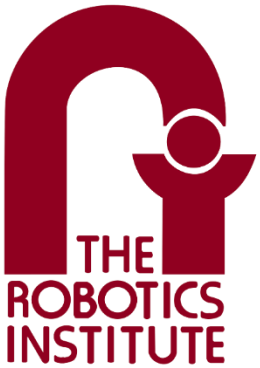
Critical Design Review

Team E: Beyond Sight
Environmental Sensing Infrastructure for Autonomous Driving

Team Members:
Rohit Murthy
Vivek GR
Oliver Krengel
Chien Chih Ho
Peng Sheng Guo

December 14, 2017



Abstract

This report discusses the progress made on the project to develop an environmental sensing infrastructure to enable autonomous vehicles to navigate safely through urban environments. The discussions focus on two major portions of a project: the technical details and the project management methodologies that have enabled us to make good progress in this fall semester. We first detail the motivation for the project and the specific use case against which we will validate the project requirements. We then explain the requirements that drive the design and development of the system. The different components of this system are outlined in the functional and cyberphysical architectures in the next section. In this fall semester we have been able to make significant inroads in the detection, tracking, prediction and electrical subsystems which is detailed in Section 6.3. In order to meet the requirements for our fall validation, we also performed rigorous testing which is documented in this section as well. The project management plans are discussed in Section 7 with focus on the work to be completed in the spring semester in order to meet the requirements for the spring validation experiments. This also includes the important risks that we have been tracking and will track in the future as well. Conclusive remarks on our achievements and key goals for spring are outlined in the last section of this report.

Contents

1. Project Description 1

2. Use Case 2

3. System-Level Requirements 3
   3.1 Functional Requirements 3
   3.2 Performance Requirements 3
   3.3 Non-Functional Requirements 4

4. Functional Architecture 5

5. Cyberphysical Architecture 6

6. Current System Status 8
   6.1 Fall System Requirements 8
   6.2 Overall System Depiction 9
   6.3 Subsystem descriptions/depictions 9
      6.3.1 Vehicle 9
      6.3.2 Pedestrian Detection 11
      6.3.3 Pedestrian Tracking 13
      6.3.4 Trajectory Prediction 13
      6.3.5 Electrical Subsystem 14
   6.4 FVE performance evaluation 15
   6.5 Strengths and Weaknesses 16

7. Project management 17
   7.1 Work Breakdown Structure 17
   7.2 Schedule 17
   7.3 Test plan 18
      7.3.1 Progress Review Milestones 18
      7.3.2 Spring Validation Experiment 19
   7.4 Budget status 20
   7.5 Risk management 21

8. Conclusions 22
   8.1 Lessons learned 22
   8.2 Key spring activities 23

9. References 23

1. Project Description

Every year in the United States, approximately 2.5 million accidents are reported at intersections, as reported by the Federal Highway Commission [1]. The same agency reports that intersection accidents account for 40% of all crashes. Worse, 50% of all serious collisions and 20% of all fatal collisions occur at intersections.

A study conducted by the National Highway Traffic Safety Administration found "obstructed view" as a primary reason for intersection accidents across drivers of all ages and genders [2]. This will come as no surprise to any driver, though. Intersections pose difficulties that drivers do not experience on 2 lane roads, such as:

- Timed signals to monitor
- Cars from the side turning in front of driver
- Cars from in front turning in front of driver
- Cars stopping in front of driver to turn
- Pedestrians crossing in front of driver
- Pedestrians crossing beside driver

With all these reasons accounted for, it is no wonder how many accidents occur at intersections. In many cases, drivers simply have too many obstacles to account for at any given time. The proliferation of self-driving cars may assist vehicles at intersections in accounting for many obstacles simultaneously, but the problem of occluded viewpoints remains. With only car-mounted sensors, vehicles driving through intersections may not be able to detect obstacles such as crossing vehicles and pedestrians if their view is occluded by larger vehicles. The use of information gathered on sensors outside the vehicle will therefore be necessary to solve the occlusion problem.

To solve the occlusion problem, we are proposing an infrastructure system at intersections developed around live 3D video capture. This system will be able to detect moving objects and predict their movements. Furthermore, the system will be designed to interface with vehicles approaching and passing through intersections, so that it can communicate with these vehicles. By utilizing path prediction and communication, the system will be able to detect would-be collisions in advance and alert vehicles, thus preventing collisions.

This project will develop a LiDAR/Camera-based system which can detect and track pedestrians and communicate with autonomous vehicles. The system has been tested on a controlled intersection with a known occluded space and is now being generalized to a four-way intersection. Our hope is that the technology we develop can be put to use in the future at all intersections.

2. Use Case

Andy is driving to work along his normal route, down Forbes avenue through Oakland. He is attending to the road but he is in autopilot, it is early in the morning and he has made this commute hundreds of times. He turns onto Bellefield then makes a left on Fifth Avenue. His car is alone in the left lane clear to the intersection at Bouquet, where the light is red. He begins to slow down 100 yards away when the light turns green.

Andy accelerates to pass the bus (shown below in orange) when a man steps out in front of the bus, 15 feet in front of his car. The car comes to an immediate halt as the pedestrian freezes in the intersection. Andy's car is 2 feet from the man, but his foot didn't reach the brake until the car was 5 feet away. As he breathes a sigh of relief, Andy wonders why his car stopped…

5 seconds earlier, when Andy was approaching the intersection, his car's computer had made a wireless connection with a LIDAR system monitoring the Fifth Avenue-Bouquet intersection. The system continuously detects vehicles, pedestrians, and other moving objects in the vicinity of the intersection. This system is the blue puck with red bar displayed at the top of the schematic.

2 seconds earlier, a man had jumped out of the bus and made a quick turn in front, hoping to cross the street before the light turned. The LIDAR system had tracked the man's movements and predicted that his path would move in front of Andy's car. The system instantaneously sent a signal to his car's computer, alerting its obstacle avoidance system to the pedestrian about to be in front of the car.

4 tenths of a second earlier, Andy's car had automatically braked, 1 tenth of a second before Andy had a view of the man and 3 tenths of a second before he could have applied the brakes. Thanks to the LIDAR system that monitors the intersection, Andy's car avoided an accident that neither he nor his car would have been able to prevent on their own. The entire situation is depicted in a simplified form in figure 1 below.

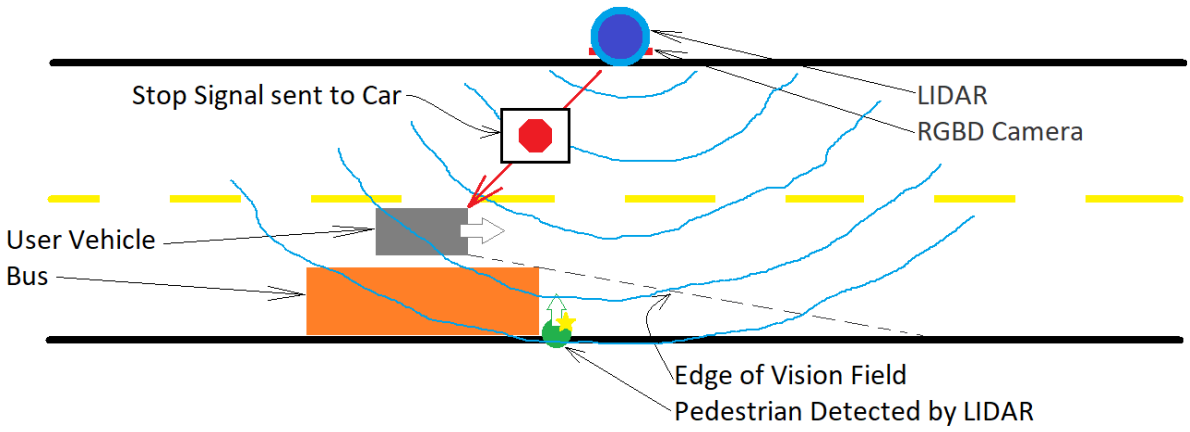


Figure 1. System graphical representation.

