WATonomous Autonomous Vehicle Design Concept for Year 2 of the SAE AutoDrive Challenge

Abstract

This report is prepared by the University of Waterloo's WATonomous student design team for the second year of the SAE AutoDrive challenge. This report showcases the progress made by WATonomous over the last year in preparing an autonomous vehicle (AV) for the second year of the competition. These accomplishments are presented in terms of hardware and software integration, as well as cost analysis.

The approach presented in this paper builds upon the Year 1 design to accommodate for more complex autonomy challenges. A relatively low-cost approach is introduced in this paper with a focus on expandability to create a Level 4 self-driving vehicle, attempting to minimize the amount of design changes in the future by designing well for the early-mid levels of autonomy.

The related areas of architecture discussed include sensor suite, sensor mounting and cooling, electrical, sensor fusion, perception, processing and path planning, associated costs and why those decisions were made.

In the previous report, it was found that the low-cost solution without sacrificing performance is to extract information from a 16-channel resolution LiDAR and a 41 FPS camera and utilize sensor fusion to obtain a spatial 3D, human eye representation of the environment. For Year 2, we amend our previous claims by including more sensors to further reduce the vehicle's blind spots. The perceived environment combined with odometry and localization data provides the AV with all of the information needed in the early and later stages of autonomy. Sensor placement remains mostly on the roof of the vehicle to maximize the field of view (FOV). In addition, we've included a front bumper mount to ensure that the AV is able to see the entire surrounding environment.

Introduction

The motivation behind this report is to fulfill the current need in society for eliminating driving accidents, decreasing environmental pollution, and increasing human efficiency. Self-driving vehicles are one of the most sought-after engineering advancements in today's society. They are often regarded as one of the most difficult engineering challenges due to the complexity involved in creating one. Human lives are at stake when operating an autonomous vehicle, and their multi-system sensor, electrical, and software dependencies in addition to handling complex real-life scenarios allow for many methods of failure. In cases of fault and failure, redundancy must be in place to avoid unsafe and undesired vehicle behaviour.

Autonomy is a heavily researched topic of which many different approaches are available. However, finding the balance between cost and safety is one of the most difficult components of the design of an autonomous vehicle. The WATonomous AV system is designed to extract the full benefits from each of the primary sensors and fuse the data together to allow the AV to have a fully perceived environment. This results in lower computational need, lower resolution sensors, and as a result, a relatively inexpensive approach to designing an AV.

The report goes in depth on the design of the middle stages of creating an autonomous Chevrolet Bolt EV and lays out the groundwork for bringing an autonomous car up to the SAE Level 4 Driving Automation standard. To do this, an in-depth analysis is done on the AV's mechanical, electrical, and software systems.

For the mechanical system, an analysis of optimal sensor placement and protective techniques are laid out. A focus is made on FOV and accommodation for additional sensors, as well as different configurations. The electrical design is broken down, analyzing power consumption, optimal power system components and explaining robust electrical wiring measures that are important for ensuring the reliability of the autonomous vehicle when in operation. Additionally, the software architecture is detailed, including an analysis of sensor fusion to leverage the key features from all sensors. Methods that are robust for creating the AV's environment, such as computer vision and 3D point cloud clustering, are analyzed and are used to perform mission planning. Motion control algorithms are then analyzed to properly navigate the AV from the environmental data.

Concept Selection Review

Hardware Design Selection

Sensor Suite Selection

The selected sensor suite was initially designed in Year 1 to be highly modular to accommodate additional system development. In year 2, the selection has vastly remained the same, although more units of each sensor were considered in our design. Redundancy, reliability and maximizing field of view were concerns in maintaining the AV's safety system requirements. To fulfill these, the vehicle must use its sensors to determine the distance, geometric boundaries, and identities of surrounding objects, while also understanding the dynamics and relative position of these objects.

Visual sensors considered include cameras, LiDAR and RADAR sensors. LiDARs are beneficial for detecting objects in a 3D spatial environment 360° around the vehicle. Cameras allow for feature extraction and classification of objects, and RADARs can detect velocity and displacements with high accuracy. Odometry and localization sensors considered were an INS/GNSS device, useful for obtaining accurate linear/angular acceleration and vehicle localization, as well as wheel encoders, useful for obtaining vehicle velocity.

Each of the sensors listed above have areas of strength and weakness. A general comparison of each sensor type is shown in Table 1 below from a scale of 1 (weak) to 5 (strong). All attributes described in Table 1 were considered to select a suite that would strongly cover all aspects when combined and fused together.

Table 1: General Comparison of AV Sensors on Performance

Feature	Camera	LiDAR	RADAR	GPS	IMU
Range	2	3	4	N/A	N/A
Resolution	5	3	2	3	3
SNR	5	3	3	5	2
All-Weather	2	3	5	5	5
Physical Size	5	2	4	3	3
Data Stream (Resolution & Freq)	4	5	2	3	3
Cost	4	2	4	1	1

LiDAR - Camera Combination Suite

The drawback of a single sensor type suite is the required information processing to infer the full range of the spatial, dynamic, and semantic characteristics surrounding the vehicle. A way that allows for extracting the useful features of both types of sensors is to use a moderately channeled LiDAR with cameras. This way, the AV can extract the 3D object detection and distance features from the LiDAR and the classification abilities from cameras.

Odometry and Localization Suite

Having GPS and IMU sensors allows for fusion of multiple data sources, such as odometers and encoders, to decrease the uncertainty of the GPS location, acceleration and pose data. This approach proved quite useful to the path planning group and did not require much changing for Year 2. The primary changes in Year 2 involve expanding the use of the GPS system to roughly localize the vehicle on the provided HD Map of the competition track.

Sensor Suite Decision

The final sensor suite selection in Year 1 consisted of 2 PointGrey Blackfly 2.3MP vision cameras, a Velodyne VLP-16 LiDAR, and a coupled Novatel PwrPak7-E1 GPS and IMU. For Year 2, the overall sensor types are the same with new variants of each added.

Camera Suite Updates

For cameras, 1 more Blackfly 2.3MP and 3 new BlackflyS 3.2MP cameras from FLIR will be added, for a total of 6 cameras. The new BlackflyS was chosen for its improved image resolution, as well as being a descendent of the original Blackfly 2.3MP, making driver integration simpler.

PointGrey cameras were the type of cameras considered in Year 1, in particular, models that leverage Sony's Pregius global shutter CMOS technology which reduces motion blur at high speeds [1]. Another critical criterion that PointGrey cameras fulfill is customizability. They provide full control over parameters such as shutter, gain, and white balance, all of which help in the tuning and performance of perception algorithms. Generally, this has proven to be a good choice, and so upgraded PointGrey cameras were acquired for Year 2 instead of another model.

The decision to continue using the Blackfly line cameras instead of other PointGrey cameras came down to the frame rate. A higher frame rate is beneficial for reducing latency and increasing the responsiveness of object-tracking algorithms. For Year 2, the dynamic challenges require tracking both static and dynamic

objects at relatively slow speeds, therefore the BlackflyS with 41 FPS is more than suitable.

LiDAR Suite Updates

On top of the original VLP-16 LiDAR, 2 new VLP-16s and 1 new VLP-32C will be added. This is primarily to achieve a far higher fidelity 3D spatial representation and to cover major blind spots. The VLP-32C will replace the existing VLP-16 as the center LiDAR, while the 2 VLP-16s will be placed on the sides of the roof rack at a 45° angle to give a better view of the sides of the car. The last VLP-16 will be mounted on the front bumper and serve to give a better view of the lower elevation in front of the vehicle.

For Year 2 challenges, the LiDARs will be used to detect and obtain distances to 3D objects in the object avoidance and navigation challenges. The cameras will be used to detect, classify, and understand the many traffic signs and symbols, as well as parking spaces and lane lines to complete all three of the dynamic challenges. The 6 cameras will be used to maximize the field of view in the environment, as well as cover blind spots. The LiDAR's superior range and point cloud clustering will allow the vehicle to see three dimensional objects further ahead.

INS Suite Updates

Finally, the Novatel GPS and IMU provide accurate location, velocity, and acceleration information, which is necessary for path planning and decision making, specifically to estimate the vehicle's position and dynamic states at all times without compromising on low speed accuracy. A new key use of the INS in Year 2 is to assist in hardware time synchronization between sensors. This was achieved through the use of a high-accuracy reference clock signal provided by the GPS. Specialized software executes a best master clock algorithm to detect the reference signal based on the hardware identifier, clock quality, and variance. Routing this signal to the designated GPIO inputs on the cameras and LiDARs will provide the required time-stamping between the GPS clock signal and the slave sensors.

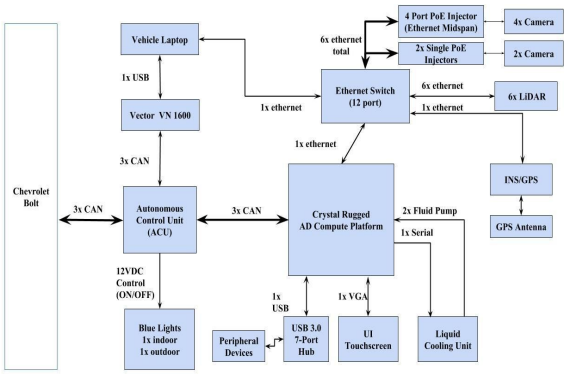
Sensor Suite: Considerations for Future Expansion

The current sensor suite provides the vehicle with significant sensing capabilities which are more than enough for Year 2. Since the sensor placement on the roof of the vehicle is customizable, the sensing configuration can be optimized for greater sensing capabilities. Expansion for future years will mainly focus on improving the Time Synchronization PTP network as well as including Continental RADARs for a greater FOV coverage. The addition of Continental RADARs will allow for object detection in varying weather environments, greater accuracy nearby detections, as well as high fidelity object tracking that is superior to LiDAR.

Computing Platform

Careful consideration was taken during the design of the electrical power architecture to make it expandable and adaptable. This was achieved, in part, by using specialized power components to provide centralized junctions that allowed for the connection of multiple components. Figure 1 shows a diagram of the structure of the computing platform.

Figure 1: Sensor and Computing Signal Architecture



Specifically, a 12 position circuit breaker block from Blue Sea Systems was used to distribute 12V DC to the various sensors and hardware elements that require it. This block can safely and easily allow for new components to be integrated into the current power architecture, including the competition scoring equipment. In Year 1, the choice of a 16 port ethernet switch was motivated through a worst-case port usage estimation with 6 RADARs, 4 LiDARs, 1 compute server connection, 1 INS, and 4 cameras, totalling to 16 used ethernet ports in future years. For Year 2 the connections would be: 1 INS, 1 Computer, 5 LiDAR and 6 cameras, totalling 13, which is sufficient. Extra hardware will be required for adding 6 RADARS in Year 3. A 4 port PoE injector switch was acquired for the cameras in Year 1. Since 6 cameras will be used, 2 more single PoE injectors were acquired.

Other considerations include a 1000W inverter with two 6-port power bars (resulting in a total of 12 AC ports to accommodate hardware that needs a 120V AC input) and a powered USB 3.0 hub to expand the USB capacity and range of the compute server. Custom hardware for the power distribution for the RADARs is being developed but will not be used in Year 2. While these choices vastly exceeded Year 1 requirements, they were chosen with future expansion and integration in mind.

The Intel Compute Platform was integrated into this power architecture through a 12V to 24V DC-DC Converter. A liquid cooling unit was acquired to help with temperature management of the Compute Platform. This was necessary as the computational

requirements of the autonomy software stack can easily cause temperature failures of the system if not properly regulated. The cooling unit communicates with temperature sensors installed inside the Compute Platform and manages the flow of coolant liquid into the system. The Compute Platform is connected to an ethernet switch for data communication between all of the sensors on the car. Three DB9 cables are routed inside the vehicle to the Compute Platform's CAN-PCIe card, enabling the ability to send and receive CAN bus communication from the vehicle on the Low Speed, Chassis Expansion, and High Speed buses. This is the interface that allows controls to be sent to autonomously drive the vehicle. A PTP-PCIe card was acquired and integrated into the Compute Platform to assist in performing hardware time synchronization between the sensors. This will help significantly in developing an expandable and modular time synchronization PTP network to improve the quality, latency, and bandwidth of the sensor data pipeline.

