Final Report

Team E: Beyond Sight

Environmental Sensing Infrastructure for Autonomous Driving

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

Sponsors:
David Held
John Dolan
Dimi Apostolopoulos

May 9, 2018

THE ROBOTICS INSTITUTE

Abstract

An oft-repeated promise of self-driving car makers and advocates is the reduction in car-related injuries and fatalities that will accompany the adoption of autonomous driving technology. While the authors of this report agree that this assessment is correct, we aim to address a fundamental problem that self-driving vehicles cannot solve on their own: occlusion. The majority of car accidents occur at intersections, and many of these involve moving objects that enter a driver's field of vision without sufficient time to prevent a collision. By themselves, self-driving vehicles face the same problem, they cannot prevent collisions with objects that they cannot sense. This report proposes a novel method of accident prevention in these occluded scenarios. By embedding sensors in occlusion-free vantage points that can communicate with approaching vehicles, accidents can be prevented with moving obstacles that the vehicles themselves cannot sense. This report covers the requirements, development, testing, and deployment of the system described. In this proof-of-concept, data is fused between the environmental sensing infrastructure and GPS data from an approaching vehicle to provide an early warning system for the driver.

Table of Contents

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. Even more worrisome, 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 that 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.

In this project, we start by outlining the system requirements in Sections 3-5 based on which the system is designed. we have developed a LIDAR/Camera-based system that detects, track, and predicts pedestrian trajectories which is detailed in Section 7. This data is fused with incoming GPS data from an approaching vehicle over radio. By using the predicted paths of the pedestrians and the vehicle, would-be collisions are detected in advance, and the driver of the vehicle is sent an alert to stop the car in order to prevent the collision.

The system has been tested on a controlled environment modelled off of a standard four-way intersection. By utilizing high vantage points capable of viewing all approaching foot and vehicle traffic, even in the event of large vehicles entering the area, we demonstrate the reliability of such a system to prevent the hardest-to-avoid accidents.

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…

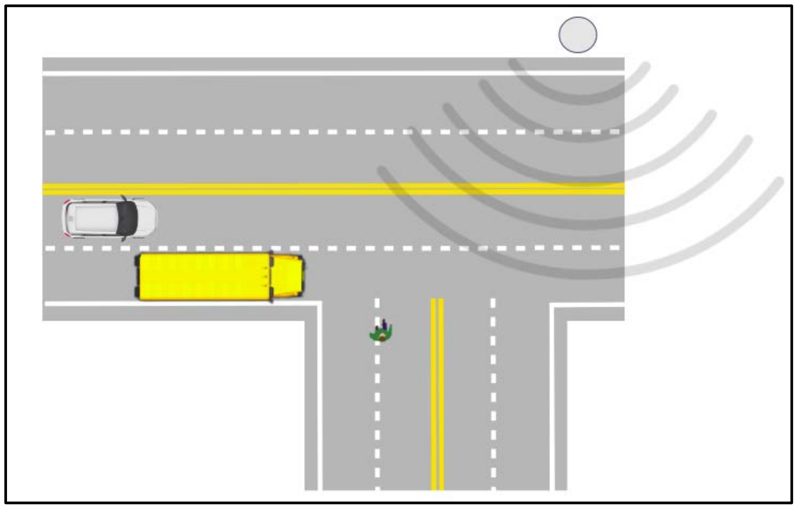


Figure 1. System graphical representation.

5 seconds earlier, when Andy was approaching the intersection, his car's computer had made a wireless connection with the environmental sensing infrastructure monitoring the Fifth Avenue-Bouquet intersection. The sensors continuously detect vehicles, pedestrians, and other moving objects in the vicinity of the intersection. The infrastructure is the grey puck displayed at the top Figure 1.

5 tenths of a 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 infrastructure's sensors had tracked the man's movements and predicted that his path would move in front of Andy's car. The infrastructure 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 environmental sensing infrastructure 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 use case is depicted in a simplified form in Figure 2a below. Figure 2b demonstrates what would otherwise occur without the environmental sensing infrastructure.

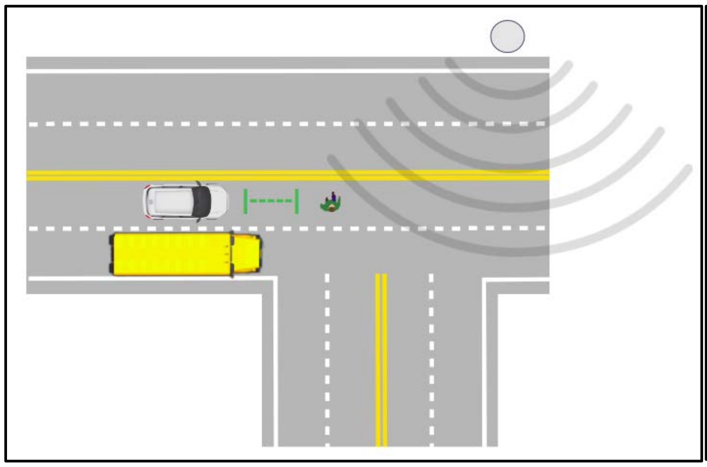


Figure 2a. Result of infrastructure in use

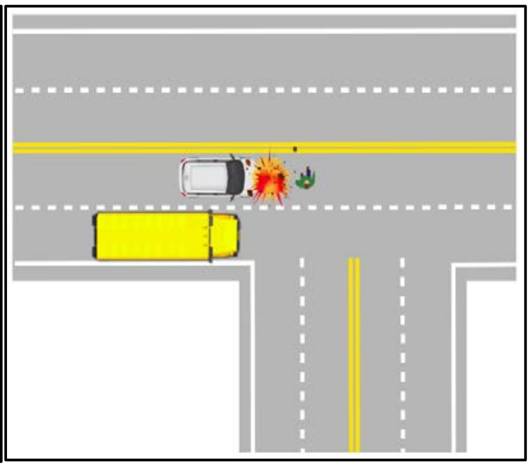


Figure 2b. Result without infrastructure

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 vehicles
	VEHICLE:
M.F.5	Alert driver to stop with enough time to stop safely

3.2 Performance Requirements

Table 2. Performance Requirements

ID	Title	Description
		INFRASTRUCTURE:
M.P.1	Detection	Detect up to 3 pedestrian centroids with Euclidean distance error < 0.3m
M.P.2	Tracking	Track up to 3 pedestrians within 20m of the infrastructure
M.P.3	Prediction	Predict up to 3 pedestrians’ trajectories 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 0.5 seconds.
		VEHICLE:
M.P.5	Vehicle	Alert driver to stop short of intersection with approaching pedestrian >1.2 seconds* ahead of entry to intersection.