3. System-Level Requirements

The System level requirements are driven by our objective of making intersections safe. We do this by preventing collisions between vehicles and pedestrians.
The system-level requirements are categorized as:

a) Functional (F)
b) Performance (P)
c) Non-functional (NF).

Where [M.] denotes the requirement is mandatory.

3.1 Functional Requirements

Table 1. Functional Requirements

ID	Title
	INFRASTRUCTURE:
M.F.1	Detect pedestrian
M.F.2	Track pedestrian
M.F.3	Predict trajectories of pedestrians
M.F.4	Publish trajectories to autonomous vehicles
	VEHICLE:
M.F.5	Subscribe to pedestrian trajectories
M.F.6	Avoid collisions with pedestrians before they enter the vehicles' field of view

3.2 Performance Requirements

Table 2. Performance Requirements

ID	Title	Description
		INFRASTRUCTURE:
M.P.1	Detection	Detect single pedestrian centroid with Euclidean distance error < 0.3m
M.P.2	Tracking	Track single pedestrian within 20m of the infrastructure

M.P.3	Prediction	Predict single pedestrian trajectory 1.2 seconds into the future with an average error of 0.5m
M.P.4	Cycle Time	Time between first frame with a pedestrian to first published trajectory should be less than 1 second.
		VEHICLE:
M.P.5	Vehicle	Stop short of single pedestrian before he/she enters the field of view

3.3 Non-Functional Requirements

Table 3. Non-Functional Requirements

ID	Title	Description
M.N.1	Stability	Shall be physically stable
M.N.2	Electrically Isolated	Shall be isolated electrically
M.N.3	Maintainability	Shall be easy to move and maintain
M.N.4	Testing	Shall be safe for testing
M.N.5	Regulations	Shall adhere to strict university and legal regulations

4. Functional Architecture

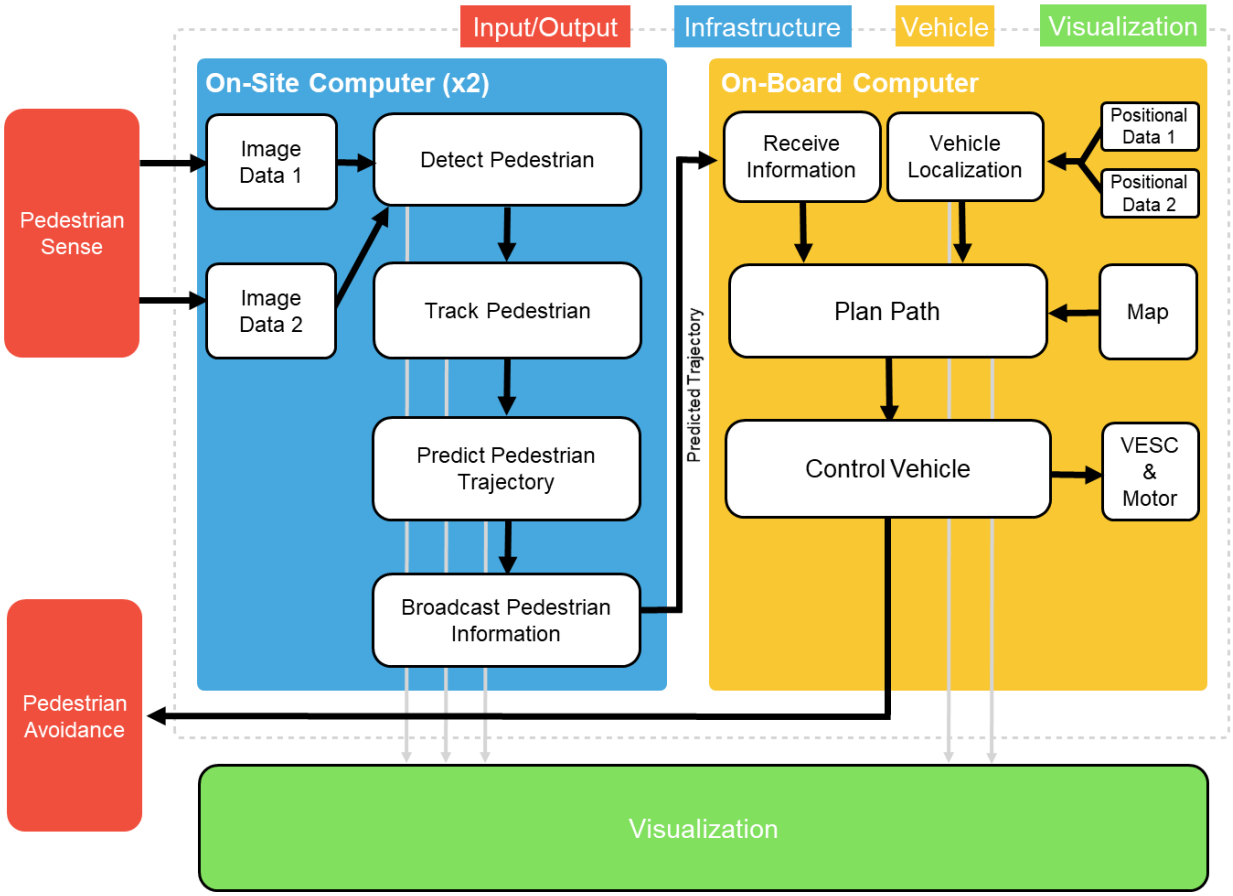


Figure 2. Functional Architecture

The Functional architecture for the project is visualized in the Figure 2 above. Our project involves two major subsystems in order to realize our project successfully.

1. **Infrastructure:** The inputs to the system are pedestrians who are within the twenty-meter range of our infrastructure. They will be detected by our detection module from image sensors. Then the tracking module will track each of the pedestrians and pass this information to the trajectory prediction module. After generating the future points of each pedestrian, the information is broadcast to the vehicle.

2. **Vehicle**: The vehicle will receive the broadcasted trajectory information, which is used as an input to the control module. In the meantime, car will localize itself with onboard sensors and pre-stored global map. The motion planning module will take all map and localization information to generate a feasible path. Control module will control the vehicle and avoid any possible collisions with pedestrians. All the process described above including pedestrian detection, prediction, localization and path planning is visualized in RViz.