Sensor Capability Analysis

Year 2 poses many newer, complex challenges that were not presented in Year 1, including dynamic obstacles and traffic flow control signals. Due to these new changes, a list of criteria was established to select appropriate models and number of sensors to use for the Year 2 dynamic challenges, including range, accuracy, sample rate, resolution, and cost.

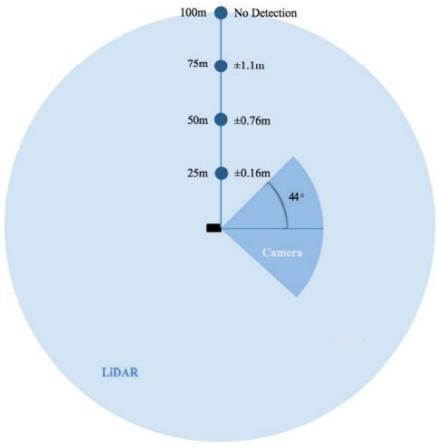
The maximum range of the sensors was an important factor to consider when selecting an appropriate sensor, as it dictated hard limitations in the vehicle's coverage to sensing the environment. The accuracy of the sensor was also included as a critical criterion, as it directly determines the quality of measurements taken by the vehicle and thus affects the output of the quality of real-time vehicle software detection algorithms. The sample rate of the sensor describes how frequently a sensor's data is able to submit new information to be used by software modules on the vehicle. This is especially important for the vehicle to detect such a dynamic and fast-changing environment, with the ability to detect sudden and possibly erratic changes in obstacles around the vehicle. Another important criterion chosen was sensor resolution, which determines precision of measurements and how much detail is present in the sensor output. Lastly, cost is an important factor in sensor selection, as it is ideal to have an affordable solution for autonomous driving and in the case if the system would become commercially available.

Sensor Verification and Effectiveness

All sensor testing and verification measures were performed at the vehicle level with the sensors mounted directly on the vehicle. This provided the means of verifying sensor capability that is constrained to the environment in which it will operate in.

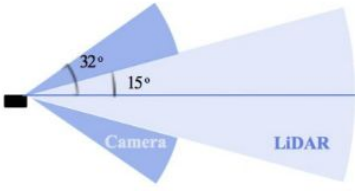
The first test involved a range test for the LiDAR, where a reflective board was held starting 100 m from the LiDAR. The vehicle slowly moved toward the board until about 2 m away. When replaying the data gathered during this test, no points were shown to have reflected off of the board until approximately 96 m away. Although this is slightly less than the VLP-16 specifications, it exceeds the minimum detection distance outlined in the sensor capability analysis by around three times the stopping distance. During this range test, an accuracy test was also performed. At four separate checkpoints, 100 m, 75 m, 50 m, and 25 m away from the reflective board, the distance from the board to the LiDAR was measured with a measuring tape. The accuracies at each distance are shown in Figure 2. Unfortunately, the accuracy level at 50 m is above the ideal value and hence this error factor must be taken into account within the software at longer distances.

Figure 2: Horizontal FOV with uncertainties for LiDAR



Similar LiDAR range tests were completed with the reflective board in the vertical direction to determine the vertical field of view of the LiDAR. The reflective board was in view until it exceeded about ±15° from the horizontal plane, which matches the specifications. Refer to Figure 3 for a visual representation of the vertical FOV.

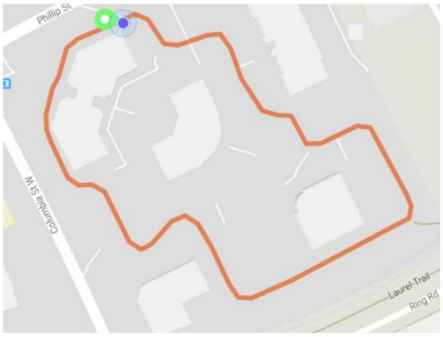
Figure 3: Vertical FOV for Camera and LiDAR



The next test a small traffic cone object was held at a close distance to the camera, and gradually pulled further away. At the point where the object was viewed as less than 10 by 10 pixels in the image frame, the distance from the object to the camera was measured with a measuring tape. The resulting range of the camera is about 46 m, which exceeds the minimum stopping distance by more than 17 m. Next, the horizontal FOV was determined by measuring the angle at which an object just disappeared from view in the image frame. The resulting horizontal field of view is ±44° from the center axis. The measured range and horizontal FOV is shown in Figure 2. The same method was applied to measure the vertical FOV, resulting in a value of about ±32° from the center axis. The vertical FOV of the camera is pictured in Figure 3 above.

The last sensor verification test involved the Novatel INS system. First, the time to first fix for a hot start of the GPS was tested, which took 4 seconds on average with low variance. Next, the susceptibility of the INS system to experience drift was tested. Starting and ending in the same location after a 5 minute drive in the route shown in Figure 4, the drift in the X and Y coordinates returned by the Novatel system was measured. In the x-axis, the drift was approximately 0.1598 m, and in the y-axis the drift was approximately 0.01348 m from the starting position. Both of these values are small enough such that they can be mostly attributed to inaccuracies of driving the vehicle to its exact starting position at the end of the route.

Figure 4: Novatel INS Drift Test Route Map



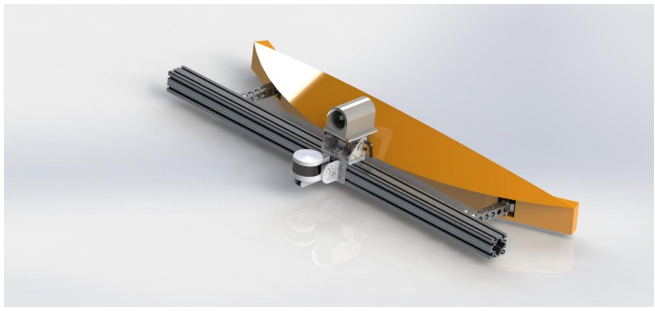
Hardware Integration Design Review

Robust Hardware Integration

Front Bumper Sensor Mount

During the Year 1 competition, the team did not have any sensors at the front of the car. But to minimize blind spots and to improve lane detection, it was decided to have a fisheye camera and a LiDAR at the front of the car. To accommodate these sensors in the front of the car, a robust mechanical structure was designed and to ensure its safety and robustness, FEA analysis was carried out.

Figure 5: Front Bumper Sensor Mount



An 80/20 3*3 inch aluminium extrusion was used as the main mechanical platform for the different sensors. This extrusion was attached with the bash bar of the car using custom designed connectors.

A camera mount with three degrees of freedom was designed for the fisheye camera. This mount was assembled onto the main mount structure. A custom designed camera shell was used to enclose the camera. An acrylic clear dome was used with the shell to ensure the view angle of the fisheye lens is not obstructed.

The LiDAR mount consists of four distinct parts in its assembly (Pivoting bracket, Rotating Arm, Binding Barrel). However, an additional secondary part can be associated to this assembly (i.e. the horizontal 80/20 quad. extrusion attached to bumper).

Two Pivoting brackets attach directly to the bumper extrusion via two screw hardware. These are speciality hardware consisting of a hex screw and sliding plate which allows for horizontal movement within the extrusion. However, the other side has holes in a circular pattern with each at a 20^0 difference. Furthermore, to achieve a tilt in between these degrees, another set of holes are drilled to achieve 10^0 difference with respect to the outer holes.

The rotating arm serves the purpose of holding the LiDAR and providing an extension from the body to allow clearance and avoid signal and mounting interference during the LiDAR's data sweep. The arm is pivoted at one end of bracket using a binding barrel to fix it while it is set to a desired angle using a second barrel.

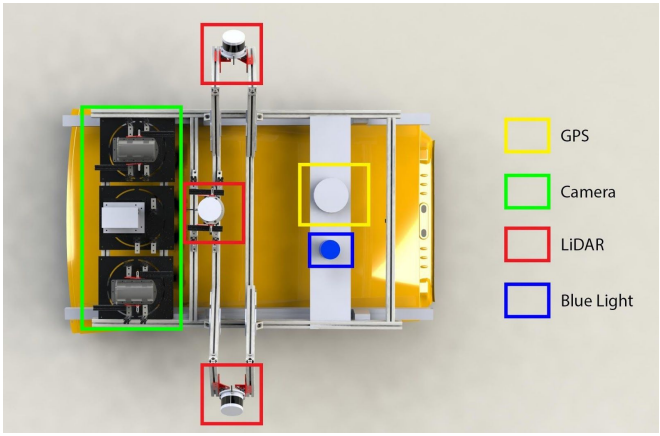
Roof Sensor Mount

In Year 1 competition, the roof rack was the only location where external sensors were mounted. A limited number of sensors could be mounted as only a small plate was attached to the top of the vehicle roof rails. The sensors were also fixed in place and could not be adjusted to satisfy the software team's needs. In Year 2,

many more LiDARs and cameras were added, and each sensor was placed on an adjustable mount.

The sensors used for Year 2 include three cameras (two of which are normal angle FOV and one with a narrow FOV), three LiDAR sensors, and an INS system. All sensors have been mounted to the exterior of the vehicle, except the IMU component of the Novatel INS system. The IMU serves best when placed in the centre of the vehicle, it has thus been mounted inside the vehicle, near the vehicle's centre of the mass.

Figure 6: Roof Sensor Mount Top-level View



NVH Robustness

To make sure that front bumper mount is robust enough, FEA analysis was carried out. it was designed such that, it can carry the weight of an adult in static loading condition and 5G forces in dynamic loading condition to survive any shocks from potholes and bumps. Moreover, natural frequencies of the mount were identified, and it was made sure they do not coincide with natural frequencies of the car, which are 45 Hz and 124 Hz.

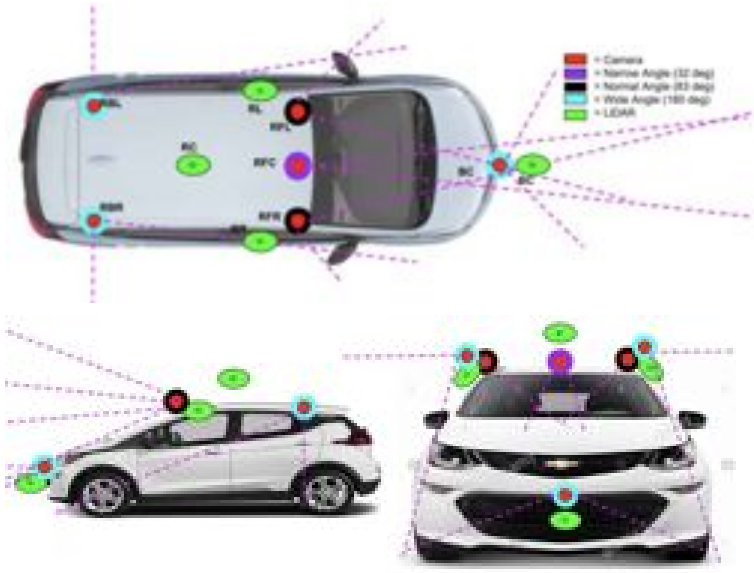
The biggest challenge in mounting sensors on the roof rack was mitigating vibration. This is needed so the effects of bumps and other irregularities in the road are not transmitted into the sensor readings. For the first prototype mount design, the main plate was fastened directly to its mounting brackets (metal on metal contact) to get an initial understanding of the required damping for the sensors. In this configuration, camera and LiDAR images were stable.

Although the original design was stable enough for its use case, rubber spacers were still added between the mounting brackets and main plate in the final sensor mount to reduce vibration that could otherwise damage sensors over long periods of operation. Stiffening members have also been added to the main plate such that road forces do not excite it at its natural frequency of 45 Hz and 124 Hz for the first and second natural frequencies respectively.

FEA and vibrational analysis were performed to verify the structural integrity of the centre and side mounted LiDAR components when subject to vibrations that the average car may experience on a daily basis.

