Final Project Report

Team F

Falcon Eye

Yuchi Wang, Rahul Ramakrishnan, Danendra Singh,
Pulkit Goyal, Pratibha Tripathi

sponsored by

Katia Sycara

May 9, 2018

Abstract

This report details the work completed by Team F for the MRSD Capstone course. The project examines the collaborative use of a heterogeneous system, comprised of a ground and an aerial vehicle, at navigating an unknown environment. First, we provide a more detailed statement of the project description and scope. Then we discuss the intended use case and motivate our system's design and purpose with a time-critical example. We proceed to break down our system's design at both a system and cyberphysical level and provide trade studies on our system's major components. We show our system's full development progress, summarize key design decisions, and evaluate our system's performance against the metrics specified in the SVE. Lastly, we take a retrospective view on our schedule, budget, and risk management and comment on possible future continuations of our project.

2. Project Description

Navigation and exploration is a central functionality in robotics applications and an ongoing problem for robotics research. To successfully navigate from one location to another, a robotic system needs to localize itself within the environment, compute the optimal trajectory, and avoid the obstacles that lie in the way. In the case that the robotic system has prior knowledge of the environment, the problem becomes deterministic. Once localization has been completed, the knowledge of the map is used to determine if a path exists and what that optimal path is.

However, if prior knowledge of the environment is unavailable, then the navigation problem is non-deterministic - that is, there is no guarantee of a solution and the true cost of any path is unknown. As a result, the path planning algorithm has to rely the information obtained from the sensors and a heuristic to estimate the true cost of the paths. Improving the navigation problem in an unknown environment effectively reduces to (1) improving the quality of information from the sensors and (2) improving the heuristic. In this project, we propose a multiagent solution to enhance the information that can be obtained from the sensor suite.

We are specifically considering navigation in unknown environments for ground robots. For this class of robotic systems, the effectiveness of their cameras, LiDAR and other visual sensors is highly dependent on the amount of physical obstructions in their local vicinities. If there are many trees, walls, or other objects that obstruct the sensor's field of view, then the information available to the path planning algorithm is substantially reduced. This can result in the robot selecting non-viable paths because the sensors were unable to detect dead ends or corners. We propose to address this problem by augmenting the robot's effective field of view with aerial data from Unmanned Aerial Systems (UAVs). This allows the robot to see areas that would be otherwise obstructed from its field of view and enables it to prune dead ends much earlier, resulting in better path planning decisions and faster navigation in unknown environments.

This problem in its entirety far exceeds the scope of an 8-month capstone project. As a result, certain aspects of the problem were simplified to reduce the complexity. First, we assume that the system is able to detect and recognize open paths, obstacles, and dead ends by representing those entities with fiducial markers, thereby eliminating the need for complex image recognition functionality. Furthermore, we assume that all valid path segments are straight, with a fiducial marker presenting each corner or bend in the paths. Last, we consider only the situations where a solution to the navigation problem exists - that is, there is guaranteed to be a traversable path to the destination.

3. Use case

In 2030, Pittsburgh is ravaged by a magnitude 3.0 earthquake and suffers heavy infrastructural damage to the buildings, pipelines, and electrical grid. Crumbled buildings, displaced streets and the resultant debris leave many some paths inaccessible and render others into dead ends. Pittsburgh is in desperate need of information on the sustained damage by the earthquake so supplies and first aid can be sent in but unfortunate weather shuts down satellite imagery while high altitude data lack the resolution to see the exact damage. Many people are trapped amidst the debris, time is of the essence, and relief teams urgently need to deliver supplies to the wounded.

As this is 2030, robotic systems have become ubiquitous in the workplace and seeing one on the streets performing some task is absolutely normal. This also means that robotic platforms have become the norm in search and rescue operations and, amongst other functionalities, are used to deliver critical supplies to patients and those in need. Thus, the city of Pittsburgh dispatches Major John Dolan with a number of FalconEye systems, a heterogeneous robotic swarm, to distribute the critical supplies to various emergency aid centers around town. John arrives at the outskirts of Pittsburgh and quickly unloads and sets up FalconEye. He inputs the location of a different aid center to each FalconEye swarm, sits back, relaxes, and monitors the progress from his monitor.

Let us examine the FalconEye system (Figure 1) which has been assigned to deliver supplies to aid center A. The amount of debris and fallen objects render many streets and alleyways inaccessible from the ground. If the robotic system only had access to ground sensors, then path planning would be a nightmare. There is no telling whether a certain pathway leads the system closer to the destination or if a giant piece of rubble lies right around the corner. However, because FalconEye is a heterogeneous platform that leverages the aerial coverage of the UAV, it is able to detect these obstructions and dead ends around corners and other intricate pathways. The UAV flies ahead of the AGV and examines the path forward for possible obstacles and debris. When it has ascertained that there is no obstruction in its extended field of view, it notifies the AGV that the path is open and gives the command to proceed forward. If there is an obstacle that the AGV cannot cross, then the UAV indicates that this path leads to a dead end, prunes it from the path planning algorithm, and continues onto the next possible path. As a result, the system is able to efficiently calculate the optimal path and navigate to aid center A in a shorter amount of time than a ground robotic system would.

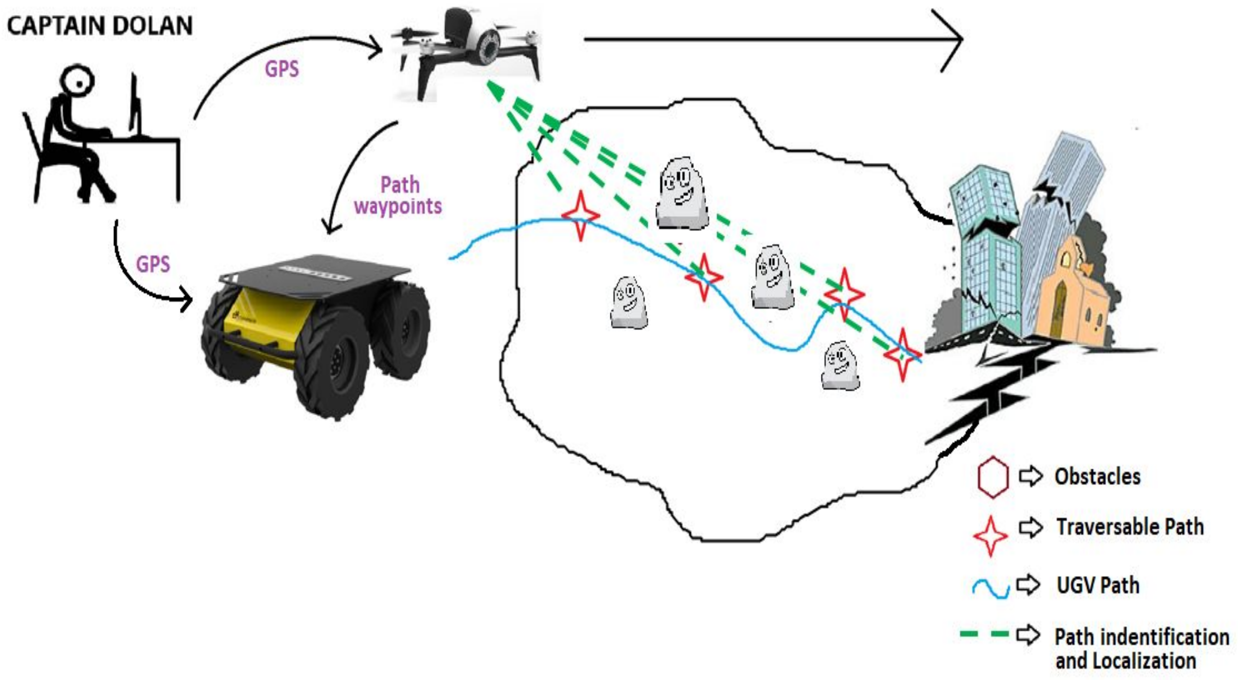


Figure 1: Use case illustration

4. System level requirements

The requirements for this project are derived by thorough discussions with all the stakeholders. They help in devising a system that is robust, realistic and reliable. The performance requirements ensure that the system functions according to the user's need and expectations.

Mandatory performance requirements

M.P.1: The AGV will navigate to each intermediate locations and the final location within 3m.
M.P.2: The UAV shall detect fiducial markers with 80% success.

M.P.3: The Base computer shall compute distance between fiducial markers with an accuracy of 15 cms.