* 1.2 seconds = 1 second (stop time of vehicle at 30mph) + 0.2 seconds (human reaction time)

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

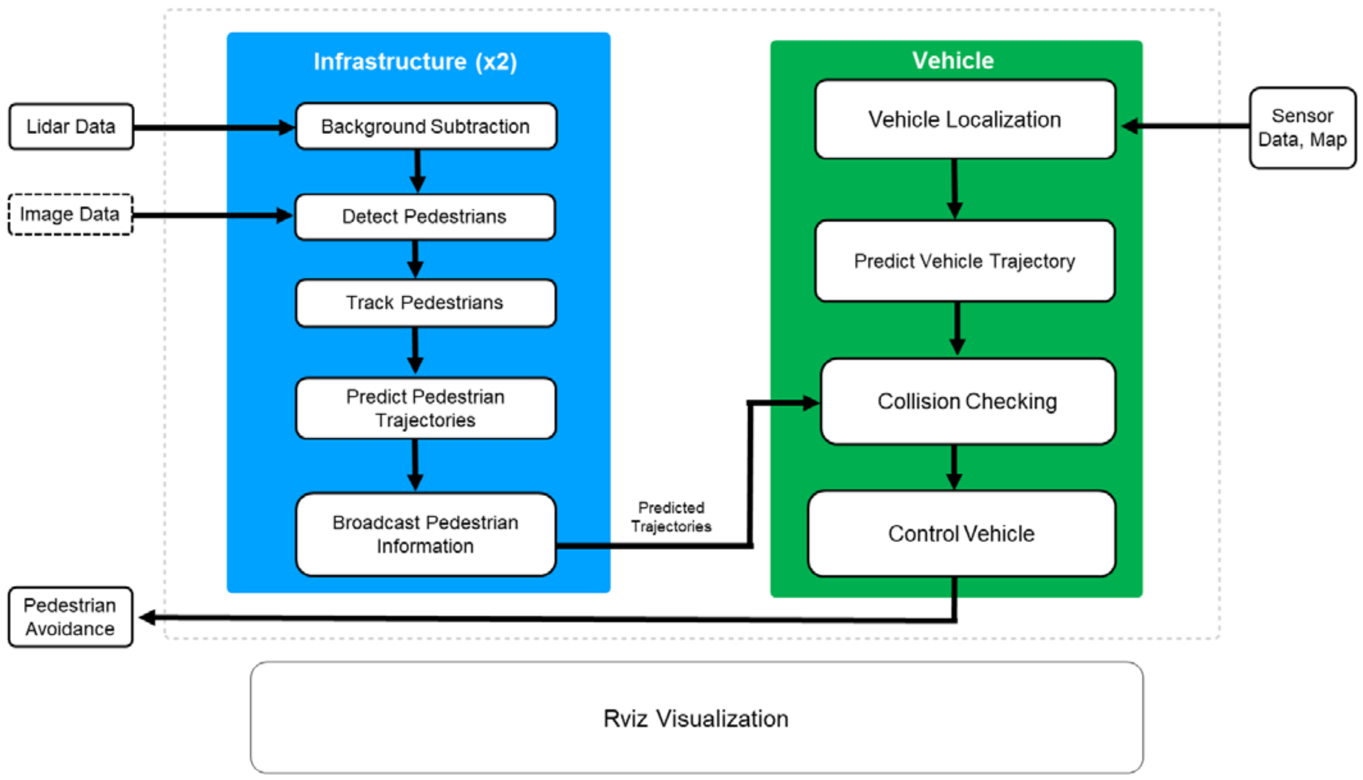


Figure 3. Functional Architecture

The Functional architecture for the project is visualized in the Figure 3 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**: Vehicles can use the published pedestrian trajectory predictions and their own kinematics to determine whether a collision is imminent. In the case of our system, we have instead implemented the collision checking node itself within the infrastructure, though this is not generally how autonomous vehicles will use the information. In our system the vehicle sends all GPS information to the infrastructure during its approach. The infrastructure uses this information to see if the trajectory of the vehicle and pedestrian will collide. If so, the infrastructure sends a *stop* message to the vehicle, alerting the driver to stop.

5. System-level trade studies

5.1. Communication

Our first choice for vehicle-to-infrastructure ("V2X") communication was DSRC radio. This is because it is the FCC-mandated standard for this form of communication. However, our investigation into this technology quickly revealed that the hardware was prohibitively expensive for our remaining $3200 budget. The other choices we considered are presented in Table 4 below.

Table 4. Communication subsystem trade study

	Hardware requirements	Range	Cost
Wi-Fi	High	~100 meters	Medium
Zigbee radio	Low	~1 mile	Low
Bluetooth	Low	~100 meters	High

While each of these technologies satisfies what would be needed for the system, the cost of bluetooth was undesirable. Furthermore, the outdoor testing that the system would require makes the hardware requirements of Wi-Fi undesirable. Our final decision was to use XBee radio modules, which can be operated in transparent mode that make software interfacing extraordinarily easy over serial communication. Finally, there is plug-and-play hardware available to connect XBees directly to an Arduino. The radio module and hardware for connecting to Arduinos are shown in Figure 4 below.

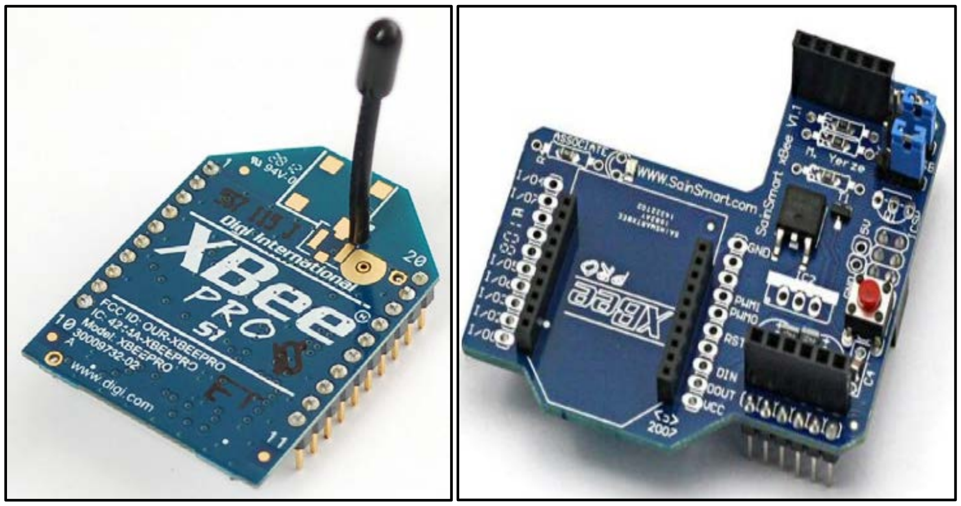


Figure 4. Xbee Radio Modules

5.2. Vehicle Localization

The vehicle we had access to for testing was Oliver's 2004 Honda CR-V. Because of the onboard computing capacity of the vehicle, it was not an option to attempt to tap into the vehicle's computer to obtain GPS data, since the vehicle does not have a GPS. Furthermore, we were unwilling to do this sort of bootstrapping, since we had found over the course of the fall semester that while it made early development easy, troubleshooting could be virtually impossible on a system that you do not have full access to. Thus, we decided our best option was to build our own localization capability and attach it to the vehicle.

Table 5. Requirements for vehicle localization

Format	Accuracy	Refresh Rate	Processor to be used
GPS required	<0.3 meters	>2 Hz	ATMEGA 2560

The format, accuracy, and refresh rate requirements leave many options available for this subsystem, but performing everything on the ATMEGA 2560 is quite restrictive. A traditional approach to this system is to use an extended Kalman filter with a combination of IMU, GPS, odometry, and other sensor data. However, the computing required to run an EKF surpasses the computational ability of the ATMEGA 2560. Thus, using only a single sensor to achieve the accuracy in GPS coordinates was our only option, unless we were to move to a bigger processor, which we did not have the resources for. Luckily, this accuracy of GPS is achievable with Differential GPS. We performed only a limited trade study of possible modules to use, and decided to go with the module already in use by Team A, who spoke well of it. The Emlid Reach GPS can be seen in Figure 5 on the right.

Figure 5. Reach Emlid GPS

5.3. Camera

Using a high resolution RGB camera was critical to our perception system, as the information from the RGB camera would be used to extract useful information from the point cloud. There were a number of parameters on which the cameras were evaluated to see which one actually suits our needs. To accelerate the process for finding a camera for our project, we decided to conduct trade study on some of the existing cameras which were present in the MRSD inventory. Table 6 shows the comparison of different cameras based on some important parameters.