5. Cyberphysical Architecture

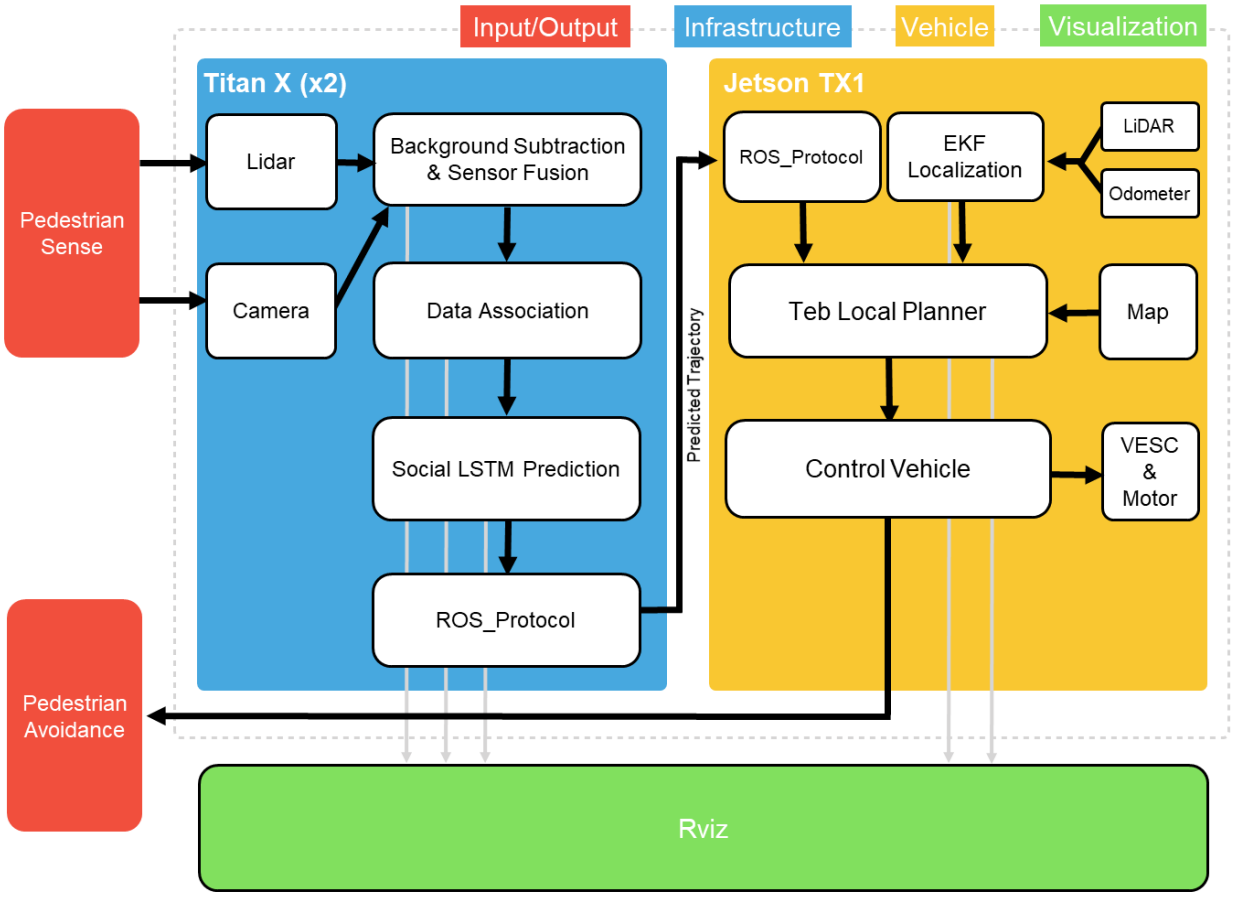


Figure 3. Cyberphysical Architecture

The Cyberphysical architecture for the project is visualized in the Figure 3 above. There are two modules in our system: infrastructure and vehicle.

1. **Infrastructure**: The "image data 1" and "image data 2" blocks from the Functional Architecture become Lidar and Camera. These are mounted on the infrastructure. Data is sent to the workstation (Titan X) through an ethernet cable. The workstation synchronizes Lidar and RGB information, then compares and subtracts foreground points from the pre-built background. After that, foreground points are clustered into groups and the centroids of each group is calculated as a pedestrian's position. However, if there are two targets too close to each other, the system might mistakenly cluster two people into one group with a single centroid. To avoid this, the system fuses the semantic information from the RGB image with bounding boxes to correct the number of groups using single-shot multi-box detector. After getting multiple target centroid coordinates, the information is passed to the tracking system. The tracking system uses a Hungarian algorithm to track the associated targets. Finally, the system predicts the trajectories of the tracked targets using Social LSTM [5] and sends the trajectories to the vehicle via Wifi. All messages are transmitted within a ROS environment.

2. **Vehicle**: The vehicle is embedded with Lidar, Jetson TX1 (embedded computer), VESC (Vector Electronic Speed Controller), and servo motors. It localizes itself using an Extended Kalman Filter given a map and laser range data. After the trajectory is passed to the vehicle through the ROS protocol, the vehicle will stop if any of the predicted pedestrians are within its path.

The algorithm flowchart is depicted below:

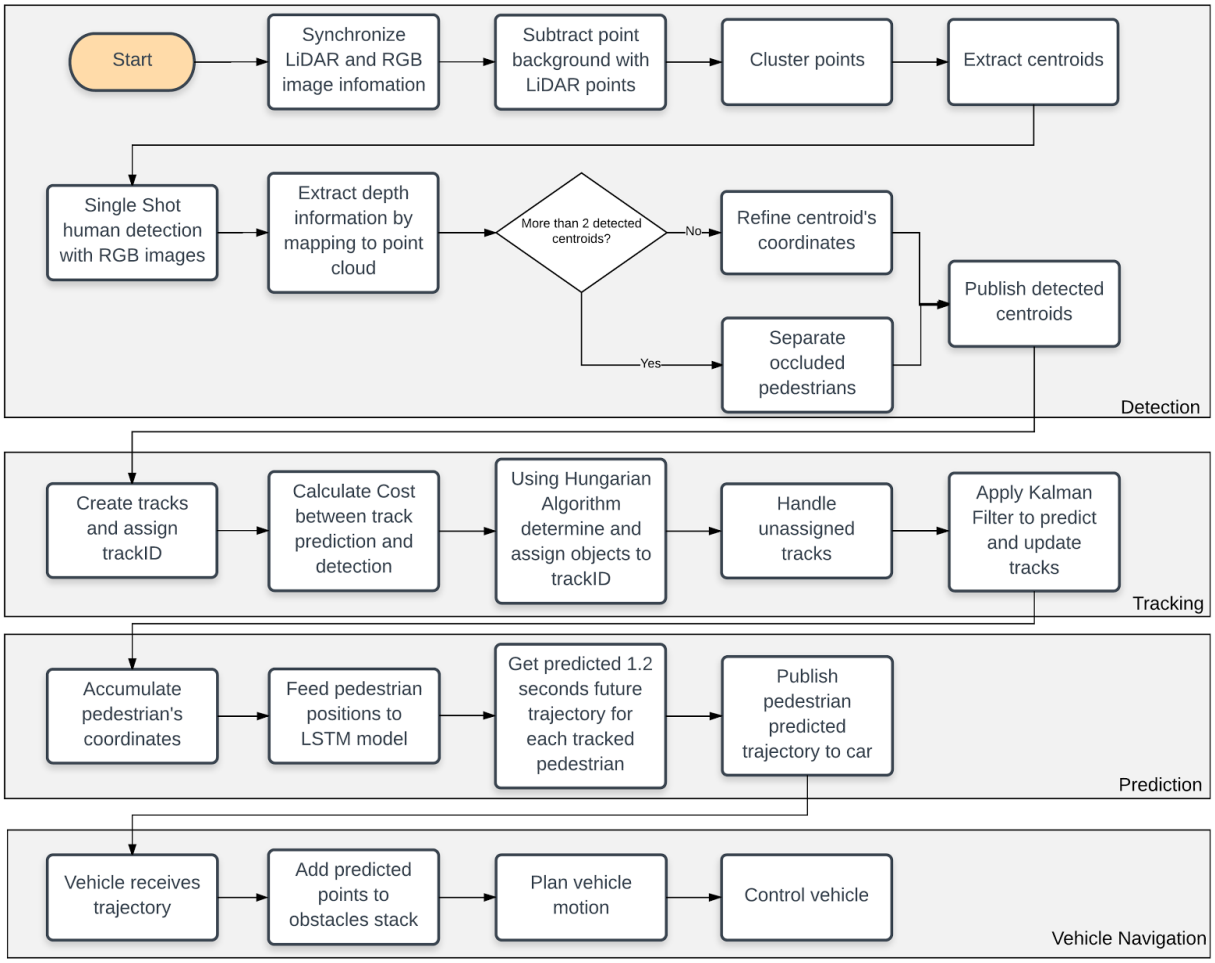


Figure 4: Algorithm Flowchart

6. Current System Status

6.1 Fall System Requirements

Table 4. Fall System Requirements

Test ID	Description	Requirement
	INFRASTRUCTURE	
Test 1	Detect single pedestrian centroid with Euclidean distance error < 0.3m	M.P.1
Test 2	Track single pedestrian within 20m of the infrastructure	M.P.2
Test 3	Predict single pedestrian trajectory 1.2 seconds into the future with an average error of 0.5m	M.P.3
Test 4	Time between first frame with a pedestrian to first published trajectory should be less than 1 second.	M.P.4
	VEHICLE	
Test 5	Stop short of single pedestrian before he/she enters the field of view	M.P.5