M.P.4: The UAV shall autonomously fly to the desired GPS location with 5m accuracy

M.P.5: The AGV will avoid obstacles 40x40x80cm or larger with 100% accuracy

Mandatory non-functional requirements

M.N. 1: The UAV shall be able to fly for the entire mission

M.N.2: The system shall be able to be easily transported from one station to other

M.N.3: The system should be easy to setup and operate.

M.N.4: The system will consist of at least 1 UAV and 1 AGV.

Desired performance requirements

D.P.1: The AGV shall combine the LiDAR data with Odometry, GPS, and AprilTag data such that GPS location of the obstacles is accurate within 2m.

D.P.2: The AGV shall be able to localize itself within the area map with 5m accuracy

D.P.3: The AGV shall provide camera feed to the user at a refresh rate of 60 Hz

Desired non functional requirements

D.N.1: The system shall have 2 UAV and 2 UGV

D.N. 2: The system should be able to operate in uneven terrain

D.N.3: The system serves for future research purpose of the sponsor

D.N.4: The system shall provide Lidar data for future research purposes

5. Functional Architecture

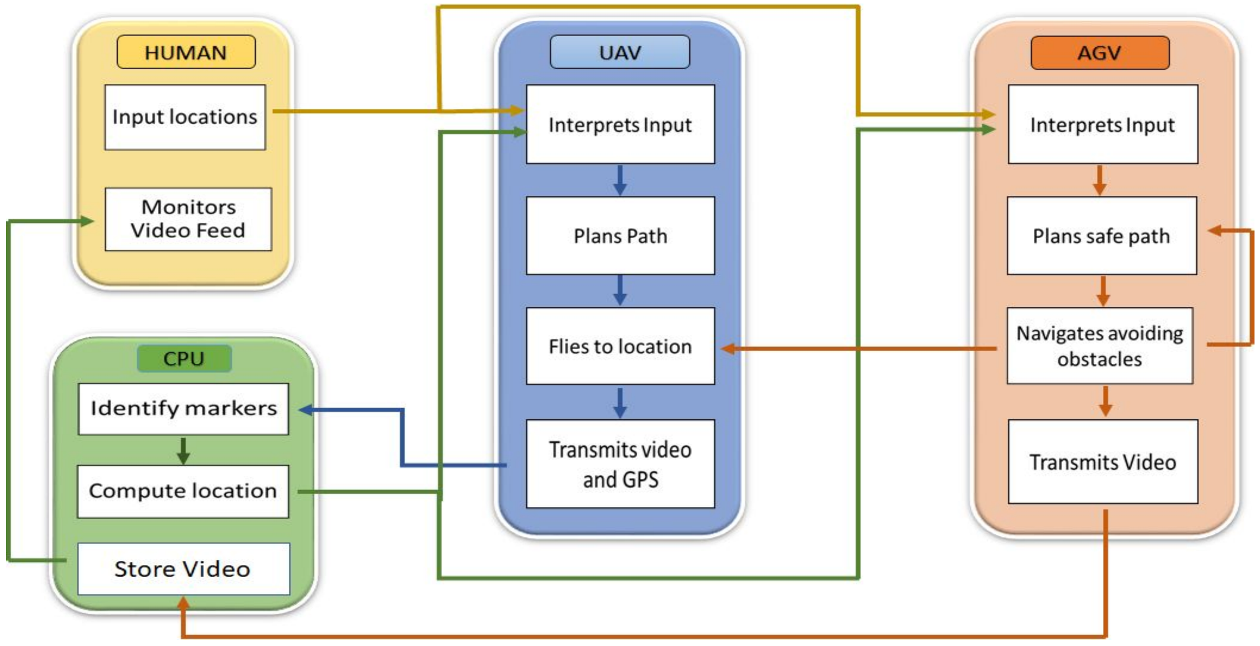


Figure 2: Functional architecture

The main functionality of the system can be divided into 3 primary components: the AGV, the UAV, and the CPU. The human plays an observer role and aside from giving the final destination and starting the system, simply monitors the system. The motivation of the functional architecture is to modularize the system such that each component can work successfully without assuming the presence of the other components. The AGV and the UAV represent the physical robots and the functionalities that they can provide as a standalone component. For the UAV, these features would be comprised of capturing and sending a video feed to the controller and navigating to GPS locations. Notably, the UAV does not make any assumptions on what the video feed is being used for, where the GPS locations are coming from, and how the next GPS location is being calculated.

The functional block for the AGV follows a similar structure. As a ground vehicle, its main objective is to navigate to a specific GPS location. As a result, the functionalities that it needs to provide address this navigation requirement. Specifically, given a target location, the AGV subsystem needs to avoid obstacles in its local vicinity and plan a local path to the location. Likewise, the AGV also transmits a video back to the controller.

The CPU acts as the bridge between the UAV and the AGV and represents the brain of the system. For the CPU block, lower level details such as UAV velocity or ground obstacles are abstracted away from its controller. Instead, the primary functions of the CPU are twofold. First, it is concerned with localizing the UAV and the AGV with respect to each other, the deadends and valid paths in the environment, and the final destination. It does so with the sensor data that it acquires from both the UAV and the AGV. Furthermore, with knowledge of the environment, it computes the most probable path to the target destination and relays this information to the UAV and the AGV.

6. System level trade studies

6.1 AGV platform

The AGV platform plays a major role in the operation of the entire system and needs to be chosen wisely based on several factors. The AGV has to be rugged enough to sustain the rough and unpredictable environmental conditions associated with disaster zones. The platform needs to be agile enough to avoid large physical obstructions in its path and while maneuvering over smaller obstacles without getting stuck. It needs to be sufficiently mobile to reach the target location in a reasonable time and while transporting the UAV payload. The AGV should be able to operate for hours at a time while performing its surveillance mission. Lastly, the cost for these platforms is considered - please note that if the platform is in inventory at CMU, it is given a cost of 0 for the purposes of the trade study. Considering all these factors (Table 1), we have narrowed down upon the Clearpath Husky.

Table 1: Trade study for the AGV platform

AGV Platform	Carrying capacity	Ruggedness	Speed	Operation time	Cost	Total
Clearpath Turtlebot	1	1	1	3	5	11
Clearpath Jackal	2	3	4	4	2	18
Clearpath Husky	4	4	3	5	5	21
Clearpath Warthog	5	5	5	3	1	19
Boston Dynamics BigDog	3	3	3	3	0	12

Boston Dynamics LS3	4	4	3	5	0	16

6.2 AGV onboard sensors

The effectiveness of path planning and object avoidance depends heavily on the sensors that are used. For this application, the sensors needs to have sufficient accuracy and sensor range such that obstacles can be detected precisely and with enough range for the AGV platform to make the necessary avoidance maneuvers. The usefulness criteria in Table 2 is indicative of how well the type of sensor data will help the path planning and localization subsystems. For instance, a 2D lidar or a stereo camera that performs suboptimally in bright light will receive a lower score than a 3D lidar. The sensor should also have a wide angle of coverage and should be easy to integrate with our system. These requirements led us to finalize on using Velodyne VLP 16 sensor to map the environment.

Table 2: Trade study for the AGV onboard sensors

Onboard sensors	Accuracy	Sensor range	Usefulness	Coverage angle	Cost	Total
Velodyne VLP 16	3	5	5	5	1	19
Sick LMS111	3	3	2	4	1	13
Microsoft Kinect	4	1	2	2	5	14
ZED 2k Stereo Cameras	5	3	3	3	4	18
Hokuyo URG-04LX-UG01	4	5	2	4	3	18

6.3 UAV platform

The UAV platform forms the basis of the aerial surveillance system and therefore needs to be robust. Its speed should be high enough to enable quickly reach the disaster zone from its deployment location and cover the surveillance area in a reasonable time. It should have high flight time to cover the maximal area and stable enough to provide clear video coverage of the target location. The package support for any given drone should also enable quick software integration and deployment. See Table 3 for our analysis. All these factors helped us to decide on Bebop 2 as our UAV platform.

