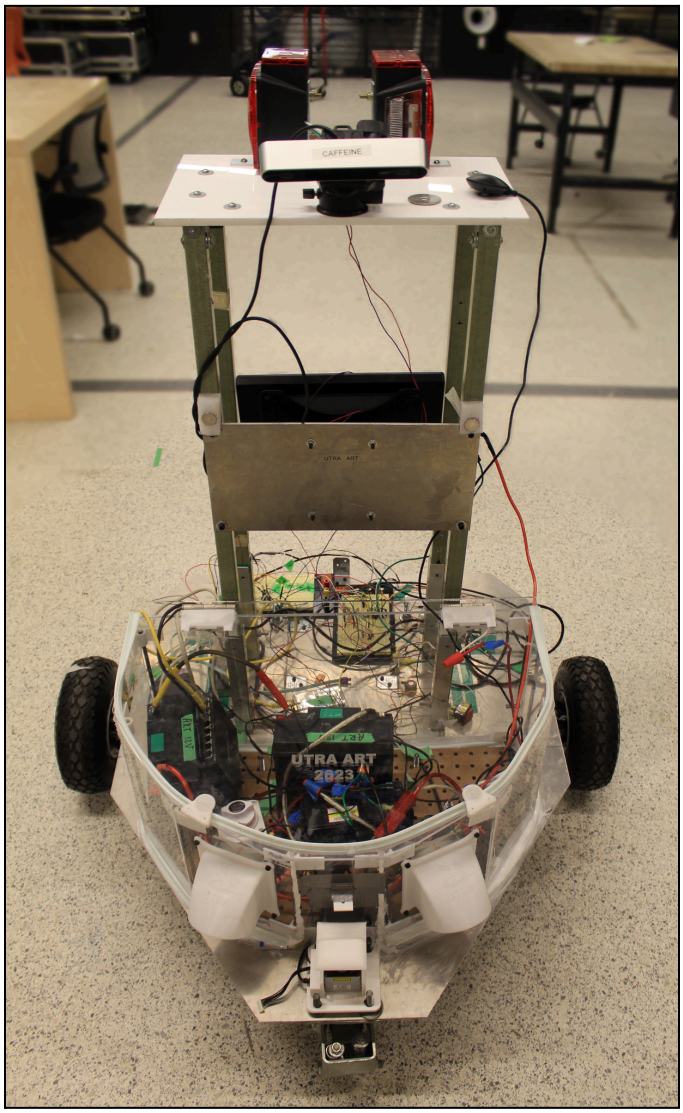
University of Toronto

UTRA Autonomous Rover Team (ART): Caffeine

May 15, 2023



ART IGVC Competition Team

Name	Email	Name	Email
Kirti Saxena	kirti.saxena@mail.utoronto.ca	Andrew Zhang	andrewzhang1000@gmail.com
Peter Shadrin	p.shadrin@mail.utoronto.ca	Ellen Shi	cyan.shi@mail.utoronto.ca
Tiger Luo	tigerluo03@gmail.com	Yifei Zhou	fei.zhou@mail.utoronto.ca
Ammar Vora	ammar.vora@mail.utoronto.ca	Grace Huan Liu	gracehuan.liu@mail.utoronto.ca
Vicky Huang	vhy.huang@mail.utoronto.ca	Angela Yu	angelaxh.yu@mail.utoronto.ca
Preet Mistry	preet.mistry@mail.utoronto.ca	Sebastian Levy	sebastian.levy@mail.utoronto.ca
Tracy Sun	tracy.sun@mail.utoronto.ca		

Statement of Integrity will be provided separately.

1. Conduct of design process, team identification and team organization

1.1. *Introduction*

The University of Toronto Robotics Association's (UTRA) Autonomous Rover Team (ART) was founded in 2008 by a group of students with a passion for robotics. The team has since grown to fill the shoes of an introductory robotics designed team at the University of Toronto. Our team is relatively large, with upward of 30 active members on a weekly basis. Our team's focus is to give students a platform to learn robotics concepts while aiming for a completed project.

1.2. *Organization*

Team Leadership			
Peter Shadrin	Project Manager	**Ivy Tan**	ROS Lead
Kirti Saxena	Project Manager	**Ammar Vora**	CV Lead
Ghamr Saeed	Embedded Lead	**Satvick Acharya**	CV Lead
Yifei Zhou	Embedded On-boarding Director	**Anushka Sethi**	Student Relations
Andrew Zhang	Mechanical Lead	**Laura Elshaer**	Student Relations
Samuel Eskandar	Mechanical On-boarding Director	**Heenal Patel**	Marketing Director
Sebastian Levy	Finance Director		

ART is divided into 4 subteams, each with their own team leads, onboarding managers, and general members. They are: Mechanical, Embedded / Electrical, Computer Vision, and ROS. Team leads are responsible for leading work sessions for each subteam. Onboarding managers prepare introductory workshops to give new members a foundation in concepts that are likely new to them. General members (in addition to team leads and onboarding managers) work on tasks assigned by the team leads with the goal of preparing a rover for competition.

