAM A: The VINS Mono SLAM algorithm was implemented on Jetson TX1 which used live data from the camera and IMU streamed by the blimp teams. The output of this algorithm were the current localization of the blimps and the current and history point clouds which would be used by the Context Awareness teams.

5) SLAM B: We were able to integrate with other teams (primarily the Husky team) to publish and subscribe all the zed camera topic and rtabmap topic which include left and right camera images, odometry data from zed camera and occupancy grid and point cloud data from rtabmap. Our Jetson was the ros-master and was publishing all the data. An example image of the output at this stage was the output seen in Figure 6 where the Husky was manually guided through an indoor hallway while mapping the environment.

6) Context Awareness - A: In the second demo we decided to improve the real time segmentation rate and so we decided to implement ENet model for segmentation purposes. We used Caffe Deep learning frame work to implement ENet model. For our task which was to perform semantic segmentation at a construction site we did not have enough annotated images for the training purpose. e gathered 1000 images from the construction facilities testing laboratory and annotated those images using the Supervisely online tool. We annotated those images with three classes namely, sky, obstacles and ground. We trained the ENet model on a server having a Xeon E5, 128GB ram, as the CPU and Tesla k40c and Tesla k20c with 12GB VRAM as GPUs. The Docker container was used to train the ENet model on the server.

Docker is a computer program that performs operating-system-level virtualization also known as containerization. We used Docker virtualization tool to overcome different version

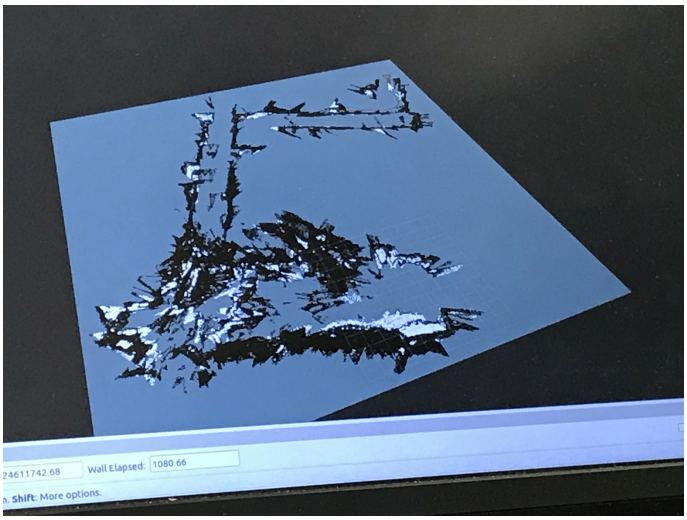


Fig. 6. SLAM B output of the visualization of traversing a hallway. The hallways are correctly mapped as linear obstacles labeled black which indicates an obstacle.

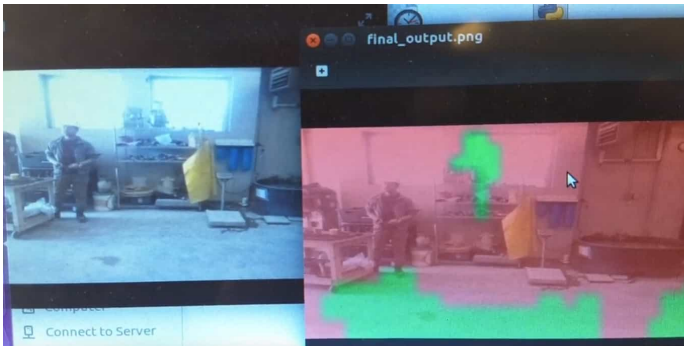


Fig. 7. Segmented Image

interdependencies while training on the server. We used the trained model weights on Jetson to carry out the inference process.

We used Enet with python initially and found that then inferencing speed obtained was around 0.5 fps. This was like what we obtained from the SegNet model. So, we decided to use ENet with C++. We found that our speed increased from 0.5 to 7 fps. The ENet model was able to achieve segmentation speed of about 150 milliseconds per frame.

7) Context Awareness - B: a) We were able to integrate with Indoor Blimp and subscribe to their topics in real time. Further we used their image and successfully segmented it into ground and obstacles. We were also able to publish both original image and segmented image back. The three step process was working as a single pipelined step which we failed to achieve in Demo 1. A sample segmentation output from Demo 2 can be seen illustrated in Figure 7 above.

8) Context Awareness - C: We setup semantic segmentation module on Jetson TX! with Nvidia digits on laptop. We tried various tensorflow based segmentation modules. They all failed to build on TX1. Finally we were successful in setting up FCN Alexnet on digits. For the demo, we trained the model

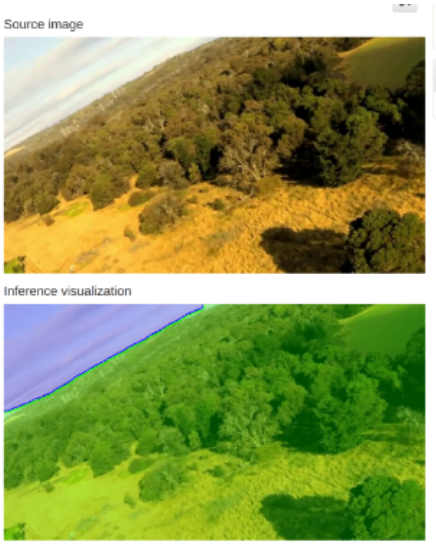


Fig. 8. Sample segmented image

with 300 images and tried segmenting the image published by outdoor blimp. Figure 8 shows the inferencing of our model on Jetson.

C. Demo III

1) Husky: The Husky demo resolved the key issues faced during the previous demo:

1) Added tilt mechanism to track blimp along with using wide angle lens.The video for same can be found here for tilt mechanism here.
2) Created mounting for the tilt mechanism.
3) The map used by Husky to navigate was provided by the input from the zed camera.
4) The map was updated real time and hence, the navigation was smoother. The video of the Husky navigating autonomously can be found here:Indoor Navigation 1 Indoor Navigation 2 and Outdoor Navigation
5) The networking issue was resolved using a router with better signal strength and range.