Table 3: Trade study for the AGV onboard sensors

Drone platform	Speed	Equipped sensors	Operation time	Developer support	Stability	Cost	Total
Clearpath Pelican	2	5	4	4	5	2	22
Parrot AR Drone 2.0 Power	3	4	2	5	2	5	23
Parrot Bebop 2	5	3	5	4	4	5	26
DJI Phantom 2	3	3	5	5	4	5	25
DJI Mavic Pro	5	3	5	5	4	3	25

7. Cyberphysical architecture

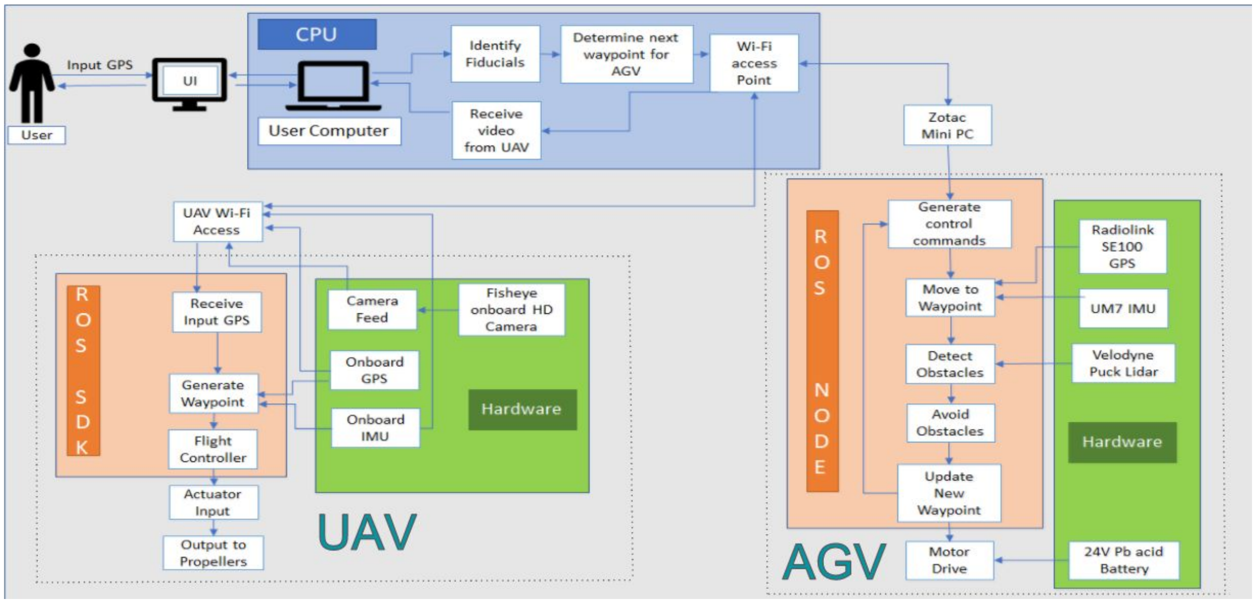


Fig. 3: Cyber-Physical Architecture

The cyber-physical architecture depicted in the Figure 3 details the specific functionality and components of the main subsystems. It elaborates the flow of information and interrelation of various functions of the architecture.

The cyber-physical architecture derives clearly from the functional architecture and is thus also divided into 3 main components:

1. AGV (Software and Hardware)
2. UAV (Software and Hardware)
3. Central Processing Unit (CPU)

The software controls of the UAV and the AGV are exposed our controller in the form of ROS nodes. The open-source property and developer culture of ROS enables us to easily integrate the functions and features that are required for those subsystems.

7.1 Autonomous Ground Vehicle(AGV)

Our AGV will be performing the following functions:

- Receive waypoints over a common network.
- Plan a local path to the input location received.
- Provide camera feed to the user in real time.
- Detect and avoid obstacles with the Velodyne Puck and adjust its local path accordingly.
- Generate a map of the area based on sensor data.

7.2 Unmanned Aerial Vehicle(UAV)

The UAV has the primary task of transmitting the video data to the CPU and navigating to GPS coordinates. Specifically, it the following operations:

- Receive GPS locations from the CPU.
- Navigate to the GPS locations.
- Transmit video feed to the user.

7.3 Central Processing Unit(CPU)

The central processing unit is a computer at the base station that interacts with all other subsystems. It has a two way communication channel between the user, AGV and UAV. Its main functionalities are

- Receive video feed from the UAV
- Detect fiducials from the video feed
- Localize the UAV and the AGV with respect to the fiducials
- Compute most optimal path to final destination and the next target locations for the UAV and the AGV
- Send the target locations to the UAV and the AGV

8. System description and evaluation

8.1 Subsystem description

8.1.1 AGV Subsystem

The main objective of AGV is drive to the closest location to the disaster zone and then trigger the UAV flying. While moving to the disaster zone the AGV avoids obstacles on the way and incorporates path planning. We will be using Clearpath's Husky (Figure 4) mobile platform that has been provided by Professor George Kantor.

Figure 4: Clearpath Husky AGV

a) AGV sensing

The AGV generates environment map using a LIDAR sensor. A LIDAR usually consists of a scanner and a laser. The laser emits a laser beam at a certain frequency and also receives the reflected laser beam from an object on the path of the laser. This can be used to estimate the distance between the LIDAR sensor and the object by estimating the time difference between the transmitted and received laser. This method is a very reliable method of estimating the distance upto a certain limit depending upon the LIDAR specifications. We have selected Velodyne VLP 16 sensor (Figure 5) due to its excellent online support and reliability.

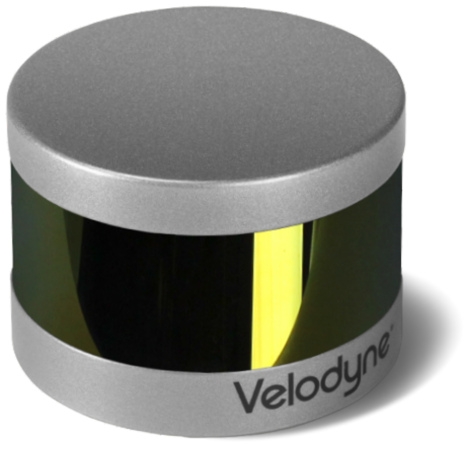
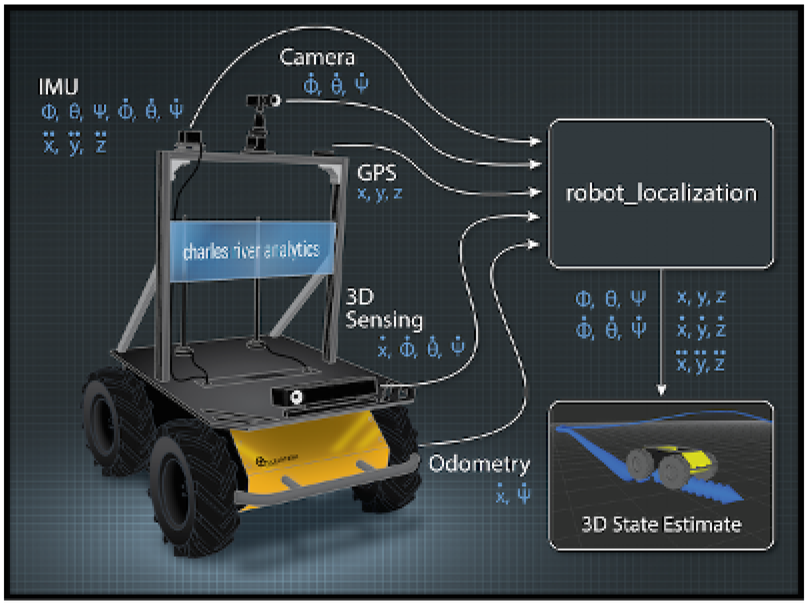


Figure 5: Velodyne VLP 16

b) Localization

We will be using a GPS system on the AGV for global localization of the vehicle. Using global waypoint generation we can navigate in a large environment space using individual GPS waypoints. We will use ROS package for husky to transfer GPS coordinate input by the user. Conventional GPS modules have an accuracy of around 5-8 meters which can be used to localize the AGV. Please refer to Figure 6 for a visualization of our localization architecture.



IMU
$\Phi, \theta, \psi, \dot{\Phi}, \dot{\theta}, \dot{\psi}$
$\ddot{x}, \ddot{y}, \ddot{z}$
Camera
$\dot{\Phi}, \dot{\theta}, \dot{\psi}$
GPS
x, y, z
charles river analytics
3D Sensing
$\dot{x}, \dot{\Phi}, \dot{\theta}, \dot{\psi}$
Odometry
$\dot{x}, \dot{\psi}$
robot_localization
Φ, θ, ψ
$\dot{\Phi}, \dot{\theta}, \dot{\psi}$
x, y, z
$\dot{x}, \dot{y}, \dot{z}$
$\ddot{x}, \ddot{y}, \ddot{z}$
3D State Estimate

Figure 6: Localization methodology

c) Object Detection

LIDAR is used to detect ground obstacles that lie in the path of the AGV. We can classify the reflected lasers that have the same bouncing distance from the LIDAR and our neighbours as one object and using centroid estimation techniques, the center of the object can be fairly estimated for object avoidance.

d) Path Planning

We are going to develop an algorithm for optimized localized-navigation. Search based planning algorithms like A*, D* and D* Lite have proven to be very effective for motion planning of a mobile robots. We have decided to use search based algorithm and test them on the AGV, finalizing the one which does not depreciate the computational capabilities of the AGV while at the same time accurately plans the correct path.