6.2 Overall System Depiction

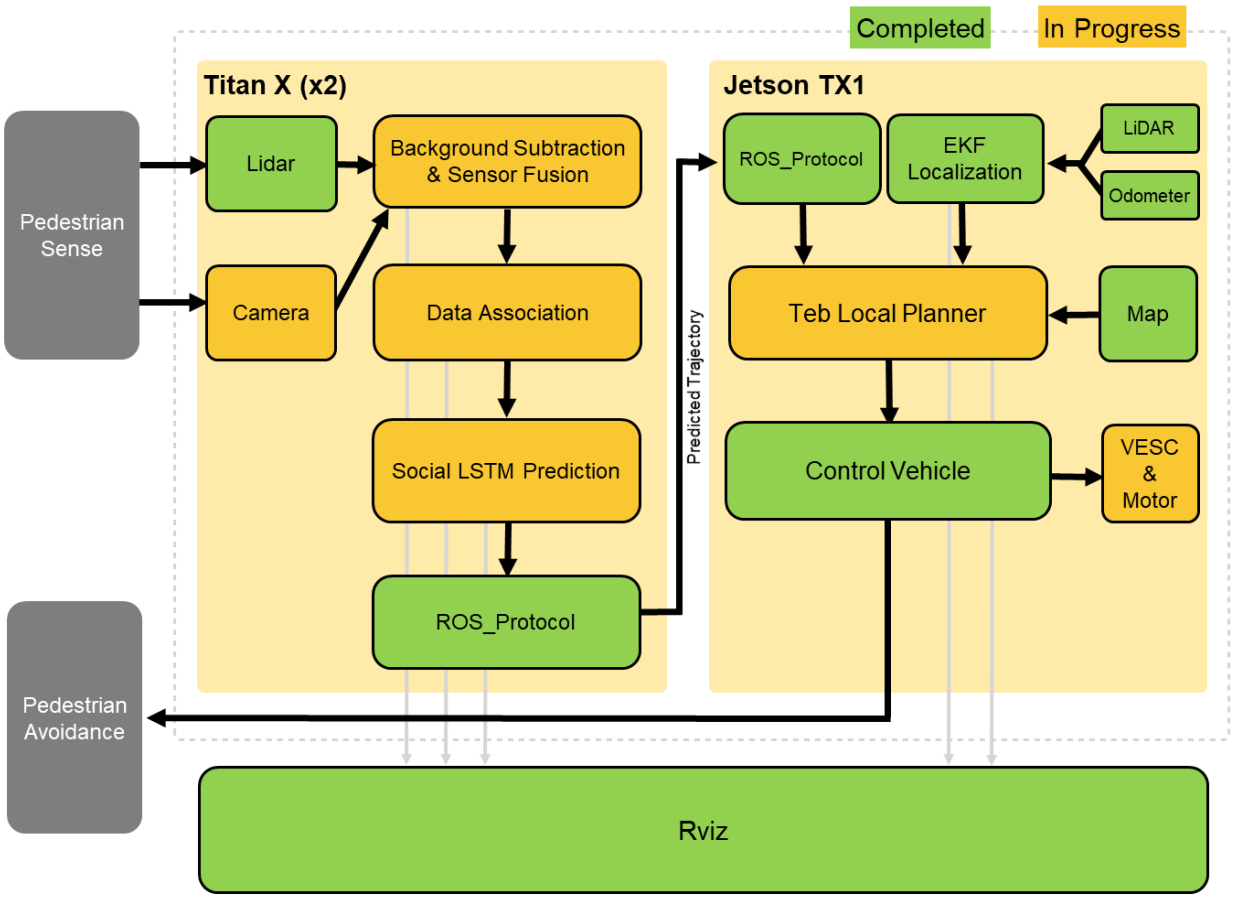


Figure 5: Current Overall System Status

6.3 Subsystem descriptions/depictions

6.3.1 Vehicle

Our vehicle subsystem for the fall semester is a miniature autonomous vehicle built by the D team last year [3][4]. We have also outfitted the vehicle with a thin wire "bumper" to increase the scale in relation to humans. The vehicle is built with a chassis from a hobby RC car and is controlled with an onboard Jetson TX1. The TX1 connects to a Teensy microcontroller which commands the vector electronic speed controller (VESC) that controls the motors of the vehicle. It also connects to a passive wireless receiver, Hokuyo LiDAR, USB hub, IMU, and power distribution board (PDB). The PDB receives power from an 11.1V lithium polymer battery and distributes it between the active components. The vehicle can be seen in Figure 6 below.

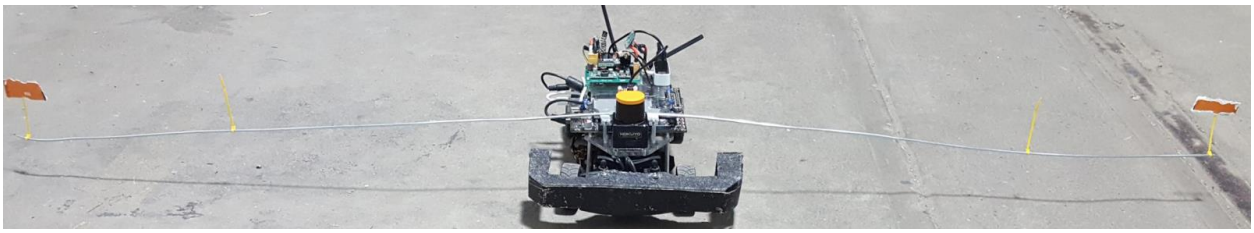


Figure 6. Vehicle subsystem

The ROS visualization tools provide an excellent view of how the car's subsystems work together to produce a working system. The rqt_graph of the fully functional system can be seen in Figure 7 below.

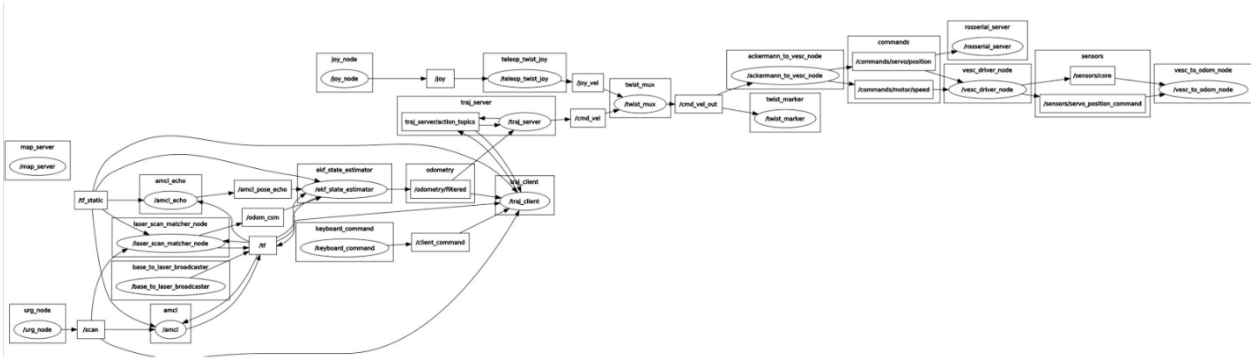


Figure 7. Vehicle rqt graph

The bottom left of this rqt_graph shows how the IMU and laser scanner feed into an adaptive monte carlo localizer (AMCL), then into a an extended Kalman filter to provide state estimation. This information is published as odometry information which is used by both the trajectory server and trajectory client. The trajectory server sends commands through the multiplexer to the Ackermann controller, which controls the motor and steering. Left out of that description is the keyboard command, which feeds into the trajectory client. The command '*a'* activates the ramp protocol, which moves the vehicle to a user-defined goal point. Specifically, this references a .yaml file (bsilqr_params.yaml) which has a predefined goal location (along with many other parameters). This planning method is executed by the trajectory client and server, which utilize the ROS action library to steer the robot to the goal.