Sensor Placement

Figure 7: Sensor Placement Diagram



Many changes were made with respect to difficulties completing some of the Year 1 challenges. The Year 2 design incorporates 3 additional LiDARs and 4 additional cameras in order to reduce the sensor suite's blind spots and for redundancy purposes. Similarly to the previous design, the IMU is placed near the center of mass of the vehicle, the GPS is mounted without any ceiling obstruction on the roof, and the front left (RFL in Figure 7) and front right (RFR) cameras remain for the use of stereo vision for depth perception.

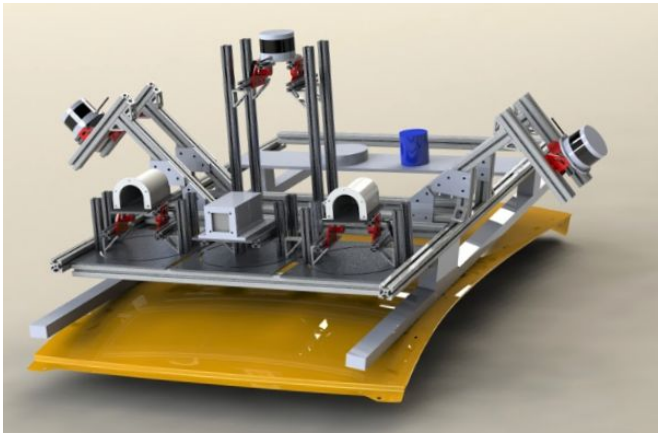
In order to address the previous blind spot extending up to 3m in front of the vehicle, the cameras were raised and angled down. This effectively reduced the blind spot to be within 1m of the vehicle, with the last meter to be addressed by the front bumper camera (BC). To compensate for the loss in long range vision, the new roof sensor suite includes a centre camera (RFC) equipped with a 12.5mm focal length lens for long range object classification.

As a result of raising the cameras, the centre VLP-32C LiDAR (RC), placed as such to make the most out of its FOV, had to be raised as well. Two additional VLP-16 LiDARs were mounted on the side of the vehicle tilted down (RL, RR), as well as one VLP-16 on the front bumper (BC) to further increase redundancy and avoid collisions.

Sensor Mounting

80/20 Extrusions were attached to the roof rails of the vehicle to create a roof rack. This allowed for the ability to mount many more sensors compared to year 1.

Figure 8: Roof Sensor Mount Front View



The centre LiDAR was centered along the width of the vehicle and mounted in the front half of the vehicle length wise. To mount the four pillars of the stand to the roof mount frame, four gusset brackets were used. This allowed for adjustability in terms of the position of each pillar, which is a recognized benefit to using T-slotted framing components. Additionally, four more gusset brackets were attached to each column, and two 6-inch aluminum extrusion cut outs were mounted on each side, acting as the base for the tilting mechanism. A pair of 90-degrees pivot brackets were utilized to create adjustability in the preferred field of view for the centre LiDAR, which allows the software teams to adjust the tilt angle based on specific future needs. Lastly, a customized LiDAR plate was designed in order to attach the LiDAR to the tilting mechanism. The overall height of the centre LiDAR plate was increased from the year one competition, allowing the LiDAR to generate mapping views without obstruction of the pillars which support it.

Two VLP-16 LiDARs were chosen to be mounted on the side of the vehicle. The sensors are placed in the middle of the vehicle lengthwise and suspended on an angle over the vehicle's B-pillar. Their placement enables a larger range of detection from the year 1 vehicle, which incorporated a single center mounted lidar. These additional side mounted sensors allow for the detection of objects on the side of the vehicle, effectively reducing the size of the side view blind zones.

The side lidars are also mounted using T-slotted extrusions and framing components. The primary extrusions are suspended off the top surface of the roof rack at a 45-degree angle upwards. They are supported by shorter upright extrusions, which are fastened to the primary extrusions using surface brackets. Surface brackets were also used to fasten the primary extrusions to the existing frame. A secondary extrusion is fastened perpendicular to the primary and allows for the mounting of the same lidar tilting mechanism, as previously discussed. The three LiDARs do not require any modifications or external housing to be temperature and weather resistant.

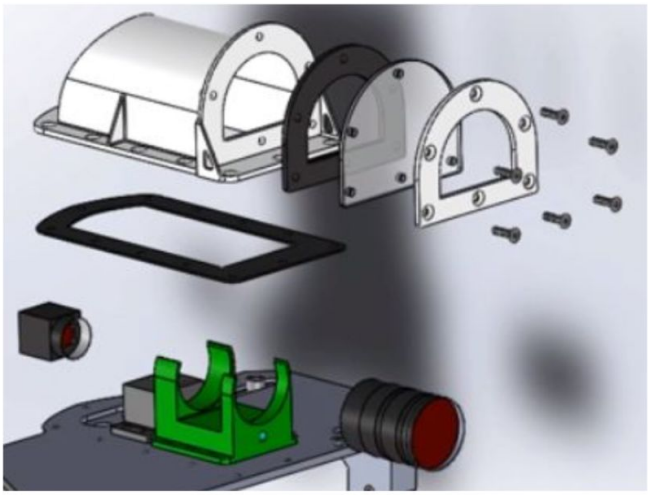
Three cameras are mounted on to the front of the roof rack equally spaced across its width. The left and right cameras use a normal FOV angle lens. The four aluminum extrusions oriented vertically act as pillars of the mounting structure and are directly bolted onto the base which is fastened to the roof rack. Gusset brackets are attached to each extrusion which act as a support for the two aluminum extrusions that lay horizontally. These two extrusions act as a support structure for the camera plate which is fastened onto them. The plate has slots on the both side through which it is attached to the pair of aluminum extrusions which allows for adjustability in the forward and backward direction. Although this adjustability can currently only be executed manually, in the future this adjustability is meant to be motorized. Finally, the camera and lens are mounted on to a copper block heat sink and a 3D printed stand respectively. Both of which are directed bolted onto the camera plate.

As shown above, the mounting design for side cameras is very similar to the front camera design. However, unlike the mounting of the front camera, the side camera mounting includes a pair of 90-degrees pivot brackets which allows the cameras to be tilted and fixed at varying angles as required by the software team. Furthermore, the 3D printed stand for the lenses differs geometrically from the one for the center camera mounting as the dimensions of the lenses and side cameras is different than that of the center camera.

The two cameras are enclosed in custom 3D printed housing with an opening in front for a polycarbonate sight window. Both the sight window and the bottom of the housing are gasket-sealed to provide waterproofing. Since the LiDAR is intended for automotive use with an IP67 rating, this level must also be sufficient for protecting the cameras, inspiring the decision to create a camera housing design that achieves IP67. This means that the camera housing must be dust tight and watertight down to a 1000 mm depth for 30 minutes. Gasket material is selected with temperature in mind, and fastener spacing, and torque specs are calculated to compress the gasket by the vendor’s recommended amount of 25%. Additionally, the cameras are sensitive to heat with a maximum operating temperature of 45°C. To assist in cooling the cameras,

they are each mounted on top of a copper plate that will act as a heat sink. The camera covers are also covered in material that reflects light such that the plastic casing does not absorb heat from the sun, keeping the cameras cooler.

Figure 9: Exploded View of Camera Housing



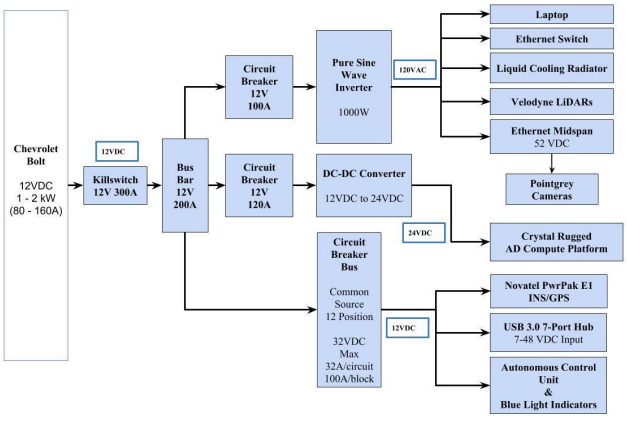
A metal plate is along the rear of the vehicle. This metal plate allows for the mounting of various antennas such as the scoring antenna, and the GPS unit that is used. Standard fasteners are used as these units are not susceptible to vibration. A blue autonomous light is also mounted using standard fasteners. This light was necessary for the competition.

Electrical Integration

A robust electrical integration of the sensors was ensured in a variety of different ways. First, the wiring of the entire electrical sensing subsystem was methodically planned out and executed. Great care was taken to source ethernet cables (RJ45 CAT6) with a high-quality mechanical clip mechanism to ensure ethernet connections cannot become dislodged or disconnected during vehicle operation. Additionally, the slack and specific path throughout the car of the various cabling was managed carefully to ensure any shifting during operation would not result in any tension, compression, or stress of any kind on the cables. This was achieved through carefully routing cables according to a routing plan the electrical team created, using a combination of zip ties, cable clips, cable glands and zip tie mounts to secure the cables running through the car and around the server rack. Any custom sensor power and data wiring was concealed with coaxial cables and rigid wiring enclosures to ensure secure and sheltered connections. For our LiDAR cabling, we ensured strong connections between the LiDAR sensors and interface box and added secure casing that allows us to debug connection problems and measure power consumption whilst guaranteeing flexibility to the system's cabling. To secure cables that cross the interior and exterior of the vehicle through the roof, watertight cable glands were used to eliminate water leakage.

The power system was also designed in such a way as to ensure connections are safe and secure, and important components are protected through the use of circuit breakers. A breakdown of the power system architecture can be seen in Figure 10. Initially, the 12V DC input from the vehicle is protected by a marine battery switch, ensuring there are no live wires or buses anywhere within the power architecture except at that single point of contact. From there, the power is distributed to a 12V DC, 150A busbar (with a parallel ground bus bar), that connects to the three main power subsystems: the DC-DC converter for the compute system, the 12V DC circuit breaker block for sensor components, and the pure sine wave inverter to generate 120 VAC. The sine wave inverter and DC-DC converter are both protected by separate, properly-sized circuit breakers, while the 12V circuit breaker block allows for each outgoing channel to be protected using small, modular circuit breakers. Most devices connected to the 120VAC bar will have integrated certified DC rectifiers to ensure a reliable and safe DC input. This is all done to ensure that in the event of an electrical failure or short circuit condition, the circuit breaker for that individual component will disconnect it from the overall system, and the rest of the power system and the vehicle's battery system will remain isolated, and thus unaffected.

Figure 10: Power System Architecture



Finally, the electrical platform also ensures robustness through the sizing of the power wires that connect the various components together. Worst case current and power calculations have been done on all stages of the power architecture, and appropriate gauges and

lengths of wire have been selected and implemented to ensure that they are rated for greater than the worst-case current draw. This ensures that there is minimized voltage drop in the power wiring, as well as reducing the risk of any catastrophic failures that might occur during a short circuit condition, such as an electrical fire.

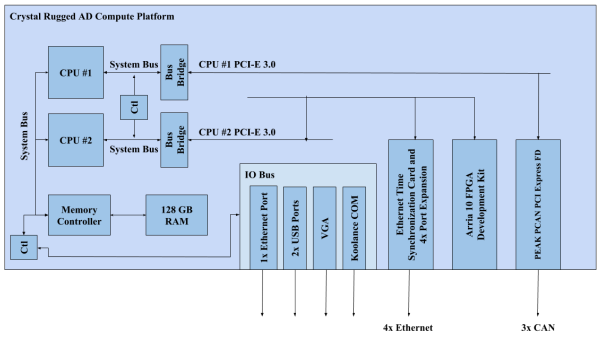
Compute Platform Integration

Serial Data Vehicle Integration

Equivalent to Year 1, a PEAK PCAN PCI Express FD card is used to create a fast and robust serial connection from the Intel Crystal Rugged to the vehicle via the CAN communication protocol. A baud rate of 500,000 bits per second is used on the High Speed and Chassis Expansion buses to achieve fault tolerance. The Low Speed bus uses a baud rate of 33,000 bits per second due to the lower priority of the traffic on this bus.