8.1.2 Central Processing Unit

A base computer is being used as a centralized unit for data transfer between all subsystems. It will also be responsible for receiving data inputs from the user and display live video feed from the UAV via an easy to use GUI. The specifications of the base unit are as follows:

- I7-7700HQ Intel Processor @2.8 GHz
- 16 GB DDR4-RAM
- NVIDIA GeForce GTX 1060 6GB GDDR5 GPU
- 1TB Hard Disk Drive, 5400rpm, 2.5”, SATA3
- 512GB Solid State Drive M.2 PCIe

8.1.3 Communication

The communication system is critical for the mission and hence should be robust, reliable and efficient. Since it will encounter multiple communication channels, the module should not interfere with other channels, avoid latency and give excellent range.

We have decided to use Ubiquiti BULLET M-5 outdoor 5 GHz WiFi radio for our communication needs.

8.1.4 UAV Subsystem

After getting deployed from the AGV, the UAV will perform autonomous aerial surveillance of the area and transmit High-Definition video frames back to the operator. We will be using Parrot Bebop 2 (Figure 7) drones, that are provided to us by the sponsor. The drones come preinstalled with a HD camera and software image stabilization.

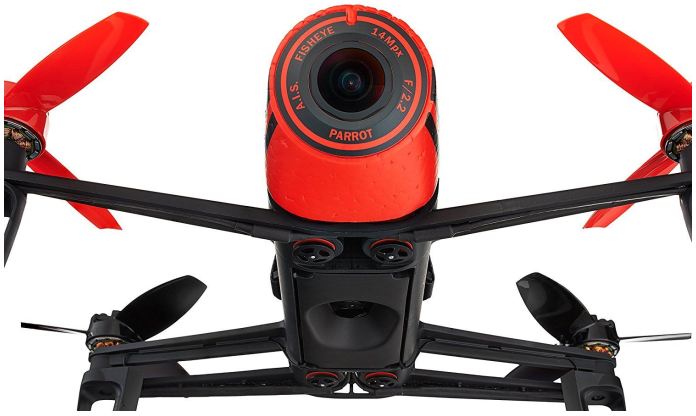


Figure 7: Parrot Bebop 2 Drone

a) UAV Sensing

The UAV comes with a High-Definition fisheye camera. This can be used to transmit live video feed from the UAV to the base station. The fisheye lens can be used as a software gimble and hence transmit video feed frames from different angles.

b) Localization and Path Planning

GPS based localization and path planning will be incorporated for autonomous surveillance of the disaster area. Multiple UAVs help in covering more area in the same time, hence allowing the user to identify area of maximum interest and focus the surveillance operation in those areas.

8.2 Modelling, Analysis, and Testing

The navigation stack for the AGV and the UAV required extensive testing to all the sensor inputs and the calculations to ensure that the heading and the required direction was correct. First, the GPS accuracy was tested in terms of stability (short-term inaccuracy) and drift (long term inaccuracy). We let the UAV and the AGV record the GPS location without moving for 1 minute and computed the maximum difference to find out the maximum drift. The plot for the UAV GPS locations is provided below. This results in approximately 3 meters of drift and 50 cm of stability.

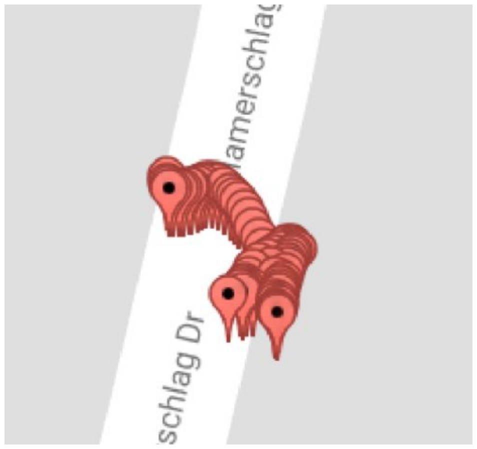
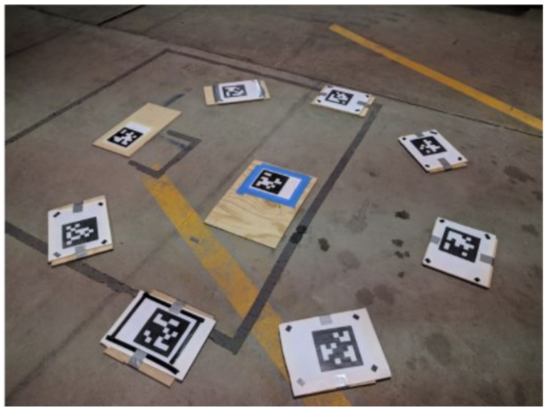


Figure 8: GPS drift of the Drone

We also tested the IMU of the Drone and of the Husky. The IMU data from the drone was very stable and we didn't notice a significant amount of drift. We were specifically looking at the yaw data since that information is required to correct the heading of the Drone but we found it to be stable and consistent over time. However, the same could not be said for the UM7 IMU module which we were using for the Husky. The yaw drift on this UM7 was unacceptable and regardless of the repeated calibrations that we performed, the value would drift without bound. Even when we held it absolutely still on the ground, the value would continue to either rise or fall. In the span of 2 minutes, the IMU's yaw value would go from 0 to 180 when it was stationary. This result is unacceptable to us and for the FVE, we addressed this issue by using the phone as an IMU.

After the individual calculations for the GPS navigation was complete, the entire controller was tested together. The first version of the controller didn't use a smoothing feature and was therefore very jerky. To address this issue, we implemented a P controller that adjusted the turn rate and the speed based on its error. To increase the effectiveness of the controller, we scaled the output nonlinear based on the input. For instance, the drone turns fast unless its heading is very closely aligned with the desired heading, thus increasing speed while not overshooting while turning.

As well, we tested the accuracy and the rate of detection for the April Tags using the drone camera. This is a crucial functionality that must be achieved in order for us to implement the rest of the system and it is also an essential aspect of our FVE. To test both criteria, we arranged the April Tags in a certain pattern and measured the system's ability to reproduce that pattern. Both the original layout and the RVIZ visualization are shown below. The blue bordered april tag depicts the home frame and hence is not visualized on RVIZ



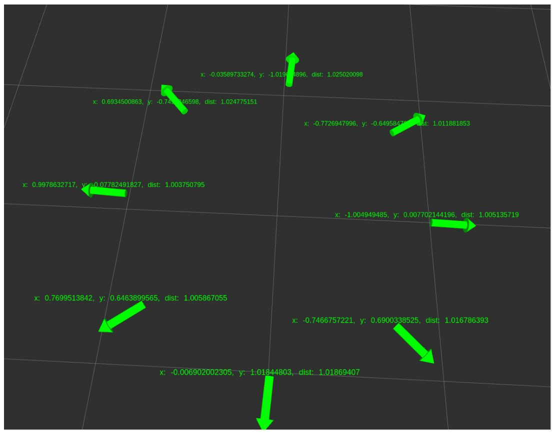

x: -0.03589733274, y: -1.019[illegible]4896, dist: 1.025020098
x: 0.6934500863, y: -0.74[illegible]46598, dist: 1.024775151
x: -0.7726947996, y: -0.6495847[illegible] dist: 1.011881853
x: 0.9978632717, y: -0.07782491827, dist: 1.003750795
x: -1.004949485, y: 0.007702144196, dist: 1.005135719
x: 0.7699513842, y: 0.6463899565, dist: 1.005867055
x: -0.7466757221, y: 0.6900338525, dist: 1.016786393
x: -0.006902002305, y: 1.01844803, dist: 1.01869407

Figure 9: Testing the localization of the April Tags

The last component that was tested was the network. The network is a central component of our project and it acts as the glue that holds all the subsystems together and enables them to interact with each other. Our first requirement was that all our devices connect to the network. This meant that the Bebop 2 has to act as a client and not a host. We have been able to do this with a script and some manual tinkering with the Bebop 2 Operating Sytem. Our second criteria for the network is that it has to work at a range of 50 meters. Technically, this requirement was met but if we consider additional constraints, then our current implementation is inaccurate. Specifically, two devices are able to connect to the same network when they are 50 m apart but the bandwidth between them decreases significantly. Initially, the network was tested with the husky and this constraint was not an issue as we did not have massive amounts of data flowing through the network. However, image processing for the Drone's camera feed happens remotely and therefore the entire video feed needs to go through the network. When the two devices were 50 m apart, we saw slow transmission rates which led to slow localization with April Tags.