Missing from the above graph is the connection to our infrastructure. When working together, the trajectory client subscribes to the "/predicted_points" topic which carries the PoseArray message that contains pedestrian trajectory data. The trajectory client includes a node that will stop if this pedestrian data is determined to be in the vehicle's path.

By connecting to a ROS core running on the vehicle's TX1, we are able to view the "odometry_filtered" and "scan" (from the Hokuyo) topics on the vehicle's map. Since the system came pre-built, we did not perform any modelling on the vehicle.

The analysis came in a rigorous poring over of the ROS nodes when the vehicle was active to determine how the subsystems worked together. Since our communication needed only to interface with the trajectory client, contained within the "ilqr_loco" package, this package was

the primary area of focus. Toward the end of the semester, when we started interfacing with the infrastructure more regularly, the ekf used for state estimation was analyzed as well.

We performed extensive testing on the vehicle in order to perform our fall validation experiment without failure. What you can see in Figure 8 below is the vehicle's odometry with respect its initial position (odom frame), the velodyne frame, and a human in its path.

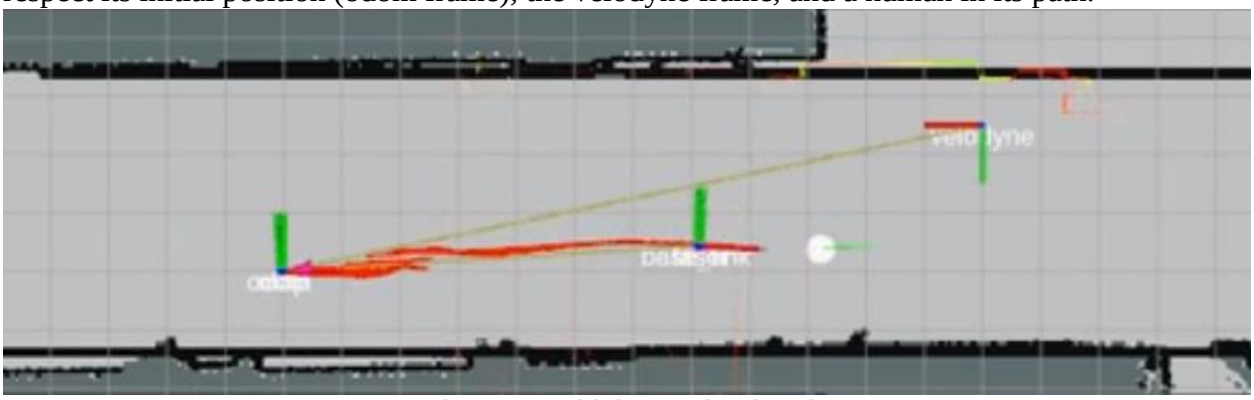


Figure 8. Vehicle running in rviz

The ability to visualize the vehicle was useful for the many iterations we performed changing the map of our test environment, running the vehicle to its waypoint, and testing the vehicle with live pedestrians. Primarily, this feature made it easy for us to determine where the vehicle believed it was with respect to the environment and the LiDAR.

6.3.2 Pedestrian Detection

The detection module is for detecting pedestrians within a 20-meter range of the infrastructure. The difference between detected pedestrian coordinates and the ground truth should be within 0.3 m based on the FVE performance requirement.

The module is able to detect pedestrians using LiDAR point cloud data. By subtracting the raw point data with pre-stored dense background, we are able to get all the foreground points and divide them into different groups using Euclidean clustering method. Centroids of each point group are then calculated as the centroid for each pedestrian.

The unit test for detection module is conducted by assigning human agent to 16 of the pre-measured points. The errors are calculated as the Euclidean distance between the ground truth and detected coordinate. All the testing data are shown below.

Table 5: Detection Performance Test

Point No.	Point Coordinate (m)	Detected Coordinate (m)	Error (m)
1	2,1	2.00,1.00	<0.0001
2	4,1	4.00,1.02	0.0200
3	6,1	5.99,1.02	0.0224
4	10,1	10.02,0.99	0.0224
5	15,1	15,0.98	0.0200
6	21,1	20.92,1.00	0.0800
7	21,3.3	20.85,3.34	0.1552
8	9,3.8	8.98,3.57	0.2309
9	8,3.8	8.04,3.72	0.0894
10	3,8	3.08,7.93	0.1063
11	3,6	3.05,5.94	0.0781
12	3,4	3.06,3.95	0.0781
13	3,2	3.03,1.98	0.0361
14	0,1	0.03,1	0.0300
15	-7,4	-6.86,4.09	0.1664
16	4,8	4.05,7.90	0.1118

The mean Euclidean distance error is 0.0779m. This proves that the detection module achieves a decent result and meets the FVE requirement.

6.3.3 Pedestrian Tracking

The tracking module is for tracking multiple pedestrians within a 20-meter range of the infrastructure. It should reliably track each of the existing pedestrians and handle new ones who are stepping into the range for the first time.

The module is currently able to track a single pedestrian reliably. The module utilizes the information from the prediction module and associates the nearest detected points to the predicted position.

The unit test for the tracking module is conducted by assigning human agent to walk between coordinate (20, 0) to (20, 3) repeatedly. The detected points' y-coordinates versus time frame are stored and visualized. The optimal shape of the graph should be sine-wave like and y-coordinate value should be within the range of (0.3, 2.7). The tracking test graph is shown in figure 9 below.

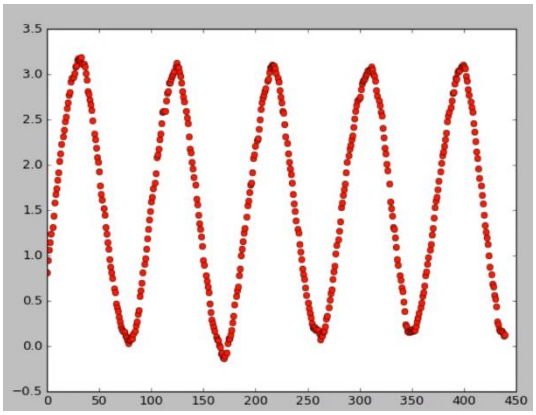


Figure 9. Tracking Test

6.3.4 Trajectory Prediction

The problem of trajectory prediction is essentially one of sequence prediction where, given a sequence of input coordinates (of a moving pedestrian), we must identify a pattern and use that pattern to predict the future sequence of coordinates of the same pedestrian. This subsystem forms the crux of our entire system since the output of this subsystem is published to the vehicle.

We have made some good progress on this subsystem although we had to settle for a different approach than the one we had initially anticipated using. The Social LSTM algorithm is the state-of-the-art in pedestrian trajectory prediction. However, it suffers from difficulties in debugging and dataset biases. Moreover, the algorithm was unsuitable for our requirements in this semester since we are focusing on a single pedestrian use case whereas the true value of the Social LSTM algorithm is found in cases with multiple pedestrians.

Hence, we shifted to a more deterministic approach using polynomial regression. Based on an observation window of 12 frames (at a frame rate of 10 frames per second), we fit a polynomial curve to input values. Using this curve, we extrapolated for the desired 12 frames (at the same frame rate) and published this as our predicted trajectory.