An ethernet expansion card allows higher ethernet bandwidth through the 8.0 GHz PCIe 3.0 bus lanes and enables external hardware and software time synchronization. Figure 11 shows a general layout of the signal architecture inside the Crystal Rugged.

Figure 11: Crystal Rugged Signal Architecture



In addition, the Vector VN1640A bus interface hardware tool is also connected to the CAN buses to act as a secondary method to communicate with the vehicle for prototyping purposes.

Computing Platform Cooling Strategy

To source the liquid cooling loop for by the Crystal Rugged, the thermal output of the various components requiring cooling in the Crystal Rugged has been considered, as shown in Table 2. The total thermal output of the computing platform in the current state was found to be 390 W. Accounting for the fact that another FPGA will be added in the future as well as other high heat output electrical devices, including power converters, the Koolance EX2-1055 Computer Liquid Cooling System was chosen as the liquid cooling solution. It should be noted that this is an all-in-one unit and provides up to 900 W of heat dissipation. Additionally, this is different from the 1450 W ALX-1450-P400 Modular Liquid Cooling System recommended by Intel. This decision was made as the former is half the cost of the latter, meets the design specifications, and consumes less power. Additionally, the liquid cooling unit is equipped with three temperature sensors at the Intel Xeon processors and Arria 10 FPGA Development Kit to monitor the compute systems temperature at critical regions. If any of these sensors exceed 55 °C, an interrupt signal is sent to the CPU to safely shutdown the system and avoid damaging components.

Table 2: Thermal Output of Computing Components

Component	Qty	Max Temp. (°C)	TDP (1 unit, W)	Extended TDP (W)
Intel® Xeon® Processor E5-2699 v4	2	79	145	290
Arria 10 FPGA Development Kit	1	45	100 (Dependent on usage)	100
Total				390

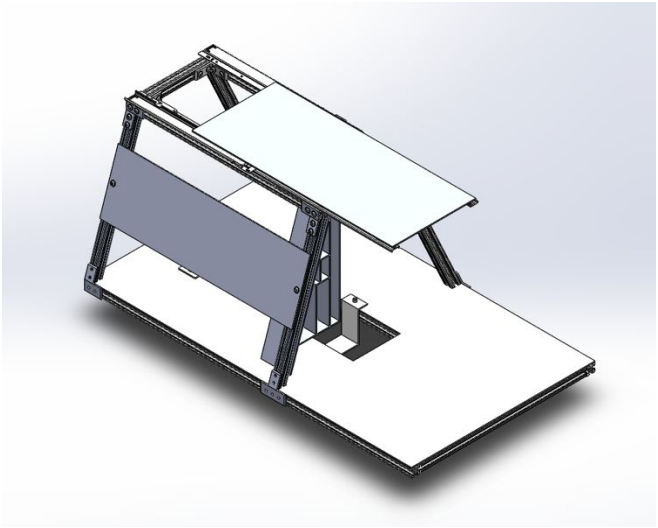
It is known that mixed metals are being used in the liquid cooling setup within the Crystal Rugged, which means that using a water-based coolant will result in corrosion within the loop. As such, it has been decided that the Koolance 705 Electrically Insulative Liquid Coolant is the best option for coolant, as it contains corrosion inhibitors and is known to be used for 2-3 years before requiring replacement. This coolant, while more expensive compared to alternative options, is expected to save time and money over the duration of the competition, requiring infrequent refills and maintenance. An additional benefit of the electrically insulative coolant is that the cooling system can be placed above the other electronics without concern. The top of the computing rack is the optimal place for the cooling system as it provides an open area for the fans to vent and allows for quick coolant refills.

Computing Platform Mounting Strategy

The compute rack was designed in Year 1 as a 3-level structure constructed using thin-walled aluminum tubing attached together by aluminum gussets and rivets. The main benefits of this construction technique were that it allowed for a very lightweight and cheap frame while also making use of exclusively off-the-shelf components. By using readily available framing, design and build time is cut down immensely. Acrylic sheets form the mounting surface for the electronics on each layer. Heavy electronic components that do not need to be accessed or serviced frequently are placed on the bottom tier, which is sunken into the trunk. The Crystal Rugged, radiator, switches and other critical components are placed near the top so that they can be easily serviced during testing. The exact component placement has been determined in CAD and the compute rack has been sized such that empty spaces are minimized.

Based on the review and analysis of the previous rack, a need was found to improve port accessibility while increasing the storing capacity of the rack. Currently, the hardware mounted on the lowest shelf is virtually inaccessible. Also, since the Crystal Rugged is front facing with ample wiring and tubing connections, the midspan and other heavily used network devices require effort and time for interaction while testing. As a result, a new design has been modeled for this purpose and is shown below.

Figure 12: Compute Platform Year 2 Mount Design



It is made of off the shelf items including 8020 square extrusion of size 1x1in. The bottom plate holds most of the electronics, while the top plate holds only the radiator. The plate on front is used to mount the kill switch, USB hub and Velodyne interface box. This simplifies interaction from both the front and the sides. All the wiring has been moved to the side by mounting the crystal on the bottom plate clamps. A dedicated shelf for network switches allows for easier alteration of sensor configurations. All the other hardware stored below the top plate can be accessed by moving it on sliders when required. A damping pad was also introduced to dampen any induced vibration during motion.

Power Consumption

Power consumption on the vehicle is highly restricted by the capacity of the 12V battery. The 60 kWh battery supplies a range of sensors, radiators, switches, and the compute platform. To ensure that the battery only needs recharging at reasonable intervals, power consumed had to be strategically restricted and regulated. This is to ensure that sensor scalability will not compromise battery life in lieu of greater precision. From a power system perspective, it is approximated that the total rated power consumption limit is 1 kW including power consumed by equipment in the front of the car. Hence, to limit the scope to additional equipment non-native to the vehicle, it is safe to say that our suite should consume no more than 800W of power. Initially, calculations were done based on equipment datasheets and technical information to determine the estimated worst-case loading. Later, these values were validated through experimental measurements on the built power system.

Both the estimated power consumption breakdown and measured values are presented in Table 3 below.

Table 3: Power Consumption by Component

Component	Estimated Power Consumption (W)	Average Measured Power Consumption (W)
DC-DC Converter	689.00	505.20
Compute Platform	659.00	-
900W Radiator	30.00	-
AC Inverter	~ 108.70	~129.98
Network Switch	~ 8.40	-
Total 6 x PoE Injectors	~ 40.80	-
Total for 6 x Cameras	15.00 (maximum)	~ 12.48
Total for 4 x Velodyne VLP-16 LiDARs	32.00	~ 28.80
Total for 1 x Velodyne VLP-32 (including interface box)	~12.50	-
Circuit Breaker Block	82.00	~ 52.84
USB 3.0 Hub	5.00	-
INS/GPS System	5.00 (1.80 Typical)	-
Total for 6 x Continental ARS430 Radars	72.00	~ 46.12
Total (worst case)	879.70	~ 688.02

There are three major power consuming modules in the design: DC-DC Converter, Sensors & AC Inverter, and Circuit Breaker Block. The DC-DC Converter powers the compute platform and liquid cooling radiator in turn. The sensors used are powered through the AC Inverter module which also powers the AC components in the car. Finally, the Circuit Breaker Block powers the other DC components in the car.

For design purposes, our system has been optimized with scalability in mind. Thus, the design supports sensors with the quantities detailed in Table 4.

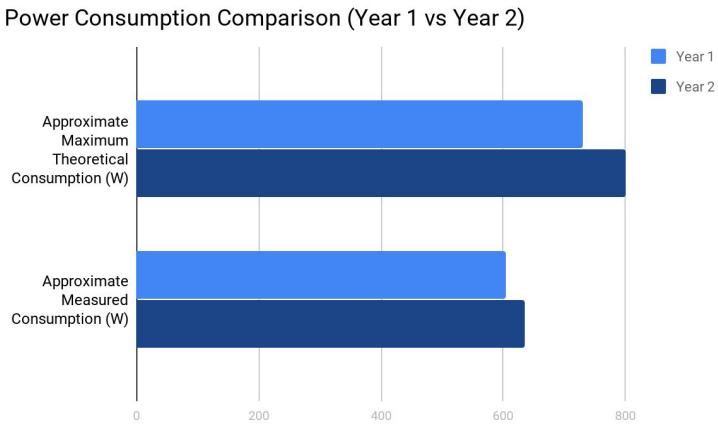
Table 4: Maximum Number of Sensors Supported

Component Name	Quantity Supported
Velodyne VLP-16 LiDAR	4
Velodyne VLP-32C LiDAR	1
Continental ARS430 Radars	6
Blackfly 2.3 MP IMX249 Camera	3
Blackfly 3.2 MP IMX265 Camera	3

It should be noted that while the maximum power draw theoretically exceeds 800W, this is only the worst case based on documented specifications. However, these values are by far away from the practical maximum power readings observed when testing. Should there be a need to reduce power consumption further, sensors impact analysis on both power and precision achieved can be conducted to decide on which sensors are to be removed for performance.

The main transition between power consumption in Year 2 from Year 1 is sensor scalability. In Year 2, we scaled up our LiDAR suite as well as our cameras as mentioned in the Hardware Design Selection section of the report. Values presented in Figure 11 represent a comparison between the power consumption (both theoretical and experimental) in Year 1 and Year 2. In particular, using experimental values, there was only a 5.1% increase in power consumption from Year 1 to Year 2. After analysing the impact of the additional sensor, it has been determined that the vehicle would be able to power the added sensors without compromising our power restrictions mentioned above.

Figure 11: Power Consumption Analysis by Year



For the compute platform itself, a stress testing program was run to achieve 100% load on all cores of the Intel Xeon processors as well as the Arria 10 FPGA Development Kit before the current was measured, and this resulted in a 500W load on the vehicle, which is much less than the theoretical estimate of 689W. For the AC Inverter, values may seem much higher than the theoretical worst-case scenario. However, this is due to the power drawn by a monitor connected which is used during development on the compute platform. Despite that, the measured values validate our assumptions about the estimated power draw. Hence, due to the available headroom for hardware integration (for example, adding additional cameras to our suite), the power consumption of the vehicle is optimized for sensor scalability.

Bill of Materials

Cost Analysis

With generous sponsorship and additional external funding, the team was able to procure numerous pieces of physical equipment and software to ameliorate the implementation of the design. The costs summary and a detailed bill of materials (BOM) can be found in Appendix A. In summary, the total cost of the vehicle, sensors, computing platforms, mounting and cooling and other miscellaneous components added to the vehicle stands at $USD 132,789.15. Similarly to Year 1, the largest portions of costs fall under the vehicle itself, followed by computing platforms, then sensing. Cost efficiency determined by performance-to-cost analysis and quality assurance were both crucial factors in each purchase decision to ensure optimal performance.

Velodyne VLP-16 LiDARs

In Year 1, the VLP-16 LiDAR was determined to be the most cost-effective sensor given its 360°, 3D distance and calibrated reflectivity measurements. In Year 2, the team was the recipient of generous donations of 3 additional VLP-16s and a VLP-32C. Although the HDL-32E and HDL-64 were considered for their higher resolution and increased FOV, it is possible to design similar solutions using the sponsored sensors.

Blackfly 3.2MP GigE Camera

Since Year 1, PointGrey released a new Blackfly model featuring the IMX 265 CMOS and a higher image resolution at the same cost [1]. While the minor FPS drop was considered, the advantages of the new Blackfly model would allow for significant image resolution improvements at long range, a key feature for the Year 2 challenges.

Sensor Mounts

The Roof Rack and the Front Bumper mount were both designed with cost efficiency in mind. Both mounts were made to be highly adjustable in order to try out various sensor suite configurations, saving cost through reusability. Additionally, some parts were listed by Canada's *Special Import Measures Act*, and as a result, another criteria of design was to be sourced from Canadian suppliers at reasonable cost [2].