8.3 Performance Evaluation against Spring Validation Experiment

Our SVE metrics consisted of the following goals:

- AGV arrives within 3 m of final destination
- AGV navigates the path in less than 10 mins
- AGV avoids 100% of static obstacles.
- UAV-AGV system navigates faster than the AGV-alone system.

During the SVE, we have been able to achieve all of our targeted requirements. Thus, at a system-level, our design and implementation were successful with regards to our metrics.

However, there are elements of our system that can be improved. They relate to the underlying sub-systems and we saw during the SVE that these areas could perform better.

8.4 Strong / Weak points

1. **Strong Points**

    a. Robust area exploration algorithm for the UAV. The UAV will always explore around the nodes to find the next April Tag and prune all dead ends.
    b. Stable April Tag Detection from a height of <=5m due to the use of 12 x 18 sheets.
    c. The goal point for the Husky can be given as a local location by clicking on the rviz GUI also as a global location in form of GPS latitude, longitude and orientation values.
    d. The localization of the Husky robot using dual EKF fusing data from IMU, Odometry and GPS is accurate and stable.
    e. Easy control of the Husky robot using Game Controller is very responsive and helpful in emergency situations.

2. **Weak Points**

    a. Unreliable communication between the AGV and the UAV due to a software bug in the global path planner. Under certain circumstances, the internal graph representation may become disjointed. If this happens, then the software fails to find a path through the graph. We were aware of this issue before the SVE but only completed fixed for the SVE encore.
    b. Exploration algorithm takes very long. Although we stated the robustness of the exploration algorithm, pruning dead ends and exploring ahead is very slow. This comes down to a number of reasons: latency in the video feed, non-PID flight controls, inefficient pruning algorithm.
    c. As the waypoints given to the Husky by Bebop are GPS values, system is heavily dependent on the accuracy of GPS signals. The performance of the system in terms of reaching the intermediate waypoints and final gps location is dependent on the weather and quality of signal received by the satellites.
    d. The waypoints given by the bebop are only latitude and longitude values whereas the navigation stack on Husky requires a orientation/heading along with all the GPS values. We were running the Husky on a fixed orientation value for the SVE but we corrected this by calculating the heading value taking into account the previous GPS value for SVE encore.
    e. Imu requires calibration based upon the environment it is being used in, which can be a limiting factor. The absolute Yaw values are heavily affected by the

calibration of magnetometer in IMU.

f. Converting the location of April Tags to latitude/longitude using GPS on the drone and then following the GPS locations using the feedback from ground robot can add error from both the GPSs.
g. Testing at night and in bad weather - The optical sensor of the drone requires sufficient amount of light to operate and stabilize the drone. Hence testing at night is not an option for us. Also testing in bad weather could result in possible danger and should be avoided.

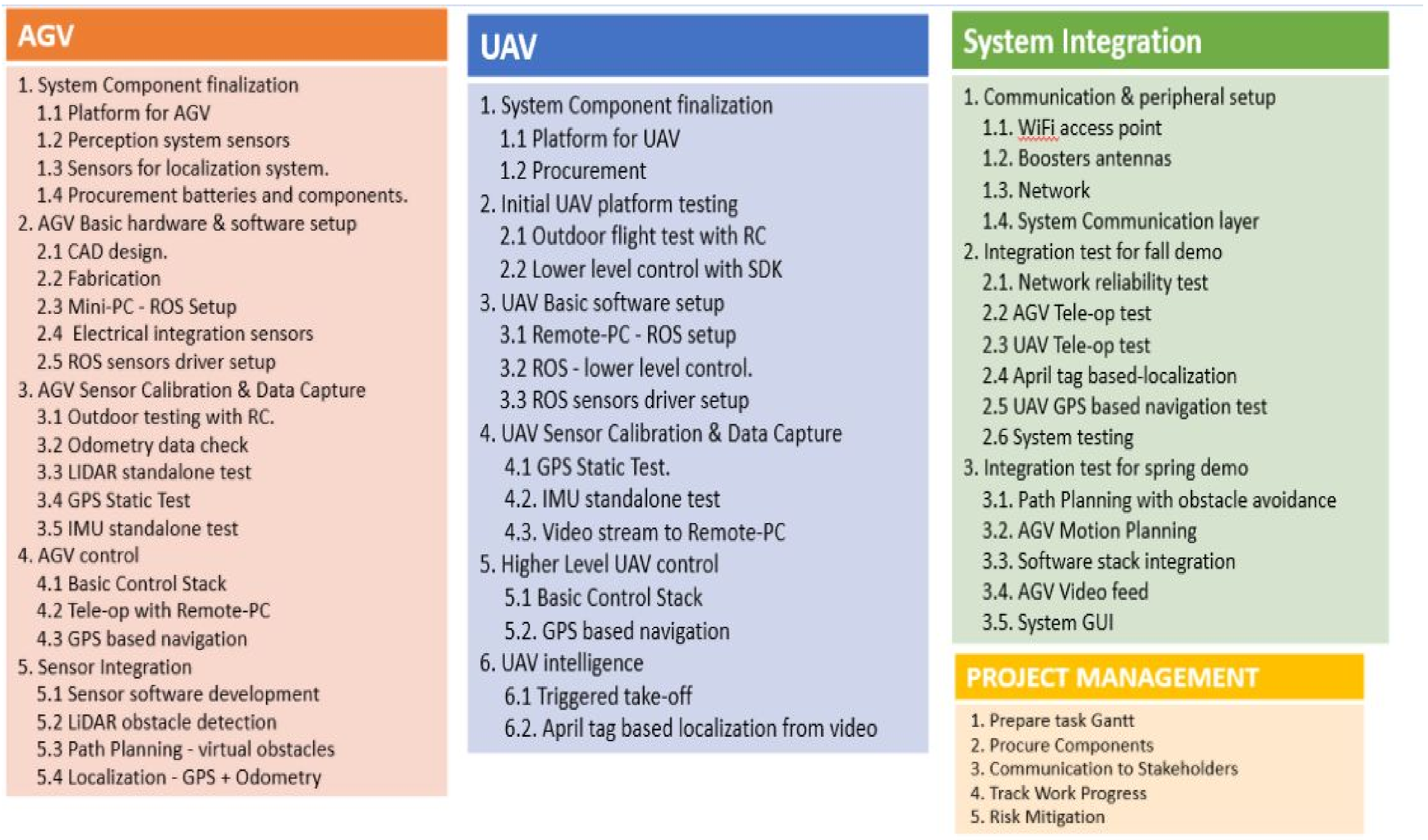
3. **Refinement Areas**

   a. Improve exploration algorithm for the UAV
   b. Increase UAV flying height with due considerations to safe ceiling.
   c. Better localization for the AGV using RTK based GPS, which can increase the GPS accuracy and reduce the dependency on weather.
   d. Instead of using the GPS values for waypoint following, bebop can provide the distance required to be travelled by Husky in terms of x and y, reducing dependency on one of the GPS.

9 Project management

9.1 Work Breakdown Structure(WBS):

We have shown our WBS below (Figure 10) as a reference for our schedule. Our overall WBS is as shown below, we have divided our system into 4 major parts : AGV, UAV, System Integration and program management.


