aUToronto SAE AutoDrive Challenge II Concept Design Report Year 2

Michael Acquaviva, Amy Chen, Brian Cheong, Kelvin Cui, Jacob Deery, Abhishek Kakad, Mustafa Khan, Alex Liu Katrina Meng, Edwin G. Ng, Chang Keun Paik, Sandro Papais, Jingxing Qian, Xiaonong Sun, Matthew Teichman, Shi Jie Wei, William Wen, Sean Wu, Jenny Xu, Stephen Yang, Jiachen Zhou, Jonathan Kelly, Steven L. Waslander
University of Toronto, Apr 30, 2023

1 INTRODUCTION

Following the ground-breaking success of the original four-year AutoDrive Challenge that ended in 2021, SAE and General Motors (GM) have again partnered to renew this international collegiate design competition, AutoDrive Challenge Series II, for another four years. The goal of the competition is to develop and demonstrate a Level 4 autonomous vehicle, capable of navigating urban driving environment, as defined by SAE Standard (J3016 [1]). In Year 1 of the Series II, aUToronto successfully designed a solid perception system with state-of-the-art sensing technologies and multi-modal computer vision techniques [2]. This year's competition focuses on vehicle controls and interface development on the new Chevrolet Bolt EUV the team received from GM, which we retrofit to integrate with last year's perception system.

With the introduction of the new vehicle, this year's Dynamic Challenge consists of four subcategories, Vehicle Integration Challenge (VIC), Perception Challenge, Intersection Challenge, and Highway Challenge. The VIC is conducted at the team's testing facility while the other three challenges will take place at MCity in Ann Arbor, Michigan. Figure 1 unveils our new autonomous vehicle, Artemis, parked on the test track at the University of Toronto Institute for Aerospace Studies (UTIAS).

Each challenge focuses on testing different autonomy components of the system. In the VIC, the team is tasked to acquire basic autonomous lateral and longitudinal controls of the vehicle by sending properly formatted Controller Area Network (CAN) commands to achieve target steering, torque, and deceleration values. The Perception Challenge, similar to Year 1, requires accurate detection of static or dynamic obstacles and various traffic signs and lights while the car is manually driven by a safety driver throughout a pre-determined path in MCity. For the rest two challenges, Artemis will demonstrate her autonomous capabilities in intersection and highway environments. The Intersection Challenge consists of a sequence of traffic intersections on a given path throughout which the vehicle must detect all traffic signs and lights, and autonomously react to the traffic controls legally. The Highway Challenge evaluates the vehicle's ability to autonomously identify traffic and speed limit signs, and to legally navigate around static obstacles or construction zones by lane change maneuvers.

Figure 1: Our autonomous vehicle, Artemis, at the University of Toronto Institute for Aerospace Studies.

In order to successfully achieve these desired functionalities, our team has built Artemis from the ground up with deployment in mind. It features a weatherproof rooftop and bumper sensor suite that provides frontal sensor coverage in multiple modalities and can be mounted on any passenger vehicle with minimum modification. Building on the perception system designed last year, this system, as illustrated in Figure 1, perceives the surrounding using 1 long-range and 3 wide-angle cameras, 4 LiDARs, and 1 automotive long-range radar, in ranges far beyond the competition requirements to ensure safety and redundancy. To make full use of the sensor suite, Artemis' onboard server-grade compute platform processes all data in real time, running the entire autonomy stack of perception, planning and control developed by our own team.

This report first introduces the sensor suite we designed in Year 1 and highlights the modifications made when migrating onto the vehicle. The following two chapters describe the mounting and wiring of sensors, accessories, and the compute platform to the vehicle power management system. Next, we present the redesigned autonomy software architecture and detail

our algorithm selection and implementation. The entire perception function group is discussed in Sensor Fusion, while Route Planning covers both planning and control. Finally, we summarize the changes from Year 1 to Year 2 and conclude with the lessons learned in developing Artemis.

The report is organized as follows:

2 SENSOR SUITE

Design of the sensor suite directly affects an autonomous vehicle's perception capability, which leads to operational safety. Our Year 1 perception cart was designed with future-proofing in mind. In fact, it was designed to be easily adapted to a 2022 Bolt EUV for the purpose of Level 4 autonomy. According to the sensor coverage analysis presented in the Year 1 report, our sensor suite design provides the necessary $\pm 80^\circ$ horizontal, -40° to $+5^\circ$ vertical field-of-view and 60 meters coverage to handle worst-case traffic scenarios, which exceeds the competition's perception requirement of $\pm 60^\circ$ and 40 meters.

2.1 SENSOR SELECTION

In order to select sensor modalities for Artemis, we considered the core capabilities necessary for the perception system based on the scored outputs of the dynamic challenges. Figure 2 shows the Pugh decision matrix outlining the options we considered for the perception sensor suite. It is clear that no single sensor is able to fully satisfy all criteria; camera-only lacks 3D localization information, while LiDAR-only and radar-only lack the appearance information necessary for classifying road signs and traffic lights. From the hybrid architectures, camera + LiDAR and camera + LiDAR + radar both perform well on detection and localization tasks, with the latter preferred due to the direct velocity measurements provided by radar. While direct velocity measurements are not strictly necessary, as the velocity can be estimated by the object tracker, they can significantly improve tracking if properly integrated into object motion modelling.

Criteria	Weight	Lidar Only	Camera Only	Radar Only	Camera + Lidar	Stereo Camera	Camera + Lidar + Radar
Perception Task Performance							
Lane boundary detection	0.5	0.5	1	0	1	1	1
Static object classification	2	0.25	1	0	1	1	1
Dynamic object classification	2	0.7	1	0	1	1	1
3D object localization	2	1	0	0.7	1	0.5	1
Object velocity	1	0	0	1	0	0	1
Integration							
Algorithmic complexity	0.5	1	0.8	0.9	0.6	0.8	0.5
Computational resources required	1	0.7	0.5	1	0.5	0.5	0.4
Other Considerations							
Power consumption	0.5	0.7	0.8	1	0.5	0.6	0.4
Sensor cost	1	1	1	1	0.5	0.5	0.3
Totals		6.45	6.3	5.35	7.55	6.7	8.15
Rank		4	5	6	2	3	1

Figure 2: Sensor suite decision matrix.

As the top option, we selected camera + LiDAR + radar as the sensor architecture for Artemis. All mechanical and electrical systems are designed to support this architecture. While this sensor architecture is largely similar to the architecture used on the Year 1 perception cart, several changes have been made to the specific sensors used. The following subsections outline the sensors selected for each modality.

2.1.1 GNSS/INS Selection

Starting from Year 2 onwards, aUToronto has partnered with NovAtel, a division of Hexagon, for Artemis' GNSS/INS solution. The primary motivation for this change was the availability of NovAtel's TerraStar-C PRO GNSS correction service, which enables centimeter-level positioning with rapid (<3 minute) convergence and instant re-convergence. This accuracy enables Artemis to rely entirely on GNSS/INS for accurate positioning during dynamic events, rather than relying development of sophisticated localization algorithms to correct the vehicle position estimate relative to the map.

Artemis uses the NovAtel PwrPak7D-E2 receiver, a dual-antenna receiver with an integrated IMU. As a fully integrated GNSS/INS system, this unit performs onboard sensor fusion between GPS and IMU information, providing Artemis with a robust, filtered estimate of the vehicle position, orientation, and velocity.

2.1.2 LiDAR Selection

Cepton is the official LiDAR supplier for AutoDrive Challenge II. In current configuration, Artemis continues to use 4x Cepton P60. However, all systems are designed to additionally support the new Cepton X90 LiDAR. The top center and bumper LiDAR mounts are designed to be compatible with both the P60 and X90, facilitating easy reconfiguration of the sensor platform.

2.1.3 Camera Selection

As with the Year 1 perception cart, Artemis uses 4x Atlas ATP071S cameras from Lucid Vision Labs. These cameras were selected for their high framerate, resolution, and dynamic range, all of which ensure consistent image quality over the diverse range of lighting conditions that a vehicle may encounter. Moreover, the IP67 rating enables testing and operation in wet conditions, commonly encountered in Canada during the winter and spring. A detailed outline of the camera selection criteria can be found in the Year 1

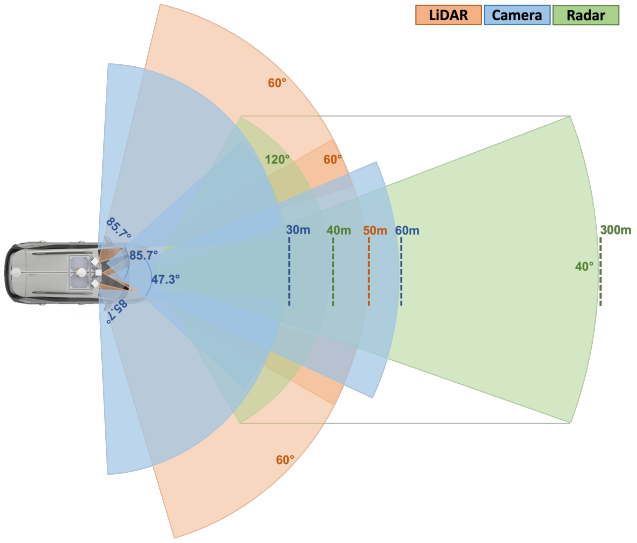


Figure 3: Artemis Sensors Coverage.

Concept Design Report.

2.1.4 Radar Selection In place of the Continental ARS 430 radar used in Year 1, Artemis uses the Continental ARS 548. The primary advantage of the 548 over the 430 is the significantly improved range of 300 m vs 60 m. The ARS 548 maintains a high FOV of $\pm 60^{\circ}$, with distance resolution of 0.22 m and accuracy of $\pm$ 0.15 m. The greatly increased range significantly enhances the ability of the system to detect vehicles and pedestrians at distances past the range of the LiDAR, which is crucial for safe operation at higher speeds. While detection past 40 m is not required in Year 2 of the competition, integrating long-range radar into Artemis at this stage ensures the car will be well prepared for future challenges.

While the electrical and mechanical systems were fully designed to support this sensor modality, procurement issues delayed the arrival of the ARS 548 significantly, and therefore the team decided not to implement it into the software stack for Year 2. Therefore, this report details the mechanical and electrical integration of the radar, but does not include it in software architecture or sensor fusion discussion.

2.2 SENSOR PLACEMENT

The Year 2 sensor placement strategy is largely similar to Year 1. As detection of objects beside or behind the vehicle is not required in Year 2, all sensors are mounted oriented forwards to ensure critical areas in front of the vehicle are covered by multiple redundant sensors.

Two notable changes were made from Year 1 to Year 2. First, the mounting angle of the top LiDARs, previously -10° relative to horizontal, was changed to -5°. This was done to prevent LiDAR points from falling on the hood of the vehicle, which extends further compared to the body of the perception cart. Moreover, it increases the density of the point cloud at ranges slightly farther from the vehicle. The bumper-mounted LiDAR ensures that there are no blind spots immediately in front of the bumper, where the top LiDARs would be occluded.

The second change was to switch from three bumper-mounted radars to just one. This change was made due to mechanical constraints detailed in Section 3.2. While removing the two side radar decreases the overall field of view, the 120° FOV is still sufficient for Year 2 competition requirements. The team is investigating alternative radar mounting points for future years. The top-down view of the sensors coverage diagram is presented in Figure 3.

2.3 SENSOR INTEGRATION

As the cameras, LiDARs, and GNSS/INS all communicate using TCP or UDP over Ethernet, it was originally intended to connect all 9 sensors to a common network switch, which would then connect to a single Ethernet port on the compute platform. However, issues encountered during development necessitated allocating a unique network interface to each LiDAR. The final configuration uses a Netgear M4300-16X switch for the 4 cameras and GNSS/INS. The switch connects to a 10 Gb network interface on the compute platform, selected for the high data throughput necessary for the cameras. Each LiDAR connects to a single port of a quad-port network interface card.

2.3.1 Time Synchronization The time synchronization strategy was greatly simplified in Year 2 compared to the Year 1 approach. The Lucid cameras, Cepton LiDARs, and Continental radar all support Precision Time Protocol (PTP) for synchronizing internal clocks. These sensors all connect to the compute platform, which provides a PTP grandmaster clock on each network interface. The compute PTP clocks are synchronized to the system clock, which is set by the Novatel GNSS receiver PPS timestamps using *gpsd*. This network ensures that GPS time is the source of truth and all sensor timestamps are computed relative to the same timebase.

3 HARDWARE DESIGN AND MOUNTING

While preparing the mechanical platform for AutoDrive Challenge II, several factors were prioritized such as ease of assembly, precise sensor placement, and weather protection. We have taken inspiration from the design language of modern Level 4 autonomous vehicles, and while some features are difficult to achieve with our limited resources, we have aimed to make our design as close to a deliverable product for the 2022 Bolt EUV as possible. In Year 1, mechanical components were designed with reusability in mind. Namely, the use of standard t-slot framing, removable brackets, and modular design structure allowed much of the previous year's mechanical structure to be transferred from the

perception cart, to the new vehicle. Our decision to reuse much of the structure has proven to be both cost and time-efficient since it minimizes the need for new materials, and allows us to maintain consistency for other sub-teams. As well, with the new vehicle platform, there have been several opportunities for new challenges and areas for mechanical innovation.

3.1 TOP SENSOR RACK MOUNTING

The main purpose of the top sensor platform is to offer a stable and secure structure for the proper placement and orientation of all primary sensors. These sensors include four cameras, three LiDARs, two GPS antennas, one GNSS/INS receiver, and a few peripheral devices. The pentagonal top sensor rack was transferred from the perception cart used in last year's competition over to the roof of the 2022 Chevrolet Bolt EUV, as illustrated in Figure 4.

This top sensor platform was made from 45mm x 45mm lightweight aluminum extrusions assembled with various metal brackets, connector plates and standard hardware. Aluminum extrusions offer several benefits, such as lightweight yet strong properties, easily machinable and greater flexibility in design, as extrusions can be easily cut and assembled in various configurations. By using aluminum extrusions, we have been able to create a modular design that can be easily assembled and disassembled, making it easier to replace or upgrade individual components. This modularity also allows for greater flexibility in testing different configurations and sensor placements, which is critical for developing an autonomous vehicle which may undergo several different sensor variations. Furthermore, aluminum extrusion construction is cost and time effective which allows the team to fabricate and design with less manufacturing time.

The top sensor platform consists of two levels, the main and upper level. The primary level of the platform is designed to protect sensitive camera sensors and the GPS/IMU device from damage caused by rain or snow.