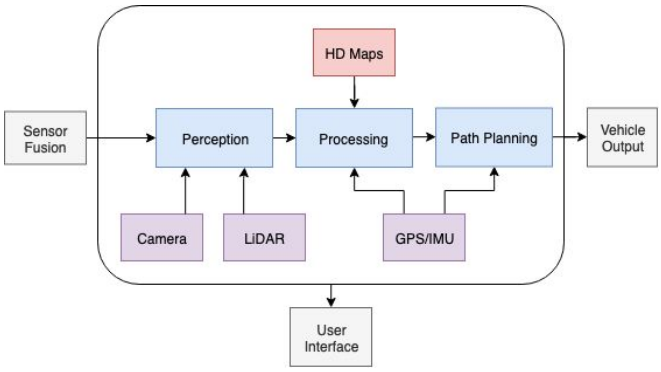
Software Design Review

This section will review the design of WATonomous' software modules.

Software Architecture

The Year 1 software architecture was designed with consideration for low-level, embedded software and higher-level, decision-making software. A data pipeline was designed to describe the interactions between different components of the autonomous vehicle's software stack. These components, referred to as *nodes*, were implemented as standalone executables using the Robot Operating System (ROS) framework. In Year 2, the functional and organizational advantages of decoupling our software tasks with respect to the ROS nodes are still leveraged, allowing our production and consumption of data to benefit from built-in concurrency, open source libraries and packages for robotic applications within the ROS community. The general software architecture is shown in Figure 12 below.

Figure 12: General Software Architecture



This effectively caused the formulation of subteams that contribute to the software architecture. These software subteams are broken down as follows.

Sensor Fusion
The Sensor Fusion subteam ensures that the raw data from various sensors is accurate, filtered and synchronized before it is provided to the software subteams.

Perception
Perception primarily deals with computer vision algorithm development required to accurately detect and classify objects of interests, such as pedestrians, cyclists, road markings and traffic control objects.

Processing
Processing is responsible for tracking the positions of objects identified by Perception, fusing the detections and classifications from Perception into a unified internal representation of the car's environment, as well as localizing the vehicle onto the provided HD Map.

Path Planning
Path Planning uses the environment it receives from Processing to plan the trajectory and velocity the car should travel based on the appropriate behaviour when encountering pedestrians, cyclists, road markings, traffic control objects. This subteam also provides the waypoint management system as well as the controllers necessary to maintain the target path.

Embedded Actuation (Vehicle Output)
This subteam is the endpoint between the software systems and the existing systems in the vehicle. Their main task for the software architecture is to provide the interface between the Path Planning controllers and the vehicle CAN bus.

User Interface
The User Interface subteam is tasked with producing a graphical interface to allow an end user to interact with the autonomous vehicle, in addition to being able to show an overview of the vehicle's status. Additionally, this component provides visualizations for each node's output for debugging purposes.

There are additional subteams that do not have their own ROS node as their work is not directly required for the vehicle to be autonomous. These are broken down as follows:

Global Mapping
In Year 2, Global Mapping remains a subteam focused on meeting the Global Mapping challenge: providing an application allowing for location search as well as routing from a point A to a point B. It is expected to integrate this module into our global path planner by Year 3.

Simulation
In Year 2, the Simulation subteam's purpose is to meet the Simulation challenge's requirements: incorporate our autonomy components into the simulated agent's behaviour within provided simulation environment.

DevOps
The DevOps subteam is in charge of ensuring that the various modules and components created by the subteams are able to be brought together. Tools are built to facilitate continuous integration and development environment setup.

Sensor Fusion Algorithm and Computer Vision

The first step in this pipeline is perception. The perception team consumes processed images from the cameras mounted on the vehicle to produce a data representation of the forward road. Overall, there are four significant road features in the challenge: roadlines, traffic signs, traffic lights, and other 3D objects.

Roadline Detection

Roadlines include lane lines, crosswalks, road boundaries, stop lines and parking lines. These detected roadlines are used to construct a map of the vehicle's surroundings, and to help the vehicle make informed decisions in its environment. To detect road lines, an image taken from the left camera is resized to 512 x 512 pixels and are inputted into a semantic segmentation neural network that produces a binary output image. The binary image highlights locations of road lines in the image, where pixels corresponding to road lines are white and all other pixels are black. A neural network approach was selected instead of heuristic computer vision

approaches because the neural network approach is more robust to changes in lighting conditions and in roadline colors, and will reduce the need for heavy calibration work to be done once the team arrives at the competition site. After numerous testing and tuning, the SegNet semantic segmentation model with skip connection was selected as the neural network to be used due to its high performance in intersection over union (IoU) when tested on validation datasets [3]. The IoU defines the overlap percent between a ground truth label and a predicted label, and is a metric used to quantify the similarity between computer vision predictions and truth labels.

After the binary roadline image is generated, the next step is to perform a bird's eye view (BEV) image perspective transform to restore proper proportion and achieve constant scale in both image width and height directions. Distances in real-world space to the detected road lines can be then calculated. To perform a BEV transformation, a rectangle is created on the flat ground in front of the camera, which will appear to be a trapezoid. The four vertices of the captured rectangle are then projected onto a rectangle in a destination image using the OpenCV perspective transform method [4]. A conversion scale from pixels to meters can be applied in both x and y directions to obtain real world distances.

After BEV transform, the Probabilistic Hough Transform is applied using the OpenCV library to detect straight road lines, including stop lines and parking spot lines [5]. The straight roadlines are then further classified into stop lines and parking lines using the slope of the straight lines. A polynomial regression algorithm is used to localize and fit curved lane lines in the image. Figure 13 illustrates the road line detection software pipeline.

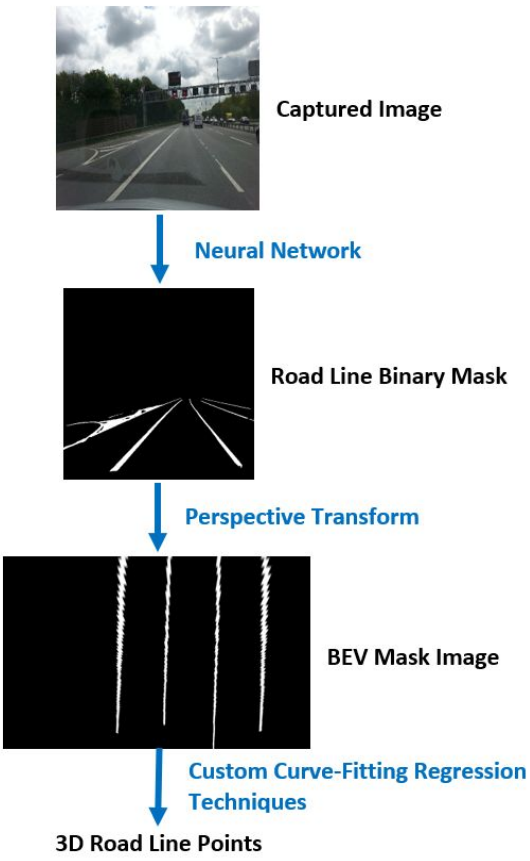


Figure 13: Roadline Detection Software Pipeline

Image Object Classification

Another challenge tackled by the perception modules of the vehicle involve both detecting and classifying objects in an image. Objects of interest for the vehicle to detect include traffic signs, traffic lights, and pedestrians as these objects greatly influence the future path planning behaviour of the vehicle. Object detection neural networks were chosen to localize and detect objects of interest, due to their robustness and invariation to image noise and ability to learn features of objects.

Due to the broad scope of the object classification problem, the challenge was divided into several subproblems, with different neural network architectures defined for different use cases of detecting specific objects. Traffic signs were detected using Single Shot Detector (SSD) with ResNet-50 feature extraction method [6]. This architecture was chosen for its lightweight computation and fast inferencing speed, as well as its accuracy. Traffic lights and pedestrians were detected using the YOLOv3 neural network architecture [7]. The YOLOv3 architecture was chosen for its fast real-time inferencing speed, as the detection requirements for traffic lights and pedestrians was strict and needed fast update rates to track objects quickly. Regardless of the neural network

architecture, each object detection network returns the pixel bounding box coordinates and classification probabilities for each object detected in the image. These bounding box coordinates are sent to the Processing software modules to be used for further cleaning and filtering of detections.

A mix of open-source datasets and data synthesis were used in training the object detection networks. For training a YOLOv3 network in detecting pedestrians, the KITTI and the Tsing-Daimler Cyclist Detection Benchmark datasets were used. Due to the limitation in data on specific Michigan traffic sign and light models required by the Year 2 Challenges, synthetic data was used to train the neural networks. The Cut, Paste and Learn method is the primary method of generating new data, and works by injecting different objects cut-outs to randomized locations in background scenes, generating automatic label annotations in the process [8]. Data augmentation is utilized to add further variation in the data to further represent real-life conditions, including illumination, motion blurring, object scaling, rotation, color variations, and pixel-wise noise. The amount of variation of the augmentations will depend on real-life data and must be estimated to best represent real-life data. Background images are provided by the BDD100K (Berkeley DeepDrive) dataset to provide realistic urban environments as taken by the vehicle's cameras. Figure 14 shows an example of the data synthesis method.

Figure 14: Example of Image Synthesis of Brightness, Motion Blur, and No Data Augmentation (Top to Bottom)

LiDAR Object Detection

To detect obstacles in 3D space for the vehicle to react to, a conditional euclidean clustering approach is used to tackle this problem. A 3D point cloud containing all points of the environment detected for a single frame captured is sent to the clustering method to be segmented, with any points from the ground removed before further analysis. This algorithm uses a flood-fill approach to create clusters by searching locally throughout the LiDAR point cloud to find points that are defined to be nearby neighbours to each other [9]. Constraints for defining the closeness of each point includes the physical euclidean distance, smoothness, and color similarity [9]. After all clusters defining different objects in the point cloud have been found, 3D bounding boxes are drawn around each cluster and outputted as detected objects to be used for further processing.

High Level Data Fusion

High Level Data Fusion (HLDF) is the module responsible for the consolidation of the discrete data sources from the Perception modules into a single data stream, as well as association between different types of objects. HLDF takes as input the obstacles and road lines detected by the various Perception modules. It produces as output a ROS message, to be consumed by the Object Tracking module, that contains all relevant data from Perception, such as roadlines and obstacles. HLDF fuses bounding boxes together from

Perception over each individual frame, and also performs roadline matching and fusion.

For roadline fusion, the incoming roadlines are described as a polynomial with 2 points describing the end points of the detected line. HLDF maintains a current best estimate of the lane lines based on previous seen lane lines. Incoming lanes are then matched with the existing lanes by subtracting the polynomials describing both lanes and integrating the remaining polynomial over the area for which the polynomial is valid. This gives an area which describes the closeness of the 2 road lines. Roadlines are then matched based on which polynomials have the minimum area between them. Subtraction and integration of polynomials is computationally efficient, and the resulting metric provides accurate lane matching for 93% of detected lane lines from data taken from the year 1 competition.

HLDF also performs bounding box association within individual frames. Separate bounding boxes are provided from both image and object-based detection methods and must be associated. The LiDAR provides unlabelled data, while the cameras provide labelled data. As a result, class is not taken into account for data association. The data association is performed solely on the basis of pose and dimension of bounding boxes. Any bounding box pairs whose centroids are further than a set threshold apart, as an efficiency measure, are ignored as possible candidates for association. For any classified bounding boxes, the Intersection over Union (IoU) score is then computed for any unclassified bounding boxes with centroids sufficiently close to it. The unclassified bounding box with the highest IoU is then considered associated with that classified bounding box. The 2 bounding boxes are then averaged together, and the label from the classified bounding box applied.

Object Tracking

After fusion and data association has been performed in HLDF, the current environment is sent to the Object Tracking module. Object Tracking is responsible for estimating velocity and acceleration for any relevant objects. It takes as input the output object messages from High Level Data Fusion, and outputs another environment message, with updated velocity and acceleration fields, to the Path Planning modules.