Evaluating the above cameras based on a set of parameters, it is clear that Lifecam is able to satisfy all the requirements. It is both cost effective and has an easy interface with ROS. So the team decided to use Lifecam.

Table 6. Trade study for Camera

Parameters	**Cost**	**Resolution**	**ROS Support Availability**	**Easy interface**
Point Gray - Chameleon	$ 365	1296x964	Yes	Yes
Ausdom AW615	$ 36.89	1280x720	No	Yes
BlasterX senz3D	$ 169.95	1920X1080	Yes	No
Microsoft Lifecam	$ 54.95	1280X720	Yes	Yes

5.4. Detection

One of the most critical part in our system is the speed of the perception pipeline. If we can speed up the perception pipeline without losing the accuracy, we can alert the vehicles earlier if there is any potential collision. Therefore, we emphasize the speed of the detection subsystem when we do the trade study. Among all the neural network approaches which usually have slow inference time like 5-20 fps, YOLO 2 (You Only Look Once) [5], a Unified, Real-Time Object Detection, outperforms all of them in its fast inference time in the range of 45 to 150 fps on the Titan X GPU. It gets 88% top-5 accuracy in ImageNet 2012 and 70.4 in mAP while the famous detection network Faster R-CNN [6] only has 70.7 in mAP.

5.5. Tracking Algorithm

The trade study required for this subsystem was related to the data association method used. The two options that we had was either using native nearest neighbor data association or Hungarian method with Kalman filtering. Since our project is safety oriented, we decided to choose the latter one as it provides more robust and accurate results for associating data between time frames. The only remaining concern is related to the speed of this approach. After full testing, we found out that the performance for both methods in our on-site embedded computer are very similar. Thus, we chose the more robust algorithm namely Hungarian data association.

5.6. Prediction Algorithm

The main trade study required for this subsystem was related to the prediction algorithm used. The two options that we had was either a more classical regression-based methodology, named Polynomial Regression, or a deep learning based approach, using Social LSTM [7]. The main consideration when selecting the method was that it should be able to give reliable output over long periods of time even when there is slightly noisy input data. It is difficult to estimate the reliability of an algorithm hence we implemented a small proof-of-concept version of both the algorithms. We learned that the polynomial regression algorithm was far more robust and scalable to input data than the Social LSTM algorithm. Moreover, transferring deep-learning systems from scenes seen in a database to actual real-world scenarios has not proven effective till date, even in state-of-the-art systems. Given that ours is a safety-critical system, we chose the more reliable algorithm in Polynomial Regression.

6. Cyberphysical Architecture

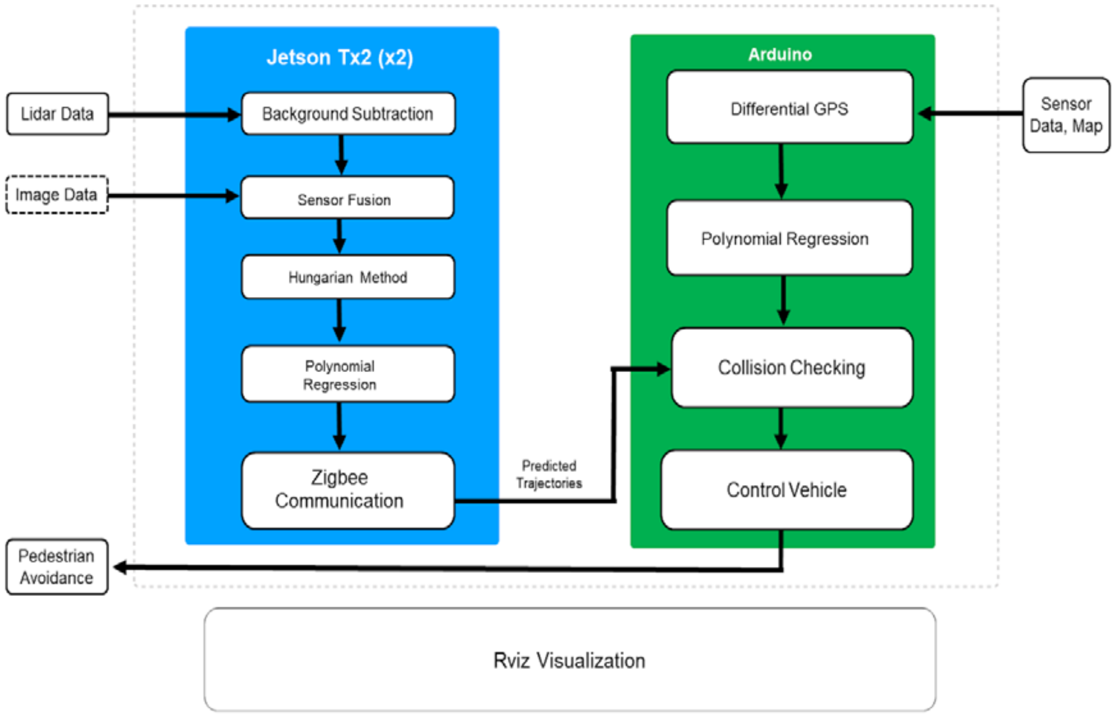


Figure 6. Cyberphysical Architecture

The Cyberphysical architecture for the project is visualized in the Figure 6 above. There are two modules in our system: infrastructure and vehicle.

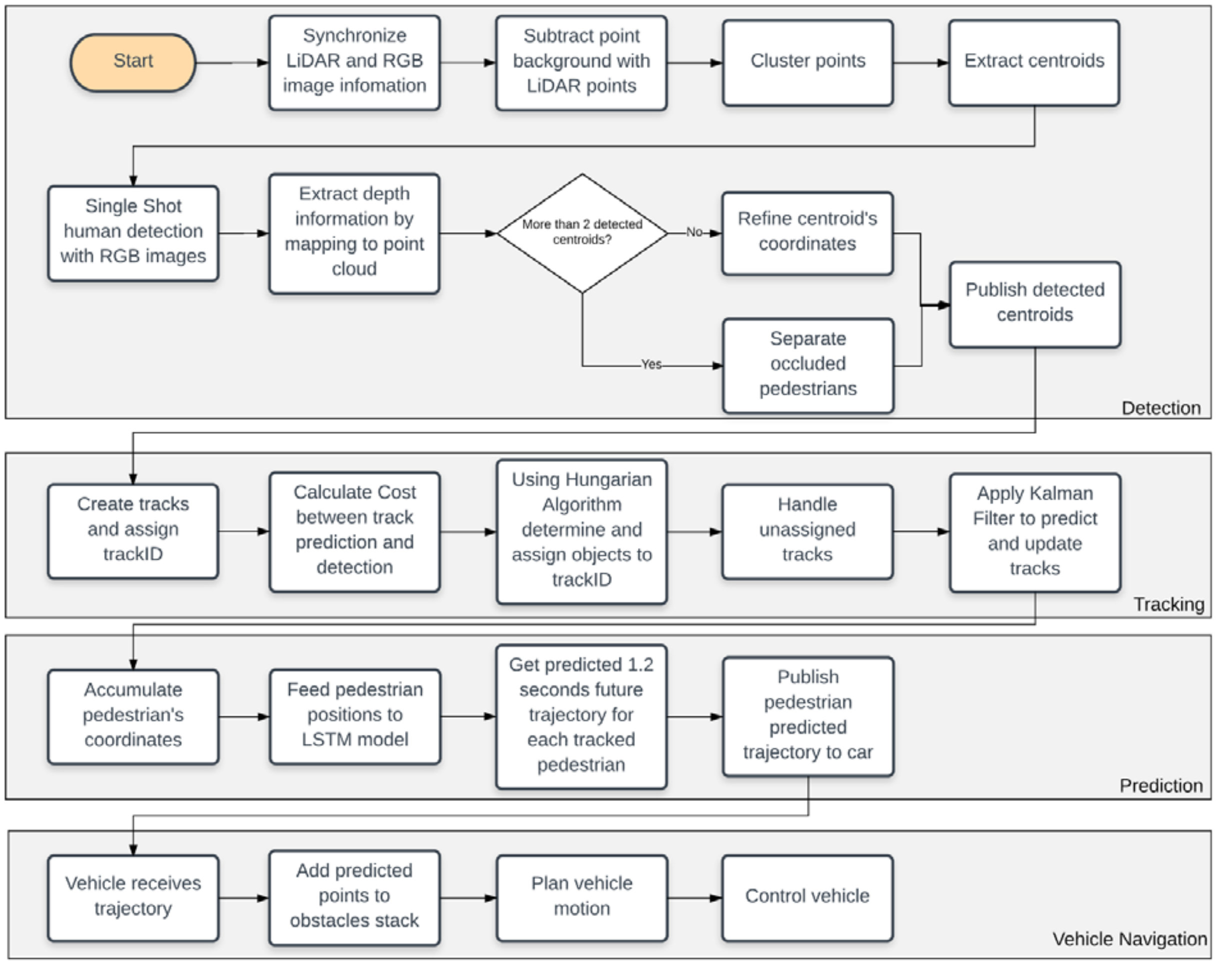
1. **Infrastructure**: The LiDAR and camera are mounted on the infrastructure. Data is sent to the Jetson TX2 embedded computers through an Ethernet cable. The workstation synchronizes LiDAR and RGB information, then compares and subtracts foreground points from the pre-built background. After that, 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 and sends the trajectories to the vehicle via WiFi. All messages are transmitted within a ROS environment.