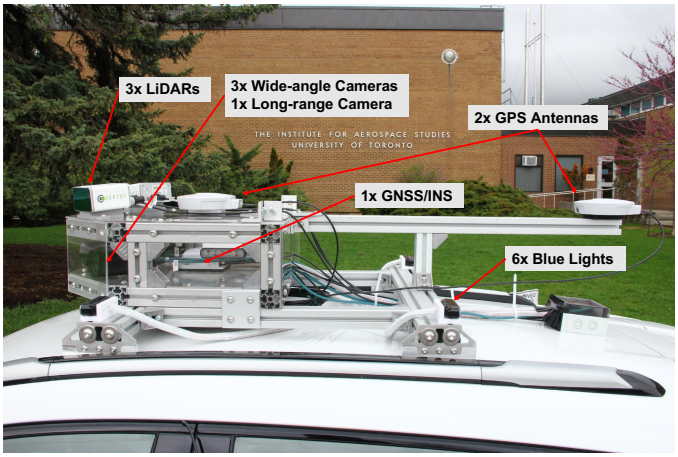


Figure 4: Top Sensor Rack Overview

The sensors are mounted on a stable 1/8" thick sheet-metal surface, and water-jet clear PVC panels were added to the walls and top to shield the sensors from extreme weather conditions. The LiDARs and GPS antennas are placed on the open upper level. This allows the LiDAR sensors to have an unobstructed field of view. As well, the two GPS antennas are placed 120 cm apart on a single extrusion beam that spans across the back side of the sensor rack.

In order to properly transfer the top sensor platform from the pushcart to the vehicle, several adaptations had to be made. First of all, a roof rack was designed using similar 45mm x 45mm aluminum extrusions which would allow the top sensor platform to be easily attached. Using CAD software, custom contoured feet were designed to fit the rails of the vehicles roof and match the mounting points for the vehicles roof rack. Using the vehicles intended roof rack mounting points ensured a secure and stable fit. To lower costs, the custom roof rack feet were designed to be manufactured as a two-part setup which allowed us to reduce the number of CNC milling operations in favour of waterjet components. This approach significantly reduced the cost of production while still ensuring high-quality parts that meet our design specifications.

3.2 BUMPER SENSOR RACK MOUNTING

The purpose of the front bumper sensor is to mount additional LiDAR and radar sensors to the front of the vehicle for more robust sensing. The front bumper sensor rack design was greatly modified from the proposed design from Year 1 which described a one LiDAR and three radar configuration. This was mainly due to restrictions stating no additional hardware must protrude from the front of the vehicle. Instead, the front bumper sensor rack for Year 2 will focus on a one LiDAR and one radar configuration of wider field-of-view. In order to ensure no sensors protrude from the front of the vehicle, the front bumper was removed and several designs were brainstormed which employed a mounting solution on the vehicles impact bar. These designs include 1) a plate design which consists of a 3/4" aluminum plate with machined holes for mounting a LiDAR and a radar

Criteria	Weight	Aluminum Plate Design	Aluminum Extrusion (curved)	Aluminum Extrusion (straight)
Design				
Simplicity (# parts)	2	4	2	3
Versatility (adapt to design changes)	1	2	4	3
Manufacturing				
Ease of manufacturing	2	4	2	3
Tolerances	3	4	2	3
Performance				
Reliability	2	4	2	3
Maintenance	2	4	2	3
Totals		46	26	36
Rank		1	3	2

Figure 5: Pugh Chart assessing three bumper sensor rack alternatives

Figure 6: Bumper sensor rack plate design accomodating one LiDAR and one radar

beneath it, 2) a curved aluminum extrusion design which would use 3 aluminum extrusions placed at an angle on which to mount the LiDAR and radar sensors, and 3) an aluminum extrusion design which employs a single straight aluminum extrusions across the front of the impact bar on which LiDAR and radar sensors would be mounted. A Pugh decision matrix for these three alternatives can be seen in Figure 5.

The plate design for the bumper sensor rack was chosen for its low profile design and simple construction. The low profile design ensured the sensors would not protrude beyond the front of the vehicle. As well, the single piece construction reduces possible assembly errors. The team also acquired a new radar sensor with a 120 degree field-of-view. As this meets competition requirements, the plate sensor mount was designed with only one radar mount instead of the three-radar set up proposed from Year 1. This was decided because documentation for this sensor recommended that each radar be placed minimum one meter apart, which would be impossible to fit 3 of them on the impact bar alone. In order to increase radar range in future years, additional radars can be mounted on the side panels of the vehicle.

As seen in Figure 6, the final bumper sensor rack mount consists of 3/4" aluminum which was water-jet, then CNC milled. Profiles were milled on the reverse side of the plate to allow radar mounting, as well as to accommodate the existing vehicle components such as the front support bar and car horn. The plate includes mounting holes for both the Cepton's P60 and X90 LiDAR sensors, which will allow for future compatibility if the sensors are upgraded. The radar is mounted below the LiDAR. It is also noted that the LiDAR sensors are off-centered from the center line of the vehicle. This is due to interference of the sensor cables with the vehicle's vertical front support bar and is accounted for during sensor calibration.

3.3 EXTERNAL/INTERNAL BLUELIGHT MOUNTING

The external blue lights as seen in Figure 4, per competition requirements, serve as a warning to others around the vehicle when it is being operated under the autonomous mode and they must be visible from at least 200 feet away in daylight conditions. Therefore, SAE Class 1 certified blue lights are mounted to the roof with two facing the front, two facing the back and one facing each side for a total of six external blue lights. This ensures 360-degree visibility of the blue lights regardless of which side a person is viewing the vehicle. The blue lights are mounted on the aluminum extrusions of the roof rack, with custom made mounts. Additionally, a blue LED strip was mounted to the interior of the vehicle, using adhesive, so passengers are also able to see the status of the blue lights that are synchronized with the external ones.

3.4 SERVER AND ELECTRONICS RACK

The server rack houses the Intel server, and electronics such as the REDARC Battery Management System (BMS), REDARC Total Vehicle Management System (TVMS), Cotek Inverter, Uninterrupted Power Supply (UPS), and 12V Battery. The server rack is designed with modularity and ease of access in mind. The server rack is made from 30mm x 30mm aluminum extrusions which allows the team to easily add additional mounts if necessary.

The electronics rack, Figure 7, consists of 3 levels. The first layer consists of a rectangular aluminum extrusion layer covered with a PVC panel. Cutouts in the PVC panel were water-jet for mounting and cable management. This layer holds the REDARC Battery Management System, and Contek Pure Sine Wave Inverter. The second layer holds the Intel Server with additional side brackets to secure the server and prevent sliding. Additional side brackets were also installed to hold the UPS. Finally, the third layer is intended for lighter electronics such as switches and fuses which require easy access. All three levels are supported by two U-shaped vertical supports which allows each of the levels to shift up or down depending on the accessibility requirements of the electronic components. Also mounted vertically is the REDARC TVMS to allow easy access to fuses.

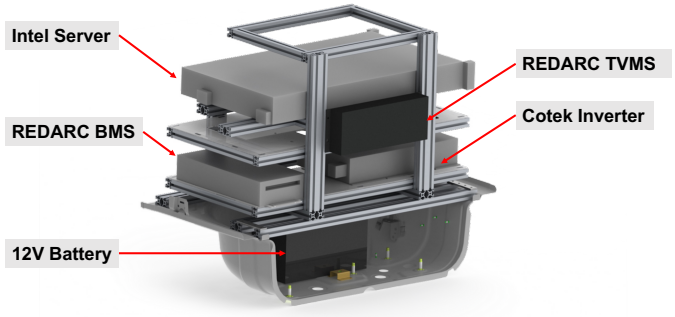


Figure 7: Three level server and electronics rack

The base of the server and electronics rack is secured by two long 60mm x 60mm extrusions. These extrusions are the exact length of the trunk and prevent the server rack from sliding. Additionally, the extra height allows a battery to be stored in the trunk area below the first level.

3.5 SENSOR CALIBRATION

Since each LiDAR and camera are mounted at a different pose, sensor calibration is required to bring the data collected from the multiple sensors into a unified frame of reference. In this system, all other sensors are calibrated with respect to the pose of the center top LiDAR. This involves first aligning the left, right, and center bottom LiDAR's point clouds with the center top frame, then computing the pose of each camera with respect to the center top frame by observing each camera-LiDAR pair. For example, the left camera's field of view overelaps with the left LiDAR, therefore its extrinsics are determined using the left LiDAR's point clouds.

3.5.1 LiDAR Extrinsics Calibration

The purpose of LiDAR to LiDAR extrinsics calibration is to determine the relative poses (positions and orientations) of multiple LiDAR sensors with respect to each other, in order to accurately transform the point cloud data from each sensor into a single, unified frame of reference.

Calibrating the left, right, and bottom LiDARs with respect to the center top LiDAR allows for accurate point cloud registration, enabling the creation of a comprehensive and accurate 3D representation of the environment. We choose to align with the center top LiDAR because it is the only sensor that has overlap with all other LiDARs. In order to perform this calibration, the Iterative Closest Point (ICP) algorithm was used [3]. ICP is an iterative method that minimizes the distance between corresponding points in the overlapping regions of two point clouds by iteratively transforming one point cloud to align with the other. Once these relative poses were found, they were injected into the driver layer of the LiDARs such that the point clouds used for the downstream perception exist in a unified frame of reference, as shown in Figure 8, where each color represents a LiDAR source such that a perfect alignment can be visually confirmed by looking at the overlap regions between each two LiDARs.

3.5.2 Camera Intrinsics Calibration

Camera intrinsics calibration is the process of determining the internal parameters of a camera, such as the focal length, principal point, and lens distortion coefficients. This calibration is necessary in order to accurately map the 3D world onto a 2D image, a required process for the object detection, tracking, and camera-LiDAR sensor fusion.

To perform this calibration, a checkerboard pattern with known dimensions is used as a target. By capturing images of the checkerboard from different viewpoints, the

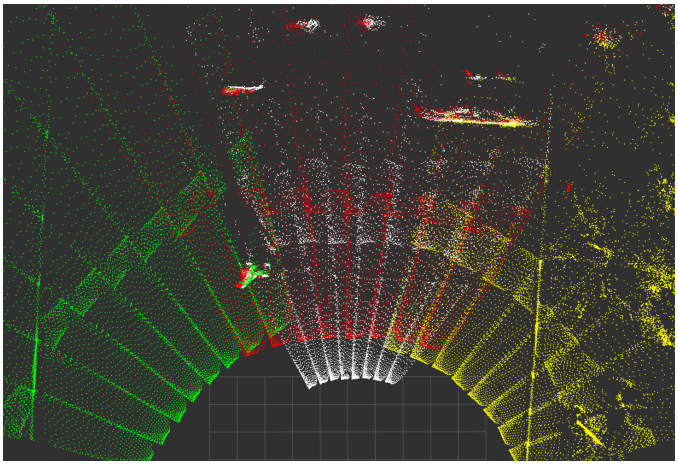


Figure 8: Aligned LiDAR Point Clouds After Calibration. Red, white, green, and yellow points in the point cloud correspond to data from the center top, center bottom, left, and right LiDARs respectively.

algorithm can determine the camera parameters that best transform the 3D world coordinates of the checkerboard corners to their corresponding 2D image coordinates. The intrinsic camera parameters are then estimated using a least-squares optimization approach through MATLAB's Single Camera Calibrator. Once the camera intrinsics are calibrated, the image can be undistorted and rectified, allowing for accurate measurements and mapping of the 3D world onto the 2D image.

The camera intrinsics matrix $\mathbf{K}$ is given by

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where f_x and f_y are the focal lengths in pixels, s is the skew coefficient, and c_x and c_y are the principal point coordinates.

This algorithm also estimates the distortion parameters are the parameters that describe the deviation of the lens from an ideal pinhole camera model. There are two types of distortion: radial and tangential. Radial distortion causes straight lines to appear curved, while tangential distortion causes the image to appear tilted.

3.5.3 Camera Extrinsics Calibration

Camera extrinsics calibration is the process of determining the 3D transformation between a camera and a world coordinate system, which is necessary for projecting world points captured by the LiDAR onto each camera's image frame.

Checkerboard-based calibration was also used for this calibration. In this approach, a checkerboard pattern is placed in the field of view of the camera and corresponding LiDAR, and images and point clouds of

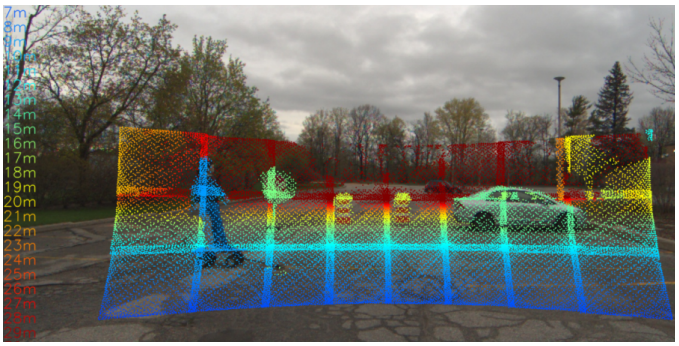


Figure 9: Static scene with 3D to 2D projection using calibrated camera extrinsics, showing camera image with corresponding LiDAR point cloud projected onto it.

the checkerboard are captured from different angles by rotating the checkerboard in the three rotational axes. Using the detected corner coordinates in the image and their corresponding LiDAR coordinates, the camera extrinsics can be estimated by finding the transformation using the Perspective-n-Point (PnP) algorithm. The process was implemented using MATLAB's LiDAR Camera Calibrator. Once the extrinsics were estimated, they could be used along with the intrinsics to project 3D LiDAR points onto the image, as shown in Figure 9.

The camera extrinsics matrix is given by

$$\mathbf{E} = [\mathbf{R}|\mathbf{t}] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

where $\mathbf{R}$ is a 3×3 rotation matrix that describes the orientation of the camera in the world coordinate system, $\mathbf{t}$ is a 3×1 translation vector that describes the position of the camera in the world coordinate system, and $[\mathbf{R}|\mathbf{t}]$ is the camera extrinsics matrix. The individual elements r_{ij} and t_i represent the rotation and translation components of the extrinsics matrix, respectively.