1.3. *Design Assumptions and design process*

Early in the season, freedom was given to members to try things out, experiment with new ideas, and iterate on proposed solutions. Later in the season (January) we took what had worked best in the experimentation phase and developed it further into a final design that we then integrated with all the other parts of the rover design.

Some design considerations and assumptions our team employs:

- We are a team with a low budget and few sponsors. Preference is given to design decisions that make use of existing components or materials whenever possible. This sometimes results in creative, albeit unconventional, solutions.
- We are limited in members that are licensed to use our school's machine shop, and even more limited in availability to use it. As such, designs are influenced heavily by

a small subset of tools all members can use. We avoid lathes, mills, CNCs, and welding wherever possible.
- Our team is entirely extra-curricular. We lose nearly every member during midterm and final exam season, and have member loss after the school year ends.

Our design decisions take a lateral approach. Once a design problem is identified, typically by the team leads, solutions are brainstormed by both general members and team leads. A subteam will produce candidate solutions that are brought to a lead meeting. At this meeting, candidate solutions are proposed and other subteam leads are able to share their concerns with each design, specifically thinking how this design will affect each individual subteam. If a candidate solution appears to satisfy every lead's constraints, a team member is assigned to the design and a prototype is made. In hardware systems this typically means a CAD model, in software, a simulation. If the design appears to work, the member in charge follows through while working with leads to implement a final version of the design.

2. Effective innovations in your vehicle design

2.1. Innovative concept(s) from other vehicles designed into our vehicle

Concepts that inspired our robot's design were a tall tower for the camera to increase the field of view accessible to our lane detection and visual odometry algorithm. We have also migrated to using a laptop instead of a Jetson, as inspired by other teams at last year's competition.

2.2. *Innovative technology applied to our vehicle*

2.2.1. Using a network switch for ROS distributed system communication

After about a year's worth of struggle with the ROS serial library to connect our Arduino to ROS effectively, we were able to figure out that computers on the same network can be very easily interfaced together through ROS, so we added a network switch and replaced our Arduinos with Raspberry Pis that now communicate between the main computer, motors, and motor encoders very quickly and effectively.

2.2.2. U-Net for lane segmentation

For lane marking segmentation, we used a custom model derived from U-Net. The U-Net model architecture is built with convolutional layers and features a contracting (encoding) left side, a symmetric expanding (decoding) right side and cross connections between the contracting and expanding layers.

We innovated by passing the U-Net model some classically derived features. Of the five input channels, the fourth and fifth input channels are Canny edge detections of the input image using a median filter on each quadrant of the input for thresholding boundaries.

2.2.3. Detection of ramp using distance difference via two LiDAR units

Ramp detection was done via two LiDAR units to detect a difference in distance from the robot's point of view. The expectation is that a ramp is the only obstacle that will generate a large enough distance difference to be detected by our software, given that the LiDARs are at two different heights. Thus, the cost of an expensive 3D LiDAR unit can be avoided by using two cheaper 2D units. This is described in greater detail in section 5.3.

2.2.4. *Generation of lane*

We rely on the Bresenham line algorithm to interpolate lines. Our inspiration to use this algorithm comes from the domain of computer graphics, as this algorithm is used in a part of the graphics pipeline to rasterize triangles. We believe the innovation here is taking concepts from an unrelated field and applying it to our problem domain by reframing the way we viewed our problem. This allowed us to take a 'cost-map' centric approach; we integrate all information about our world into our costmap in separate costmap layers.

2.2.5. *U.S. Digital E3 encoders*

Choosing these optical encoders allowed us to measure the actual angular speed of our wheels and conduct odometry.

2.2.6. *TV Display*

To assist in software visualization during course navigation, a TV display was installed on the tower of caffeine. The TV display allows users to see the computer display as caffeine runs and identify software issues. To mount the TV monitor, a custom bracket was fabricated to attach the TV securely.

3. Description of mechanical design

3.1. *Overview*

Much of the mechanical design for this year's rover was inherited from prior design team efforts. The mechanical team's focus this year was on improving the weather-proof enclosure of Caffeine and making new sensor mounts and covers for additional electronic components. Red trailer lights were added to ensure compliance with the IGVC 2023 rules concerning safety lights.

3.2. *Description of drive-by-wire kit*

The navigation stack controls the rover using a speed signal and a direction signal for each motor controller. The command linear and angular velocity is published by the central computer to a ROS topic. The topic is then interpreted by one of the Raspberry Pis which in turn sends signals to the motor controllers to execute the command.

Caffeine’s drive-by-wire kit implements a differential drive system. Steering is conducted by controlling the rotational direction of two independent east and west mounted motors. Our current motors were inherited from last year’s design and can achieve the competition maximum speed of 5 mph. The motors power and torque were also deemed to be sufficient.