2. **Vehicle:** The vehicle is equipped with a differential GPS, Arduino Mega microcontroller, and XBee radio module. The GPS itself is also connected to an antenna and a transceiver (for communicating with the base GPS for real-time kinematics). The Arduino reads the GPS data at 5Hz and broadcasts over our XBee radio network. All subscribed nodes, i.e. the infrastructures, notify the vehicle via XBee when they are receiving the information. The GPS coordinates and timestamp are then used to detect whether a collision is imminent. In the event that it is, the infrastructure sends a stop message to the vehicle. When the vehicle receives a stop message, it illuminates the *stop* LED's for the driver.

7. System description and evaluation

7.1 Overall System

The overall system can be depicted using this helpful flowchart that shows the processing performed on input LiDAR and RGB data to the point that the vehicle receives the predicted pedestrian trajectory and takes an action based on that information.



Start
Synchronize LiDAR and RGB image infomation
Subtract point background with LiDAR points
Cluster points
Extract centroids
Single Shot human detection with RGB images
Extract depth information by mapping to point cloud
More than 2 detected centroids?
No
Refine centroid's coordinates
Yes
Separate occluded pedestrians
Publish detected centroids
Detection
Create tracks and assign trackID
Calculate Cost between track prediction and detection
Using Hungarian Algorithm determine and assign objects to trackID
Handle unassigned tracks
Apply Kalman Filter to predict and update tracks
Tracking
Accumulate pedestrian's coordinates
Feed pedestrian positions to LSTM model
Get predicted 1.2 seconds future trajectory for each tracked pedestrian
Publish pedestrian predicted trajectory to car
Prediction
Vehicle receives trajectory
Add predicted points to obstacles stack
Plan vehicle motion
Control vehicle
Vehicle Navigation

Figure 7: Algorithm Flowchart

7.2 Subsystem Descriptions

7.2.1 LiDAR-based Detection

One of the biggest difficulties for the self-driving car perception technologies is that how the LiDAR can detect the moving objects while the cars are still moving. The dynamic of the vehicle movement and the environment will cause the detection errors. However, our system doesn’t have such issue since our infrastructure is static and the environment is also static. To do the LiDAR-based detection on the fixed infrastructure, we record the pointcloud of the background when there are no moving objects, and do the background subtraction to get the foreground pointclouds when there are any moving objects. After getting the foreground pointcloud, we use the nearest neighbors to cluster the pointclouds into groups. Finally, we find the centroids of these clusters and output their coordinates.

7.2.2 Camera-based Detection

For detecting pedestrians using the camera data, we are using the YOLO v2 neural network. The data from the camera comes at nearly 30fps however we chose to inly use the compressed format of the image rather than the raw image in order to make the algorithm faster. The neural network outputs the bounding boxes of 20 different objects with their detection confidence to the LiDAR-Camera Fusion subsystem.

7.2.3 LiDAR-Camera Fusion

Our whole fusion algorithm is based on the fixed nature of the infrastructure. Thus, we can treat all the points subtracted from background as true foreground. However, due to the lack of context information, there are chances that multiple pedestrians would be classified as a single pedestrian. This false detection will affect the performance of tracking and trajectory prediction in later stages. This is the reason why we need RGB image to separate those wrongly clustered points at least within the camera field of view.

The LiDAR 3D bounding box will be projected into camera frame with extrinsic and intrinsic matrices given by calibration results. Simultaneously, image 2D bounding box would be generated from either state-of-the-art deep learning detection network. For each overlapping pair of LiDAR and RGB bounding boxes, we re-cluster all the point cloud data into K groups, with K equals to number of RGB bounding box.

The fusion results are visualized below. There are 2 pedestrians walking within camera frame. Since they are too closed to each other, LiDAR clustering node will treat them as a single pedestrian as indicated by the red point. However, after fused the detection result from YOLO network, the system can accurately detect each of them correctly as shown by the white points.

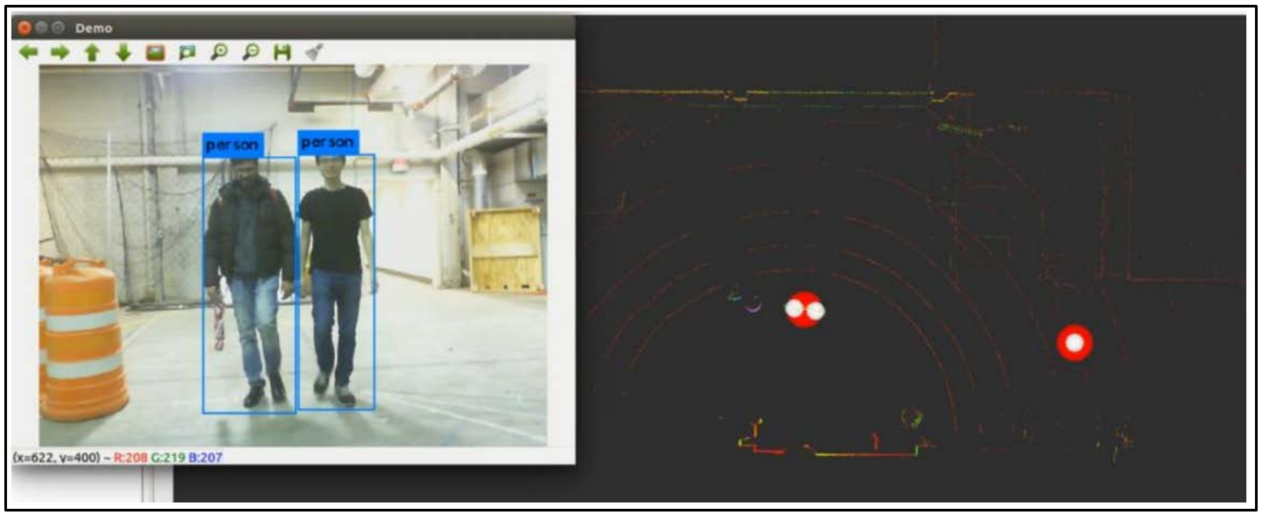


Figure 8. Fusion results.