To project points from 3D to 2D, the following perspective projection equation is used $s \begin{bmatrix} u & v & 1 \end{bmatrix}^T = \mathbf{K} \cdot \mathbf{E} \cdot \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ where (X, Y, Z) are the coordinates of a 3D point in the world coordinate system, (u, v) are the corresponding 2D coordinates of the point in the image plane of the camera, s is a scaling factor, and matrices $\mathbf{K}$ and $\mathbf{E}$ are the intrinsics and extrinsics we obtained earlier.

Figure 9 shows that after careful camera-LiDAR calibration, the LiDAR point cloud when projected onto the corresponding camera image plane will match perfectly with the 2D image pixels. The alignment can be best observed on the foreground objects in the scene, namely the pedestrian dummy, stop sign, barrels and the vehicle parked on the right.

4 VEHICLE ELECTRICAL DESIGN

The design philosophy of the electrical components is to ensure the highest standards of electrical safety and reliability while keeping power usage to a minimum. The team has chosen the REDARC System as the central electrical management system due to its ability to efficiently distribute power to each component while maintaining control over the system. The team has also taken measures to reduce power usage, which not only reduces the carbon footprint but also prolonging battery life.

The REDARC system is made up of two components, the battery management system (BMS) and Redivision total vehicle management system (TVMS). The BMS manages not only the charge of our secondary battery which is located in the trunk of the vehicle, but also the power draw between the provided 12V vehicle battery located in the front and our secondary battery. The Redivsion TVMS is a digital relay fuse box that manages the power distribution to all our 12 VDC sensors. The TVMS powers the GPS, LiDARs, radar and blue lights, we are able to toggle power to our sensors using REDARC display located beside the driver seat. To power our AC devices, such as an Uninterruptible Power Supply (UPS) and a network switch, we use the Cotek 200 0W inverter. The UPS provides power stability while switch between power draw loads. This inverter converts our 12 VDC battery voltage into 120 VAC at 60 Hz. Figure 10 shows the electrical system schematic of the vehicle and Figure 11 is the view of our electrical system from the back trunk.

Table 1: Measured Power Consumption of Electrical System

Load Device	No. of Devices	Voltage Requirements (V)	Power Consumption (W)
Computer	*1*	*110V*	*500*
GPS	*1*	*12V Automotive*	*5*
Radar	*1*	*12V Automotive*	*12*
Camera	*4*	*12V Regulated*	*22*
LiDAR	*4*	*12V Regulated*	*36*
Monitor	*1*	*110V*	*15*
Network	*1*	*110V*	*41*
Blue lights	-	*12V*	*32*
Total			*663*

Power consumption for this year was determined through measurements using REDARC system and a summary of average power consumption can be found in Table 1. To account for the vehicle's power consumption of 663 W, our vehicle's internal battery can supply a maximum of 1200 W, while our secondary battery can supply a total of 2,400 W of power. Thus our vehicle has excess of 2,937 W to spare, which opens the door for future sensor expansion and possible cooling.

4.1 CAN BUS SYSTEM

Controller area network (CAN) bus is a communication

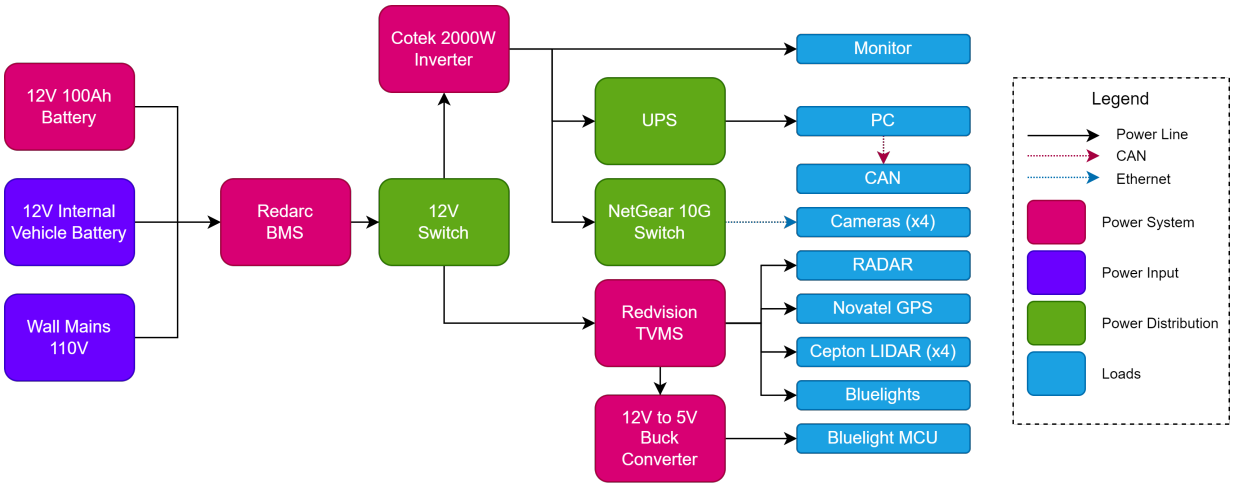


Figure 10: Power Distribution System Diagram

protocol used in vehicles and machines to enable microcontrollers and devices to communicate with each other. The Bolt EUV has an ODBII breakout harness located to front passenger side which interfaces with vehicles CAN Buses. In order for the CAN Bus signals to reach the server, a custom harness was constructed to connect the front passenger side to the rear of the vehicle. The CAN Bus wires were made using 20 AWG single core copper, the pairs were twisted using a electric drill, and custom ODBII breakout connectors were made to the J1962 and CGM pinouts. The CAN Bus harness is made up of four separate CAN Bus: high-speed (HS), chassis expansion (CE), low speed (LS), and scoring (SC). The CAN Bus channels HS, CE, and LS are used to control the vehicle, while SC outputs competition scoring data and autonomous vehicle state.

4.2 BLUELIGHT CONTROLLER

The bluelight controller controls internal and external blue lights of the vehicle. These safety lights indicate the state of the autonomous vehicle: flashing indicates non active CAN Bus communication between vehicle and the server, light off indicates the vehicle is in standby mode with an active CAN Bus interface, and solid lights indicate

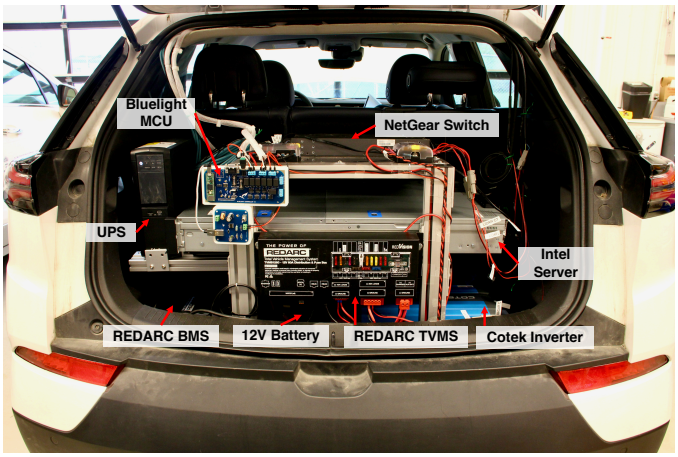


Figure 11: The vehicle power management system wiring.

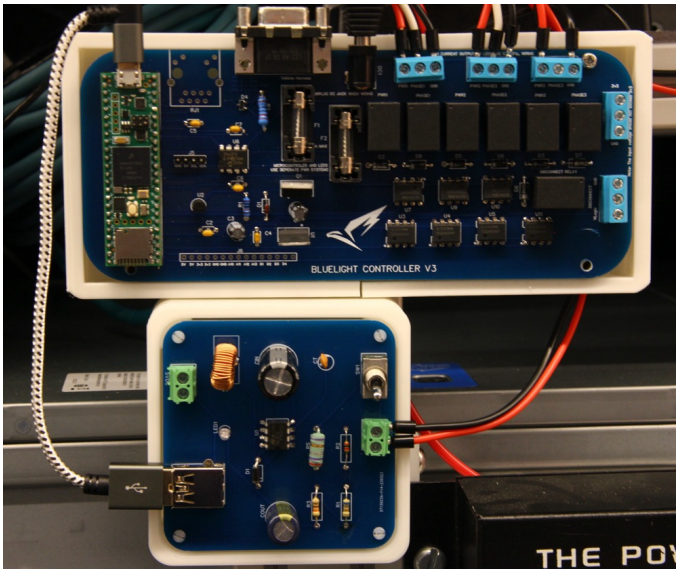


Figure 12: *Top*: The bluelight controller board; *Bottom*: 12V->5V Buck converter board.

that the car is currently in active autonomous mode. The bluelights are controlled by the scoring CAN Bus interface. A Teensy 4.1 microcontroller unit (MCU) is used to interface with the Scoring CAN Bus to detect transitions in autonomous vehicle state.

Figure 12 depicts the boards responsible for powering and controlling the bluelights. The printed circuit board (PCB) above is the bluelight controller with Teensy MCU and the PCB below is a buck converter, which converts automotive 12 V into smooth 5 V for the Teensy. A switch was added to the buck-convert to toggle the power supply to the bluelights. The bluelight controller board makes use of relays to switch the bluelights on and off. Our vehicle uses Whelen ION Series bluelight which are Class 1 SAE vehicle lights. These lights require specific signal-timing which are controlled by the relays.

4.3 ELECTRICAL SAFETY

Several electrical safety issues need to be addressed, to

ensure our team's safety and the safety of participants in the competition. These can be broken down into four different categories:

1) Battery Safety: Battery systems must be designed and built to be safe and reliable to prevent issues such as overheating, short circuits and other fault-causing fires and other hazards.

2) Electromagnetic Interference (EMI): The vehicle's electronic motors and power inverters generate EMI which produces unwanted electrical noise or interference on sensors, computing platforms and communication channels (e.g. CAN Bus).

3) Electrical System Reliability: Power disruptions must be prevented during normal vehicle operation. Additionally they are designed and built to handle various rough roads and temperature conditions.

4) Human Operation Error: Safety measures must be in place to prevent unintentional operator misuse of the vehicle.

Our vehicle battery system was designed and build to balance the power draw between the provided 12 V battery and the rear 12 V trunk auxiliary battery. As a result, we are able to prevent overheating during vehicle operation and charging. The battery management system by REDARC has temperature and voltage sensors which interface with the battery providing important data on battery capacity and health. By configuring the REDARC settings we are able to control the rate of discharge between batteries and provide an optimal power draw between batteries during vehicle operation. The battery sensor enables our team to monitor the capacity, ensuring we do not fully discharge and damage the battery. In addition, we have configured alarms to indicate low battery levels. Lastly, we have inline current fuses specified based on a maximum operating amperage limit, which are placed near the batteries to prevent short circuits.

To minimize the effect of EMF interface on our power system, we incorporated a number of design strategies. While designing and building our high current power distribution system in the bottom truck, we ensured that all power lines were less than 1 meter in total length and we kept the positive 12 V lines and their corresponding ground cables side by side. Additionally, we placed the DC-to-AC inverter away from the key power systems and the server to prevent interference. To eliminate crosstalk interference in the CAN Bus system we twisted our CAN Bus wire pairs in a counter-clockwise direction with exception of cable connectors areas.

To ensure our vehicle's reliability, we spec our copper wires to ensure we are using the proper gauge. We mounted all electrical components to the t-extrusions of the server rack. The auxiliary battery in the truck is tied down using a buckle tie-down. Vibration rubber dampers were added to the server rack and the battery to prevent damage from vehicle vibrations. Lastly, all power connections for the sensors use vehicle-rated power connectors, such as the Deutsch DT series connectors.

To prevent operator error or misuse we configured REDARC setting to provide both audio and visual alerts to the driver, to notify of any power failures or low battery status. We can also use the REDARC system to ensure the power system has been safely disabled. Moreover, during any vehicle operation, we actively monitor the current draws of various vehicle sensors to determine possible power failures.

5 SOFTWARE ARCHITECTURE

Figure 13 depicts the overall software architecture for Artemis. The system was designed with a focus on modularity; all major components are implemented as separate processes using the Robot Operating System 2 (ROS2) framework and communicate using message passing. This isolates the internal implementation details of each process from the public interface and allows for easier integration, as each process only needs to be aware of the message types that it must send and receive.

5.1 SUBSYSTEM OVERVIEW

In this section, we provide a high-level overview of each software subsystem. Sections 6, and 7 outline the perception systems and planning and control systems in more detail, respectively.

5.1.1 Driver Layer The driver layer is responsible for direct communication with the sensor suite. The primary responsibility of each driver is to convert data retrieved from each sensor into the message formats expected by downstream modules.

5.1.2 Detection Layer The detection layer comprises 5 distinct detectors. The lane detector consumes camera images and detects both lane lines and stoplines. Object detection is delegated to the remaining 4 detectors. Three of these operate on camera images and perform 2D object detection (2DOD) in image coordinates. Traffic light detection, static object detection (signs, barrels, barricades), and dynamic object detection (car, pedestrian, deer) are performed by separate detectors. The remaining detector operates on LiDAR point clouds and performs 3D object detection (3DOD) in the sensor frame.

We chose to adopt a *late fusion* strategy for our perception stack, where detectors operate on one sensor modality only and detections are fused at a later stage. While early fusion systems that perform detection on multiple modalities simultaneously can sometimes

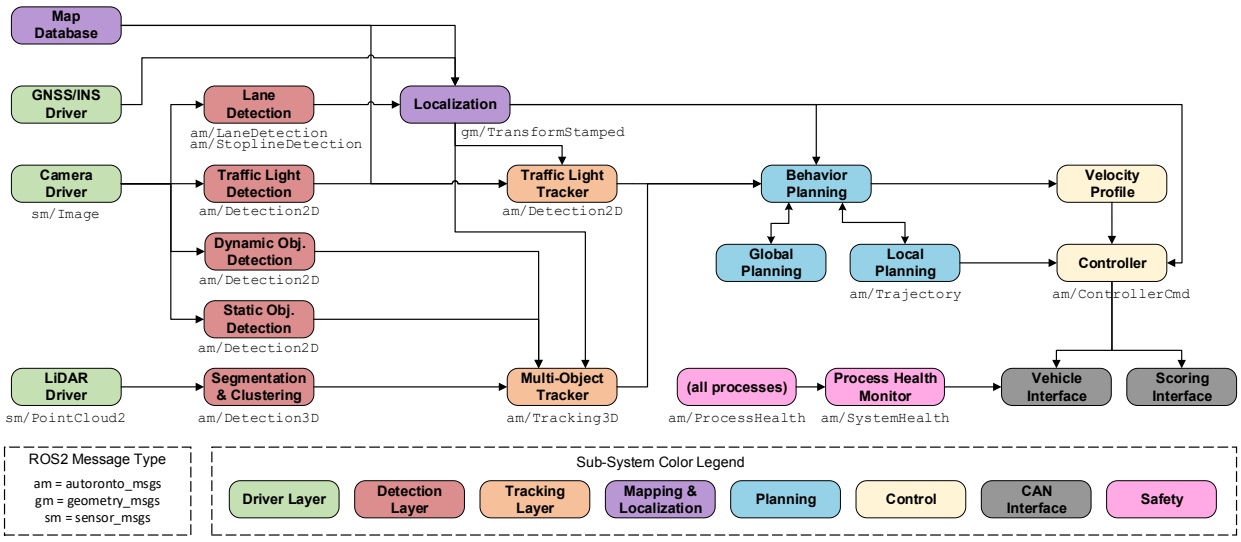


Figure 13: Artemis software architecture

achieve higher performance, the advantages of late fusion are numerous. This approach is more modular by default, allowing different image and LiDAR detectors to be developed, evaluated, and integrated independently.