2) SLAM A: The VINS Mono SLAM algorithm was successfully implemented on the Jetson. The live video feed published by the raspberry pi cameras on the Blimps and the IMU data were used by the code to find and track key points. These key points were stored to form the point cloud which was published and later used by Context Awareness teams. The output can be visualized in Figure 10 and 11. Also, the demonstration can be viewed here and here.

3) SLAM B: The key aims of the final demo were to demonstrate the autonomous navigation of the Husky using the SLAM output generated by our module while also visualizing the generation of the map using rviz. Figure 12 shows the generated occupancy grid ($octomap\_grid$) of the limited construction site area which was employed in the autonomous navigation of the Husky. Performing a comparison between the actual and generated trajectory yields the insight that whenever

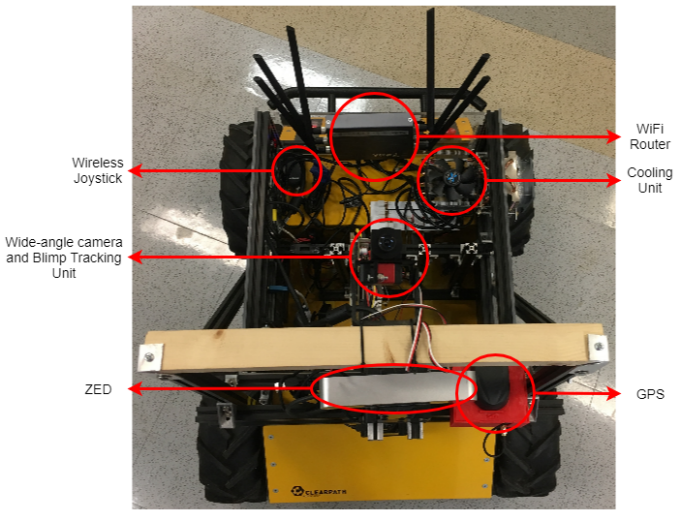


Fig. 9. Top view of Husky prepared for Demo III

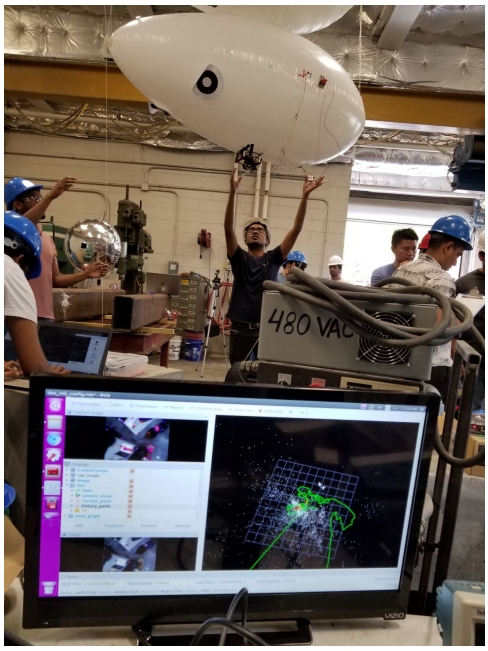


Fig. 10. VINS Mono Final Live Demo

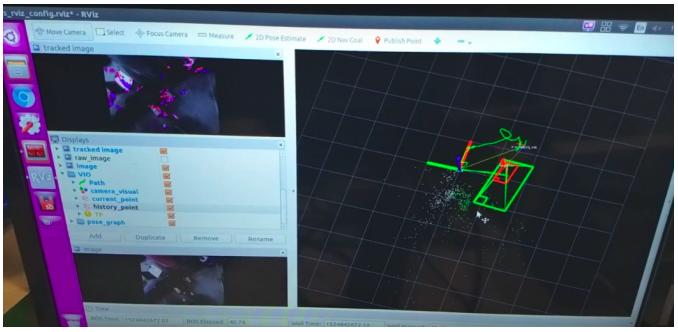


Fig. 11. VINS Mono RVIZ Final Demo with Indoor Blimp

the Husky remains in the same place it is difficult to update the map through deletion of older data points in the map with the updated more recent data. For instance, if a human passes briefly in front of the Husky, the Husky has a hard time deleting the ”obstacle” of the person after they have passed. This effect is exacerbated by inactivity of the Husky as we have found that if the Husky is moving at a fairly rapid speed the received data is more precise and accurate in the case of objects temporarily passing in front of the sensors.

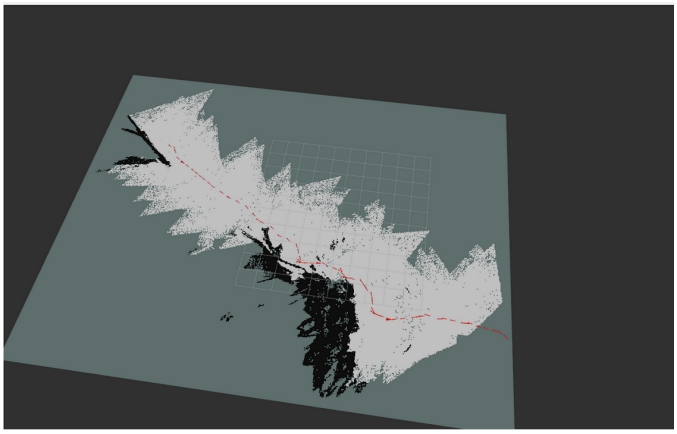


Fig. 12. Generated occupancy grid of the demo construction site at CFL. The obstacles to the left of the start location as seen in Figure 13 are clearly labeled as obstacles.

4) Context Awareness - A: For the Third Demo we decided to concentrate on integration with other teams. We used ROS kinetic for this purpose. We subscribed to ROS image topics published from the ZED camera by the SLAM-B team. We subscribed to the left camera image topic and carried out the segmentation on that stream of images. We publish our segmented images and the corresponding depth images as an image stream using ROS so that the occupancy grid can be obtained by SLAM-B team. Though segmentation was done at 7 fps it was published at 1 fps because of latency in ROS network.