AGV

1. System Component finalization
   1.1 Platform for AGV
   1.2 Perception system sensors
   1.3 Sensors for localization system.
   1.4 Procurement batteries and components.
2. AGV Basic hardware & software setup
   2.1 CAD design.
   2.2 Fabrication
   2.3 Mini-PC - ROS Setup
   2.4 Electrical integration sensors
   2.5 ROS sensors driver setup
3. AGV Sensor Calibration & Data Capture
   3.1 Outdoor testing with RC.
   3.2 Odometry data check
   3.3 LIDAR standalone test
   3.4 GPS Static Test
   3.5 IMU standalone test
4. AGV control
   4.1 Basic Control Stack
   4.2 Tele-op with Remote-PC
   4.3 GPS based navigation
5. Sensor Integration
   5.1 Sensor software development
   5.2 LiDAR obstacle detection
   5.3 Path Planning - virtual obstacles
   5.4 Localization - GPS + Odometry

UAV

1. System Component finalization
   1.1 Platform for UAV
   1.2 Procurement
2. Initial UAV platform testing
   2.1 Outdoor flight test with RC
   2.2 Lower level control with SDK
3. UAV Basic software setup
   3.1 Remote-PC - ROS setup
   3.2 ROS - lower level control.
   3.3 ROS sensors driver setup
4. UAV Sensor Calibration & Data Capture
   4.1 GPS Static Test.
   4.2. IMU standalone test
   4.3. Video stream to Remote-PC
5. Higher Level UAV control
   5.1 Basic Control Stack
   5.2. GPS based navigation
6. UAV intelligence
   6.1 Triggered take-off
   6.2. April tag based localization from video

System Integration

1. Communication & peripheral setup
   1.1. WiFi access point
   1.2. Boosters antennas
   1.3. Network
   1.4. System Communication layer
2. Integration test for fall demo
   2.1. Network reliability test
   2.2 AGV Tele-op test
   2.3 UAV Tele-op test
   2.4 April tag based-localization
   2.5 UAV GPS based navigation test
   2.6 System testing
3. Integration test for spring demo
   3.1. Path Planning with obstacle avoidance
   3.2. AGV Motion Planning
   3.3. Software stack integration
   3.4. AGV Video feed
   3.5. System GUI

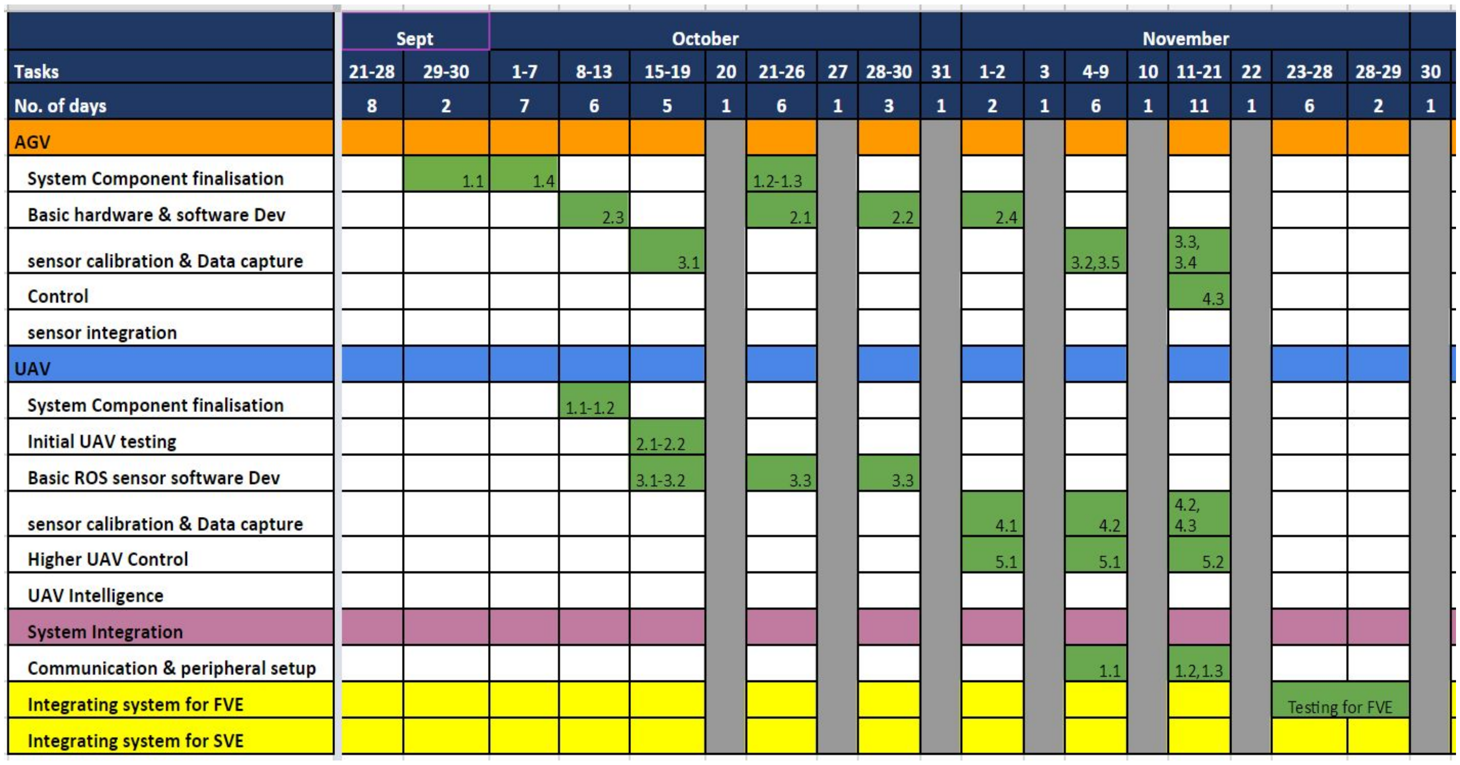
PROJECT MANAGEMENT

1. Prepare task Gantt
2. Procure Components
3. Communication to Stakeholders
4. Track Work Progress
5. Risk Mitigation

Figure 10: Detailed Work Breakdown Structure

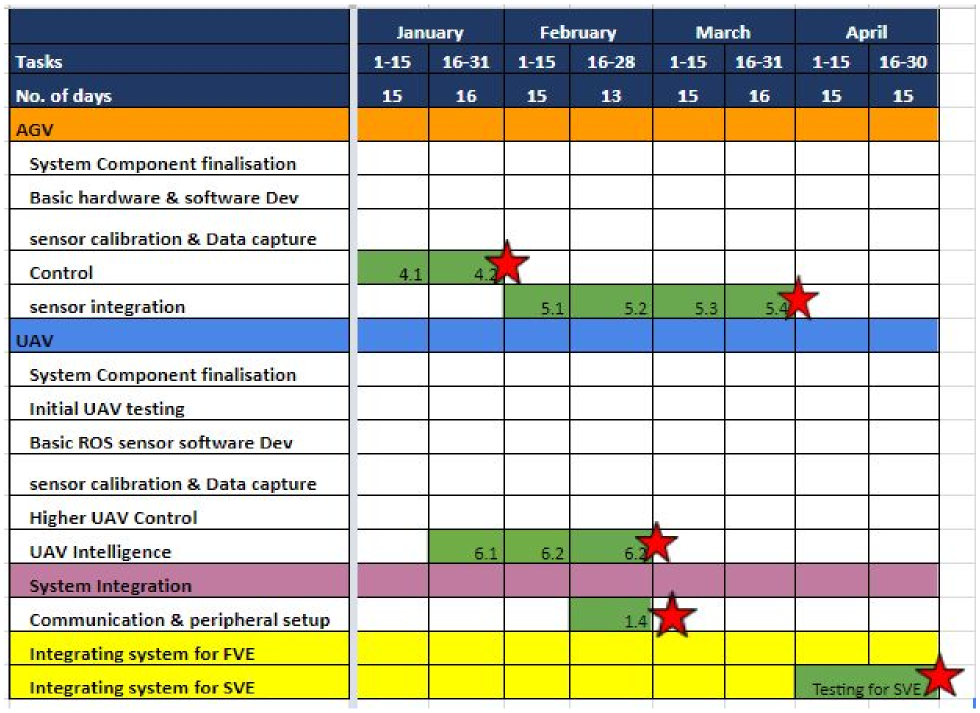
9.2 Schedule

Our schedule which references the WBS is shown below (Figure 9 and 10). In retrospect, we have remained mostly faithful to our original schedule until the last two months. Sensor calibration, integration and control for the AGV took more time than expected. We faced difficulties with consistent performance up until the two weeks before SVE. This pushed back our integration testing by half a month which was probably not enough time to fully test the entire system. Additionally, we did not fully understand the links between subsystems since we placed sensor integration after control in the AGV but clearly that should have came before. Something we did well was that we modularized our subsystems very robustly. When it came time for integration, and bugs or issues that we faced were mostly in the linkage stage. Lastly, our breakdown of the subsystems in the beginning remained relatively accurate to what we actually implemented. This in turn meant that we could follow our schedule as faithfully as possible since we did not need to reformat the schedule as the semesters went on.