5.1.3 Mapping and Localization The localization subsystem is responsible for publishing corrections to the received GNSS/INS pose in order to ensure accurate reporting of the vehicle position relative to the map. The Year 2 Artemis software stack does not include an operational localizer; we assume the correctness of the GNSS/INS pose and feed this estimate forward to downstream systems. The modular architectural design facilitates the easy implementation of a localizer; no downstream systems would require modification, as they would automatically begin using the more accurate estimated pose. Localizer corrections, along with all other frame transformations, are published using the ROS2 tf2 library. This library enables easy lookup of transformation matrices between any two frames in the transform tree.

5.1.4 Tracking Layer The tracking layer is responsible for fusing detections from multiple detectors over time to produce smooth, accurate estimates of static and dynamic object states. This layer comprises two subsystems: the multi-object tracker and the traffic light tracker. Traffic lights are processed separately due to the additional logic required for determining light states.

The multi-object tracker consumes detections from the 2D and 3D object detectors, associates them with each other and with existing object tracks, and performs track updates for each object accordingly. The output of the tracker is a set of object tracks, comprising the 3D position, orientation, velocity, dimensions, and class category of each object in the scene. These tracks are expected to remain consistent over time to facilitate reliable motion planning.

The traffic light handling pipeline is responsible for performing localization and state estimation of traffic lights. This module serves three purposes: 1) associate each traffic light detection with a mapped traffic light and a set of controlled lanes, 2) track traffic lights over time to refine class estimates and detect flashing state of the lights, 3) reject spurious or incorrect detections. This is important because noisy or incorrect output can cause abrupt and dangerous autonomous driving behaviour. To address this issue, the traffic light handling pipeline uses multiple sensor modalities and probabilistic priors to improve perception accuracy.

5.1.5 Planning The motion planning stack is responsible for determining, given the states of the ego vehicle and all other detected objects, the path that the ego should follow. This is implemented by three different hierarchical planners, each responsible for planning motion at a different level of granularity. The global planner determines, given the start position and goal position, the sequence of lane segments that the vehicle should traverse. The local planner generates a kinematically feasible path that routes around obstacles. The behavior planner coordinates between global and local planners, determining vehicle behaviors like reacting to traffic lights, stopping at signs, and processing obstacles. It oversees high-level management in Intersection and Highway challenges, ensuring safe and legal navigation.

5.1.6 Control The control subsystem comprises two components: the velocity profile generator and the controller. The velocity profile takes in the generated trajectory and produces a sequence of target velocities for all path points that respect the speed limits as well as limits based on road geometry and vehicle dynamics. The controller is responsible for translating the plan into the low-level commands used by the vehicle: axle torque,

deceleration, and steering wheel angle. This is further decoupled into lateral control (ensuring the vehicle remains in the center of the lane) and longitudinal control (ensuring the vehicle travels forward at the desired speed).

5.1.7 CAN Interface The CAN interface processes are responsible for interfacing with the vehicle and scoring CAN buses. The scoring interface reports all dynamic challenge information (lane detections, object detections) over the scoring CAN, while the vehicle interface handles all other interface tasks: relaying controller commands, reporting back the vehicle states, and implementing the state machines required for autonomous control.

5.1.8 Safety The safety subsystem comprises processes which serve to monitor the integrity of the software stack. While this system does not directly perform perception, planning, or control functionality, it serves a critical role in detecting and handling upstream failure conditions that may result in degraded driving performance.

The safety system is designed around per-process heartbeat signals. Every process in the system is expected to output a pulse: a periodic *process health* message indicating the process status (either "healthy" or "unhealthy"). A process should pulse healthy if it is operating nominally: all inputs are provided, all outputs are being published at the desired rate. Conversely, if a process is missing inputs or is unable to produce the nominal output, it should pulse unhealthy and indicate the reason.

The process health monitor is responsible for aggregating all process heartbeats into an overall system health message. The system as a whole is considered healthy if every process is in the healthy state. As a non-functioning process may simply freeze rather than correctly pulsing unhealthy, the process health monitor additionally maintains per-process watchdog timers. If a process fails to send a heartbeat signal within the configured timeout, it is automatically marked as unhealthy until it resumes sending healthy pulses. This allows the system to automatically recognize when an upstream process has failed rather than allowing failures to propagate undetected into downstream processes. The system health message is processed by the vehicle interface and used as an autonomy state check - if the system state is unhealthy, the vehicle will transition out of autonomy and will be prevented from resuming autonomous control until the system becomes healthy again.

6 SENSOR FUSION

In this section, we describe how Artemis leverages its various onboard sensors to perceive and understand its surroundings. Sensor Fusion is a concept centered around aggregating perception information across multiple sensor modalities. As introduced earlier, Artemis employs a *late fusion* strategy for our perception stack, which means all detectors process data in their own modality first and are then fused to corroborate each other, resulting in an improved detection robustness. It is both a computer vision technique, and more important, a safety feature to mitigate single sensor failure.

As such, the rest of the section starts with 2D and 3D Object Detection on their own, and then provide details of how these detection results, together with other GPS and High Definition (HD) Map data, are fused to establish object tracks, handle traffic light recognition, and identify lanes and limit lines for the downstream route planning module.

6.1 2D OBJECT DETECTION

6.1.1 Deep Learning Model This year, all 2D object detection tasks used a pure machine learning approach. Several models were considered including YOLOv3, YOLOv5, YOLOv8, YOLOX, and NanoDet. The decision was made to use YOLOv5 due to its ease of implementation as we already had the model architecture and training pipeline setup from the previous year. As well, YOLOv5 provides comparable performance with the current state-of-the-art in terms of accuracy and inference speed, as analyzed in Figure 14.

After choosing the YOLOv5 model, we experimented with different model sizes including nano, small, medium, and large. While the Year 1 stack used models of different sizes and input resolutions, we opted to reduce all Year 2 models to the nano size for faster inference while still achieving similar accuracy. With a smaller model, we increased the input image resolution to 1600x1088 which provided an increased detection range of 20 m to 30 m and for the wide-angle cameras and 40 to 65 m for the

Criteria	Weight	YOLOv3	YOLOX	YOLOv5	YOLOv8	NanoDet
Principal Limitations						
Reported model performance	3	4	5	5	5	3
Reported model runtime	1	3	4	4	4	4
Tested model performance	3	4	4	4	5	3
Tested model runtime	2	3	3	3	3	5
Model size	1	3	3	3	3	4
Ease of Implementation	3	3	3	5	3	5
License Permissions	0	2	5	2	2	5
Totals		45	49	55	52	51
Rank		5	4	1	2	3

Figure 14: Pugh matrix for deciding the network architecture for 2D OD.

Criteria	Weight	4 Models	3 Models	2 Models	1 Model
Principal Limitations					
Tested model performance	3	5	5	3	2
Tested model runtime	2	2	3	4	5
Model size	1	2	3	4	5
Ease of Implementation	3	5	5	3	2
Totals		36	39	30	27
Rank		2	1	3	4

Figure 15: Pugh matrix for deciding the number of models to split up the 2D OD detection tasks.

long-range camera. To reduce redundant computations, the deer model was merged with the car/pedestrian model and new obstacles were added to the model with traffic signs. This trade-off is shown in Figure 15 and the final implementation is shown in Table 2 uses three models: lights (traffic and railroad), dynamic objects (pedestrians, vehicles, and deer), and static objects (signs, barrels, and barricades).

To improve performance and meet competition requirements, we performed extensive data inventory, collection, and labeling of new data. This included revising our traffic light classes and adding classes for three Right Turn Only signs, three Left Turn Only signs, railroad crossing signs, railroad lights, barrels, and barricades. In total, we added 67K new training labels, 54K new validation labels, and 21 curated competition test set videos, resulting in a total of 1.4M training and 0.3M validation labels which is summarized in Table 3.

6.1.2 Deep Learning Inference Dedicated deep learning inference architecture is required in order to efficiently run the three 2D OD models across multiple cameras with limited computational resources and integrate with downstream tasks. Multiple optimizations were taken to best utilize the resources and achieve the

Table 2: Model summary, where all models use YOLOv5 nano with 1600x1088 input resolution.

Model	Network	Cameras	Classes
Dynamic Obj.	YOLOv5-n-1600	LW, CW, RW	Car, Ped, Deer
Static Obj.	YOLOv5-n-1600	LW, CW, RW	2xObstacle, 15xSpeed, 8xSign
Lights	YOLOv5-n-1600	CW, LR	4x3L, 5x4L, 9x5L, 2xRailL

Table 3: Training and validation data set samples.

Classes	Samples
3 Light	30.9K
4 Light	13.1K
5 Light	18.7K
Railroad Light	8.2K
Speed Sign	47.2K
Stop / Yield / Railroad	18.2 K
Barrel / Barricade	8.5K
Turn Only	11.2K
Vehicle / Pedestrian	647K
Deer	2.7K

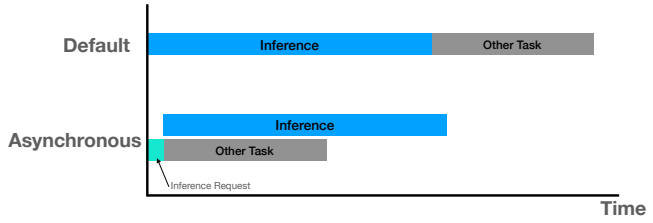


Figure 16: Illustrative comparison between default and asynchronous inference

best performance. The chosen inference framework is OpenVINO[4], as it provides a very streamlined integration with the given Intel GPU.

Inference uses asynchronous inference to optimize the usage of limited computational resources available. As shown in Figure 16, the default scheme for performing inference waits for the inference to be completed before other code can be executed, which is referred to as *blocking*. In contrast, the asynchronous API (async API) that is provided by OpenVINO allows for other code to be executed while the inference is being performed on the GPU, with very small overhead for invoking the inference request, and is referred to as *non-blocking*. This allows the program to perform other tasks such as receiving and sending messages, and post processing of the data, while waiting for the inference to be completed. Overall, the usage of async API proved to be crucial in the implementation to prevent dropping incoming messages, faster throughput speed, and easy integration with ROS2 framework used by the vehicle.

Figure 17 illustrates the system architecture for the inference pipeline. The input image and its ROS2 topic name are compiled together to form a new object called Task, which is directly used to invoke an asynchronous inference request. The program immediately goes to the post-processing step where it checks the output queue for any outputs that need to be processed and published for the downstream tasks. If the output queue is empty, it waits until a new output is filled. Meanwhile, the inference request performs inference and appends the output to the queue once completed. To run all three 2D OD models there are **three independent pipelines** that are

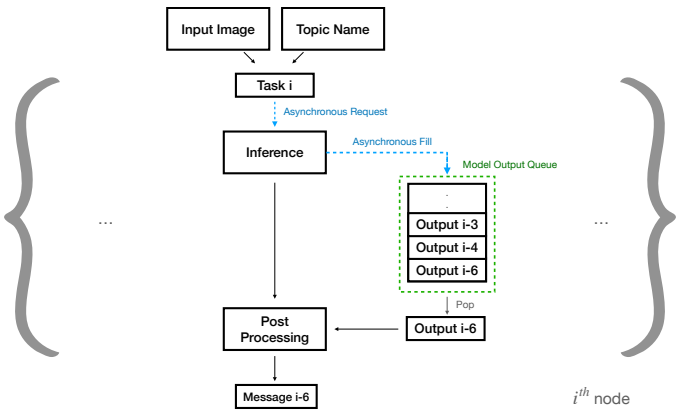


Figure 17: Inference pipeline architecture

Figure 18: 2D OD inference results for static objects (left), dynamic objects (middle), and traffic lights (right) networks.

run, each with different models for dynamic objects, static objects, and traffic lights. Images from different cameras are entered sequentially into the pipeline as they arrive.

6.1.3 Results The throughput of the 2D OD inference pipeline is highlighted in Table 4 showing all inference pipelines running together on all cameras simultaneously. The entries of the table are the mean and standard deviation of the throughout frequency in Hz, evaluated with 8 different test data. Different models only run on 2 or 3 of the 4 cameras depending on the required detection coverage area, hence some of the entries are not applicable and indicated by N/A. The results show that when all model-camera combinations are run, the inference pipeline can consistently process image data over 10 Hz.

Table 4: Inference throughput in frames per second, Hz, mean $\pm$ std. deviation

Detection Type	Center Long Range	Center Wide	Left Wide	Right Wide
Dynamic Objects	N/A	11.46 ± 0.71	11.98 ± 0.70	11.98 ± 0.70
Static Objects	N/A	11.84 ± 1.03	11.66 ± 0.88	12.23 ± 0.77
Traffic Lights	11.62 ± 0.89	10.54 ± 0.94	N/A	N/A

Table 5: Validation performance for 2D OD models.

Class	mAP (%)
All	79
3 Light	72
4 Light	83
5 Light	86
Railroad Light	72
Car	45
Pedestrian	37
Deer	76
Speed Sign	85
Yield/Stop Sign	75
Turn Only Sign	80
Railroad Sign	87
Barrel/Barricade	85

The validation performance results of 2D OD are shown in Table 5. The validation dataset includes over 10,000 images that were labeled by our team and not seen by the model in training. The metric used to assess performance is Average Precision (AP) on the labelled validation set across all classes. Our model achieves 96% AP50, with a true positive defined as 0.5 intersection over union with the ground truth labels, which should be representative of competition performance. The model was also tested qualitatively on a larger test set that covers a broader range of difficult scenes, such as is shown in Figure 18.