5) Blimp Indoor: : For the demo III the goal was to perform a couple of new implementation.

The first goal was to design and implement the closed loop Yaw control. The second goal was to incorporate the obstacle detection in the open loop velocity control program. The third goal was to design a voltage regulation system that could allow us to use two small lipo batteries in series and get rid of the heavier battery pack of 5V. Finally, integration with the SLAM and CA team was carried out. For integration with SLAM and CA the PAN-TILT mechanism was removed from the system as the teams demanded a fixed position of the camera, also the IMU was fixed with the desired orientation with respect to the camera for efficient functioning of SLAM algorithm.

6) Blimp Outdoor: The goal in the final demo was to autonomously rotate the blimp and camera to the required direction provided by slam and context awareness teams. Besides, it was necessary to develop a reliable way for the wiring, therefore the PCB was developed.

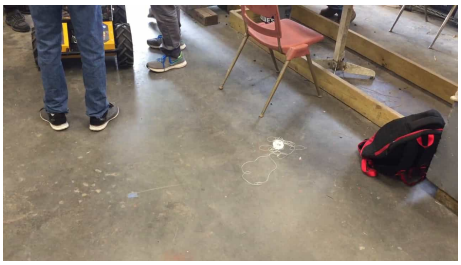
Fig. 13. Outdoor blimp on top of Husky.

7) Context B: For the final demo we were able to accomplish the following major tasks:

- Real Time Image Segmentation: The trained model was able to perform segmentation on an unsupervised image sequence with reasonably good accuracy for a good number of frames. One of the illustrations of the above can be seen from the figures below. There were some instances where the model was not able to come up with a proper segmentation of the perceived scene which was primarily due to two major reasons. Firstly, the training image sequences were collected at a different angle compared to the angle at which the blimp was streaming the image sequences. Secondly, the size of the image sequences published by blimp were way too small in regard to the actual image sizes used for training. The other marked difference we observed was the better results obtained for most of the images sequences published by blimp. This was attributed to the increased number of epochs used while training the model using DIGITs platform. A sample segmentation result from the trained model can be seen in Figure 14 where the actual ground truth and segmentation result obtained by running the same model on DIGITs platform has also been put to serve as reference for model comparison by running on different platforms.
- Image to Occupancy Grid conversion: The other significant contribution was the ability to project the segmented image onto a grid map that can be used by the Husky to identify the spatial features in the environment. One of the missing items in the failure to incorporate the depth information from the SLAM team. This was because of the visible time lag in segmentation task in comparison to the SLAM's localization technique. In addition to us being able to generate the map, we were not able to update the map with its past history of previous maps and localization. One disadvantage of this is the inability to paint a whole picture of the environment right from the starting point. Figure 15 illustrates a sample 2D occupancy grid obtained from the segmented image. Each pixel represents a safe or unsafe zone and has a resolution of about 10 centimeters per pixel.

D. Context Awareness - C

For the final demo, we integrated the I/O through ROS with Outdoor blimp. The outdoor blimp published the image that is seen at CFL site. Our trained model segmented out the image with decent accuracy. One of the segmented images can be seen in figure 16. To improve the speed of the segmentation pipeline, we dropped two out of every three frames. This could increase the segmentation to almost real time. At few instances, the segmentation didn't give out good results. According to our analysis, our dataset had to be more diverse. More details about this failure is described in section III of the report. Along with just segmenting out the image, we also projected the results in meaningful way on rviz for husky to use this data and navigate. We converted the segmented output to binary form and projected the same on rviz. One sample output is shown in figure 18. We also published the same for the blimp/husky to use this information to navigate.



(a) Image captured from blimp setup

(b) Ground truth generated for the captured image

(c) Segmented image generated by DIGITs platform

(d) Segmented image obtained by inference model running on Jetson TX1

Fig. 14. An illustration of real time segmentation output obtained from Jetson. The ground truth and DIGITS output have been added for comparative evaluation.

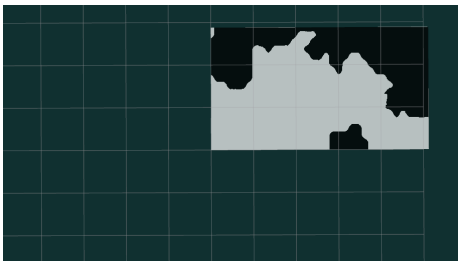


Fig. 15. A 2D occupancy grid map for the segmented image. The center of the map corresponds to (0, 0, 0)