7.2.4 Multi-pedestrian Tracking

Our tracking algorithm is based on Hungarian method (to assign pedestrian's ID in consecutive frames) and Kalman filter (to fuse prediction by dynamics and observation from LiDAR point cloud). The input to the system are the pedestrians' detected centroids in Cartesian space. The output will be a list of pedestrian with unique id and historical positions.

Within the system, Kalman filter is used to refine pedestrians' positions using noisy sensor data and predict pedestrians' positions at next time step for data association. Every pedestrian will have a dynamics model with constant velocity between each time frame. Every time a new observation comes in, it will update the Gaussian distribution for the pedestrian. Since detected centroid positions are highly sensitive to pedestrian configuration like orientation and waving hands, we also assign a large noise matrix to such observation. By doing so, Kalman filter would utilize internal dynamics model to correct the observation noise.

Hungarian method is a combinatorial optimization algorithm that solves assignment problem in polynomial time. Specifically, in our system, it is used to achieve data association from consecutive frames. We constructed a cost matrix table in which each row associated with an existing tracked object and each column associated with a new detected object. The elements inside the table are calculated by the Euclidean distance between each pair of the points. After that, Hungarian method will how to associate the point pairs in order to reduce the overall association cost.

The final tracking results are shown below. Different color indicates different pedestrians. As seen from the graph, our algorithm can track the pedestrian reliably with a frame rate of 10.

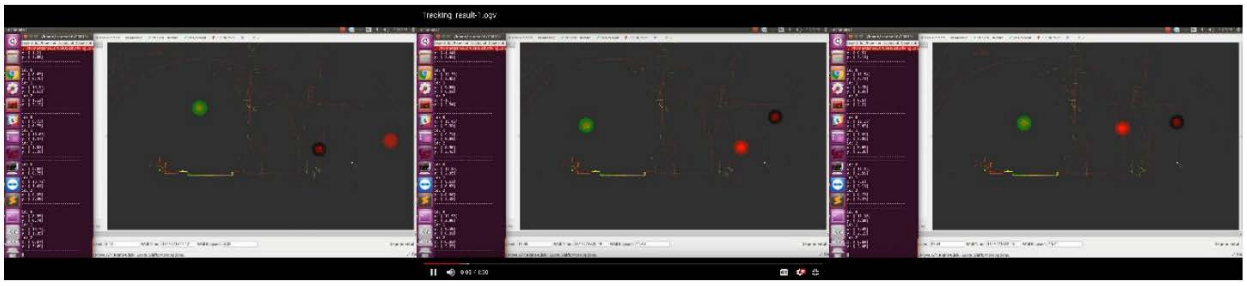


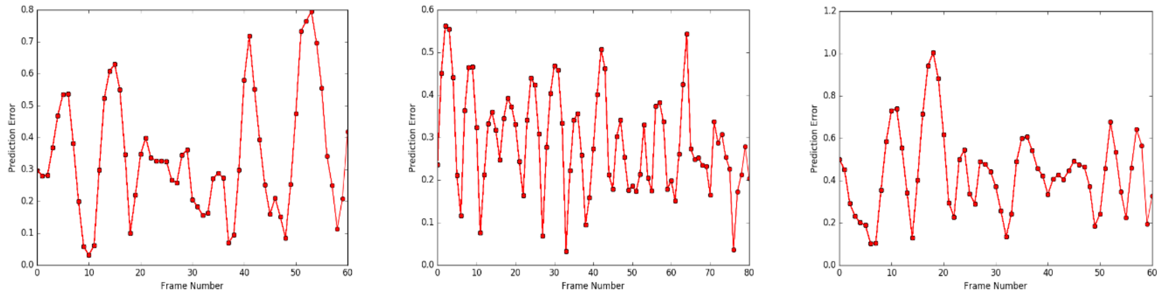
Figure 9. Tracking results.

7.2.5 Multi-pedestrian Trajectory Prediction

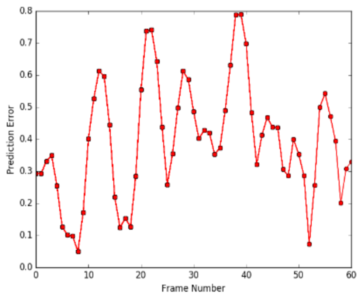
The problem of trajectory prediction is essentially one of sequence prediction where, given a sequence of input coordinates (of a moving pedestrian), we must identify a pattern and use that pattern to predict the future sequence of coordinates of the same pedestrian. This subsystem forms the crux of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

Our polynomial regression algorithm receives an input sequence of locations for each pedestrian in the scene. One strength that we have embedded in the system is the complete abstraction to the kind of sensors used and the rate at which they are publishing. This allowed us to extend the system from LiDAR-only system at 10Hz to a LiDAR-camera fused system at 6Hz. Based on a minimum observation window of 3 data points, we fit a polynomial curve to input values. Using this curve, we extrapolated for the desired 1.2 seconds and published this as our predicted trajectory.

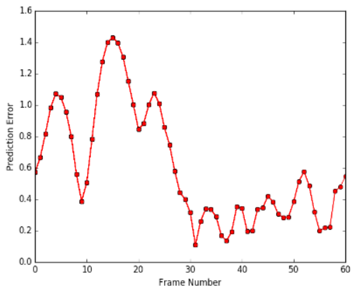
This algorithm also has the advantage that we can adapt parameters to improve the results. We played with different degree polynomials and observations lengths. We finally settled on a second degree polynomial fit with a maximum observation length of 1.2 seconds. We tested this algorithm on live pedestrian 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.



a. (b) (c)



(d)



(e)

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)

The algorithm also required minimal addition to extend from single pedestrian trajectory prediction to multiple pedestrian trajectory prediction. The Fig. 11 below shows an RViz display where the trajectory of three pedestrians is being predicted to a high level of accuracy.

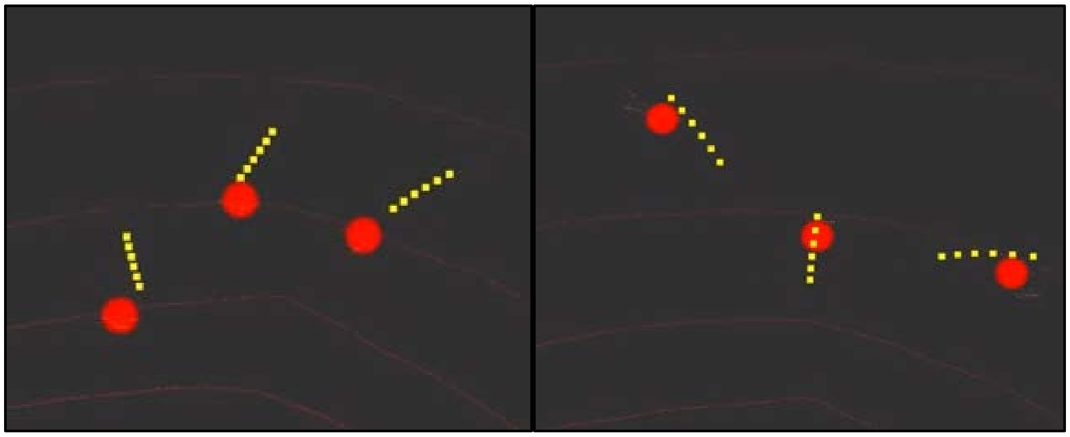


Figure 11. RViz screenshot of predicted trajectory for 3 pedestrians

7.2.6 Sensing Infrastructure

An integral portion of our project is the mounting and placement of the sensor suite in the environment. The initial idea is to have a cluster of a LiDAR and cameras, as described below, located at an intersection. We are using a sturdy tripod as the base for the sensor mount since this gives us the sturdiness of a fixed infrastructure as well as the mobility to test in any location that we need. The tripod also allows for a platform to house the electrical subsystem as described now.