For the fall semester, we proceeded with a second degree polynomial fit. We tested this algorithm on live pedestrian data and were satisfied with the results that we were getting. The graphs in figure 10 below show the performance for some specific pedestrian trajectories.

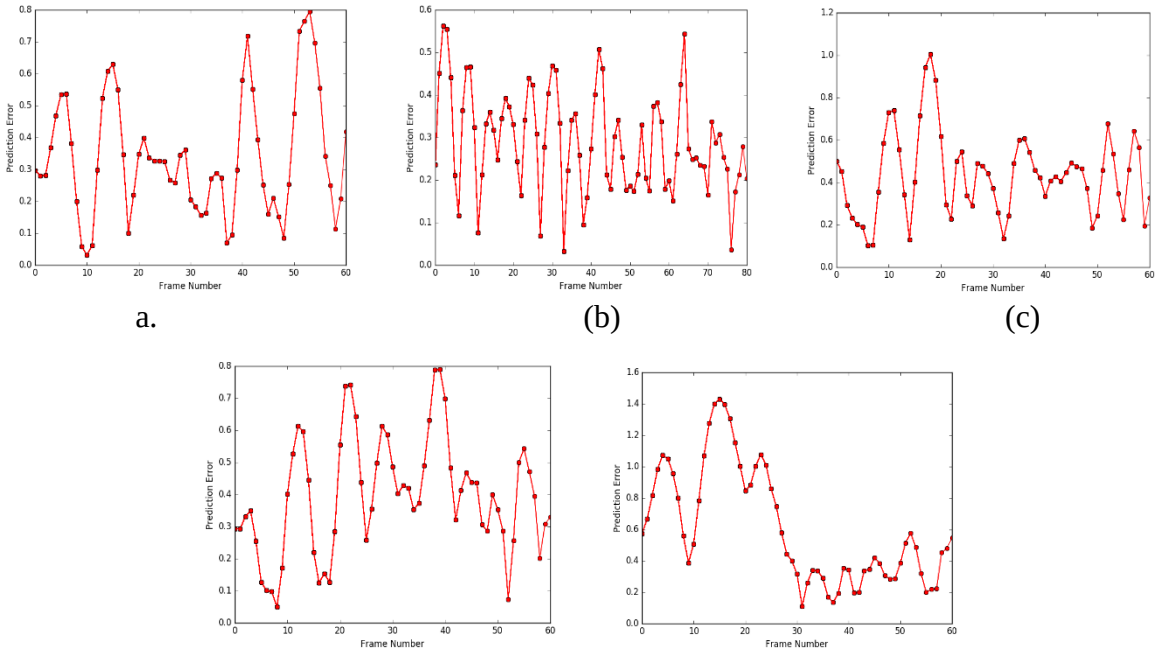


Figure 10. Graphs of trajectory prediction for trajectories of different radii of curvature (r). (a) r = ∞, (b) r = 3m, (c) r = 2m, (d) r = 1m, (e) r = 0m (right angle)

6.3.5 Electrical Subsystem

The Power Distribution Board (PDB) was designed to power the LIDAR, Jetson TX2, WiFi router, and ZED camera from an 11.1V battery. The PDB involves overvoltage, overcurrent, and reverse voltage protection. A block diagram representing the current electrical system is shown in figure 11 below.

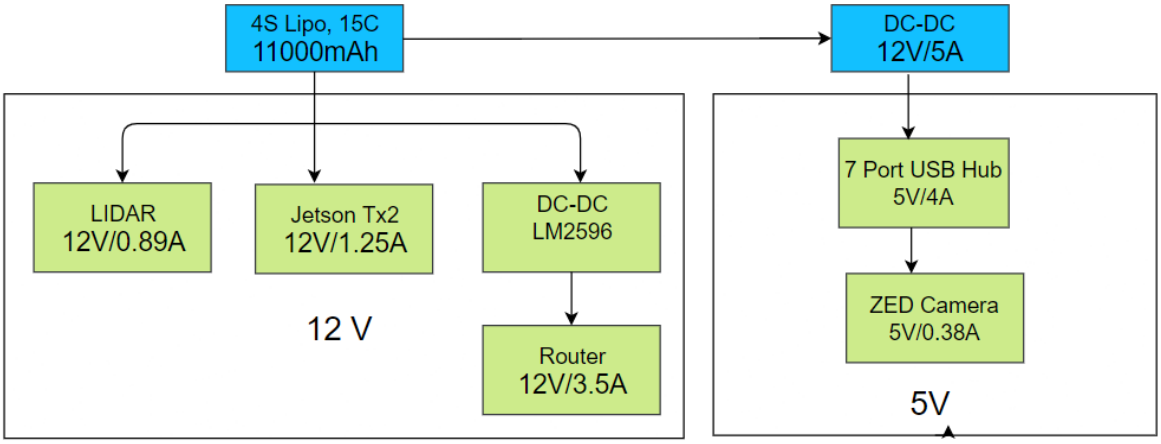


Figure 11. Electrical System

The PDB is operational and below are the results for the test conducted on the PDB.

Table 6. PDB Testing

Device	Test Input Voltage (V)	Input Current Capacity(A)	Rated Output Voltage (V)	Observed Voltage(V)
VLP-16 LIDAR	11.1	0.89	11.1	11.09
Jetson TX2	11.1	0.95	11.1	11.09
Router	11.1	1.5	11.1	11.02
Zed Camera	11.1	0.4	5	5

6.4 FVE performance evaluation

The FVE and FVE encore were successful because we validated all of the requirements mentioned in Section 6.1. Looking at it in greater detail, the detection subsystem worked remarkably well given that we were working just with the LiDAR. The average detection error of 0.09m was well below the required 0.3m (M.P.1), working well even for the range of 20m (M.P.2). The trajectory prediction was a very basic implementation, but it worked well for the FVE as our error of 0.45m satisfied the 0.5m error according to M.P.3. Our cycle time came out to 0.19 seconds, which was significantly less than the 1 second promised (M.P.4). This will probably be tougher to meet in the spring with more computationally demanding algorithms and less processing power. The last test, which involved the entire system, worked sufficiently. Unfortunately, it lacked the repeatability that all the other subsystems had. We were able to get the car to stop (M.P.5), which satisfied the requirement. However, it required careful calibration and tuning that should not have been necessary.

6.5 Strengths and Weaknesses

Strengths

a. Detection Accuracy: The detection accuracy tested below 0.1m. This was excellent, considering our requirement of less than 0.3m. We are particularly happy with this because of the amount of noise in the point cloud data.
b. Good Cycle time: Time taken between when the first pedestrian is detected to the first published trajectory was around 0.19 seconds. This is a terrific refresh rate for our system.
c. Controlled Environment: We are very happy with how smooth our validation experiments went in contrast with the frequency of interruptions during development. This is because we did a good job controlling our surroundings and preventing interruptions.
d. Graphical User Interface (GUI): The GUI developed for demonstrating our performance requirements showcased our requirements clearly and effectively.
e. Platform: The initial prototype of our project is complete. This is a strength because it will serve as a platform for us to improve on our shortcomings.
f. Team: We have an excellent team with a wide range of skill sets. This is a key strength that helped us in developing this project in a very short span of time.

Weaknesses

a. Localization: The localization done by the RC car was not to our expectations. There is great room for improvement here, which we plan to do for the SVE.
b. Communication: Intermittent loss of WiFi signals made the communication between the infrastructure and the RC car difficult.
c. Robustness: The system at present is not that robust to the noisy data coming from the sensors. Our goal is to make it so robust that the system is as good at detecting and tracking humans as other humans.
d. Background Registration: The physical infrastructure is not fixtured firmly enough to prevent the high frequency vibration in the Velodyne from shaking it. Our aim for the SVE is to make it totally stable.