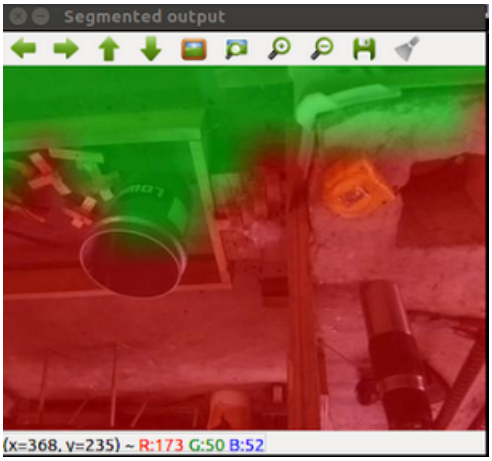


Fig. 16. Segmented output

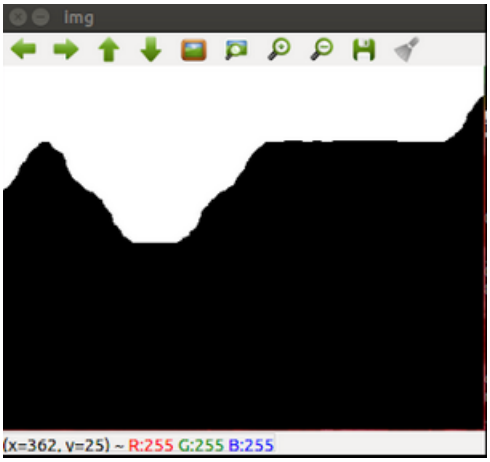


Fig. 17. Binary output

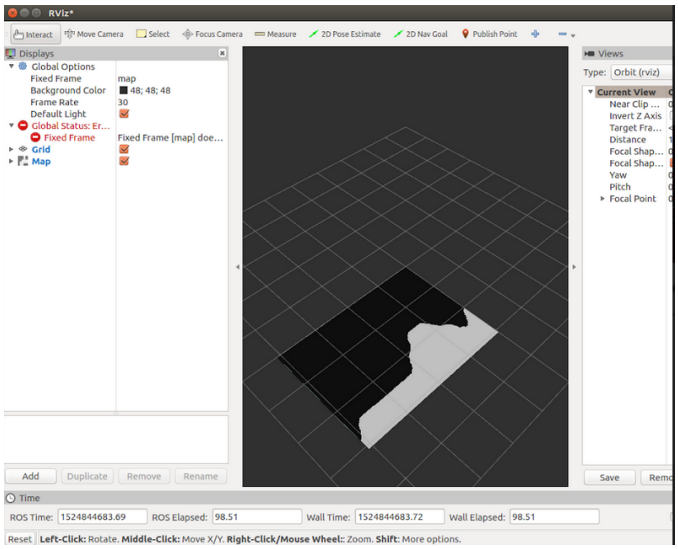


Fig. 18. projection on rviz

III. Analysis of Failures

A. Demo I

1) Husky: The below issues were noted as failures as a part of Demo I:

1) Inability to track the blimp using web-camera: The range of the blimp was not completely covered with the web-camera which was low in resolution as well as had to Pan/Tilt unit to track long ranges.

2) Tracking algorithm causing false detections: The tracking algorithm using the color markers caused many false detections. Hence, it turned out to be an unreliable mechanism to detect the blimp. Figure 19 demonstrates that multiple blimps were detected despite having just 1 blimp in the FOV.

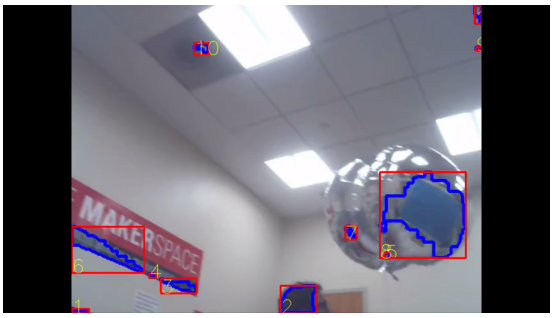


Fig. 19. False Detection using color algorithm

3) GPS driver issue: The GPS purchased was incompatible with the version of Linux installed on Jetson. The cp210x drivers for GPS module are not installed by default in NVIDIA Jetson TX1. Kernel has to be re-configured in order to install the drivers. Changing the kernel would mean all the current packages would need to be re-installed and hence, this issue was parked until a better solution was found.

2) Indoor Blimp:

Following are the issues faced by Indoor blimp during the Demo I followed by its analysis:

1. Processing capability of Raspberry pi zero:
Compatibility issue with ROS for video streaming while using raspberry pi zero.
Analysis:
With a point of view of having a small payload, raspberry pi zero W was selected as the controller. Considering the ease of integration, ROS was decided as the common platform for all teams. The controller was suppose to handle ROS and stream the data to SLAM and CA team concurrently over WiFi. However it was seen that the selected controller was not able to handle such concurrent processing and resulted in compatibility issues with ROS that made the controller to fail multiple times. It was therefore decided to use raspberry pi 3 controller but with additional burden for the payload.

2. Payload and weight constraints:
Stacking of multiple envelopes during the demo.
Analysis:
The necessity to switch from raspberry pi zero to raspberry pi 3 increased the payload substantially. Also, unforeseen weight due to wires and velcro increased the weight. The increased payload forced the team to stack multiple envelopes to support the weights of all components.

3. Inaccurate and unstable navigation of blimp:
Actual blimp movement was not as per the input command given. For example, given a command to move straight, the blimp movement would be at an angle and was unstable.
Analysis:
The stacking of multiple balloons and slight misalignment in the placement of rotors caused the inaccurate movement.

3) Outdoor Blimp:

The issues below outline the failures of the outdoor blimp during Demo I:

1. The processing capabilities of the Raspberry Pi Zero. The microcontroller was chosen due to its low weight and processing ability, but the Raspberry Pi Zero could not handle running ROS and streaming images concurrently. This deficiency led to the implementation of a Raspberry Pi 3 as an extra microcontroller to obtain images for the system.

2. Payload and weight constraints. One of the biggest issues was ensuring that the envelope could carry the payload. Even with weight calculations, extra grams would be added due to unforeseen items such as jumper wires to connect the systems and Velcro to hold the payload carrier to the envelope. These issues caused the addition of extra mylar balloons to lift the system.

4) SLAM A:

a) Long time for initialization of ORB SLAM2 for monocular camera on both the blimps (approximately 5 minutes).
Analysis:
ORB SALM2 uses Oriented, Fast and Rotated Brief features for tracking, which takes a long time for initialization owing to the large amount of processing required at the back-end.

b) Unable to detect key-points during fast transitions
Analysis:
ORB SLAM2 requires smooth transition from frame to frame for detecting new key points and maintaining the previous track history. Rapid changes in the view when the blimp moves was a major issue for ORB SLAM2 to run in real-time.

5) SLAM B:

a) We first attempted to implement stereo slam using the ORB SLAM2 framework and installed it onto the Jetson but we ran into compatibility issues when attempting to make it work. This lead us to investigate other frameworks which lead to the implementation of RTAB-MAP on our Jetson.

b) With the limited storage on the internal drive of the Jetson we were unable to install both ROS-kinetic and RTAB-MAP at the same time. This critical error necessitated the use of an external drive to ensure functionality.