The Power Distribution Board (PDB) was designed to power the LIDAR, Jetson TX2, Microsoft Lifecam 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. An Xbee USB module will also be attached to the Jetson TX2 which requires power.

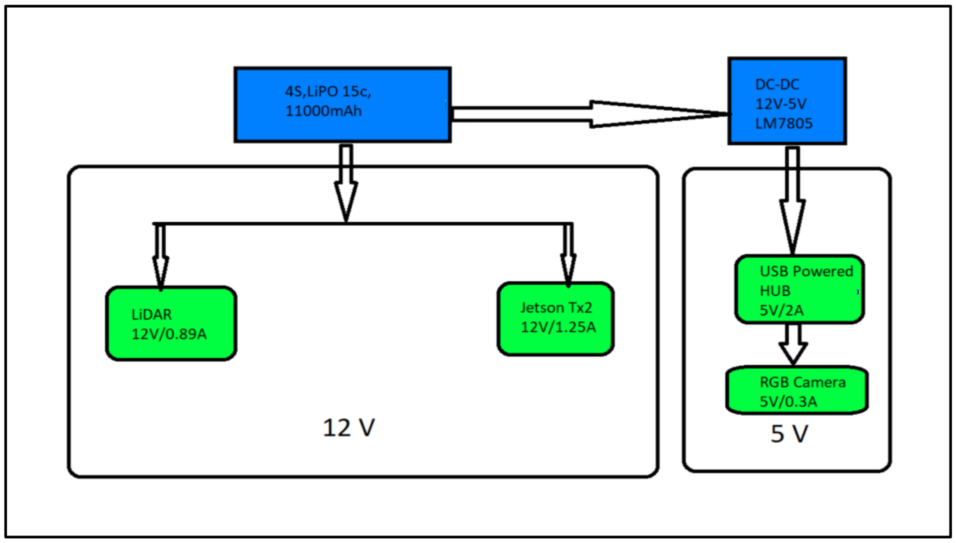


Figure 12. Electrical System

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

Table 7. 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
Camera	11.1	0.3	5	5

7.2.7 GPS-based Localization

In this semester we had to localize an actual car in its environment to predict whether there would be a collision between the car and pedestrian. To perform this, we decided to use the Reach Emlid GPS which is an RTK module giving high accuracy location information. It requires a *Base* module to be set up as well as a *Rover* module to ensure high accuracy. This is convenient for us since we already have a static infrastructure to place the *Base* module. The system is fairly easy to work with and we were able to receive Serial data from the GPS modules with some additional work. The *TinyGPS* library was invaluable in helping us to parse the NMEA format to extract the latitude and longitude information. This was then published to the infrastructure from the car as explained in the next section.

7.2.8 Communication

The vehicle was equipped with a simple Arduino Mega microcontroller attached to an XBee radio and the GPS. The XBee hooked up to the vehicle was used in broadcast mode on a private network at a baud rate of 9600. Broadcast mode allows the vehicle to connect with any infrastructure that can receive its signal. The XBee radios on the infrastructures use the MAC address of the vehicle's radio to send their information directly to the vehicle. The information communicated consists of only three small string messages: *stop, go, <latitude,longitude>*. Thus, this small bandwidth network was able to communicate with a small baud rate at a large range.

7.3 Spring Validation Experiment (SVE) Performance Evaluation

For the SVE we had five tests which displayed the performance of each important subsystem corresponding to the performance requirements as well as the performance of the entire system. We were able to pass all performance requirements as stated in Section 3.2. The exact performance is recorded in the Table 8. It is important to note that these are repeatable results as we were able to achieve similarly positive results in several tests as well as in both the SVE and SVE Encore.

Table 8. SVE Test Performance

SVE Test	Quantitative Performance
Detection Accuracy	0.12m
Tracking Consistency	Continuous tracking
Pedestrian trajectory prediction	0.2745m
Cycle Time	0.24s
Vehicle Collision Avoidance Success Rate	100%

7.4 Strengths and Weaknesses

7.4.1 Strengths

1. Detection Accuracy: The detection accuracy tested at an average of 0.12m. This was excellent, considering our requirement of less than 0.3m. We are particularly happy with this because we the test cases were difficult with multiple pedestrians standing close to each other.

2. Good Cycle time: Time taken between when the first pedestrian is detected to the first published trajectory was around 0.29 seconds. This is a terrific refresh rate for our system with the additional and costly image detection and fusion algorithms.
3. Robustness to Environment: We tested the entire system in heavy rainfall and wind for several hours and were still able to achieve adequate results. This is important to a real-world implementation of our system since the system should work in these conditions as well.
4. Graphical User Interface (GUI): The GUI developed for demonstrating our performance requirements showcased our requirements clearly and effectively.
5. Extensibility: Our software platform has now been designed such that we have the ability to work with one or many sensors
6. Scalability: Our infrastructure is currently self-contained; this means that if we wanted to have multiple instances of this infrastructure it would be extremely easy to do so.
7. 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.

7.4.2 Weaknesses

1. Communication: While our Zigbee-based communication system was adequate for our requirements, it is not robust enough to extend to several cars as would be typically required of the system.
2. Robustness: The system is good enough to work for multiple pedestrians but it is a difficult problem in general to make the system to work for many more pedestrians.
3. Background Registration: To make the system completely autonomous for long periods of time, we would need to develop an automatic system of registering new backgrounds to the old one periodically.

7.4.3 Opportunities for improvement

1. The range of the detection algorithm satisfied the requirements that we had defined but the performance dipped towards end of the range. Hence, we intend to solve the issue with fusion of camera data.
2. The trajectory prediction subsystem worked well and was able to recover from erroneous predictions very quickly. However, the performance when using the fused data is not as good as the performance when using only the LiDAR data. This shows that there is scope for improvement in this subsystem.
3. The speed when using the fusion algorithm is slow causing some lag in the system. This is the reason we attempted to keep the cycle time in check but there is still some scope for this issue to be solved.