6.2 3D OBJECT DETECTION

The 3D object detection module takes a ROS2 point cloud message published by each LiDAR sensor and outputs accurate bounding boxes and localization for the objects of interest within 50 meters in range. This module is responsible for detecting barrels, barricades, traffic signs, pedestrians and deer. The pipeline is shown in Figure 19. First, we use the Himmelsbach ground plane removal algorithm [5] to remove LiDAR points belonging to the ground. Second, 3D Euclidean clustering generates 3D bounding boxes. Lastly, the detections are classified according to the dimensions of the bounding boxes and the average intensity of the points in the bounding box. The output is a list of 3D bounding boxes $[x, y, z, l, w, h, \theta, type, confidence, \rho]^T$, where (x, y, z) are 3D object centroid location, (l, w, h) are length, width and height, θ for object's yaw angle on the $x - y$ plane with respect to the ego vehicle. $confidence$ for the classification is calculated according to the shape of the bounding box and ρ, which represents the average intensity of all LiDAR points within each bounding box.

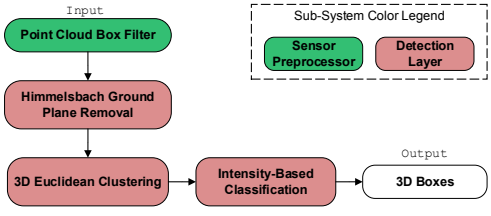


Figure 19: 3D object detection pipeline architecture.

6.2.1 Ground Plane Segmentation Ground plane removal aims to increase the accuracy of 3D Euclidean clustering and accelerate the runtime by reducing the number of points processed during clustering. The Pugh chart in Figure 20 shows the main factors we considered in our design decision. Compared to RANSAC [6], Himmelsbach [5] ground plane removal algorithm is able to handle sloped terrains and achieve better ground plane segmentation performance. The top row of Figure 22 shows the results of ground plane removal for all 4 LiDARs. Because our approach runs in real-time on CPU

and is not a machine learning-based model, we have chosen it for practicality and tunability.

Criteria	Weight	RANSAC	Himmelsbach
Implementation			
In need of data annotation	2	0	0
GPU dependency	1	0	0
Tunability	2	2	6
Performance			
CPU runtime	1	3	2
Ability to handle uneven ground	2	1	4
Ground Plane Segmentation Performance	2	2	4
Totals		13	30
Rank		2	1

Figure 20: Pugh chart for ground plane removal.

6.2.2 3D Euclidean Clustering In Year 1, we explored ML-based 3D object detection methods, namely CenterPoint [7] and PointPillars [8]. Since there aren't publicly available datasets for Cepton LiDAR, we used the traditional 3D Euclidean clustering. This year, we re-evaluated our options. For the same reason, the lack of annotated data for Cepton LiDAR, we focused on improving our 3D Euclidean clustering method, especially by leveraging the intensity channel of the point cloud.

The Pugh chart in Figure 21 shows the three methods we implemented and compared. Our final implementation is the third method "3D Euclidean Clustering + Intensity-Based Classification", which outperforms the baseline method "3D Euclidean Clustering" that was the design in Year 1.

Since the intensity of 3D points is higher for reflective objects like barrels, barricades and traffic signs, the second method, "3D Euclidean intensity-based clustering", aims to use the intensity of each 3D point to better distinguish these objects from background objects, pedestrians and deer. The input point cloud P is first divided into two point clouds based on an intensity threshold, which results in an increase in the computational cost. Then the baseline "3D Euclidean Clustering" is performed on each point cloud separately. The two clustering results are concatenated at the end. However, this method performs poorer in object clustering and localization than the baseline because the intensity of the points turns out to be highly dependent on the viewing angles and lighting conditions. For example, the first step of dividing P into two point clouds based on a fixed intensity threshold splits the original point cloud of a barrel into two separate smaller point clouds, making clustering more challenging.

Therefore, instead of incorporating intensity-related information in the clustering algorithm, we only use it for

Criteria	Weight	3D Euclidean Clustering	3D Euclidean Intensity-Based Clustering	3D Euclidean Clustering + Intensity-Based Classification	CenterPoint
Implementation					
In need of data annotation	2	0	0	0	-1
GPU dependency	1	0	0	0	-1
Tunability	2	2	1	3	0
Performance					
CPU runtime	1	3	2	3	1
Object localization performance	2	3	2	3	4
Object classification performance	2	2	3	4	4
Totals		10	8	13	7
Rank		2	3	1	4

Figure 21: Pugh chart for 3D clustering.

classification, which constitutes the third method, "3D Euclidean Clustering + Intensity-Based Classification". On top of the baseline, we give higher confidence to classes "barrel", "barricade" and "sign" if the average intensity of the 3D points inside a bounding box is higher than a set threshold t. Figure 22 shows the clustering and classification of the traffic sign, barrels, car and pedestrian in the scene. Whereas the baseline and "3D Euclidean Intensity-Based Clustering" might confuse the sign for the pedestrian, the method of our choice is able to better distinguish between them because the painting of signs make them more reflective than a pedestrian. As shown in the Pugh chart 21, this method increases the object classification performance compared to the previous attempts, which is an improvement over our Year 1 baseline approach.

In summary, our LiDAR-based 3D object detection pipeline uses the Himmelsbach ground plane removal algorithm to reduce noise and accelerate the subsequent 3D Euclidean clustering. Our pipeline produces accurate object localization and incorporates intensity-related information for improved object classification.

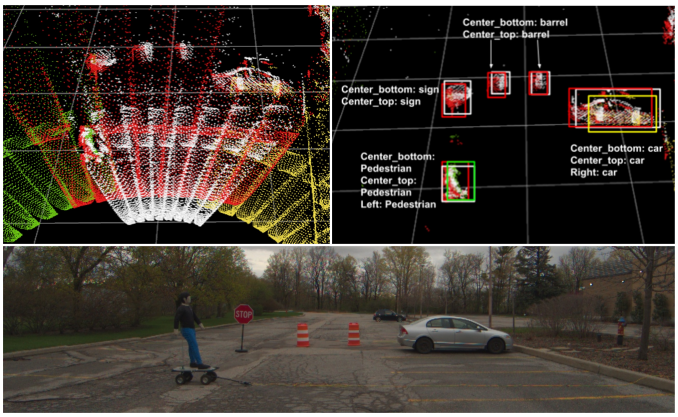


Figure 22: 3D object detection results illustration. Top left: original LiDAR point clouds from all 4 LiDARs. Top right: detections performed on ground plane removed point clouds. Bottom: scene setup.

6.3 OBJECT TRACKING BY FUSION

To stabilize the predicted positions, sizes, and types of the objects in the scene, an object tracking system that fuses cameras and LiDARs is used. This tracking is performed on the dynamic objects (vehicles, pedestrians, animals), static objects (barriers, barricades) and traffic signs. Traffic lights are handled by a separate system (see Section 6.4) because they are not detected by the LiDARs and the light states are time dependent. The objective of the tracking system is to fuse the object detections from the 2D and 3D detectors into a unified and processed set of objects of interest in the scene, as illustrated in the architecture in Figure 23.

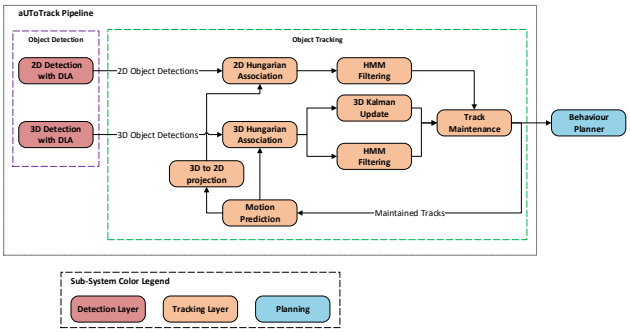


Figure 23: aUToTrackV3 system architecture.

The key improvement to aUToTrack [9] from Year 1 is the handling of detections arriving out of order. Out-of-order detections occur due to the LiDAR sensors being triggered asynchronously, resulting in detections captured earlier in time potentially arriving later than another detection that arrived earlier (see Figure 24). This can cause issues with the Kalman filter smoothing, since the object positions are time dependent and Kalman filter updates are required to occur in monotonically increasing time order.

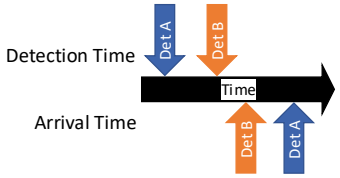


Figure 24: Example of out-of-order detection arrival issue. "Det A" and "Det B" refer to detections captured by Sensor A and Sensor B respectively, such as an individual camera or LiDAR.

6.3.1 Tracker Rollback To handle the issue of out-of-order arrivals, a tracking rollback system is implemented. This method was selected over the alternatives of discarding detections and using the method described in [10] based on the Pugh chart shown in Figure 25. A detection queue maintains a number of detections that have arrived in the order that they were captured. A tracked objects queue is implemented to retain the history of the tracked objects in a scene at a

Criteria	Weight	Drop detections	Tracker Rollback	Out of Order Kalman Filtering
Simplicity				
Ease of development	1	1	0	-1
Algorithm simplicity	2	2	1	0
Accuracy				
Accuracy of localization	3	-2	1	1
Efficiency				
Time complexity	2	1	-1	0
Totals		1	3	2
Rank		3	1	2

Figure 25: Pugh chart to determine how to handle out-of-order detections in the tracker.

previous point in time. When a new detection arrives, it is inserted into its corresponding place in the detection queue. The tracked objects are then reset, or "rolled back" to the corresponding point in time using the tracked objects queue. Lastly, the tracker recomputes the tracker updates (Section 6.3.2) using detections from the detection queue, repopulating the rolled back portion of the tracked object queue up to the latest detection in the detection queue.

6.3.2 Tracker update To update the tracked objects, the tracker takes in both 2D object detections from the cameras and 3D object detections from the LiDARs at their respective rates. An overview of the tracker update algorithm can be found in Figure 23 First, each tracked object's position estimate is updated to the timestamp of the detection used in the update using a velocity motion model and the current best estimate of the object's motion. The detections and tracks are matched using a Hungarian association algorithm [11] based on a cost function between the track and the detection. For 3D detections, each detection is transformed into the inertial map frame. The matching cost metric used is the weighted sum of the Euclidean distance between the detection and track bounding box centers, the percentage difference in the size of the bounding boxes, a 3D Intersection-over-Union (IoU) cost between the boxes and a focal loss cost [12] between the type prediction of the detection and the type of the track. This sum can be seen in equation (1). Following this association, each matched 3D detection is used to perform a Kalman update on its associated track [13].

$$C_{match} = \lambda_{dist} L_{dist}(b, \hat{b}) + \lambda_{size} L_{size}(b, \hat{b}) + \lambda_{iou} L_{iou}(b, \hat{b}) + \lambda_{cls} L_{cls}(p, \hat{p}) \quad (1)$$

To match the 2D detections, the motion predicted tracks are projected from the inertial map frame to the image plane and the matching is performed between the projected positions of the tracks and the 2D bounding box detections. The matching cost metric in this case is the weighted sum of the Euclidean distance between the track and detection bounding box centers in the image

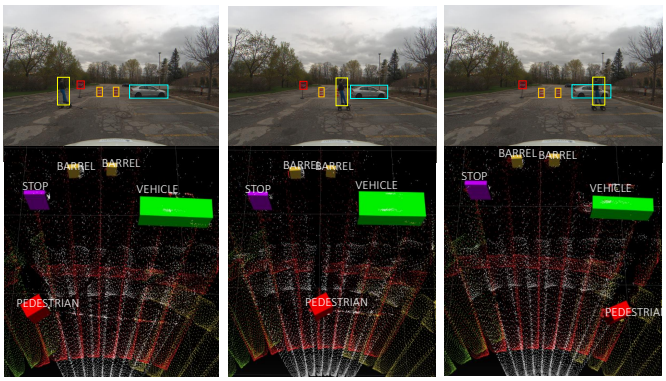


Figure 26: Visualization of the tracker performance along with the corresponding 2D Object Detection input.

plane, the percentage difference in the size of the bounding boxes in the image plane, a 2D IoU cost between the boxes, and a focal loss cost [12] between the type prediction of the detection and the type of the track. The Kalman filter update is not performed with the 2D object detections; only the Hidden Markov Model is applied to the types of each tracked object.

6.3.3 Performance of aUToTrackv3 With the rollback incorporated, the aUToTrackV3 system is able to achieve a throughput rate of 30 Hz and a latency of 50 ms without discarding any detections. Regarding the performance, the tracker is able to maintain mostly consistent tracks for both the static and dynamic objects in a scene, as shown in Figure 26. Further improvements can be made to avoid ID switches in the case of poor incoming 3D detections, such as an occluded vehicle being detected as two separate bounding boxes. As well, potential improvements can be made to incorporate the 2D bounding box information into the Kalman update, further stabilizing the positions of the objects in the scene.

To stabilize the object types, a Hidden Markov Model (HMM) is used to update the type of each track based on the type detected and the previous estimate of the object type [14].

Following the Kalman and HMM updates for the known tracks, track maintenance is performed to initialize and remove tracks. Detections that were not matched to an existing track are initialized as new "trial" tracks, representing new objects entering the scene, while tracks that have not had an associated detection for the past t_{expire} seconds are demoted from full tracks to trial tracks, suggesting the objects might have exited. Trial tracks that have not been reobserved after t_{prune} seconds are removed from the list of tracked objects. If a trial track has been observed for t_{birth} seconds, then it is promoted to a full track. Only the full tracks are published to the planner, preventing false positives from being published.

6.4 TRAFFIC LIGHT HANDLING PIPELINE

Single-frame single-camera approaches for traffic light perception (OD only) are efficient, but fail to handle difficult scenarios. Due to their single frame limitation, these approaches are sensitive to detector noise and cannot handle temporary occlusion or detect flashing lights. Furthermore, single-camera approaches have insufficient sensor coverage for long or wide intersections.

To address the limitations of single-frame single-camera approaches, we developed a multi-camera traffic light tracking pipeline based on our Year 1 approach. Our aUToLights pipeline (Figure 27) is designed to be robust and uses multiple sensor modalities for redundancy. This robustness is accomplished by using a HD map, Hidden Markov Models (HMMs), and flashing light detection.