Object tracking receives input from High Level Data Fusion. It then takes the received bounding boxes, converts their position relative to the car to an absolute position, and attempts to match bounding boxes using a Euclidean distance heuristic. A Euclidean distance heuristic was chosen for ease of computation and use. Euclidean distance may be unsuitable for future years, when scenes may become more cluttered, but when tested in sparser scenes like those expected in competition, it performs accurately. If there is no bounding box within a set distance threshold to match to, then the object is registered as a new object. Once all existing tracked bounding boxes have been matched to new measurements, extra checks are performed to see if an object is still relevant, or if it was a false positive, based on how many frames it has been since the object was last seen, and how many frames it has been seen in total.

As each object is detected and added, an associated Extended Kalman Filter (EKF) is also created. After matching occurs, the EKF takes as input the updated position measurement of any detected object, and uses that to estimate the true position, velocity, and acceleration of the object. An EKF was used due to the known robustness of the algorithm for estimation in this type of task, as well as the relatively low computational load as compared to other, more complicated object tracking methods. The positions, velocities, and accelerations of all tracked objects are then added to an environment message and sent to the Path Planning modules for use.

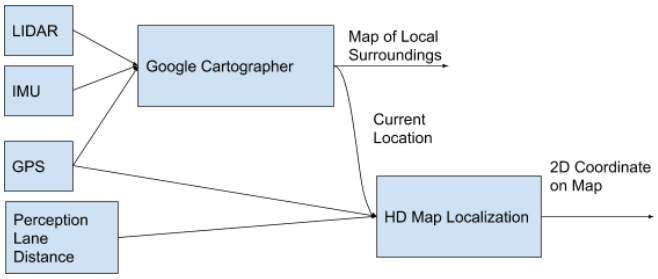
Localization

The localization aspect of the pipeline estimates the car's pose on the HERE HD map and to generate a local map of the car's surroundings. Both outputs are passed directly to the path planning module, where the behavioral planner will use the generated map to set the vehicle's goal and the costmap will be constructed accordingly. Considering how important it was to accurately follow the roads, estimating the car's location in relation to the HERE map is a main focus of the algorithm.

The localization algorithm uses four sources of input: LiDAR, IMU, GPS, and lane detections from the perception module. Localization also uses 2 third party packages, in Google Cartographer and the ROS "robot_localization" package. The LiDAR, IMU, and GPS data are passed to Google Cartographer which generates an occupancy grid of the car's surroundings and a location estimate. This estimate and the lane detection information are then passed to the "robot_localization" package to be integrated into a final output.

LiDAR data was necessary to allow Cartographer to generate the occupancy grid. It also significantly contributes to the accuracy of Cartographer's estimate. To use the full LiDAR scan, we run Cartographer in 3D mode which necessitates the use of the IMU, as Cartographer requires IMU data to initialize the car's orientation. The GPS data and lane detection source are also used to help ensure the location estimate agrees with the HERE map.

Figure 15: The Localization Software Flow



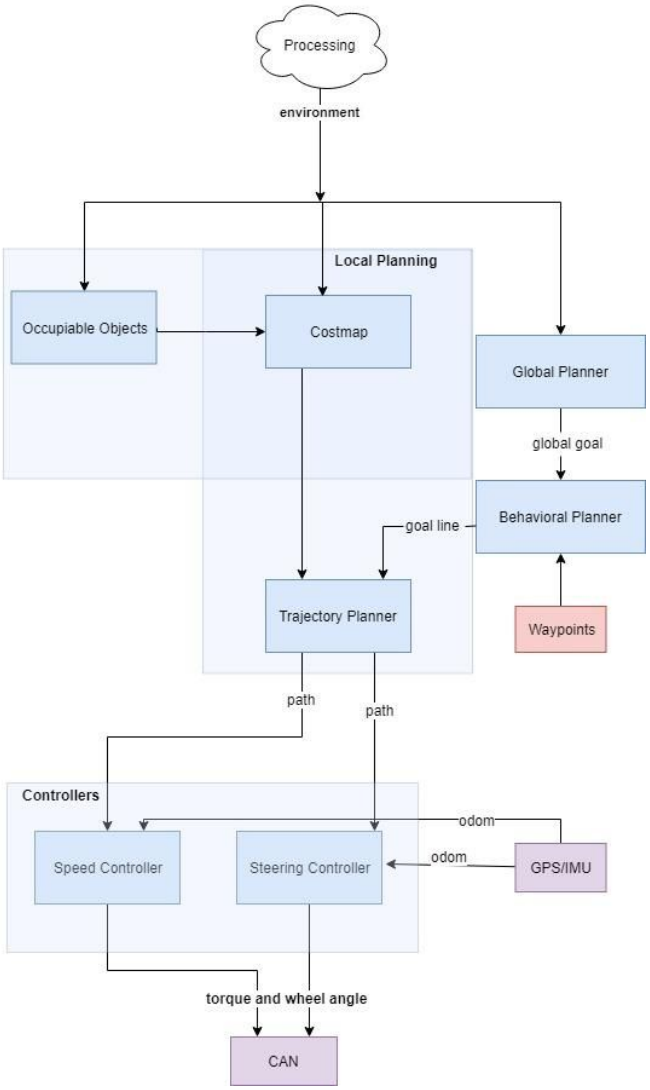
Cartographer and the "robot_localization" package were selected to be able to use all of the available inputs. The "robot_localization" ROS package was selected to augment the Cartographer estimate with the lane detection information. When tested with internal data, Cartographer was also found to have the highest accuracy, and ran in real time, as opposed to other considered options.

After receiving the location of the road centre line from the perception modules, our algorithms adjust our current location estimate to agree with this input. The new location estimate is created by shifting the car's position closer to or further from the corresponding lane line in the HERE map. It also recalculates the car's orientation depending on orientation of the lane line relative to the vehicle.

Mission Planning Algorithm

The motion planning algorithm is addressed by two subcomponents of Path Planning: Global Planning and Local Planning. In Year 3, Global Planning will happen at the HD Map level; the end user would provide some destination and the global planner will route from the vehicle's current position to the destination. For Year 2, it is instead asked to integrate a waypoint system; the global route consisting of waypoints encodes the navigation information required for the vehicle to reach its destination. Once the global plan is completed, the Local Planning module is responsible for planning a path within the local environment, i.e. the region of the world that the vehicle's sensors can perceive.

Figure 16: The Motion Planning Software Pipeline



Global Planning

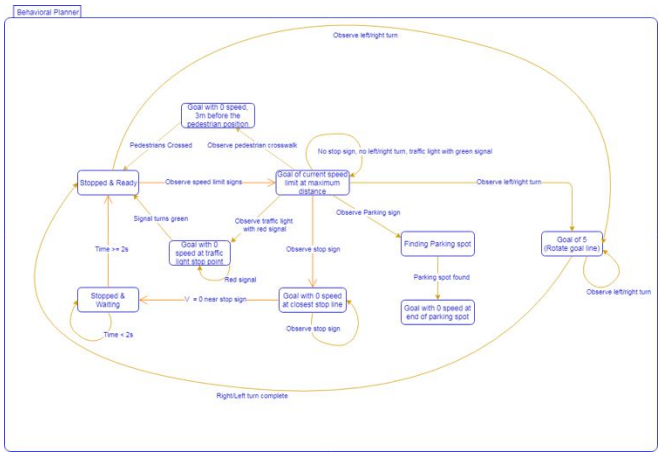
The purpose of the Global Planning module is to precompute relevant data for each Year 2 challenge given the HD map. The result of this module is an approximate list of goal states used by the Local Planning module. For Year 2, the global plan is determined by either traffic signs, such as in the case of the traffic control challenge, or by waypoints. When given waypoints, the global planner pre-computes a list of turning directives using the HD Map, this list is subsequently used to determine the lane the vehicle should follow.

Local Planning

Given the vehicle's perceived environment from Processing and the desired lane from the global planner, the Local Planning module finds the optimal path forward within the environment while abiding to traffic constraints.

Behavioral Planner

Figure 17: Behavior Planner's Finite State Machine



The first step to create a path with the given environment from Processing is to decide on a goal line (which is a line segment, velocity, and acceleration), that lies somewhere in the provided environment. The goal line decision algorithm relies on a finite state machine (FSM) to keep track the current state of the vehicle. The vehicle's state includes attributes such as acceleration and maximum speed. The Path Planning team chose to use a FSM to keep track of vehicle state since it allows state transitions to be dependent on both current state and an external trigger. Specifically, the FSM transitions are triggered by certain characteristics of the consumed env struct (environment structure); for example, an env struct with a vehicle velocity of zero would transition from the slowing state to the stopped state. Additionally, the transitions are also triggered by predefined internal triggers, such as a timer that triggers when the vehicle has been at the stop line for two seconds. Goal lines are generated based on the current FSM state and a newly consumed env struct. For example, FSM(stopped) + env_struct(open_road) would produce a goal line zero meters ahead with a velocity of zero, and FMS(accelerating) + env_struct(open_road) would produce a goal line at points of the env struct with maximum velocity.

Costmap Generation

The next step in the Path Planning task is to plan a trajectory through the augmented environment structure to the goal line. In order to use graph search algorithms, which is explained in detail in trajectory planning section, the environment has to be converted into something that can be reasoned about computationally: in other words, a costmap. The high-level idea of a costmap is that things that the vehicle should not move over (e.g. road borders and obstacles) are assigned a high cost. Therefore, the area of the costmap where a road border lies is assigned a higher cost than the area where a lane line is. Obstacles are assigned the highest possible cost, and open road is assigned a minimal cost. Similarly to Year 1, the Path Planning team chose to use a costmap representation of the vehicle's environment because it works well with the trajectory rollout algorithm [10].

To understand the complex environment, we need a way to convert the environment into a data structure which the computer can understand. A Costmap is a 2D matrix which encodes the semantic "cost" value of features into the environment as numbers or "cost values".

Everything that affects the costmap is referred to as an "object" (roadlines, pedestrians, traffic signs, etc.). WATonomous has decided to draw a distinction between two classes of objects: Static objects and Occupiable objects.

Static objects do not have state, and only influence the region of the costmap that they physically occupy. An example of a Static object is a parked vehicle, or a Road Line.

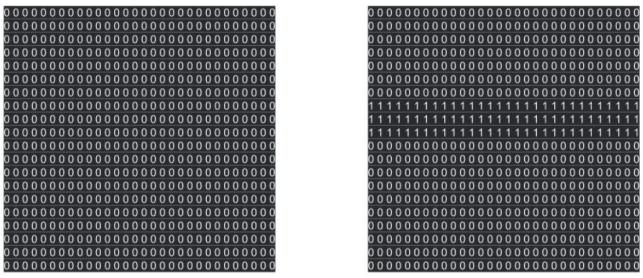
Occupiable objects on the other hand, do have state, and their region of influence depends on that state and is not restricted to the space they occupy. An example of an Occupiable object is a Stop Sign. The state of a stop sign (blocking or passable) depends on how long the vehicle has been waiting at the stop sign. The region of influence of the Stop Sign is not only where it stands in the ground, but also in the road next to it.

Static objects are represented by bounding boxes with a pose (position and rotation). There are multiple steps taken to draw a Static object onto the costmap. First, padding is applied to each object or line based on the car's size and the importance of the obstacle. Objects that the car should not hit generally have higher cost. For example, parked cars may have a wide, high cost padding to avoid collisions. Crossable lane lines might have a thinner, low cost padding. After padding, the object is translated to the desired location and orientation and added to the costmap using matrix addition. To optimize the Year 2 costmap, Eigen was chosen for its efficient matrix manipulation tools. These tools allow the team to efficiently blur the obstacle, rotate the obstacle and add the obstacle to a specific area of the costmap without changing or affecting the other parts.

Drawing an Occupiable object is more complex because it depends on the object's state. Each class of Occupiable object (Traffic Light, Pedestrian, Stop Sign, etc.) contains its own FSM which gets updated every time Path Planning receives a new Environment. (insert at least one FSM diagram example of an Occupiable Object FSM).

The region of influence of an Occupiable object class can be queried at any time, returning a Costmap Layer (which is just a 2D matrix of cost values). The returned layer depends on the state of the object. For example, a Stop Sign in the "blocking" state returns a layer containing a maximum cost strip blocking the road next to the sign, while a Stop Sign in the "passable" state returns an empty layer.