8. Project management

8.1 Schedule

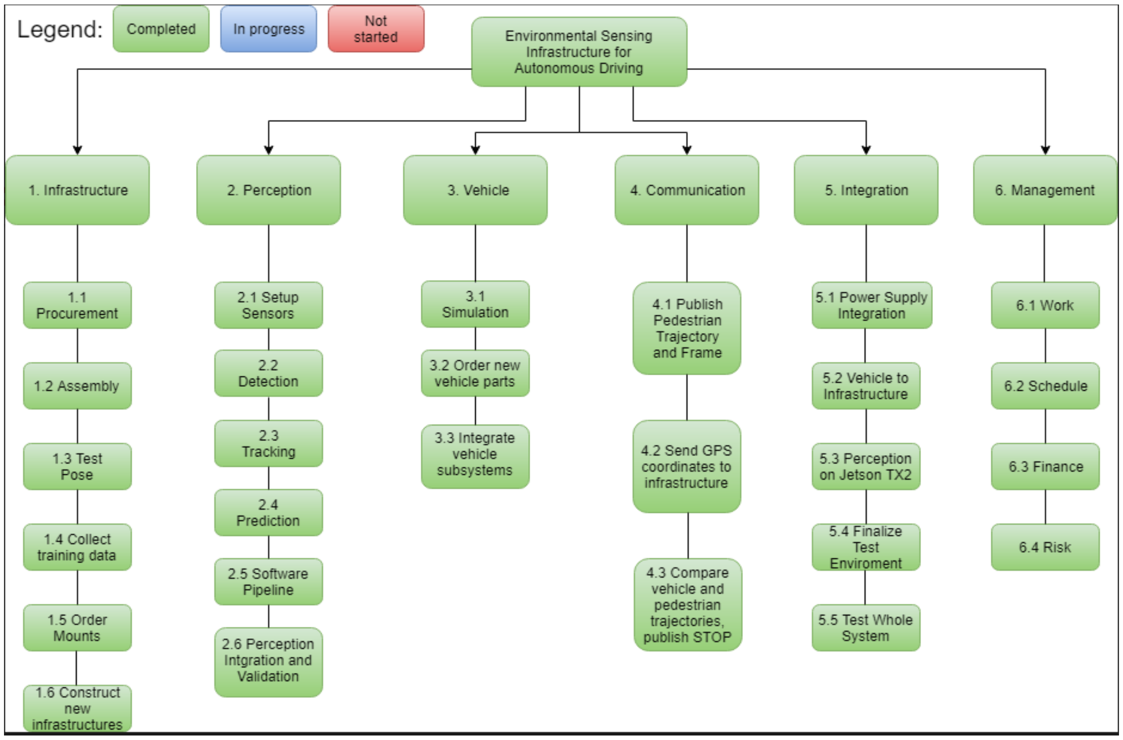

Legend:
Completed
In progress
Not started
Environmental Sensing Infrastructure for Autonomous Driving
1. Infrastructure
2. Perception
3. Vehicle
4. Communication
5. Integration
6. Management
1.1 Procurement
1.2 Assembly
1.3 Test Pose
1.4 Collect training data
1.5 Order Mounts
1.6 Construct new infrastructures
2.1 Setup Sensors
2.2 Detection
2.3 Tracking
2.4 Prediction
2.5 Software Pipeline
2.6 Perception Intgration and Validation
3.1 Simulation
3.2 Order new vehicle parts
3.3 Integrate vehicle subsystems
4.1 Publish Pedestrian Trajectory and Frame
4.2 Send GPS coordinates to infrastructure
4.3 Compare vehicle and pedestrian trajectories, publish STOP
5.1 Power Supply Integration
5.2 Vehicle to Infrastructure
5.3 Perception on Jetson TX2
5.4 Finalize Test Enviroment
5.5 Test Whole System
6.1 Work
6.2 Schedule
6.3 Finance
6.4 Risk

Figure 13. Work breakdown structure

We have followed a deliverable oriented WBS with 4 functional branches - infrastructure, perception, vehicle and communication. The last 2 branches apply 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. At the time of this report, the work is completed, thus all items are green.

The table below depicts a simplified version of the schedule for the spring semester, which maps directly to the work breakdown structure above. We completed everything that we intended to complete for the Spring semester by nearly keeping to the schedule laid out below. It was mainly between Progress Review 10-12 that we lagged behind slightly. Luckily we had kept a buffer period of a week and hence were able to complete the validation experiments at the end of the semester successfully.

Table 9. Spring schedule (Yellow = Progress Review, Red = SVE, Blue = Allotted time for task)

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					🟨				🟨				🟨				🟨				🟨				🟨				🟥
3.1	Simulation	Rohit				🟨				🟨				🟨	🟦	🟦	🟦	🟨				🟨				🟨				🟥
3.2	Order new vehicle parts	Ernie	🟦	🟦	🟦	🟨				🟨				🟨				🟨				🟨				🟨				🟥
3.3	Construct and integrate vehicle subsystems	Oliver	🟦	🟦	🟦	🟨	🟦	🟦	🟦	🟨	🟦	🟦	🟦	🟨				🟨				🟨				🟨				🟥
3.3.1	Integrate power system, communication, and TX2	Oliver	🟦	🟦	🟦	🟨				🟨				🟨				🟨				🟨				🟨				🟥
3.3.2	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				🟨				🟨				🟨				🟨				🟨	🟦	🟦	🟦	🟨	🟦	🟦	🟦	🟥
6	Management					🟨				🟨				🟨				🟨				🟨				🟨				🟥
6.1	Work	All				🟨				🟨				🟨				🟨				🟨				🟨				🟥
6.2	Schedule	All				🟨				🟨				🟨				🟨				🟨				🟨				🟥
6.3	Finances	All				🟨				🟨				🟨				🟨				🟨				🟨				🟥
6.4	Risk	All				🟨				🟨				🟨				🟨				🟨				🟨				🟥

8.2 Budget*

* Full parts list is included in the appendix

The budget of $5000 that was provided to us was substantial. Through judicious use of existing MRSD resources and the budget, we were comfortably able to purchase all essential items (and some non-essential items). We used two Velodyne VLP-16 LiDARs from existing MRSD inventory which allowed us to stay within budget. All our other purchases were made when it was an absolute necessity and by searching for the best cost-effective option. Our best purchase was the DeWalt Power Hub which allowed us to successfully test our system outdoors for extended periods of time. Moreover, this will be a valuable commodity to all teams in coming years whenever they need to do outdoor testing. Another very useful purchase was the Reach Emlid RTK GPS which allowed us to successfully localize our vehicle using just one sensor and without the additional overhead of fusing multiple sensors. It will also be a very useful purchase for the incoming batch.

Starting Budget: **$5000**
Budget Left: **$600 (12%)**

Table 10. Big ticket expenditures from budget

Item	Price
Zed Camera	450
Jetson TX2 x 2	640
Electrical Components	350
Emlid Reach GPS and antennae	960
Communication hardware	200
Tripods	300
DeWalt Power Hub	500
Vehicle Costs	250
Other (Primarily testing)	750
TOTAL	**$4400**

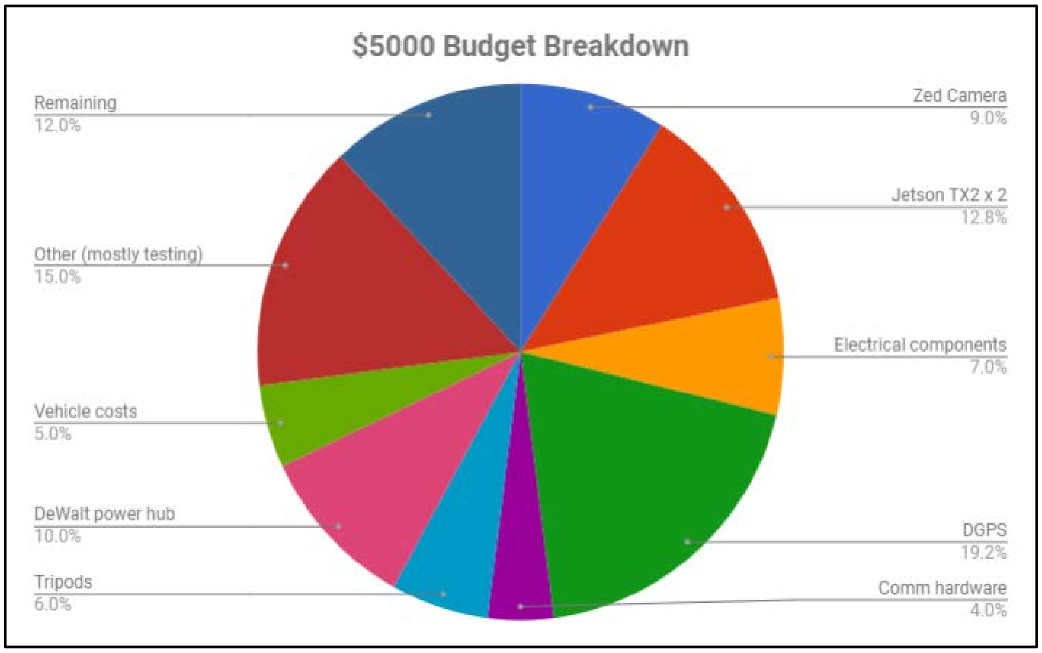


Figure 14. Pie chart showing the major expenditures

8.3 Risk management

Our risk management over both semesters has been fairly accurate although we have not kept to strictly following and updating the list. We usually sat down and tried to predict what might affect us adversely in the semester and kept a watchful eye on those risks. Our mitigation strategy usually came down to determination and quick thinking to find a solution that would work robustly and reliably.