Opportunities for improvement

The range of the detection algorithm satisfied the requirements that we had defined but the performance dipped towards end of the range. This could be problematic going into spring. Hence, we intend to solve the issue with fusion of camera data.

The trajectory prediction subsystem worked well and was able to recover from erroneous predictions very quickly. However, the performance was at the edge of the corresponding performance requirement and there is still a lot of room for improvement.

7. Project management

7.1 Work Breakdown Structure

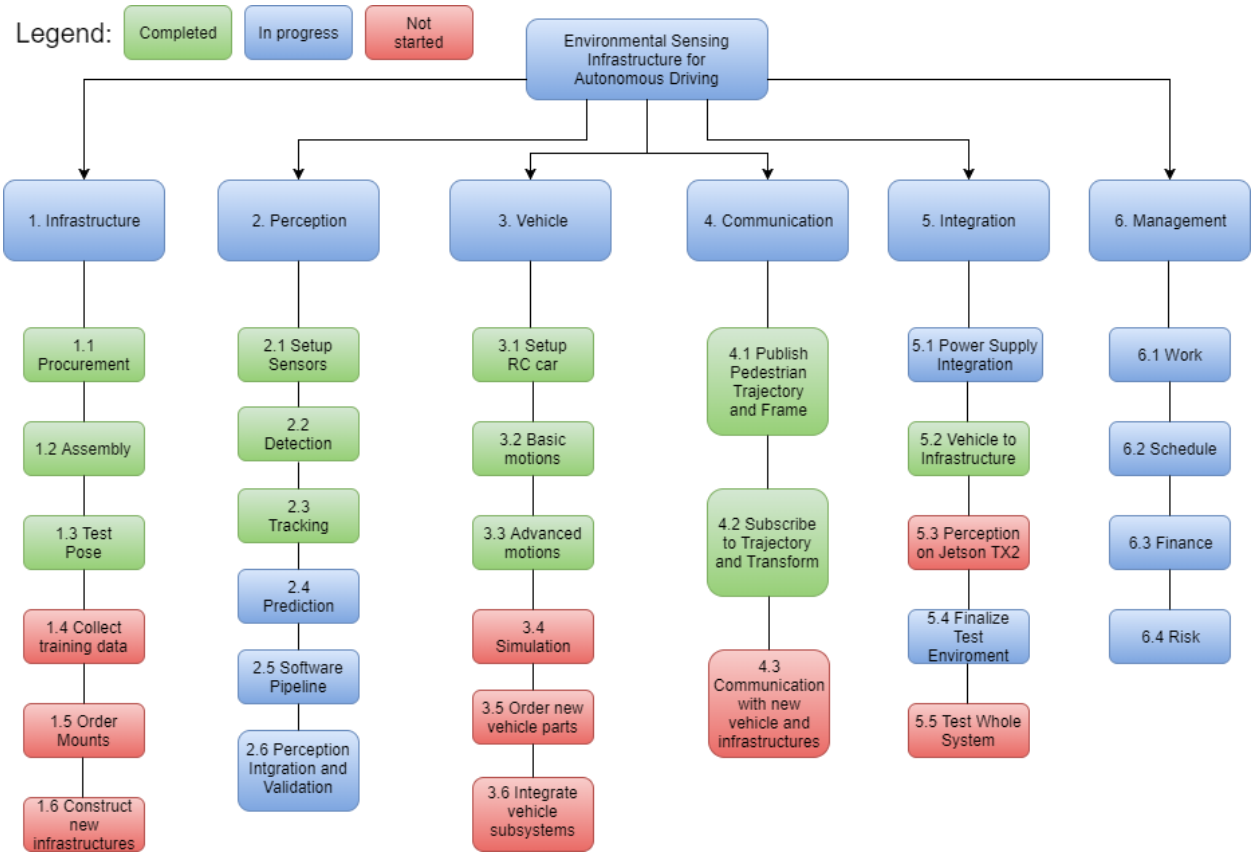


Figure 12. Work breakdown structure

We have followed a deliverable oriented WBS with 4 functional branches - infrastructure, perception, vehicle and communication. The last 2 branches are trivial to all systems - integration and management. The green work packages have been completed, the blue work packages are in progress, and the red work packages have not been started yet.

7.2 Schedule

The table below depicts a simplified version of the schedule for the spring semester. We completed everything that we intended to complete for the Fall semester and even developed a good platform for some of the tasks in the Spring. This will give us the time to attempt to improve some of the subsystems developed in the Fall, especially the performance of the vehicle. Some of these are newer work packages have been included in a little more detail now that we have a clearer idea of how to implement each individual sub-system.

Table 7. Spring schedule

WBS NUMBER	TASK TITLE	TASK OWNER	Progress Review 7	Progress Review 8	Progress Review 9	Progress Review 10	Progress Review 11	Progress Review 12	SVE
1	Infrastructure								
1.4.1	Collect Multiple Pedestrian Training Data	All							
1.5	Order tripods	Oliver							
1.6	Construct new infrastructures	Oliver							
2	Perception								
2.2.2	Multiple Pedestrian Detection	PS							
2.3.2	Multiple Pedestrian Tracking	Ernie							
2.4	Prediction	Rohit							
2.4.1	Multiple Pedestrian Prediction	PS							
2.5	Software pipeline on TX2								
2.6	LiDAR + Camera calibration	Vivek							
2.7	Perception Integration & Validation	PS							
3	Vehicle								
3.4	Simulation	Rohit							
3.5	Order new vehicle parts	Ernie							
3.6	Construct and integrate vehicle subsystems	Oliver							
3.6.1	Integrate power system, communication, and TX2	Oliver							
3.6.2	Implement Kalman filter with odometer, IMU, and LiDAR	Oliver							
4	Communication								
4.3	Communication with new vehicle and infrastructures								
5	Integration								
5.1.2	Power Supply Integration	Vivek							
5.2	Vehicle to Infrastructure	Oliver							
5.3	Perception on Jetson TX2	All							
5.3.1	Detection and tracking on TX2	PS, Ernie							
5.3.1	Prediction on TX2	Rohit							
5.4	Finalize Test Environment	Oliver							
5.5	Test and Calibrate Whole System	All							

7.3 Test plan

7.3.1 Progress Review Milestones

Table 8 shows the milestones that we will attempt to achieve en route to completing the project before the spring validation experiment.

Table 8. Progress Review Milestones

PR7	• Construct new infrastructure • Multiple pedestrian detection • Multiple pedestrian tracking • Design new autonomous vehicle subsystems • Start calibration of camera with LiDAR
PR8	• Port the detection and tracking modules to Jetson TX2 • Start the multiple pedestrian trajectory prediction • Work on EKF, localization of new vehicle • Finish calibration of camera with LiDAR
PR9	• Finish multiple pedestrian trajectory prediction • Finish developing new vehicle
PR10	• Refine software pipeline • Test the integrated perception module • Communication from multiple infrastructures • Finish Vehicle Simulation

PR11	• Processing of data from multiple infrastructures • Complete testing of perception module
PR12	• Finalize the test environment and make required changes • Calibrate the infrastructures and vehicle to the environment

7.3.2 Spring Validation Experiment

Location: Wiegand Gym, "north" points away from equipment desk, this is the +y axis
Equipment: Sensor Infrastructures, Vehicle, 4 human agents wearing non-black clothing