	Sept		October								November								
Tasks	21-28	29-30	1-7	8-13	15-19	20	21-26	27	28-30	31	1-2	3	4-9	10	11-21	22	23-28	28-29	30
No. of days	8	2	7	6	5	1	6	1	3	1	2	1	6	1	11	1	6	2	1
AGV																			
System Component finalisation		1.1	1.4				1.2-1.3												
Basic hardware & software Dev				2.3			2.1		2.2		2.4								
sensor calibration & Data capture					3.1								3.2,3.5		3.3, 3.4				
Control															4.3				
sensor integration																			
UAV																			
System Component finalisation				1.1-1.2															
Initial UAV testing					2.1-2.2														
Basic ROS sensor software Dev					3.1-3.2		3.3		3.3										
sensor calibration & Data capture											4.1		4.2		4.2, 4.3				
Higher UAV Control											5.1		5.1		5.2				
UAV Intelligence																			
System Integration																			
Communication & peripheral setup													1.1		1.2,1.3				
Integrating system for FVE																	Testing for FVE		
Integrating system for SVE																			

Figure 11: Fall semester Gantt chart



	January		February		March		April	
Tasks	1-15	16-31	1-15	16-28	1-15	16-31	1-15	16-30
No. of days	15	16	15	13	15	16	15	15
AGV								
System Component finalisation								
Basic hardware & software Dev								
sensor calibration & Data capture								
Control	4.1	4.2 ★						
sensor integration			5.1	5.2	5.3	5.4 ★		
UAV								
System Component finalisation								
Initial UAV testing								
Basic ROS sensor software Dev								
sensor calibration & Data capture								
Higher UAV Control								
UAV Intelligence		6.1	6.2	6.2 ★				
System Integration								
Communication & peripheral setup				1.4 ★				
Integrating system for FVE								
Integrating system for SVE							Testing for SVE ★	

Figure 12: Spring semester Gantt chart, the stars represent the milestones

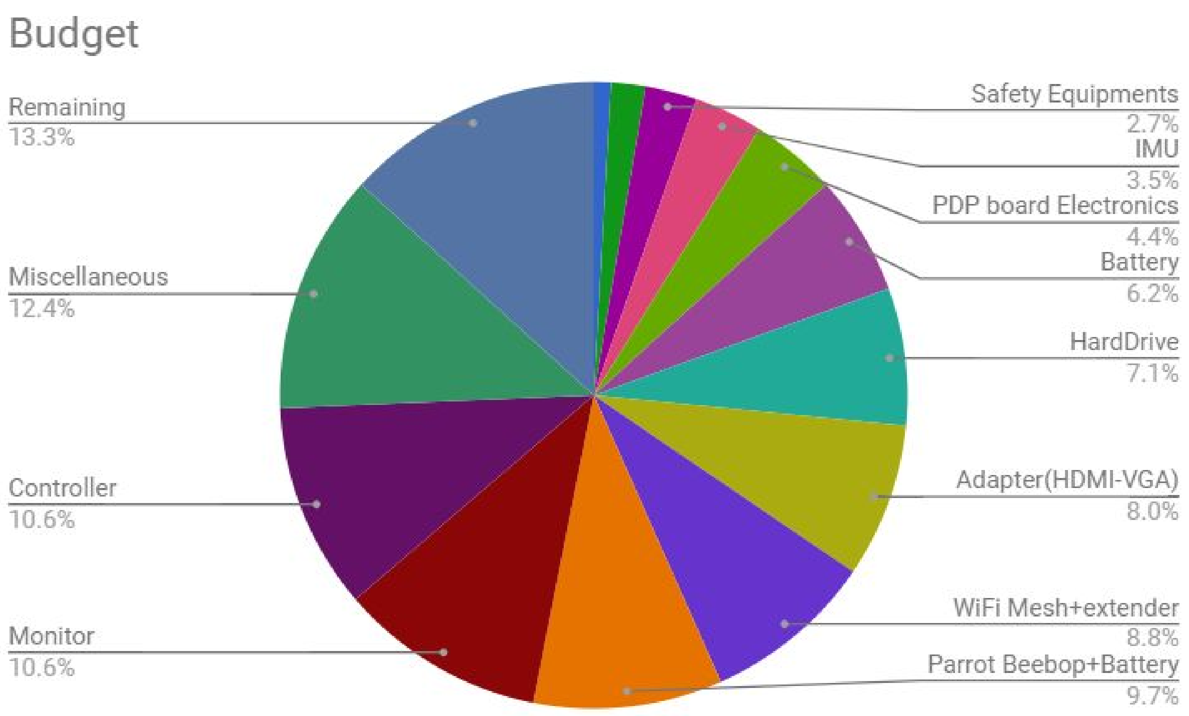
9.2 Budget

See Table 4 for a list of our purchase and Figure 13 for a visualization of our costs.

Table 4: Budget Breakdown

S.No.	**Components**	**Sub-Components**	**Q.**	**Unit Cost-$**	**Total Cost-$**
Sponsored Components					
1.	Parrot Bebop 2 (Aerial Drone)	-	1	400	400
2.	Clearpath Husky (Ground Bot)	Husky Robot	2		
		Li-Ion-Phosphate Battery	2		
		Battery Chargers	4		
		Zotac Mini-PC	2		
		GPS	1		
3.	Velodyne Puck	-	1	8000	8000
4.	WiFi-Router	-	1		
5.	USB-Camera	-	1		
6.	Monitor	-	1		
7.	Laptop	-	1		
Bought Components					
S.No	Components	Sub-Components	No of units	Total Amount	
1	Upgrades for Mini-PC	RAM	5	341.97	
		SSD	2	293.96	
		Mouse-keyboard	5	73.92	

2	External HDD	-	1	99.99	
3	Safety Equipments	Caution Tape	3	22.48	
		Drone catching net	3	41.98	
4	IMU	Razor IMU	3	99.85	
5	PDP board Electronics	-	-	166.53	Many Components
		Charger	3	46.99	
		cable	8	183.18	
7	Battery	-	2	71.98	
8	Hard Drive		2	393.95	
9	Adapter(HDMI-VGA)		4	47.98	
10	WiFi Mesh+extender		3	228.44	
11	Parrot Beebop+Battery		3	604.98	
12	Monitor		2	357.99	
12	Controller		1	37.11	
14	Miscellaneous		5	107.69	
	Total Spent			**2879**	**Total Budget: 5000**



Money Spent: $2879
Balance Left: $2121

Figure 13: Budget Breakdown

9.3 Risk Management (After FVE)

Table 5: Risks Identified before FVE

S.No.	Risk	L	C	Type	Mitigation strategy	Owner
1	Unexpected component requirement	2	4	Technical	Frequently conduct requirement reviews for major subsystems	Pulkit
2	GPS testing not possible at night	2	5	Technical	Shift work schedule of members working on drone to maximize daytime testing	Pulkit
3	Drone testing not possible at night	2	5	Technical	Shift work schedule of members working on drone to maximize daytime testing	Danny
4	Outdoor testing problems due to weather	4	4	Schedule	Setup indoor testing platform and test often	Pulkit
5	Sensor data not	3	4	Technical	Test all required sensors early and finish testing by FVE.	Rahul

	sufficiently accurate				Order new sensors if required.	
6	Insufficient ROS support available from Bebop online community	2	2	Technical	Take guidance from the Robotics Institute's members working with drones	Yuchi
7	Limited system network bandwidth	3	2	Technical	Compress images received from the cameras	Pratibha
8	System integration takes up lot of time	4	3	Schedule	Start system integration early and integrate subsystems whenever possible	Pulkit
9	Team Member unavailable due to sickness, personal matters etc	1	4	Schedule	Double Pulkit's workload	Pulkit
10	Husky localization not sufficiently accurate	5	4	Technical	Implement sensor fusion. Use visual odometry with LiDAR, increase April Tag size, use better GPS	Pratibha
11	Team Member unavailable due to sickness, personal matters etc	3	3	Schedule	Assign secondary holder to major tasks	Danny
12	Availability of Drone Testing Location and Licensing	1	4	Schedule	Acquire FAA licensing & acquire permissions to fly on campus	Yuchi
13	Network architecture setup with range 50m	5	4	Technical	Setup multi-master ROS network and improve the network quality	Pulkit
14	Safety issues when drone flying outside	4	4	Technical	While testing, use caution tape and drone capture net	Danny

We were able to mitigate most of the risks during FVE (Table 5), some of the system limitations we tackled during SVE.