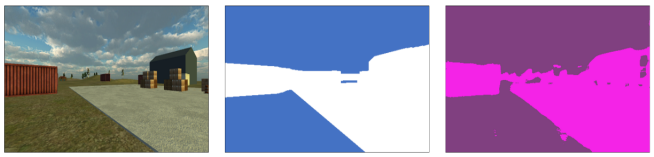


Fig. 20. Segmentation done with Recurrent Convolutional Neural Network for Scene Labeling on host computer

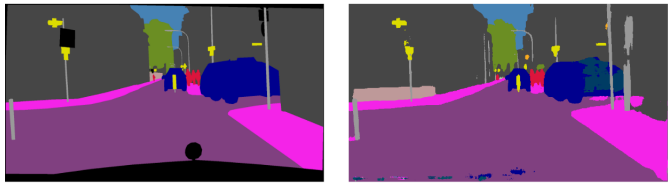


Fig. 21. Segmentation done with ERFnet on host computer.

c) Unable to determine a method to make *rtabmap* subscribe to zed camera published data and use it. The primary cause of this failure was the limited time spent using RTAB-MAP at this demo due to the relatively quick transition to this framework from the initially selected ORB SLAM2 framework.

6) Context Awareness - A: For the first demo we used SegNet for the segmentation purpose. output of the SegNet was 0.5 FPS which was really slow and we were not able train it properly using the available dataset. So, we used cityscape data for the training purpose. we also were not able to subscribe to the images published by SLAM-B team through ROS, so we used Jetson's on board camera for the segmentation purpose.

7) Context-Awareness B:

- Setting up different frameworks on Jetson was challenging: Searching for different Algorithms and finding its corresponding framework took a lot of time. Many of them not supported on Jetson. Had ERFnet, using Pytorch, working on host system but we were not able to deploy Pytorch on Jetson. ENET using Torch and Lua gave Memory Issues on Jetson. Recurrent Convolutional Neural Network for scene Labeling worked excellent on host computer using Tensorflow, but unable to get Tensorflow working on Jetson. Finally settled with FCN alexnet which uses caffe. Figure 20 shows running Scene labeling task on a simulated dataset mimicking a construction site while Figure 21 shows ERFNet model output on Cityscapes dataset.
- Segmentation was performed on simulated dataset: No integration with Blimp had happened till the first demo. Training dataset was not collected and annotated as we had not finalized which context awareness team will work with husky or Blimp, so model was trained on Aerial dataset.

8) Context-Awareness C:

- Blimp team published images in compressed format. But we were expecting the images in raw format. We couldn't convert the compressed image into raw image for further processing

B. Demo II

1) Husky: 1. Limited by LIDAR: Demo I involved navigation of Husky in a small area. However, when taken to the construction site, the LIDAR based navigation was severely impaired the navigation of Husky in a relatively bigger surrounding such as the CFL.

2. Movebase libraries: The start and end points entered by the user over RVIZ resulted in the generation of a navigation path for the Husky. This path used Movebase libraries and hence the path was generated randomly. This resulted in the Husky being stuck and unable to navigate further in a dynamic environment such as the CFL.

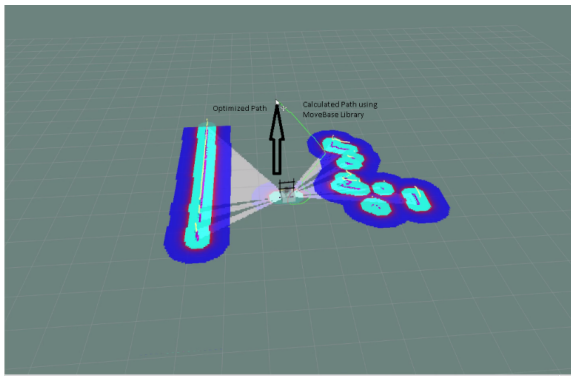


Fig. 22. Random path for navigation

3. IMU interfacing issue: The IMU purchased came with an interface that required UART interface. However, to interface the IMU with the Jetson, a USB protocol was needed.

Fig. 23. Garbage values from IMU

4. Networking Issue: The router was unable to provide sufficient bandwidth to support multiple concurrent connection at a variety of range. Since most of the teams were still in a naive stage of implementation of their respective systems, the Jetsons were not mounted on the Husky. Hence, the Jetsons were individually controlled by the respective teams that were located at various locations on the CFL. Hence, as a result, the router mounted on the Husky was unable to provide work efficiently at a longer range. Hence, the demonstrations that relied on the streaming of images was slower.

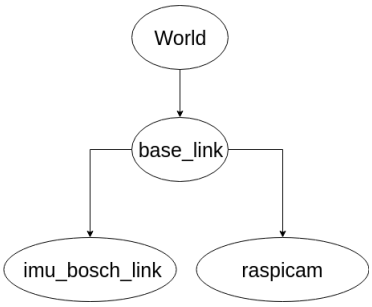


Fig. 24. Incorrect Frame Structure

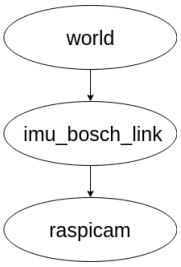


Fig. 25. Correct Frame Structure

2) Indoor Blimp:

The issues faced by Indoor blimp during the Demo II are listed below.

1. Hardware failures and inconsistent behavior:

The problems were primarily observed with one of the motor drivers which failed to work during the demo as well.

Analysis:

The design of the PCB had few drawbacks.The IMU and one of the motor drivers were placed side by side. So one reason that the Indoor team could come up is interference caused by high frequency I2C signals in the motor driver terminal. Hence this design was revised in the third demo where we successfully resolved the issue for the motor driver by placing the motor driver terminals away from the I2C pins of IMU.

3) Outdoor Blimp:

The issues below outline the failures of the outdoor blimp during Demo II:

1. Inaccurate navigation of the outdoor blimp system. The PID for the yaw system was not properly tuned, and logical errors were present in the control system for the motors. This was the reason why the blimp was unstable when attempting to reach a setpoint using the DC motors.

2. Instability of the system in wind. The outdoor blimp could not navigate outdoors due to the excessive wind during demo II. The outdoor blimp was not equipped with control systems to stabilize the blimp system outdoors due to weight constraints, and therefore, the system was found to be unstable in the excessive winds present during the demo.