Our final traffic light pipeline design was selected after we evaluated the performance of each proposed alternative on data collected last year at MCity and at the University of Toronto. To emphasize safety, we give highest priority to False Positive Rate and False Negative Rate. We also consider 3D localization accuracy, flashing detection, and tracking performance. Based on our analysis in Figure 28, our aUToLights pipeline (OD + Map + HMM + Flashing detection) was chosen.

The overall pipeline shown in Figure 27 accomplishes 3 major tasks: 1) traffic light fusion, 2) traffic light tracking, 3) traffic light state estimation, which we briefly describe in the following sections, but readers are encouraged to read our recent publication for more details [15].

6.4.1 Traffic Light Fusion We run our 2D detector separately on the center long-range (LR) and center wide-angle (WA) cameras to obtain 2D traffic light observations (type and light bulb state). From the HD map, we obtain the precise 3D position of all traffic lights. We then fuse the 2D traffic light observations from the detector with the HD map. This enables us to combine state estimates with precise 3D position from the map. If our 2D detector fails, we can avoid a false positive (erroneous traffic light detection) or a false negative (missed traffic light detection) by using the HD map as a fallback.

We perform the 3 following fusion steps separately for each camera.

1. Determine expected traffic lights from the map.
2. Project 3D traffic light position from HD map to 2D bounding box in each camera sensor's image plane.
3. Determine optimal association of projected traffic lights (map) to 2D observations.

Figure 29 shows an example of light fusion. With accurate sensor calibration, the projected traffic lights from the HD map are close to the 2D detections. During

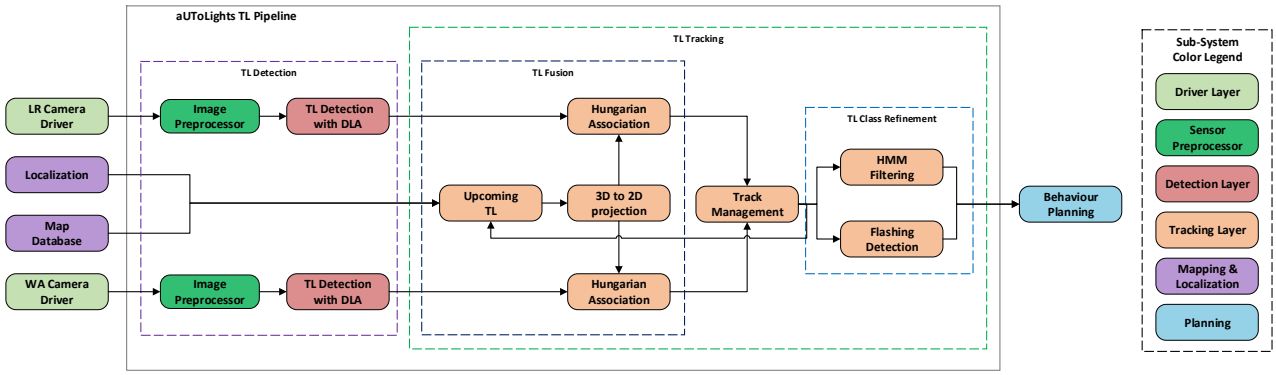


Figure 27: Traffic Light pipeline architecture.

Criteria	Weight	OD Only	OD + Map	OD + HMM	OD + Map + HMM	OD + Map + HMM + FLASH
Implementation						
Ease of Implementation	1	1	0.5	0.5	0.25	0.25
Additional Dataset Requirements	1	1	1	0.5	0.5	0.5
Performance						
False Positives Rate	3	0.25	0.5	0.5	1	1
False Negative Rate	3	0.25	1	0.25	1	1
3D Localization Accuracy	2	0.25	1	0.25	1	1
Flashing Light Detection	2	0	0	0.5	0.5	1
Object Tracking Performance	2	0	0.5	0.5	1	1
Totals		4	9	5.75	11.75	12.75
Rank		5	3	4	2	1

Figure 28: Traffic light handling architecture comparison.

traffic light association, we pair up the traffic lights from the map and the 2D detector based on the bounding box similarity and euclidean distance. Successful pairs of the map and 2D traffic lights are fused to create candidate traffic light tracks.

6.4.2 Traffic Light Tracking Following fusion of traffic light observations, we track traffic lights temporally across frames over the two cameras. Each candidate traffic light track must be detected for N_{birth} successive frames before we establish a traffic light track. All traffic light

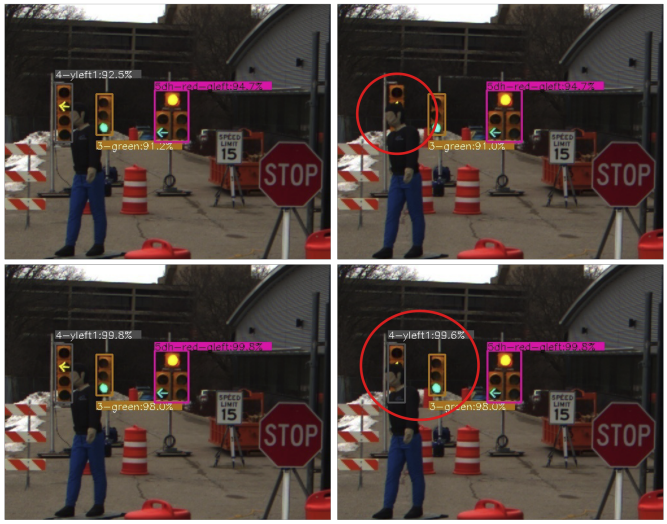
Figure 29: An example of traffic light fusion using Year 1 data collected at MCity. Red and green boxes represent 2D traffic light detections and their states, indicating red and green lights respectively. Cyan boxes indicate traffic lights projected from the HD map. Associations are depicted by green lines.

tracks are assigned a track ID to uniquely identify it. For each observation of an existing traffic light track, we add the 2D detected state to the track's history. By creating traffic light tracks and updating the track's history, we can handle 2D false negatives due to temporary occlusions or 2D detector limitations. Any traffic light tracks that are not observed for N_{death} frames are discarded.

6.4.3 Traffic Light State Estimation For each traffic light track, we use the track's state history to estimate the most likely current state. We use two approaches for state estimation: 1) Hidden Markov Model (HMM) state filtering and 2) duty cycle threshold flashing light detection. Both of these approaches rely on using statistical priors derived from traffic light regulations.

For HMM state filtering, we incorporate the expected traffic light sequences from regulations (e.g. green light to yellow light to red light) into transition probabilities between different states. We also model the 2D detector using observation probabilities based on its test performance and its tendency to mispredict specific classes. The transition and observation probabilities allow us to perform an HMM update step to propagate our current belief state of the most likely traffic light state. For each new observation, we incorporate its predicted state and confidence using the HMM update step. The HMM filter also enables state estimation when there is no 2D detection for a given frame. Figure 30 shows an example where the HMM enables us to infer a state when the 2D detector fails due to occlusion on the 4-light. Note that the detection confidences are also higher when using the HMM filter since the state estimate is based on multiple frames rather than a single frame.

For flashing light detection, we use a simple duty cycle threshold approach. Based on the US federal highway regulations, we set an upper and lower threshold of ($\frac{1}{2}$, $\frac{2}{3}$) for the ratio of on and off traffic light states over the last N states [16]. The traffic light is predicted to be flashing only if both the on and off state ratios are within this threshold.



t_1 t_2

Figure 30: Visualization of traffic light bounding boxes and predicted class labels over a sequence of frames (two frames are shown) for the detector only approach (*Top*) and the proposed approach (*Bottom*).

6.5 LANE AND STOP LINE HANDLING PIPELINE

Reliable lane and stop line handling is crucial for localizing Artemis, navigating safely, and following road rules. Single lane and stop line detectors are an efficient way to achieve this but fail to provide a complete understanding of a driving environment. Due to their reliance on computer vision methods these approaches are highly susceptible to lighting conditions, the quality of road markings, and occlusions. Additionally, by only making detections in the ego lane these methods provide a limited understanding of the road and restricts downstream capabilities.

To address the limitations of computer vision based single lane and stop line detectors, we developed a pipeline that fuses multi-lane detections with the vehicle's HD map and GPS position. By using a HD map, we exploit a prior that greatly improves the safety and reliability of our system while satisfying our computational constraints. At the same time, we still utilize our lane and stop line detection capabilities to verify the accuracy of the HD map and make necessary

Criteria	Weight	Computer Vision Approach	Deep Learning + Post Processing	Deep Learning + Post Processing + Map
Implementation				
Ease Of Implementation	1	1	0.5	0.25
Requires Annotated Data	1	0	1	0.5
GPU Dependency (Artemis Compute)	2	0	1	0.5
Tunable	1	1	0.5	0.8
Performance				
Lane And Stop Line Handling Accuracy	3	0.5	0.75	1
Operation Speed	3	1	0.5	1
Robustness After Environment Change	2	0.5	0.75	1
Totals		7.5	9.25	10.55
Rank		3	2	1

Figure 31: Lane and stop line architecture comparison.

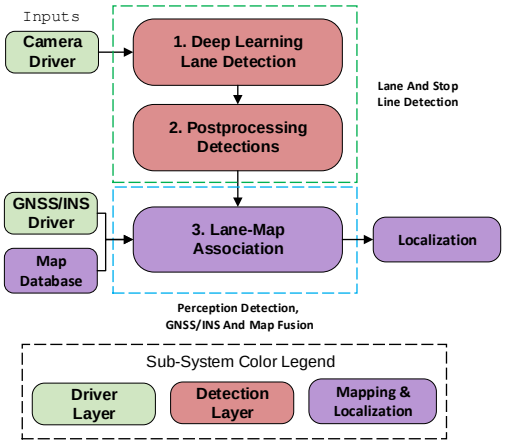


Figure 32: Lane and stop line handling pipeline.

offline corrections based on discrepancies between the two. This capability is particularly useful when road markings have changed due to roadwork or repainting.

After thorough evaluation of proposed alternatives on data collected at MCity and testing at the University of Toronto, we selected the most reliable system based on detection accuracy, robustness to environment changes, and operation speed. Our final lane and stop line handling pipeline (combining Deep Learning Detections + Computer Vision Post Processing + HD Map) was determined to be the most effective option, as shown in Figure 31.

Our pipeline is shown in Figure 32, in which we perform the following 3 steps:

1. Detect all road lines using an end-to-end neural network.
2. Post-process the detections using computer vision methods to identify attributes such as lane type, color, and distance to lane lines and stop lines.
3. Fuse the post-processed detections and HD map to localize the vehicle.

6.5.1 Deep Learning Detections And Post-Processing

We use YOLOPv2 [17], a state-of-the-art road line detection network trained on BDD100K dataset [18]. In order to overcome computational limitations and scarcity of annotated data, our neural networks do not aim to detect and classify multiple lane classes. Instead, we utilize deep learning to detect crucial road lines such as but not limited to lane lines, stop lines, crosswalks, and curbs as a single, unified class. These are then post-processed using conventional computer vision methods which are fast and efficient to determine specific attributes such as the color, type, and distance of the line from the vehicle. In the end, these detections are compared to the HD map.

The post-processing of these deep learning detections involves several steps, shown in Figure 33. After

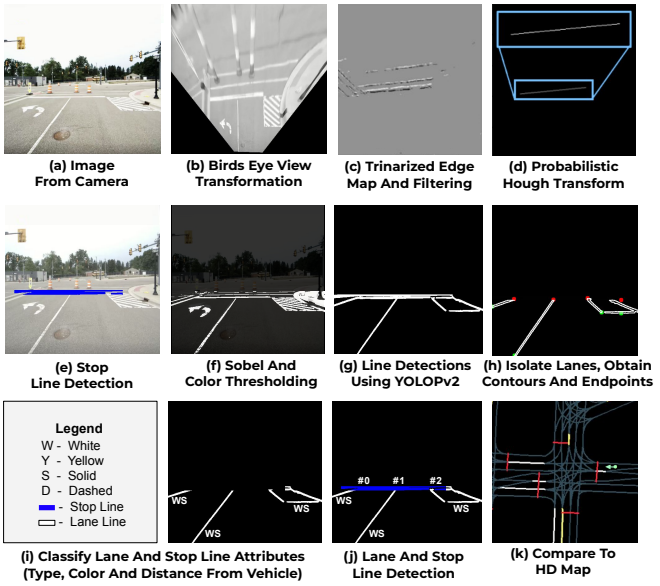


Figure 33: Lane and stop line detection pipeline.

transforming the camera image into the bird's eye view using a homography transformation, we apply contrast enhancement and edge detection to improve the quality of the image for feature extraction. Using prior knowledge of road marking widths, a probabilistic Hough transform, and line segment clustering, we consistently detect stop lines, which we then isolate by masking out other pixels. Color thresholding is used to accurately identify important attributes of these detections such as whether the lanes are yellow or white and if they are solid, dashed or double lines. Reporting characteristics of the lanes permits downstream systems to gain a better understanding of available navigation options. We also predict the relative distances between Artemis and detected lines by analyzing pixel differences, which is crucial for safe lane keeping. Finally, we cross-reference the detections with the HD map and flag any discrepancies for human review. Validating the map offline improves its reliability. This ensures that Artemis can confidently handle lanes and stop lines using the HD map when driving autonomously.

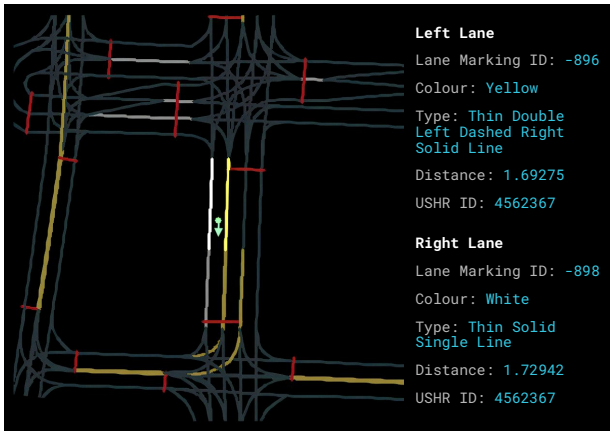


Figure 34: Lane and stop line handling using the HD map.

6.5.2 Lane-Map Association And Localization With the HD map validated and fixed using our deep learning lane detection approach, it can be queried based on the vehicle's GPS location. When Artemis is under autonomous driving mode, it relies on the HD map associated at its real-time GPS location to obtain information about nearby lanes including their relative distance, color (e.g. yellow or white) and marking type (e.g. solid, dashed, or double), as shown in Figure 34. In addition to lane handling which will be used by the route planning module, the HD map is also used to detect stop lines, which is crucial for safely halting the vehicle when approaching intersections where other road users may be crossing.