1) Outdoor testing issues we solved mainly in SVE - until FVE we were facing issues due to really bad weather and we were facing constraints for outside GPS testing, we were able to find good location later on, at B-Level we realized that due to presence of too many structures the GPS, we were getting was really bad.
2) We further improved on network range during SVE.
3) Drone testing at night was a consistent issue, which was more of a hardware problem, we did some testing in night using floodlights, but we never preferred to fly in night due to safety issues.
4) Unavailability of team members, mostly due to overwhelming coursework was also a risk we tried to mitigate using better time management, but that was also a constant constraint.

5) We tested and calibrated all sensors separately and tested them to ensure that all of them are giving accurate data. We faced a lot of issues in integrating IMU and GPS, it took us more than the anticipated time. Mainly during integrating the subsystems we were able to realize more issues with respect to these sensors, and their integration issue extended till SVE.
6) We had some issues related to beebop connecting to network, as it is designed to act as a host by default, this was a issue for which there was not much support available online, but the team was able to resolve this too before FVE.

Risk Before SVE

Table 6: Risks Identified before SVE

S. N	Risk	L	C	Type	Mitigation strategy	Owner
1	Drone testing not possible during night and in bad weather	4	5	Schedule	Optimize weekends and good weather for testing	Danny
2	Drone getting stuck in Trees (not wanting to come down)	4	3	Technical	Buy more drones and allocate reserve budget	Yuchi
3	Limited system network bandwidth	3	2	Technical	Analyze offline planning	Pratibha
4	System integration takes up lot of time	4	3	Schedule	Start system integration early and integrate subsystems whenever possible	Pulkit
5	Availability of Drone Testing Location and Licensing	1	4	Schedule	Acquire FAA licensing & acquire permissions to fly on campus	Rahul

We were able to plan and mitigate all the risks we identified for SVE (table 6). The mitigation strategy was as follows:

1) For testing outside in bad weather, we brought two big canopies, so that we can bring the stuff inside the canopies during shot rains and then continue testing after that subsides.
2) We brought two extra drones, because we lost two drones. We allocated budget very carefully till the end, even we were able to save 13% of our budget.
3) We made two WiFi long range meshes to enhance the range and communication between our subsystems (AGV and UAV).
4) We did an extensive testing of our combined system.
5) We also registered our drone, to ensure that we can safely fly it on the campus.

Figure 14 shows the Risk Analysis table for our complete set of risks.

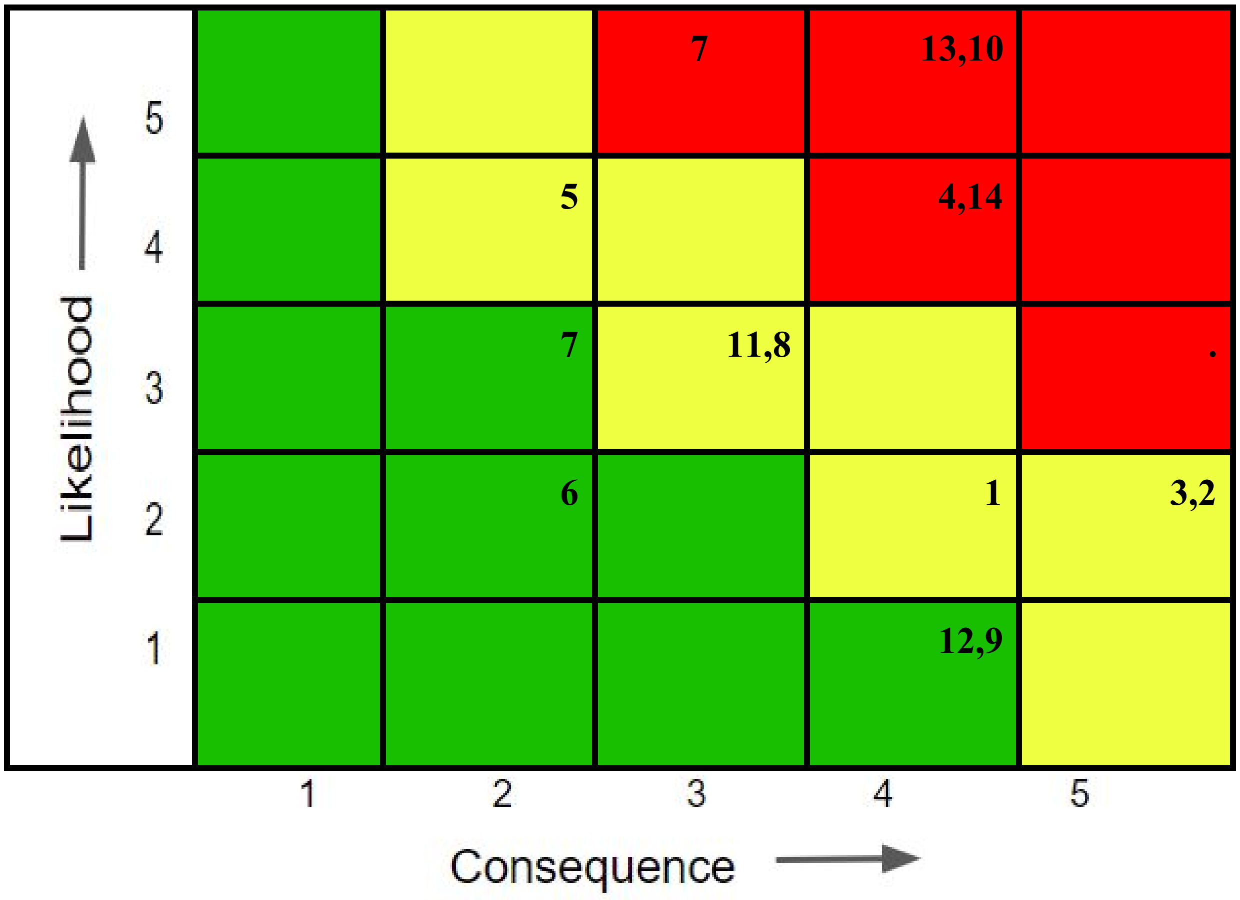


Figure 14: Risk Analysis

10 Conclusions:

In this report, we have presented a project that aims to combine UAV autonomy with AGV navigation in unexplored environments. In the end, we have demonstrated that our proposed heterogeneous system is able to traverse a simplified and constrained unknown environment faster than a AGV-only system. Along the way, we have faced some key technical and collaborative challenges. Here we summarize the most major difficulties and how we overcame them.

10.1 Lessons Learnt

In terms of technical difficulties, it must be said that most of the system relied heavily on technical knowledge that we did not learn until the end of the first or second semester. These included knowledge such as EKF implementation, ROS subtleties, transforms and coordinate systems, and general computer vision. The transform and coordinate system knowledge was key to understanding how to use April Tag ROS while EKF was required to debug the AGV's ground navigation. Although in the end we were able to address all of these difficulties, had we known the details in the beginning, we could've undoubtedly chosen more appropriate solutions to our problems. This is discussed in more detail further below.

In terms of collaborative and teamwork challenges, we faced interpersonal roadblocks like any other team. Perhaps the biggest factor is the mismatch of our coursework schedule. For both semesters, 4 people in the team were taking one set of courses while the last member (Yuchi) was taking another set. This meant that getting the entire team together to perform testing proved to be a rather challenging endeavor. We addressed this issue by assigning most of the UAV development to Yuchi so he could work independently. Another challenge was distributing the workload with regards to the different level of technical expertise amongst our group members. Our project is mostly software based and thus we had to find a balance between letting everyone on the team learn new material regardless of their background. We believe that in the end, we are all very satisfied with the amount of experience we acquired.

10.2 Future work

There are many improvements that could be done if we were to continue this project or do it again from the beginning. On the UAV side, we would've chosen a better drone - preferably one with a more accurate GPS, higher resolution camera, and a configurable loadout. We would also leverage the computer vision knowledge we acquired to not rely on April Tags for localization. On the Husky side, perhaps implementing the EKF on a lower level would give us finer control of our localization as opposed to using a inflexible prebuilt package. Team management can also be improved. We should have a centralized git repository and assign someone as team captain to delegate tasks as opposed to going at our own individual paces.

References

[1] http://library.isr.ist.utl.pt/docs/roswiki/Robots%282f%29Husky.html . [2] http://velodynelidar.com/vlp-16.html .

[3] https://www.clearpathrobotics.com/tag/husky/ .

[4] https://www.amazon.com/Ubiquiti-BULLET-M5-HP-Outdoor-802-11n-M5HP/dp/B002SYTPMU . [5] MRSD Fall 2016 Conceptual Design Review document of Team Rescue Rangers (F).