Figure 18: Costmap Example



Costmap layer when Stop sign in "passable" state

Costmap layer when Stop sign in "blocking" state

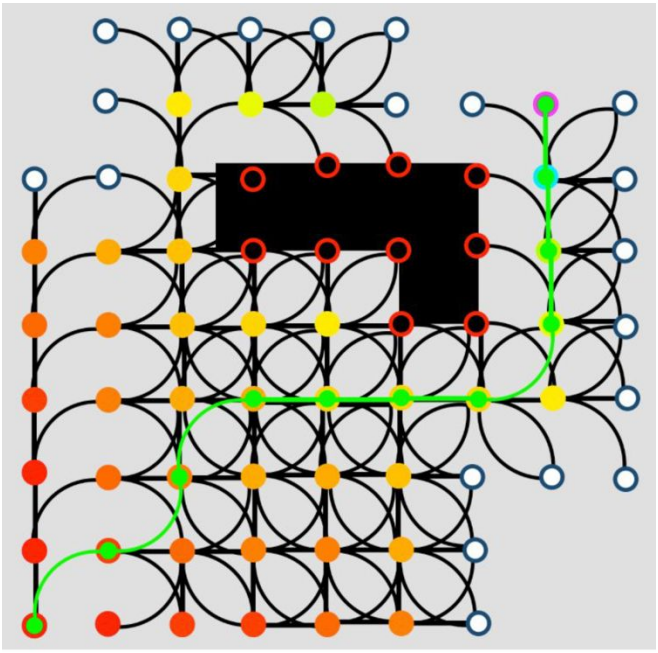
Trajectory Planning

Another requirement of graph search algorithms is a discrete set of next vehicle states (a 2D position). Therefore, now that the current environment is shown in the costmap, a selection of optimal (minimum cost) next vehicle state can be computed given a current vehicle state. The discretization of possible next vehicle states is done by interpolating over the car's possible turning angle, at some constant radius magnitude.

The cost approximation of moving from the current vehicle state to the next state is done by interpolating over intermediate vehicle states of the twist the vehicle would follow to get to that next state, assuming a constant turning angle. In Year 2, we've further tuned the parameters relating the vehicle speed to its maximal twist angle by distance in order to ensure that the vehicle states generated are possible for the vehicle. The number of intermediate interpolated states is equal to the arclength of the twist, divided by the magnitude of the interpolation, thus a smaller magnitude means a more accurate discrete approximation of the vehicle's path along the twist. The cost at each intermediate vehicle state is summed, and that sum is used to approximate the cost of the twist necessary to get from the current state to a next state. The distance from the goal line of each next state is also factored into the state's estimated cost, since Path Planning wants to choose states that move the vehicle close to the goal line. Pseudocode for this state generation is available in Appendix B.

The A* search implementation that actually generates the optimal, minimum-cost path (i.e. a vector of vehicle states) is given the costmap, an initial vehicle state, and a goal line [11]. From there the algorithm generates the possible next vehicle states and puts those states into a priority queue ordered by minimum cost. The algorithm then pops the minimum cost state off the queue and recursively calls itself, finding the optimal next state from the first optimal next state. The recursive A* search continues until the vehicle state popped off the queue is close enough (within 1 meter) to the goal line, and then the path of optimal vehicle states that were taken to reach that last vehicle state is returned. This algorithm is shown visually in Figure 19 below.

Figure 19: Visual Representation of A* Trajectory Rollout



The last step in the Path Planning task is to ensure that the physical vehicle follows the optimal path found as closely as possible. This is done using feedback controllers, one to control the car's wheel angle, and one to control the car's speed. The implementation of the controllers is described in the Motion Control section of this report.

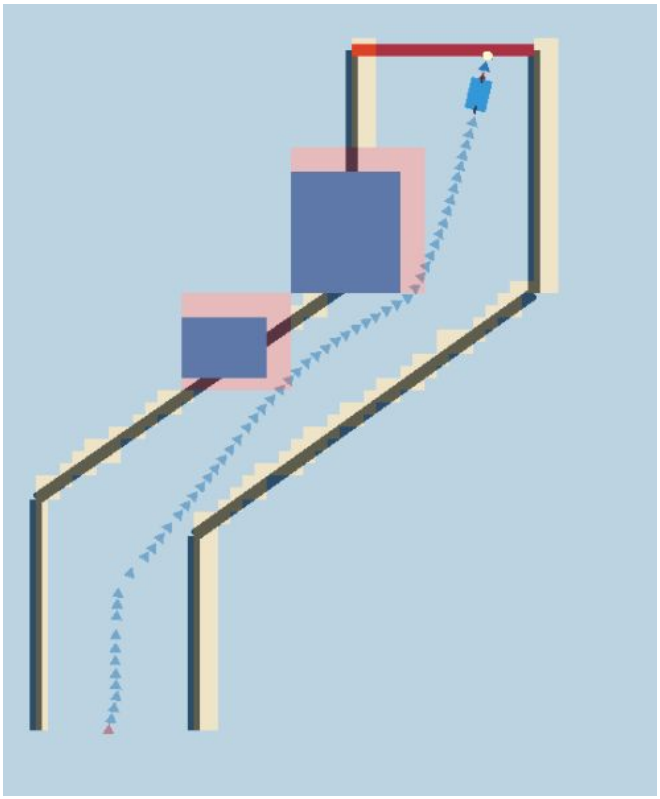
Local Planning Simulator

WATonomous has developed a custom simulator implemented in C++ using GLUT. The simulator is used to test the local planning algorithms in real time using a manually drawn environment. The simulator provides a GUI on which the user can draw an environment using key presses and mouse clicks. For example, the user is able to add a lane line into the environment by clicking to add points to the polyline that represents the lane line.

After the environment has been drawn, the costmap produced by the static and occupiable object generation algorithms is rendered on top of that environment. After the costmap is rendered, the trajectory planning module is called, and the resulting path is

plotted on top of the environment and costmap. This allows developers of those local planning algorithms to iterate on their design, while quickly confirming that they have not regressed the output costmap in a simple and visual way.

Figure 20: Local Planning Simulator



Motion Control Algorithm

Feedback Control

Similar to Year 1, two Feedback Controllers are invoked to maintain the speed and heading of the vehicle by correcting tracking errors. Both Controllers take the current Path and the Current State of the vehicle as input. The Current State is a reading of the vehicle's current location and velocity from the GPS and the IMU, whereas the Path is passed down from the Trajectory Planner module.

A PID Controller is used to correct tracking error for the speed. To compensate for the actuation latency (the delay between the time of command sent and the time of physical activation), the torque output is calculated as a combination of three error terms: Proportional, Derivative, and Integral. The Proportional term aims to correct the error between the current and target speeds, the Derivative the rate of change in the current speed, while the Integral the accumulation of these errors over time (regardless of how small each error is at its point in time).

A Pure Pursuit Controller is used to correct tracking error for the steering angle [12]. Similar to the logic used in Year 1, the Next State is chosen based on the Current State's speed. The higher the speed at which the vehicle is driving, the farther away the Next State can afford to be. There is, however, one significant change introduced into this year's implementation. After the tentative Next State is selected in the previous step, the steering angle required to arrive at this tentative State given the Current State is calculated. The difference between this steering angle and the Current State's angle acts a scaling factor to reselect the Next State closer to the Current State to avoid cutting the corner on sharp turns. The output of this Controller is thus the steering angle in degrees that would take the vehicle on an arc to hit the carefully chosen Next State.

Further post-mortem analysis and reflection upon the results of year 1 brought out multiple critical failures that were later improved upon thoroughly and systematically in year 2. The wheel angle and speed controllers were originally implemented from scratch in C++, which necessitated custom error-prone implementations such as finding the Next State along the Path polyline. This led to an increase in the time needed to fully test the controllers' functionality, and to tune their integration with Trajectory Planning. In addition, the lookahead value used to find the Next State only took into account the speed of the vehicle, and not how it was turning. This resulted in erroneous edge cases, for example some maneuvers, (e.g. taking a sharp turn), require a much smaller lookahead than others, (e.g. driving straight), in order to not cut the corner of the turn. Furthermore, there was no replicable and reliable way to test the controllers in simulation. To gauge the performance of the controllers under certain parameter values, the vehicle had to be physically towed to the test track and repeatedly run in the same turn manually with different parameter values. This led to a huge time loss spent on the test track in order to tune the parameter values that were proven to be suboptimal during the competition.

For this year, to circumvent the aforementioned issues, the design and implementation process of the two Controllers is strategically migrated to MATLAB/Simulink [13]. The program provides ease of use in modifying the gains in *real time* and allows incorporation of *Vehicle Dynamics* blocks for further tuning that takes into account the physical limitations of the vehicle. The *Ziegler-Nichols method* is employed to select the initial values for the tuning parameters. More specifically, *Kd* and *Ki* are initially set to 0 while *Kp* is increased gradually from 1 till the vehicle's speed oscillates consistently around a certain speed. The oscillation range and *Kp* are used to derive the starting values for *Ki* and *Kd* by following the

mathematical formulas shown in [14]. These parameters are subsequently fine-tuned through a manual trial and error process. The same process is repeated for 5 different speeds ranging from 5 mph to the maximum speed limit of 25 mph to obtain a set of PID parameters. The set is then linearized to account for all speeds within said range. The block diagram of the Feedback Control module is available in Appendix C.

CAN Interfacing

In order to properly execute the calculated trajectories and receive feedback control signals, a robust communication interface is required to structure and encode controls commands to the vehicle, as well as receive, decode, and relay feedback information from the vehicle's internal controllers. This interface was created using the well-known CAN communication protocol. The robust communication is achieved through concurrency and parallelism in decoding and encoding messages. Each state of the vehicle is maintained through its own thread or process, to mitigate data loss and corruption, as well as greatly decrease latency between the vehicle and the computing system. Information is channeled through the vehicle's three CAN buses: High Speed (HS), Chassis Expansion (CE), and Low Speed (LS). For receiving feedback information from the vehicle's internal sensors, corresponding addresses are queried and queued to gather the desired message bit packets. These message packets are filtered and manipulated to extract the physical signals from the vehicle (e.g. acceleration, steering angle, brake pressure, etc.). This data is then provided to the feedback controllers for real-time use, as well as to sub-teams such as local mapping, planning, and sensor fusion.

In addition to effectively receiving information from the vehicle, safety-critical control commands need to be sent to and understood by the vehicle for real-time control. To achieve this on a low-level, commands for steering, braking, and torque must be periodically transmitted to keep the vehicle in a controllable state. Once the vehicle is in the correct state, physical signals (e.g. torque requests) are translated into a valid message byte packet (which includes protection signals and active rolling counts) and sent to the receiving control unit on the CAN bus. There are separate feedback signals specific to the communication interface to ensure messages have been sent and received correctly at the desired timestamp.

In tandem with the safety-critical control, important indication signals are controllable with the CAN interfaces. Indications including left/right blinker lights, hazard lights, and high beam lights are controllable with requests from the path planning module. These indicators will be triggered during lane changes, intersection turnings, parking, and under tunnel navigation.

Safety Concept Review

Building on top of the testing safety protocol of Year 1, each clause was revised and deemed to be relevant for Year 2 as well.

One of the additions of Year 2 is the front bumper rack. The main structure of the front bumper mount was designed to carry 122 kg of weight in static loading conditions and 5G forces in dynamic loading conditions. Using a factor of safety of 1.2, shocks and vibrations from bumps in the road are mitigated.

Figure 21: Front Bumper Mount Bolts

While designing the front bumper mount, it was made sure that aluminum bolts will be the failure point in case of any accident, and a metal cable was used, as shown in Figure 21 above to hold the mount from falling off the car, in case of impact.

For electrical safety, thecompute and power systems have been equipped with fuses throughout to protect the vehicles against the risk of excess current draw or faulty equipment. For compute safety, the motherboard holds three temperature sensors and cooling units to protect the vehicles compute unit against the risk of extreme internal temperatures. Proper wires with sufficient gauge and insulation were selected to protect the vehicle against risk of cable splicing and power surges, preventing the event of an electrical fire.