Table 9. Fall Validation Experiment Tests

Step	Description	Requirement
0.1	Set infrastructure #1 in location near door, oriented 45 degrees east of north. This is point (0,0). Set infrastructure #2 14 meters east and 14 meters north of infrastructure #1, oriented directly toward infrastructure #1.	none
1.1	Measure points (4,6), (4,7), (10,3), and (1,1) from origin. Mark these points on the floor with blue tape.	Detection error
1.2.1	4 human agent stands at floor markers. Display centroids on desktop – measure error as distance from points (7.1,1.4), (7.8,2.1), (9.2,-4.9) and (1.4,0).	Detection error
Test 1	Measure error from ground truth at each point. • Test passes if all 4 pedestrians are detected and have average error less than 0.3 meters.	Detection error
2.1	Human agent stands at point (0,0), then walks to (14,0), then (14,14), then (0,14), and back to (0,0).	Tracking area
Test 2	Display recorded pedestrian trajectory on desktop. • Test fails if centroid disappears from human in tracking area.	Tracking area
3.1	Human agents walk simultaneously between points (3,3), (3,11), (11,11), and (11,3) in a clockwise fashion.	Trajectory prediction
Test 3	View all timestamped paths • Test passes if all 4 paths have timestamped information 1.2 seconds into the future.	Trajectory prediction
4.1	Set two 2 meter wide by 4 meter tall black curtains between (3,3),(3,1) and (3,1),(3,-1).	none
4.2	Human agent stands at (2.5,2.5), then walks back into view of infrastructure #2.	Cycle time
Test 4	Display tracking information on desktop. • Test passes if first timestamped centroid and first published trajectory are less than 0.5 seconds apart.	Cycle time
5.1	Set vehicle at center at point (-15, 4.5). Set vehicle waypoint to (10,4.5). Human agent starts walking north from (3,-1) two seconds after vehicle starts.	Stop short
5.2-5.4	Reposition curtains and repeat test symmetrically for each point (11,3), (11,11), and (3,11).	
Test 5	Observe vehicle trajectories in all 4 cases. • Test passes if vehicle stops short of pedestrians at each corner of intersection.	Stop short

7.4 Budget status

Total Budget: $5,000
Spent: **$1,362.35(26.527%)**

Budget Left: **$3637.65(72.753%)**

Table 10. Project Expenditures

Item	Price
Seville Classics Industrial All-Purpose Utility Cart, NSF Listed	$69.99
Jiffyloc Heavy Duty Extension Pole, 4 - 8 feet, Made In USA	$19.99
Jiffyloc Quick Release Adaptor	$10
Jiffyloc angle adaptor	$29.99
Jiffyloc male thread adaptor	$4.99
Laser Pointer	$35.00
PCB Materials	$59.92
9 Gauge Wire	$10.98
Lipo Battery explosion proof case	$12.99
Voltage checker	$9.54
Barrel Jack connector	$7.99
DC barrel pigtail connector	$6.99
BIG TICKET ITEMS	
Zed Camera	$449
NVIDIA Jetson TX2 Developer Kit	$304.00
Wifi Extender	$129.00
Multistar 11.1 V (4 Batteries)	$165.98
TOTAL	**$1,362.35**

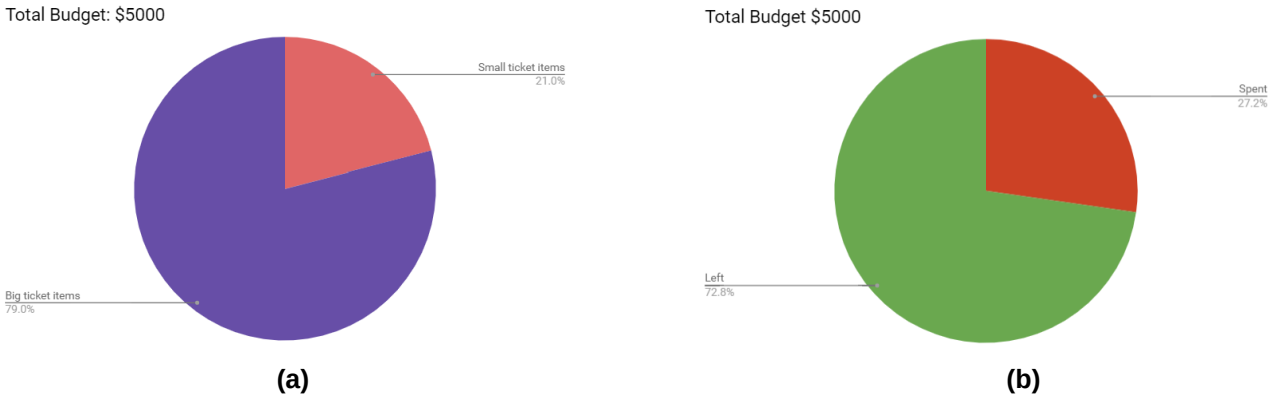


Figure 13. Pie charts depicting: (a) Proportion of budget spent on big and small ticket items, (b) Proportion of budget left vs budget spent

7.5 Risk management

Table 11. List of risks and corresponding mitigation strategies

Risk ID	Risk Definition	Type	Likelihood (1-5)	Consequence (1-5)	Mitigation Strategy
1	School work Overwhelming	Schedule	5	3	Help each other with work, get ahead when possible
2	Personnel Availability	Schedule	2	2	Share schedules ahead of time, plan work accordingly
3	OS Compatibility	Technical	4	4	Dual booting system/ Backup system
4	RC Car Failed	Technical	2	5	Simulation/ Build Back up robot
5	Network Failure	Technical	2	4	Redundancy Communication

6	System Environment Changes	Technical	4	3	Document the changes/ Backup system// Use docker
7	Jetson TX2 Performance	Technical	3	4	Make the Origin Chronos GPU portable
8	WiFi Range Insufficient	Technical	3	3	Use a WiFi extender
9	Multiple Sensors/ Infrastructure Integration Failed	Technical	2	4	Design the system where each subsystem can function independently

8. Conclusions

8.1 Lessons learned

We learned a lot this semester. In the project, it seems, we learn more from our mistakes than our failures. Here is a list of some of the issues we encountered and proposed mitigation strategies.

1. System Environment Changes Issue:
    - Problem: Some system variables or environments are changed. This causes other sub-systems to crash.
    - Lesson learned: Back up the system using the restore point that restores Ubuntu to previous state (eg. Systemback in Ubuntu).
2. Vehicle and Velodyne Network Issue
    - Problem: The system cannot connect to the Internet and the Velodyne at the same time.
    - Lesson Learned: Change the IP address of the router.
    - Problem: The vehicle wifi is locked to a specific wifi network.
    - Lesson Learned: Use the root account to modify the network manager settings.
3. Vehicle Localization Issue
    - Problem: The vehicle cannot localize itself given the map and laser scanner data.
    - Lesson Learned: Verify you know how a system is working before trying to fix it.
4. Camera and Velodyne Calibration Issue
    - Problem: Calibration between the camera and LiDAR was unsuccessful after weeks of effort.
    - Lesson learned: Requirements are paramount. If a subsystem is unnecessary, abandon it (camera).

We also develop some tools to improve the productivity and speed of the development and testing. Here are some useful tools.

1. Automate repeatable tasks with shell script: Write shell scripts that can launch the nodes, launch files and some specific commands (eg. ssh into other computer, launch the nodes with specific order and timing).
2. Write alias of functions in bashrc to speed up experimentation.

8.2 Key spring activities

1. Build second infrastructure
2. Build the racecar and its navigation stack
3. Build the PCB for the racecar and the infrastructure
4. Enhance tracking pipeline to track multiple targets
5. Enhance prediction pipeline to predict multiple targets' trajectories
6. Calibrate and fuse the camera and the pointcloud information
7. Visualize the result in the simulation
8. Set up the demo environment in the gym

9. References

1. https://www.autoaccident.com/statistics-on-intersection-accidents.html
2. Crash factors in Intersection-related crashes: an on-scene perspective. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811366
3. https://mrsdprojects.ri.cmu.edu/2016teamd/
4. https://github.com/mrsd16teamd
5. Alahi paper. A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, S. Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In CVPR, 2016.