3. Inability to move pan-and-tilt. This issue stemmed from the prioritization of tasks. Between Demo I and Demo II, the team focused on switching to the new Raspberry Pi 3 hardware, utilizing the new envelope system, and developing the PID for the yaw control. The team did not have enough time to complete the pan-and-tilt system for the demo.

4. Image network issue. During the demo, the context awareness and SLAM teams were not able to connect to the nodes published by the Raspberry Pi. The outdoor blimp team discovered that the reason was due to the inability for the Raspberry Pi to publish separate nodes to each subscriber. The processing power was not enough to meet the demand for every subscriber.

4) SLAM A:

a) No Localization obtained due to wrong orientation of Camera and IMU

Analysis: The Camera and the IMU orientation play an important role for proper localization. Because of the mismatch between their orientations, no trajectory was obtained.

This was resolved later by fixing together the camera and the IMU sensor by not allowing any relative motion between them when the blimp moved.

b) Point cloud was not obtained because of the wrong transform publisher structure

Analysis: The transform frame structure for the VINS Mono code to operate on is shown in figure 25. This was not matched with the frame structure of the data sent from the raspberry pi. The Camera feed and the IMU messages were out of order in the frame structure as shown in figure 24.

This was resolved by coordinating with the blimp teams and changing the frame structure of the ROS rostopics, before publishing it to the network.

c) No synchronization between the camera feed and the IMU

Analysis: Due to a lack of synchronization between the camera feed and the IMU, the VINS Mono algorithm was unable to get initialized as it received frames out of order.

This was resolved by employing Approximate Time Synchronizer in ROS, which subscribes to the two ROS topics, compares their time stamps, and republishes the messages which occur at the same time stamp. Though this results in loss of few frames, it performs well by synchronizing the two feeds irrespective of the traffic on the network.

d) System Clock of Jetson TX1 and Raspberry Pi not synchronized

Analysis: Jetson TX1 and Raspberry Pi did not have the same time in the system. Due to this, in-spite of sending the timestamps, VINS Mono algorithm was unable to initialize.

This problem was resolved by synchronizing both the systems (Jetson and Raspberry Pi) with the Internet time before starting them.

5) Context Awareness - A: In the second Demo we used ENet for the purpose of semantic segmentation. The output of the ENet segmentation obtained was 7 frames per second. For the second demo we were able to subscribe to ROS Camera topics published by the SLAM-B team. We subscribed to the topic which consists of an RGB rectified image stream from the left camera of a ZED Camera. We were able to subscribe to ROS topics but not publish the segmented image. This was

rectified by the end of the final demo.

6) Context Awareness - B:

- **Received Gray scale images from Blimp:** Analysis: ROS packages issues. Fix applied: Setup a standard for images to be published
- **Segmentation was slow:** Analysis: Frame rate at which the images were being published was too high compared to the segmentation rate, which led to images being accumulated in queue. Fix applied: Taking into account the low variation in view of images, some of the incoming frames from the blimp were dropped.
- **Inability to subscribe to the images being published:** Due to high load on the network, we were unable to subscribe to the images being published. Fix applied: This issue was resolved by the addition of an additional router.

7) SLAM B:

a) Our Jetson was unable to receive power from the Husky for no discernible reason. This was likely due to a hardware failure on the original Jetson as several software fixes were attempted without success. Luckily, there was an extra Jetson left over so we were able to replace the device and follow the same installation procedure employed on the first one.

b) Had a hard time figuring out the correct version of all open source libraries which can work properly with each other. This situation occurred primarily due to the independent development of each open source library employed. A recent update to the Jetpack library broke the compatibility with other open source libraries we were using. Please make sure to follow our installation instructions provided in the previous reports to avoid this issue.

C. Demo III

1) Husky: 1. Testing of Blimp Tracking: The blimp tracking could not be thoroughly tested due to the limited navigation of the blimp. Hence, the testing was done as shown in the video here:

2. Odometry data mismatch: The Husky uses odometry data sent by the encoders on the wheels whereas the zed camera has its own odometry data. The map generated by SLAM used the Zed camera's odometry data whereas the map tracking the navigation of the husky used the odometry data from the encoders on the Husky. Hence, there was a mismatch in the RVIZ simulation of Husky and SLAM. This issue was noticed one day before the demonstration and due to the integration complexity, was parked aside.

2) Indoor Blimp:

1. Inability to implement the closed loop velocity control: During the demo III, the Indoor blimp team implemented the open loop velocity control and not the closed loop velocity control.

Analysis:

The team did implemented a PID controller which we used for the yaw control but were not able to use the same for velocity control as we faced some issues with the actual velocity estimation from IMU. The issue emanated from the other higher priority tasks that demanded immediate attention. From demo II to demo III the team completely focused on improving the design of the PCB to resolve the motor controller issue and make the system as compact as possible. Also most of the time the team worked on the integration with SLAM team. During this time one of the IMU stopped working so the spare IMU with the Indoor team was shared with the SLAM team to check the thoroughness of their algorithm. Hence, it was quite difficult to manage time and resources and we were unable to design the estimation system to calculate the actual velocity from the linear acceleration values of the IMU.

2. Implementation of complete obstacle based navigation:

The indoor team was able to implement a simple obstacle detection system with only one sensor.

Analysis:

The obstacle detection demanded placement of multiple sensors on the blimp along its periphery. This was difficult from the point of view of the payload weight that the envelope could handle. Hence we planned to implement a simple obstacle detection in front and rear which would halt the system once an obstacle is detected on either sides. We successfully implemented the program for the obstacle detection in the open-loop velocity control. However, we ran into some issues when we electrically connected a second sensor to the controller/PCB. The issue caused some spurious data that detected an obstacle even when no obstacle was detected. We tried implementing a low pass filter in the software and also calculated moving average of the readings over a time interval, but still the problem was not resolved. Finally, given the time constraints we performed the demo with a single ultrasonic sensor.