7 ROUTE PLANNING

The rapidly evolving domain of autonomous vehicles demands sophisticated and efficient route planning solutions that not only ensure safe navigation but also optimize the overall travel experience. To address these requirements, our Route Planning section incorporates three key modules: (1) the Planning module, which focuses on generating an optimized planned path, (2) Velocity Profile Generator module which smooths out the speeds in the planned path to product a trajectory, and (3) the Vehicle Control module, which is responsible for executing the trajectory as closely as possible. This cohesive system ensures a seamless transition from high-level route planning to low-level control commands, ultimately delivering a safe and efficient driving experience.

To facilitate understanding of key concepts that underpin our Planning and Vehicle Control modules, we define the following terms:

- pose: position and orientation $P = \{x_{ref}, y_{ref}, \theta_{ref}\}$
- planned path: a sequence of poses with max speed
- trajectory: a path with a target velocity v_{ref} associated with each pose
- lanelet: a distinct, variable-length segment of a road lane, characterized by a centerline, boundary, and direction of travel, which connects to other lanelets through parent-child relationships
- sibling lanelet: a lanelet that is immediately adjacent to and shares the same direction of travel with the current lanelet
- global path: a sequence of connected lanelet ids obtained from the HD map
- road segment: a section of the road network consisting of interconnected lanelets and associated traffic signs and signals that together facilitate traffic flow and navigation
- connectivity graph: a set of all road segments that form the HD map

7.1 PLANNER

Our proposed solution is a hierarchical design that comprises three interconnected sub-planners: 1) Behavioral Planner (BP), 2) Global Planner (GP), and 3) Local Planner (LP). The hierarchical design allows for efficient and modular operation while ensuring the robust and safe navigation of self-driving cars. This design ensures that each sub-planner can focus on its specific task, ultimately leading to more reliable and optimized route-planning solutions. This planner architecture has been inspired by the success of hierarchical designs in various real-world tasks, including top-ranked teams in the DARPA Urban Challenge[19] that employed similar strategies for tasks such as high-level routing planning and low-level trajectory generation.

The Planner architecture, as shown in Figure 35, takes as input the target sequence of waypoints to traverse, map data, vehicle localization, traffic light states, and detected objects. Its output is a desired planned path, which is subsequently sent downstream to the Velocity Profile Generator module. BP serves as the central hub of our architecture, managing the inputs and outputs, facilitating bilateral communication between GP and LP, and dictating reactions to detections from perception. Each cycle of the planning module, which consists of re-processing in BP and invoking LP occurs every 100 ms (10 Hz) while GP is invoked by BP once on every change to the destination. Given that this year's competition scope does not involve dynamic rerouting, the global path leading to the destination remains unchanged.

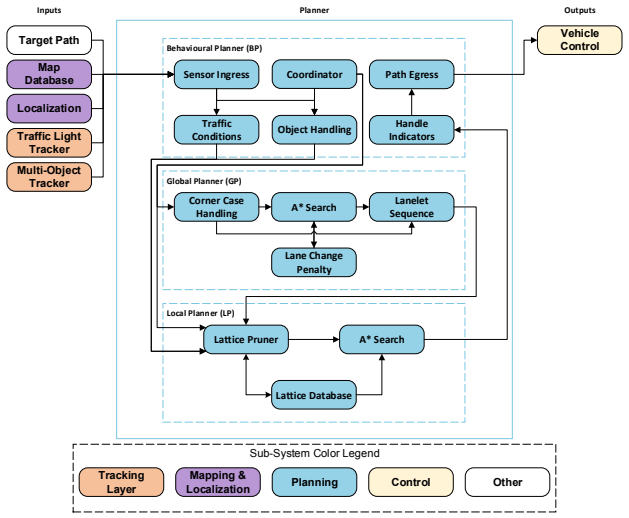


Figure 35: Planning module architecture.

7.1.1 Behavioral Planner (BP)

BP contains two main components: a main Coordinator module and the rest blocks as the Behavior module as shown in Figure 35. The Coordinator module acts as a high-level manager for the three sub-planners whereas the Behavior module is responsible for handling traffic signals, traffic signs, and detected obstacles and determining the correct behavior for the vehicle including toggling the indicator lights.

The Coordinator module reads sensor data at a fixed frequency of 10 Hz. At the start of each planning cycle, Coordinator calls the three sub-planners in the following order: BP, GP, and LP. First, the Behavior module acquires updated information on Artemis's position, current road traffic state, and potential obstacles from the Coordinator module. Next, GP is triggered to generate the global path using Artemis's current location and destination. For this year without dynamic rerouting, GP only needs to be called once to generate the path. If no path is provided or an emergency arises, the Behavior module generates a safe planned path to bring Artemis to a safe stop. Otherwise, the global path from GP is forwarded to LP, which creates a planned path to guide Artemis to its desired destination.

When BP is initially run, data from the HD map, sensors, and vehicle localization are combined to determine legal behaviors for the car. For example, if any speed signs are detected on the side of the road, a virtual line is drawn across the road at the position of the sign, and the speed limit is observed when the vehicle crosses the virtual line. Any obstacles detected by the sensors are first checked to be valid, and validated obstacles are then passed to LP to trigger lane changes if necessary.

When Artemis approaches an intersection with traffic lights, the Behavior module will generate a virtual stop line in front of the intersection upon a red signal, inducing an empty planned path which brings Artemis to a stop. For stop signs, the Behavior module will pause for a minimum stopping duration and only check for obstacles before calling LP again to move the vehicle.

Furthermore, car indicator lights such as blinkers are also triggered by the Behavior module during lane changes or turns.

7.1.2 Global Planner (GP)

Given a sequence of target waypoints to traverse from the start to destination, GP searches through the connectivity graph to determine the sequence of connected lanelets with the shortest total distance to be traversed. This forms the global path.

Despite this year's competition providing a predetermined road sequence which allows pre-computation of the global path offline, we opted to develop GP for future adaptability and to demonstrate our system's resilience. BP triggers GP at the start of the planning cycle with Artemis's current location and destination. GP then generates a global path that BP forwards to LP, and is not triggered again until a new destination is given.

The global path includes lane change signals. However, LP can override these signals based on perception outputs, such as avoiding obstacles. However, the planned path strives to adhere to GP's recommendations

with deviations being penalized.

GP is implemented using the A* algorithm, which searches over lanelets in the connectivity graph to find the most efficient route. In addition, we considered corner cases where the start and destination locations are within the same lanelet. Under these scenarios, the lanelet travel direction determines if the destination is before or after the starting point; if it's after, the current lanelet is returned, otherwise, a path with more than one lanelet is computed. Moreover, the GP design discourages frequent lane changes and lane changes into the destination lanelet to ensure road safety, avoid confusion, reduce the risk of accidents, and maintain route efficiency by providing a smooth, predictable vehicle trajectory.

By incorporating GP into our Planner architecture, we have designed a system that is not only well-suited for the current competition but also adaptable for future iterations, showcasing the robustness and flexibility of our solution.

7.1.3 Local Planner (LP) LP takes the global path generated by GP via BP and creates a kinematically feasible, obstacle-free planned path for the controller. To achieve this, various planners were assessed, as depicted in Figure 36. Ultimately, a lattice-based planner was chosen for its kinematic feasibility assurance, runtime performance, and ease of implementation.

Criteria	Weight	Centerline Follower	Lattice-Based Planner	Free Space Planner
Implementation				
Complexity	5	0	1	-2
Ease of testing	2	0	3	-2
Performance				
Runtime Performance	3	0	3	-2
Reliability (amount of edge cases)	1	0	2	-1
Capabilities	1	0	1	8
Totals		0	23	-13
Rank		2	1	3

Figure 36: Pugh chart comparing different approaches to local planning.

For each HD map, we pre-generate its lattice version offline. To construct the lattice, we create lattice edges connecting pairs of points on the HD map, from each point to three points situated 10 to 15 meters ahead in the current and sibling lanelets. Edges are connected by designating any edge beginning at the end of another edge as the second edge's child.

Each lattice edge represents a potential planned path segment, facilitating transitions between initial and terminal states. These segments are generated using a pair of quintic Hermite splines for each dimension. The splines are constructed to ensure the initial and terminal poses accurately match the lattice edge's initial and terminal positions, while other parameters are chosen to satisfy curvature and collision avoidance constraints. The

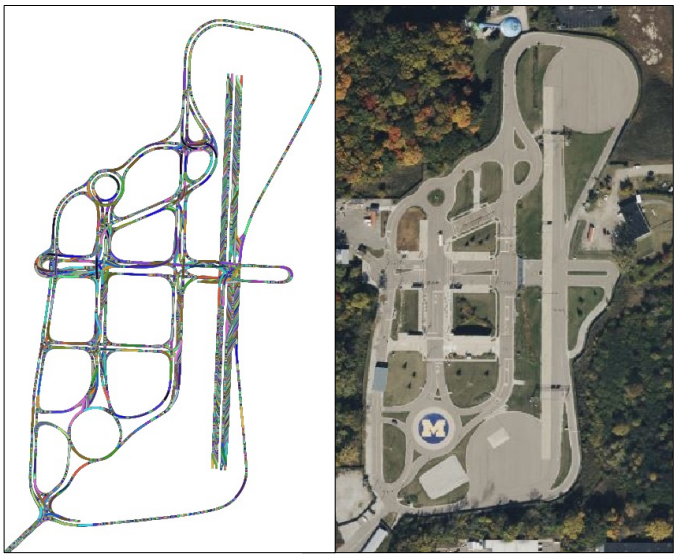


Figure 37: *Left*: Full lattice generated from the MCity map; *Right*: Satellite imagery of MCity.

generated lattice for MCity is shown in Figure 37. Details of how lattice edges look like can be best viewed in color in Figure 38.

Since the area of the HD map is relatively small, a complete path from the vehicle's current pose to the destination can be generated each cycle. An A* search algorithm is employed on the lattice to determine a path given starting and ending positions. The search utilizes Euclidean distance as a heuristic and applies a penalty for lane changes and deviations from the global path, resulting in the lowest cost path through the lattice.

To handle obstacles, lattice edges intersecting with an obstacle are pruned, making them impassable for the A* search. Since this process occurs online, it is computationally infeasible to check for intersections with every lattice edge due to runtime constraints. To reduce runtime, we only check for intersections within a certain

Figure 38: An example of a planned path on a test lattice generated from our UTIAS map. On the left is a path in red without obstacles and on the right is a path with obstacles on the pruned lattice. Segments in other colors represent lattice edges.

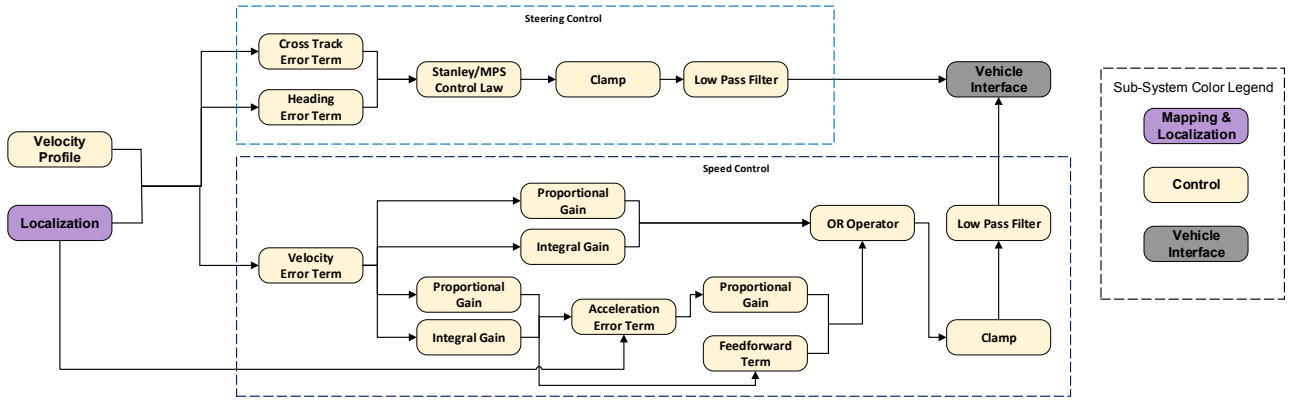


Figure 39: Decoupled Controller Diagram depicting the steering and speed control

distance from the vehicle's current pose. We find the nearest edges by employing a Breadth-First Search (BFS) from our current edge and only check those edges for intersection. This approach significantly reduces the computational effort required to prune the lattice. Obstacles also have a timeout and will disappear after a fixed amount of time if not re-detected. This prevents issues from erroneous detections from the object tracker. Figure 38 shows paths generated on our UTIAS test lattice before and after injecting obstacles.

We designed the Planner architecture to satisfy the requirements posed by the Intersection and Highway challenges. For the Intersection Challenge, we are provided with a sequence of target waypoints ending with the destination. BP ensures appropriate reactions to stop and yield signs along with traffic lights while LP ensures a kinematically feasible planned path for the car free of hitting curbs. For the Highway Challenge, since no explicit destination is provided, it instead terminates at a full blockage of all legal lanes. Here, BP manages a temporal record of detected static obstacles from perception and dynamically adjusts the destination. LP uses this temporal record to generate kinematically feasible planned paths with obstacle avoidance and lane changes. In both challenges, GP translates the high-level sequence of waypoints into the optimized global path. All these considerations are necessary to ensure that Artemis can meet all the competition requirements and successfully complete both challenges autonomously.

7.2 VEHICLE CONTROL

This section explains our vehicle control strategy. The controller takes in a desired path and the vehicle's localization information as inputs to compute axle torque, deceleration rate and steering wheel angle commands. These resulting control commands are taken as inputs by the vehicle interface module, which sends these commands to the vehicle over CAN messaging protocol.The desired path is obtained from the Path Planner and is a sequence of waypoints consisting of 2D poses and speed limits. This planned path is converted into a trajectory for the controller to track by the Velocity Profile Generator(VPG), which generates a kinematically feasible velocity profile over the planned path by substituting the road speed limits with the desired velocity values. The vehicle's current state is obtained from the NovAtel GNSS/INS sensor and provides the measured 2D pose, current velocity and current acceleration.