For CAN communication with the vehicle, many safety layers are implemented on both a software and hardware level. Error states and diagnostic codes are tracked by the computing platform at a

high frequency to detect any faults in the internal control units. If the error state is non-critical to the controls of the car, or non-indicative of corrupt data, the error is simply logged for later analysis. In any other case, the vehicle will alert the central state machine to safely shut down the process and return control to the safety driver. All states for the central state machine are displayed on the user interface for driver information. In the event where communication is cut off from one of the three CAN bus lines, the vehicle will return control to the safety driver and the entire software pipeline is temporarily terminated. The safety driver also has full manual override capabilities while in an autonomous state, and all actuation commands are blocked when a manual override is triggered.

In addition, an autonomous control box allows for a physical electrical disconnection between the vehicle's ECUs and the computing platform, while ensuring power to each CAN bus line is adequate. When in the vehicle is in an autonomous state, meaning the connection is established to the vehicle's ECUs, the blue safety light will be turned on to warn bystanders of the vehicle's state. In addition, high insulation and twisted pair DB9 cables are used to prevent any data corruption caused by electromagnetic interference between the computing system and the control MCUs.

Aesthetic Design and Usability

Figure 22: WATonomous Decal Coating

The above image demonstrates the approach WATonomous has taken to the exterior design. The silver vehicle in the top left of the above image is the original Chevrolet Bolt. The team colour, blue, was chosen as the main base for it represents reliability, integrity and unity. It also brings out a sense of safety which has always been WATonomous's priorities and ensures the consumers feel secure and protected in the presence of the vehicle. The team name is placed stylishly in white on the left side of the vehicle along with multiple decals from various sponsors, which further emphasizes stability thus building rapport and instilling trust with the consumers in WATonomous.

When integrating additional parts to the vehicle, the mechanical team was mindful of symmetry. The room rack does take away from the symmetry of the car because it was designed in way that is aesthetically pleasing to the eye by not drawing much attention. The wing shape lidar design has two cameras facing outward on each side as to not break the established pattern of symmetry. The outward direction of the wing shape lidar design adds a rotational effect around the focal point. The front bumper was architected in a way that keeps the idea of balance in the mind's eye as the symmetrical design is easier for the mind to process. It provides a focal point that draws the eye's attention immediately and balances out the visual weight of the car.

On November 22nd, 2018, WATonomous and St. Paul's Greenhouse came together and hosted a Greenhouse event discussing social responsibility with a focus on autonomous vehicles. Social responsibility events help to educate the local community and possible consumers on the benefits of autonomous vehicles. At the Greenhouse event, the WATonomous team was able to answer questions regarding potential dangers of autonomous vehicles by providing research proven solutions. Public engagement at events such as the Greenhouse event helps to eliminate possible mental barriers that may stop potential consumers from purchasing autonomous vehicles.

Equally important, WATonomous recognizes the importance of government support in commercializing autonomous vehicles. Earlier this year on January 22nd, 2019, the Minister of Transportation Ontario, Hon. Jeff Yurek, visited the University of Waterloo, where he formally announced that the province is easing restrictions on its autonomous vehicle pilot program. WATonomous was able to present its technology before the Minister to further demonstrate the capabilities of autonomous vehicles to the government. The team was then featured in an article on *The Record* along with Hon. Jeff Yurek with the title "Ontario driving ahead with expanded autonomous vehicle program" [15].

Additionally, WATonomous engages with potential consumers on social media through weekly postings and updates. Customers are more likely to trust our autonomous vehicle because WATonomous provides transparency to the general public. The WATonomous Facebook page has a five-star rating and grew more than 54% since the Year 1 challenge. The Facebook page now broke the 1,000 follower milestone in the Year 2 phase and totals 1,485 posts as of March 24th, 2019. WATonomous' active marketing strategy acts as the bridge between consumers and autonomous vehicles.

Conclusion

This report presents a detailed description of the technologies used by the WATonomous team in preparation for Year 2 of the SAE AutoDrive Challenge. The hardware was chosen to be performance-effective and cost-effective. Redundancy in selected sensors provided the element of safety. Electrical and thermal analyses were conducted to determine power draw and concluded that our previous cooling system was still appropriate. Building upon the design philosophy of Year 1, design considerations were made in anticipation of future challenges as the competition progresses, such as the adjustable sensor mounts.

The Software design was architected in a modular way to allow multiple teams to tackle the problems in isolation and to easily integrate the components once they were completed. Implementing the software stack as a set of ROS nodes improved runtime performance by leveraging their built-in concurrent nature and had managerial benefits by mapping directly to the team structure.

This year, a safety-first, robust and high-performance approach to the Year 2 challenges was accomplished. The WATonomous team is proud of their work and looks forward to the competition and making the world a safer place.

References

1. *Blackfly 2.3 MP Mono GigE PoE (Sony Pregius IMX249) Technical Specifications*, PointGrey. [online] https://www.ptgrey.com/blackfly-23-mp-color-gige-poe-sony-pregius-imx249.
2. *Special Import Measures Act*, Government of Canada. https://laws-lois.justice.gc.ca/eng/acts/s-15/.
3. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*, 2016. https://arxiv.org/pdf/1511.00561.pdf
4. Geometric Image Transform, OpenCV Open Source Software, Intel Corporation, USA, 2017
5. *Feature Detection*, Feature Detection - OpenCV 2.4.13.7 documentation. https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html.
6. Liu, W. et al. (2016). *SSD: Single Shot MultiBox Detector*. arXiv. https://arxiv.org/pdf/1512.02325.pdf.
7. Redmon, J. and Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. [online] University of Washington. https://pjreddie.com/media/files/papers/YOLOv3.pdf.
8. D. Dwibedi, I. Misra, M. Hebert. *Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection*, The Robotics Institute, Carnegie Mellon University, August 2017. arXiv e-print services, https://arxiv.org/pdf/1708.01642.pdf.
9. *Documentation - Point Cloud Library (PCL)*. Pointclouds.org. (2019). http://pointclouds.org/documentation/tutorials/conditional_euclidean_clustering.php.
10. *base_local_planner*, ROS Wiki, http://wiki.ros.org/base_local_planner.
11. *Introduction to A**, Introduction to A*, http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html.
12. *Automatic Steering Methods for Autonomous Automobile Path Tracking* , Snider, Jarrod M. (2009).
13. *PID Controller Tuning in Simulink*. Mathworks.com, MATLAB & Simulink. https://www.mathworks.com/help/slcontrol/gs/automated-tuning-of-simulink-pid-controller-block.html.
14. *Ziegler–Nichols Tuning Rules for PID*, Microstar Laboratories, http://www.mstarlabs.com/control/znrule.html
15. Brent, Davis. *Ontario Driving Ahead with Expanded Autonomous Vehicles Program*, https://www.therecord.com/news-story/9137924-ontario-driving-ahead-with-expanded-autonomous-vehicle-program/

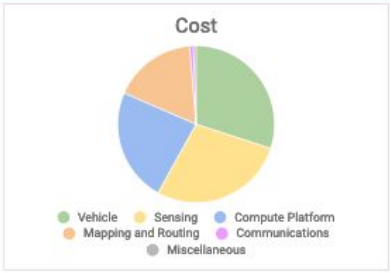
Appendix

Appendix A: Bill of Materials (BOM)

AutoDrive Challenge Cost Summary

University	University of Waterloo
Year	2
Total Vehicle Cost	$132,789.15

Area Totals	Area of Commodity	Cost	
	Vehicle	$	39,825.48
	Sensing	$	37,306.52
	Compute Platform	$	31,085.36
	Mapping and Routing	$	22,780.00
	Communications	$	1,079.08
	Miscellaneous	$	712.71



AutoDrive Challenge Bill of Materials (BOM)

University	University of Waterloo
Year	2
Total Vehicle Cost	$132,789.15

Area of Commodity	Component	Descriptions/Details	Unit Cost	Quantity	Total Cost
Vehicle	Chevrolet Bolt	Provided to Teams	$ 39,825.48	1	$ 39,825.48
Vehicle	**Area Total**				**$ 39,825.48**
Sensing	Velodyne VLP-16 LiDAR	Provided to Teams	$ 3,999.00	4	$ 15,996.00
Sensing	Velodyne VLP-32C LiDAR	Provided to Teams	$ 15,999.00	1	$ 15,999.00
Sensing	Roof mounting	Steel fasteners, T-slotted frames, brackets, screws	$ 832.59	1	$ 832.59
Sensing	Bumper mounting	Carbon steel tubes, screws, camera dome, stainless steel binding barrels	$ 467.61	1	$ 467.61
Sensing	Radar Circuit	Sensors and linear regulators	$ 76.69	1	$ 76.69
Sensing	Camera	Fujinon lens, FLIR systems Inc camera lens (thermal imaging, infrared, night vision)	$ 3,702.67	1	$ 3,702.67
Sensing	Electrical	Sunforce 1000 W Pure Sine Wane Inverter	$ 231.96	1	$ 231.96
Sensing	**Area Total**				**$ 37,306.52**
Compute Platform	Crystal Rugged RS363S15F Casing	Provided to Teams	$ 15,500.00	1	$ 15,500.00
Compute Platform	Arria 10 FPGA	Provided to Teams	$ 4,500.00	1	$ 4,500.00
Compute Platform	Cooper Bussmann Transporation 12055CD2 12/24V to 24V Converter	Provided to Teams	$ 1,500.00	1	$ 1,500.00
Compute Platform	Intel Xeon Processor	Provided to Teams	$ 4,000.00	2	$ 8,000.00
Compute Platform	Heating and Cooling	Computer Liquid Cooling System	$ 399.99	1	$ 399.99
Compute Platform	Electrical	Circuit Breakers, Wire Seal connectors	$ 672.23	1	$ 672.23
Compute Platform	Mounting	Aluminium Sheets, Mounting Screws for Crystal Rugged, Cable Glands, Machining costs	$ 513.14	1	$ 513.14
Compute Platform	**Area Total**				**$ 31,085.36**
Mapping and Routing	Novatel - PwrPak7-E1 GNSS/INS Enclosure, GNSS-502 VEXXIS Antenna, and RF Cable	Provided to Teams	$ 22,780.00	1	$ 22,780.00
Mapping and Routing	**Area Total**				$ 22,780.00
Communications		Express gigabit network card-to synchronize sensors, 4 Port Gigabit Midspaon PoE, Crucial MX500 SSD, Ethernet Cables	$ 1,079.08	1	$ 1,079.08
Communications	**Area Total**				**$ 1,079.08**
Miscellaneous	Electrical	AV Warning Lights, Digi-Key Electric Components, Power strips, Compression Fittings, Installation Tools	$ 712.71	1	$ 712.71
Miscellaneous	**Area Total**				**$ 712.71**

Appendix B: Trajectory Planning Pseudocode

```
costForTwist(twist, initial_position, costmap) {
  final_pos = addArc(twist.radius, twist.delta_theta, initial_position);
  total_cost = costmap.getCost({final_pos_.x, final_state.y});
  for (state in interpolateTwist(twist, curr_state, mag = 1)) {
    total_cost + = costmap.getCost({state.x, state.y});
  }
  return total_score;
}

addArc(rad, theta, &to_state) {
  new_theta = addAngle(theta, to_state.heading);
  return {
       {to_state.x+(radius*sin(new_theta)-radius*sin(to_state.heading)),
       to_state.y+(-radius*cos(new_theta)+radius*cos(to_state.heading))},
       new_theta
       };
}

Path interpolateTwist(twist, initial_state, magnitude) {
  path = Path();
  twist_length = twist.arcLength();
  interpol_angle_mag = twist.delta_theta * magnitude / twist_length;
  angle = interpol_angle_mag;
  while (fabs(angle) < fabs(twist.delta_theta)) {
    path.push_back(addArc(twist.radius, angle, initial_state));
    angle += interpol_angle_mag;
  }
  return path;
}
```

Appendix C: Feedback Controller Algorithm