3. Inability to showcase a proper demo in Demo III:

The indoor team was unable to perform demo to the desired expectation because of some hardware issues that the system ran into at the last moment. But we do have the videos for the same that can be found here. The same are available in the shared drive that can be found here

Analysis

One of the motors failed to work during the final demo. We observed that it was heated and came to a conclusion that it was short circuited or damaged due to continual use. We did tried to replace the motor but it seemed that the problem also propagated to motor controller as well. Secondly we faced some issues with the raspberry pi 3. Due to amount of processing beyond its capacity the processor was heating up very fast and this resulted in two problems. One it could be a possibility that it might not be giving correct signals to the motor controller that failed the operation of the motor. Secondly, it was entering a thermal shutdown approximately after every 15 minutes. This also caused inconvenience to CA and SLAM team during the demo.

3) Outdoor Blimp:

The issues below outline the failures of the outdoor blimp during Demo III:

1. Inability to navigate autonomously. This failure results from the lack of a stable platform to create and implement an autonomous navigation algorithm. The entire focus for

the team was to first create a stable blimp platform before moving efforts to develop and test an autonomous path-finding system. The long development times of the hardware cost software development in implementing an autonomous navigation system.

2. Multiple launch files for ROS. This failure stems from communication issues between the outdoor blimp team and the context awareness and SLAM teams. One team required an OpenCV compressed image transmission while the another team required a non-OpenCV raw image transmission. The Raspberry Pi could not handle sending two separate types of images. A solution would have been to have the Jetson take the OpenCV image and republish as a regular image, but since this requirement was given too late in the development process, this feature was not added to the system. Therefore, two separate launch files were created to satisfy the requirements for both teams.

3. Lack of GPS and barometer integration into the software. This failure is a result of requirements of other teams. Since the context awareness and SLAM teams were not in need of GPS and altitude measurements, the barometer was not integrated and the GPS system was disabled to save processing power for the Raspberry Pi.

4) SLAM B:

- Integration with the context awareness team proved unattainable. The reason for this issue was that the pipeline to get the segmented image, convert it to video, and use that video to make a map of the environment yielded a major obstacle. This obstacle was that there was a considerable lag between the raw images given and output segmentation received back which resulted in major issues integrating the depth data with the received image due to said time lag.
- The frame rate used by SLAM B and the frame rate used by context team were incompatible. We directly used the video at a high frame rate of 60 fps but the context team was converting the images to frames and then segmenting it and then converting it back to video at a sluggish 7 fps. Our SLAM algorithm requires a much higher frame rate than 7 fps so we were unable to integrate with them. The reason they were constrained to use the low frame rate was due to the low processing power of the Jetson when running their computer vision algorithms. In the future, it may be possible to avoid using all the frames by passing each through a queue but limited time prevented the testing of this method before the demo.
- Odometry data localization error between Husky sensors and Zed camera localization. Our scripts utilize two different sources of localization data which were shown to be slightly mismatched when changing between the two. This is likely due to slight errors in both the absolute distance between the sensors and an error in the coordinate transformation between the two sensors.

5) Context Awareness - A: For the final demo we created a pipeline to subscribe to ROS topics and publish our segmented images as a stream of camera images. Also, we subscribed to depth camera image topic from the ZED and synchronized our segmented image to publish both segmented output as well as its respective depth image so that the SLAM-B team can create an occupancy map based on our data. In the final demo we had some bandwidth issue because of which we could not echo the synchronized depth topic that we published.

6) Context Awareness - C: Major failures of Ctx-C in final demo are as follows -

- Couldn't update Occupancy map dynamically - We couldn't subscribe to the point cloud published by SLAM to compute the occupancy map by reading camera orientation at every frame. We projected our findings on grid assuming the initial camera location.
- Slow segmentation - Segmentation with FCN Alexnet took about 2 seconds to segment every frame. Though we took care not to accumulate the delay by dropping few frames, we couldn't reduce the inherent delay in segmentation
- Change in resolution of training data and test data - We had captured images at 1280x720 resolution. We reduced the resolution to 640x360 for training and faster segmentation. But on the day of demo, the outdoor blimp team published image in 410x380 resolution. There were errors in segmentation results because of this.
- Incomplete integration - Though we projected the segmentation findings on grid and published the same, husky/blimp couldn't use that information to make navigation decisions.

7) Context Awareness - B: Some of the major drawbacks suffered on the final demo are highlighted below:

- Slow segmentation: The average time taken to segment an incoming image published by indoor blimp was around 300ms. This leads to the assumption that the actual frame rate for segmentation should be 3 Hz but the delay incurred due to the network latency on account of multiple publish and subscribe operations meant that we could only achieve a far lesser rate. To add to this, the segmented image had to be converted to a 2D occupancy grid map which incurs additional CPU cycles. On account of this, we were able to achieve a frame rate of 1 image every 3-4 seconds. To ensure that the segmentation does not seriously lag the entire module integration, we made sure that every 30th frame was only considered for segmentation while the other frames were ignored.
- Inability to update the occupancy map: Though we were able to generate the 2D occupancy grid map, due to the unavailability of localization information, we were not able to accurately say where the blimp was at any given moment of time. The origin would always be taken to be (0, 0, 0). Furthermore, at any point in time, the map would not be built in conjunction with the historical data available. It would overwrite the previous map, thus leading to incorrect occupancy grid updates.
- Synchronization of point cloud with SLAM: As men-

tioned above, due to the slow overall segmentation process, we couldn't really achieve proper synchronization with the occupancy grid being generated by the context awareness module and the point cloud data being generated by SLAM.
- Improper camera orientation: The camera module attached to the indoor blimp did not have a pan tilt mechanism for camera orientation. The data collected by context awareness team was at an orientation of 30° from the vertical. The camera feed obtained from the blimp was as good as a top view image which did not align well with what the model had been trained to work with. Due to this, there were significant deviations from the expected behavior for the segmentation module.
- Disparity between annotated images and live camera feed: The image sequences being sent by the blimp team was 410*308. This was considerably smaller that what our segmentation model was trained (1280*720) to work upon which were collected using Apple iPhones. Though we did resize the image to appropriate size, the results just weren't up to the mark in certain scenarios.