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	4	Help each other with work, get ahead when possible
2	Personnel Availability	Schedule	2	2	Share schedules ahead of time, plan work accordingly
3	Localization	Technical	3	5	Use LiDAR as a backup
4	Loss of Communicatio n	Technical	4	2	Build vehicle trajectory prediction to be robust to packet loss
5	Camera LiDAR Calibration	Technical	5	5	Control the Environment
6	Jetson TX2 Performance	Technical	3	4	Optimizing Algorithms
7	Weather	Non-Technical	5	5	Have a video backup of functional system
8	Multiple Sensors/ Infrastructure Integration Failure	Technical	2	4	Design the system where each subsystem can function independently

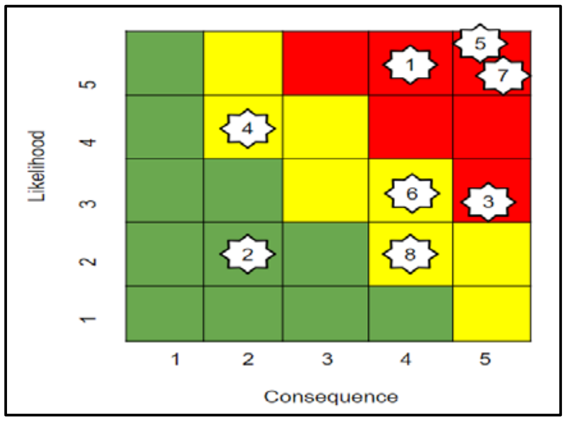


Figure 15. Likelihood-Consequence Risk Table

9. Conclusions

9.1 Lessons Learned

A team project of this scale brings up several challenges which provides a lot of scope for learning. Some of the lessons that we learned are summarized below:

Management:

- Requirements should drive design, not the state-of-the-art methods
- Requirements should be defined not based on what is possible but on what the stakeholders ask for
- Division of work should be done based on capability and interest; but it's important each member in the team understands his/her role in the team
- It is beneficial if each person has a "buddy" so that if the lead on a subsystem is busy or unable to perform the work for any reason, there is someone who has a general idea of what has been implemented in the subsystem and what work is left
- Defining intermediate goals (e.g. Progress Review goals) are very helpful in making incremental progress without feeling daunted by the work at hand

Software:

- Version control is important as it provides a safeguard against possible mistakes as well as easy access to the codebase
- Maintaining a uniform package requirement is important as it ensures team members do not inadvertently install packages that might conflict with another part of the system
- Maintain a detailed README file to ensure that the basic requirements of the system are listed
- Record lots of Rosbag files of raw sensor data which will allow unit testing of each subsystem
- Robustness is extremely important for each subsystem; keep testing conditions in mind during development and aim to meet it even in the worst-case scenarios, not in the best-case or average-case scenarios

Testing/Validation:

- Ensure that the list of commands required to start the system is well documented so that even a person new to the system will be able to use it
- Perform testing from a completely unbiased viewpoint, this allows you to realize errors in the system rather than sweep them under the rug
- After testing the system once, it is important to branch your code and work in a "demo-oriented" manner to ensure that the system has more robustness
- When performing the validation experiment, it is important that you control the environment and the experiment - stick to the script as best you can
- It is important to achieve repeatability of the system before considering that the system has been completed; the aim should be to get the system to a point that it works when you switch it on, not on the third or fourth attempt

9.2 Future Work

- Structure infrastructure in a client-server architecture instead of performing the computation on-board
- Integrate multiple infrastructures to allow for a better error-catching as well as gathering more useful information
- Switch the communication to a faster and more robust system such as DSRC
- Optimize the deep learning networks to allow for a faster system since this is critical in safety systems
- There are more sensors that can be useful additions to this system such as infra-red or thermal cameras; the system has been built keeping modularity in mind so adding more sensors should be considerably easier
- Make the electronics of the system self-contained and attached to the infrastructures

10. References

1. https://www.autoaccident.com/statistics-on-intersection-accidents.html
2. Crash factors in Intersection-related crashes: an on-scene perspective.
3. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811366
4. https://mrsdprojects.ri.cmu.edu/2016teamd/
5. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-time Object Detection
6. Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
7. 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.
8. https://github.com/mrsd16teamd

Appendix: Parts List

1	Seville Classics Industrial All-Purpose Utility Cart, NSF Listed
2	Jiffyloc Heavy Duty Extension Pole, 4 - 8 feet, Made In USA
3	Jiffyloc Quick Release Adaptor
4	ZED Stereo Camera
5	Set of 10 5/32" pins
6	Jiffyloc angle adaptor
7	Jiffyloc Heavy Duty Extension Pole, 4 - 8 feet, Made In USA
8	Jiffyloc Quick Release Adaptor
9	ZED Stereo Camera
10	<-(set of ten) 5/32" pins
11	Jiffyloc angle adaptor
12	Jiffyloc male thread adaptor
13	NVIDIA Jetson TX2 Developer Kit
14	50 ft, 9 gauge wire
15	range extender
16	Green toggle laser pointer
17	Barrel Jack Connector
18	10.0µF ceramic capacitors
19	TVS DIODE
20	TVS DIODE
21	TVS DIODE
22	SCHOTTKEY DIODE
23	TVS DIODE
24	TVS DIODE
25	LITTLEFUSE
26	LITTLEFUSE
27	LITTLEFUSE
28	LITTLEFUSE
29	SIMPLE SWITCHER® Power Converter 150 kHz 3A Step-

	Down Voltage Regulator
30	INDUCTOR
31	RESISTOR, American symbol
32	LED
33	12V to 5V DC/DC Converter
34	Multistar 11.1V battery
35	Colcase LiPo battery explosion proof case
36	Voltage checker
37	Barrel jack connectors
38	DC barrel pigtail connector
39	Multistar 11.1V battery
40	Tripod
41	Mount stock
42	Xbee pro
43	Xbee dongle
44	Xbee shield
45	REACH RTK KIT
46	3DR 915 Mhz (US) Telemetry Radio
47	GNSS Antenna Pack with Cables
48	Green toggle laser pointer
49	3DR 915 Mhz (US) Telemetry Radio
50	Xbee shield
51	Tripod
52	Jetson TX2
53	Keyboard
54	HDMI Cables
55	MZ-7KE256BW
56	CONN ADAPT MCX PLUG TO SMA JACK
57	STDR2000101
58	HKPilot transceiver telemetry radio set

59	Laser-to-Lidar Nut
60	LITTLEFUSE
61	LITTLEFUSE
62	10.0µF ceramic capacitors
63	Barrel Jack Connector
64	XT-60 Male/Female Pair
65	SMD LED - Green 1206 (strip of 25)
66	250 Ohm Resistor
67	750 Ohm Resistor
68	HDMI to VGA Connector
69	USb to Mini-USB
70	Button Head Hex Drive Screws (Passivated 18-8 Stainless Steel, M3 x 0.50 mm Thread, 6mm Long)- McMaster Carr
71	CABLE ASSY STR 2.5MM 6' 24 AWG
72	Spiral Cable Management
73	Spiral Zipper Cable Management
74	Barrel Jack Connector
75	LITTLEFUSE