Criteria	Weight	Pure Pursuit	Stanley	MPC
Ease of implementation	8	7	10	3
Ease of calibration	8	9	10	2
Computation time	3	10	10	0
Constraints	5	4	4	10
Lanekeeping performance	8	6	6	10
Disturbance rejection	6	5	5	10
Steady State Error	5	0	0	10
Totals		256	288	280
Rank		3	1	2

Figure 40: Pugh chart for comparing different lateral control approaches

7.2.1 Control Architecture Overview

The Pugh chart in Figure 40 compares the different geometric and optimization approaches that were considered for the lateral control. Although the Model Predictive Controller (MPC) is the clear winner in terms of criteria such as ability to add constraints, lanekeeping performance, disturbance rejection, and steady state error, the goal for this year was to implement a control strategy which is easy to develop and calibrate on the vehicle to enable us to maximize our testing time on Artemis. This led us to select Stanley controller as the baseline and an accompanying PID controller for longitudinal control.

The controller is decoupled into a steering control module and speed control module as shown in Figure 39, allowing for strong independent control performance for steering and speed.

7.2.2 Velocity Profile Generator

The VPG forms the connection between the planner and controller by taking as input the local path from the LP and assigning a kinematically feasible trapezoidal velocity profile over the path for the controller to follow. An example velocity profile can be seen in the velocity tracking plot in

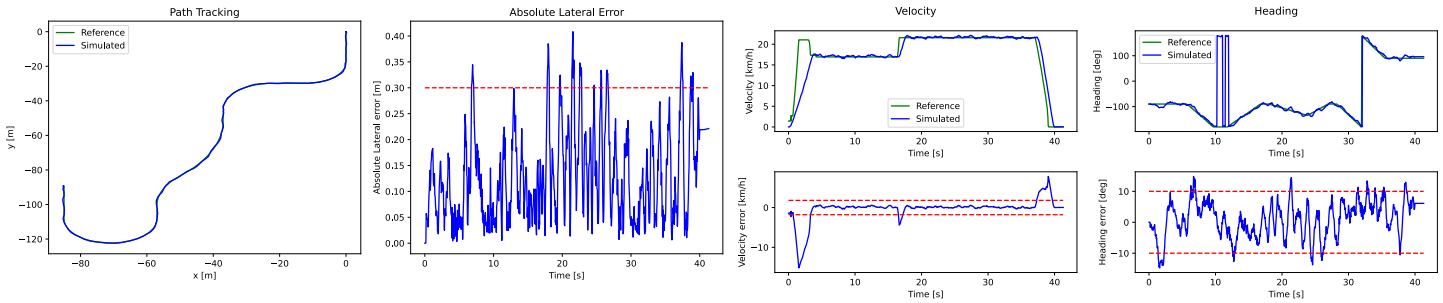


Figure 41: Experimental control results in simulation consisting of a path tracking plot, absolute lateral error plot, velocity tracking/error plots and heading tracking/error plots.

Figure41. The deceleration phase ensures that every local path ends with a 0 m/s velocity which is critical for safety reasons. VPG also reduces the desired speeds depending on the road curvature and allowable lateral acceleration limit.

7.2.3 Speed Control Our investigation into speed control involved evaluation of two distinct approaches: a singular PID system and a cascaded PID system. The former approach determines the torque command via a PI controller based on the velocity error. The latter approach, however, uses a first PI controller to derive the desired acceleration based on the velocity error, while a second integral controller computes the torque command by taking into account the acceleration error and feed-forward term. The feed-forward term is computed by multiplying the desired acceleration by a constant, which was obtained by empirical analysis. Both methods exhibit a near-identical tracking performance.

An additional consideration for the speed control architecture design was the inclusion of the negative deceleration values required by the vehicle interface. We approached this requirement by experimenting with two methods: (1) multiplying the negative torque by a constant (obtained by empirical analysis similar to the feed-forward constant) or (2) using the desired acceleration value from the first PI controller when the output torque matches the sign of the desired acceleration value, otherwise multiplying the negative torque by a constant. Both methods result in adequate speed control tracking and have similar performance.

7.2.4 Steering Control For lateral steering control, the Stanley controller used by Stanford in the DARPA Grand challenge[20] was used as a baseline to design the Model Predictive Stanley (MPS) controller. The base Stanley controller follows a geometric proportional control law which converges to the closest waypoint along the path. Using the vehicle's front axle as the reference point, the Stanley controller calculates the heading error and cross-track error between the reference and the closest waypoint. The final steering angle is a combination of both the errors -

$$\delta = \psi + \arctan\left(\frac{k * e}{k_s + v}\right) \tag{2}$$

Where ψ corresponds to the heading error and e corresponds to the cross-track error. The gain k determines how aggressively we reduce the cross-track error and k_s protects against low speed singularities.

MPS applies the Stanley control law over the prediction horizon while simultaneously propagating the vehicle's states through time using a kinematic bicycle model. This results in improved performance as we are not only eliminating error at the closest reference point but along a fixed horizon of points on the planned path. Compared to the base Stanley controller, we get a much smoother steering response. However, having a large prediction horizon during sharp turns results in the vehicle cutting corners due to the waypoints near the end of the path affecting the resulting δ. This drawback was mitigated using 2 methods - gain scheduling to vary the MPS horizon with road curvature and reducing the weights on the the individual δs as we move away from the vehicle's current position. The MPS control law is as follows -

$$\delta = \sum_{i=0}^{N} K_i \left[\psi + \arctan\left(\frac{k * e}{k_s + v}\right)\right]_i \tag{3}$$

7.2.5 Safety Constraints To ensure safety of the controller commands, we place constraints on numerous parameters: the maximum velocity, minimum/maximum acceleration, minimum/maximum torque, maximum steering wheel angles and maximum steering wheel angle rate. Additionally, low pass filters were added for all the controller commands to ensure a smooth torque, deceleration and steering command request to the vehicle interface module.

7.2.6 Experimental Results Our evaluation of the decoupled controller involved numerous scenarios to fully test the controller's efficacy, which included a driving scenario complete with straight paths, left and right turns, and a stop. We illustrate the path tracking performance, lateral results, longitudinal results, and heading error results in Figures 41 respectively. The longitudinal control performance was evaluated based on the velocity profile tracking and the lateral control performance was evaluated based on the heading error and lateral error with respect to the reference trajectory. Our controller demonstrated a maximum lateral error of 40 cm, while the speed tracking performance proved adequate for

precise longitudinal tracking. Overall, our evaluation attests to the efficacy of our controller in navigating a rigorous driving scenario.

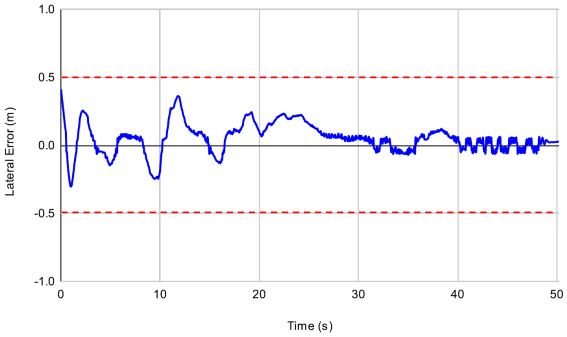


Figure 42: Vehicle Lateral Error for UTIAS parking lot path

7.2.7 Vehicle Testing Figure 42 depicts the vehicle's lateral error which is an indicator of the controller's lanekeeping performance. This data was collected during autonomous testing at our UTIAS campus and the path followed by the vehicle for this test can be seen in Figure 38. During this test, the planner, VPG, controller and the vehicle interface module were running on Artemis. As we can see, the MPS controller is able to achieve lateral errors of less than 50cm which was one of our core design requirements. The error is less than 25cm on straight paths and peaks at around 40cm on high curvature turns which also involves a tight U-turn!

8 CHANGES FROM YEAR 1 TO YEAR 2

In closing, we summarize the changes from Year 1 to Year 2 in two categories, hardware and software. Hardware changes were mostly motivated by the integration with the new vehicle, new sensors, and the Intel server. Software changes were driven by lessons learned in Year 1 developing the perception system as well as the introduction of route planning and vehicle control modules.

8.1 HARDWARE CHANGES

8.1.1 Vehicle Retrofit The pentagonal top sensor rack stayed the same as the Year 1 design, while the bumper sensor rack was changed to a plate design for its low profile such that the bumper sensors do not protrude from the existing vehicle contour. A new bluelight controller unit was built to automatically toggle bluelight states as a warning light or an autonomous mode indicator. A custom harness made up of four separate CAN bus lines was constructed to communicate between the vehicle and our server.

8.1.2 Sensors The bumper sensor suite was modified to support a single Continental ARS548 radar instead of the 3 ARS430 due to the significant sensor improvement that a single ARS548 suffices the coverage requirement. In addition, the new X90 LiDAR by Cepton has been supported but was not used to replace an existing P60 because having all LiDARs the same model allows us to tune their parameters in a unified fashion. As for the GNSS/INS unit, aUToronto has switched over to the NovAtel PwrPak7D-E2 receiver, using TerraStar-C PRO GNSS correction service allowing for rapid convergence and centimeter-level accuracy. While the Lucid cameras originally used hardware synchronization in Year 1, we decided to remove this additional hardware and opted for software-based PTP sync instead, allowing for simplified time synchronization with the GPS system.

8.1.3 Intel Server The computing platform in Year 2 has been changed from the consumer desktop with an NVIDIA GPU to an Intel data center server with an Intel GPU. As a result, many software architectural changes were made to leverage the high CPU core count as well as the deep learning acceleration capabilities of the server. Most notably, the integration of OpenVINO [4] allowed not only for deep learning acceleration, but asynchronous inference in order to improve performance.

8.2 SOFTWARE CHANGES

8.2.1 Perception Sensor Fusion Our development on deep learning acceleration using Intel's OpenVINO in an asynchronous way enabled us to fully commit to deep learning approaches for 2D object detection. In Year 2, all objects were detected using a YOLOv5 nano model, including barrels and barricades which were detected using conventional pattern matching in Year 1. In addition, new signs such as various turn only signs and railroad signs and lights were also supported by the neural network. 3D object detection this year incorporated LiDARs' intensity channel to enhance traffic sign detection. Moreover, from the Year 1 competition, we noticed that 3D detections would sometimes arrive out of order due to each Cepton LiDAR's different capture timestamps. This year we implemented a tracker rollback function to specifically handle such scenario which resulted in a more time consistent object tracker output. Aside from tracker, the traffic light handling pipeline was redesigned in Year 2 to improve state estimation by including observations that were previously discarded and to better handle ego motion with more accurate traffic light projection. Lastly, in addition to apply lane-map association for localization, we devised a strategy to run deep learning based lane detection model offline to validate the HD map against any discrepancy.

8.2.2 Route Planning and Vehicle Control Route planning and vehicle motion control are the entirely new addition to our autonomy software stack since we operate on a real vehicle in Year 2. To gain vehicle control, 3 CAN bus channels (HS, CE, LS) were added in addition to Year 1's scoring CAN interface. A sophisticated vehicle interface layer was developed according to the GM document to send autonomous driving commands to the vehicle.

9 PAPERS AND CONFERENCES

This section lists all publications and conferences attendance by aUToronto members since the commencement of AutoDrive Challenge II.

Conference papers published:

1. S. Wu, N. Amenta, J. Zhou, S. Papais, J. Kelly, "aUToLights: A Robust Multi-Camera Traffic Light Detection and Tracking System," 2023 20th Conference on Computer and Robot Vision (CRV), Montreal, QC, Canada, 2023

2. J. Xu, S. Waslander, "HyperMODEST: Self-Supervised 3D Object Detection with Confidence Score Filtering," 2023 20th Conference on Computer and Robot Vision (CRV), Montreal, QC, Canada, 2023

Conference attended and to-be attended:

1. Attended and presented a Workshop Presentation at the Conference on Robots and Vision (AI · CRV 2022), Toronto, ON, Canada, May 30, 2022

2. To attend and present an Oral [15] and a Poster [21] Presentation at the Conference on Computer and Robot Vision (CRV), Montreal, QC, Canada, June 6-8, 2023

10 CONCLUSION

In this concept design report, we summarized our design of Artemis for the Year 2 competition of the AutoDrive Challenge II. We detailed the mechanical and electrical integration with the new Bolt EUV, sensor placement and configuration, and software architecture. We described each module of our software architecture in details, including multi-modal perception sensor fusion strategy, route planning and vehicle control. In addition to the theoretical justification of the design decisions, we demonstrated each module's performance running live on Artemis under various internal testing scenarios to show our progress towards Level 4 autonomy.

References

[1] *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles*. DOI: 10.4271/j3016_201806.

[2] C. Qian, "On the design and validation of an autonomous vehicle perception system for the sae/gm autodrive challenge ii," M.S. thesis, University of Toronto, 2022.

[3] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[4] *Openvino toolkit overview*. [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html.

[5] M. Himmelsbach, F. v. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 560–565.

[6] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981, ISSN: 0001-0782.

[7] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 784–11 793.

[8] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[9] K. Burnett, S. Samavi, S. Waslander, T. Barfoot, and A. Schoellig, "Autotrack: A lightweight object detection and tracking system for the sae autodrive challenge," in *2019 16th Conference on Computer and Robot Vision (CRV)*, 2019, pp. 209–216.

[10] G. L. Plett, D. Zarzhitsky, and D. J. Pack, "Out-of-order sigma-point kalman filtering for target localization using cooperating unmanned aerial vehicles," Advances in Cooperative Control and Optimization, pp. 21–43, 2007.

[11] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1–2, pp. 83–97, Mar. 1955.

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2999–3007.

[13] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[14] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[15] S. Wu, N. Amenta, J. Zhou, S. Papais, and J. Kelly, "aUToLights: A robust multi-camera traffic light detection and tracking system," in *2023 20th Conference on Robots and Vision (CRV)*, 2023.

[16] U. D. of Transportation Federal Highway Administration (FHWA), *2009 mutcd with revisions 1, 2, and 3 incorporated, dated july 2022 (pdf)*.

[17] C. Han, Q. Zhao, S. Zhang, Y. Chen, Z. Zhang, and J. Yuan, *Yolopv2: Better, faster, stronger for panoptic driving perception*, 2022.

[18] F. Yu, H. Chen, X. Wang, *et al.*, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2636–2645.

[19] J. McBride, "Darpa urban challenge," May 2007.

[20] S. Thrun, M. Montemerlo, H. Dahlkamp, *et al.*, "Stanley: The robot that won the darpa grand challenge.," *J. Field Robotics*, vol. 23, pp. 661–692, Jan. 2006.

[21] J. Xu and S. Waslander, "HyperMODEST: Self-supervised 3d object detection with confidence score filtering," in *2023 20th Conference on Robots and Vision (CRV)*, 2023.