8) SLAM A:

Improper localization at certain instances

Analysis: The VINS Mono algorithm best performs when the system uses a global shutter camera and a synchronized high-end IMU. In our case, the system lacked both. The Raspberry Pi camera is a rolling shutter camera because of which faster movements create distortion in the image captured. Also the IMU sensor is unsynchronized with the image feed. These two factors combined to give a poor performance by the algorithm. This hardware constraint, resulted in the localization to go wrong occasionally and subsequently affect the future point cloud generation.

The only possible remedy to this issue was to use a sophisticated sensor which combined the capabilities of both the global shutter camera and a synchronized high-end IMU.

IV. Future work

A. Husky Hardware

In its current state, the Husky can navigate autonomously. However, here are some additional features that will make it more flexible:

- The map is generated as the Husky travels the path in real time. However, it can only be given a point that has been mapped. Hence, it is limited by the range of FOV of the Zed camera. If the Husky is made to navigate manually in a given area so that it generates a total map of the area, then that would make it flexible to traverse the given terrain.
- The data from the blimp was not integrated into the system. Hence, the Husky was only dependent on the Zed camera for navigation.
- The LIDAR and Zed camera were not integrated into the system for generation of the occupancy grid. Due to inherent drawbacks in Zed and LIDAR, the Zed works better for fast moving obstacles. Hence, the system can switch between LIDAR and Zed camera

B. Outdoor Blimp

- Increasing the envelope size to allow for more lift. Increasing lift would allow more components such as stabilizing systems and more powerful motors to be implemented onto the system.
- Reducing weight of the system. Using smaller gimbal servo motors and lighter plastics for the gimbals are just examples of how the system can reduce weight and increase the lift of the envelope.
- Redesign the method the payload and motor stands are attached to the blimp envelope. Currently, the system is attached to the envelope through tape and industrial Velcro. A more permanent solution such as glue could be investigated.
- Design and implement autonomous movement and path planning. Since all the development time went towards building the hardware system, the outdoor blimp team was not able to implement autonomous navigation and path planning. A stable platform was necessary before transitioning the focus on software algorithms.
- Integrate altitude control, barometer, and GPS back into the system. Since these systems were not utilized by any other team, they were deactivated for the final demo. For a future system, these systems should be re-activated and integrated back into the overall outdoor blimps system.

C. Indoor Blimp

- The design of the envelope can be modified to make it more compact. Currently, the dimension of the envelope is large enough to carry the required payload and at the same time it can navigate through broad passages. But the shape can be redesigned to make it navigate through confined areas at the same time providing sufficient payload.
- Can implement velocity control in closed loop system by fusing data from various additional sensors.
- Resolve the software issues with Raspberry pi zero to make it compatible for video transmission such that the weight can be reduced since raspbery pi zero is lighter than raspberry pi three.

D. Context Awareness - for Husky

The present work, with context awareness for autonomous navigation could be extended in several ways. The tasks listed are with the focus of making the model more integrated and robust from the existing state.

- The model implemented now has been trained on data from one environment Construction Facility site (CFL) . In doing so the model gets over fitted with this environment and may not be robust when tested with other environments. Good collection of input data from

multiple environments and corresponding labeled images are needed for the model to be robust.

- The present pipeline is designed to extract the ground plane from the labeled segmentation map, which is used to draw the frontal obstacle boundaries. In this process the valuable available data, about object classes is ignored. The obstacles detected in the image can be categorized as static or moving objects. Relationships between sequence of frames in the regions of moving obstacles could be drawn with an intension of obtaining motion vectors, which would help path planning and updating of occupancy map.
- The present segmentation model considers only the RGB data for training the network. Depth information based learning models can help get better segmentation outputs. The depth information would be useful in cases where ground plane is similar obstacle example.
- The computational load put on the module corresponding to the segmentation task runs heavily on the GPU. The size of the model and high memory usage along with the need for real-time performance restricts the speed of the Husky. Improving the segmentation model in order to reduce the computational load in the Context Awareness Module will address this issue.

E. Context Awareness - for blimps

- Calculating the projection of segmented output should be done by taking input from slam team about the camera matrix. This makes the information more meaningful and usable for husky to navigate
- Segmentation results should augment the occupancy grid thus making the robot more aware of its environment. Integrating the semantic segmentation output with findings from SLAM will help taking major navigation decisions
- Finding distance information is very crucial. Context awareness team of husky has the leverage of zed camera to find the depth. But the context awareness teams for blimps should find a way to calculate depth using different sensors.

F. SLAM A

- Since the Raspberry Pi camera is a rolling shutter camera it adds distortion to the images captured which degrades the performance of the VINS Mono SLAM algorithm. Possibly by using a global shutter camera the overall performance of the system can be improved
- The IMU and the camera used in the setup were discrete units because of which their data streams were not synchronized. The Asynchronous Time Synchronizer package in ROS did give better performance but at the cost of loosing frames. A better alternative for this is to use an integrated sensor combining the camera and the IMU which will possibly reduce the delay latency between the two streams.
- The project was implemented at a prototype level where the hardware was not required to perform intense calculations. At an industrial level, the hardware and the software need to be optimized. This will require the use of GPU and a CUDA compatible code.

V. Conclusion

Over the past few years, camera-mounted unmanned vehicles have gained significant popularity for developing vision-based data acquisition and construction monitoring systems; however, there are still numerous open problems for further research such as autonomous navigation in a construction environment and development of low computational complexity platforms to streamline the data collection process. To address the above-mentioned needs and increase the degree of automation for progress monitoring, this report presents an integrated mobile robotic system that runs multiple vision-based components in real-time. The proposed system implements monocular SLAM and contextual understanding of a scene, which creates a 2D spatial map with detected obstacles. This system showcases the importance of a modular framework which can include the latest SLAM and context-awareness algorithms in a plug and play format. This system is effective and can be run in real-time on multiple embedded platforms that are integrated as a system. The hardware utilization can further be improved by sharing computing resources for certain nodes, also potentially decreasing the network delay in exchanging ROS messages. The proposed system is a step forward in making intelligent and contextually aware robots ubiquitous. The results also demonstrate the potential for enabling a computer vision system for autonomous navigation to lead to automatic construction performance monitoring and condition assessment purposes.