Last year’s 300W brushless DC motor with a 16:1 planetary gearbox attached was used. This motor can output a speed of 188 rpm and 14.4 N-m torque. The mounting plates and shaft extensions for the motors from last year were also used. New sensor mounts were designed for the wheel encoders and 3D printed using polycarbonate material instead of PLA for additional rigidity.

3.3. *Suspension*

The suspension mechanism uses a pair of 4” coil springs and a 4-bar mechanism holding the motor and the wheel at a downward angle from the chassis of the rover. The spring provides a tension force, which allows the rover to absorb some shock as it traverses bumps in the course.

3.4. *Weather proofing*

Weather proofing of our robot is split into two main subsystems: the main enclosure for caffeine's electronics systems, and additional covers for external sensors.

The main enclosure for our electronics systems was re-designed this year to increase space, improve accessibility, and improve ventilation. A new enclosure was cut and assembled from 1/4" acrylic sheets and sealed with caulking to prevent leaks. The sealed joints were water-tested to ensure that no leaks were present on the enclosure. The extra space was used to make a new arrangement of our electronics. Our enclosure also includes a lid to make our electronics more accessible without compromising water protection. Lastly, ventilation fans were also installed inside of the enclosure to cool the electronics. Vent duct covers were 3D printed and attached to the fans to ensure that rainwater would not enter the enclosure through the fans.

Peripheral sensors such as the camera and lidar had differing mount location and visibility needs. An additional sensor mount was created for our secondary LiDAR sensor so that it can continue operating in the rain.

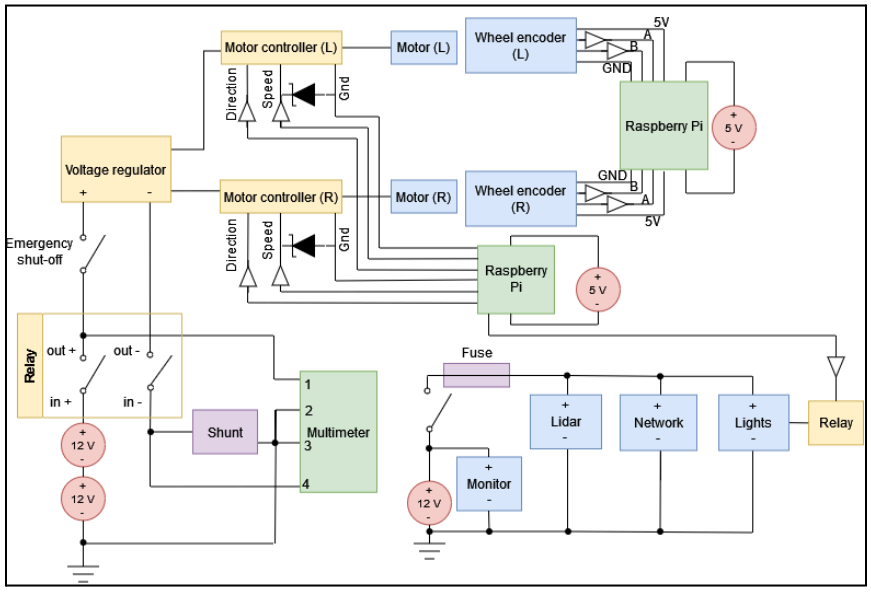
3.5. Safety Lights

Highly visible trailer lights were added on top of the tower mast of Caffeine as safety lights. These lights were selected because they could give a clear visual indication of when the rover is turned on. They will show a solid color when the rover is powered and turn to flashing when the rover is under autonomous mode.

4. Description of electronic and power design

4.1. Overview

Caffeine's circuit consists of three main branches. One operates at 12V on a lead acid battery to power the sensors, lights, and monitor. Another operates at 24V on two lead acid batteries to power the motors and motor controllers. The third consists of two Raspberry Pis, each powered by a 5V portable phone charger.



Caffeine electrical representation

4.2. *Power distribution system (capacity, max. run time, recharge rate, additional innovative concepts)*

Due to the current state of the rover, we have not been able to empirically measure its power consumption, max. run time, or recharge rate. Based on a lump sum of the power consumption estimates found on each component's datasheet, the operational power consumption of the rover is estimated to be 850 watts. This estimate yields approximately 15 minutes of runtime when the robot is paired with the 12 volt, 20 amp hour battery. The recharge rate of the system is simply the recharge rate of the battery which is 6 Amp hours at 15 volts.

4.3. *Electronics suite description including CPU and sensors system integration/feedback concepts*

Caffeine uses a laptop to do heavy computation. It is directly connected to the following list of sensors:

IMU	PhidgetSpatial 3/3/3 Precision 1044_1
GPS	BU-353S4
Lidars	Hokuyo URG-04LX and RPLidar A1
Stereo Camera	Zed Stereo Camera

List of sensor modules not including the encoders (all of which are connected directly to the computer.)

The sensor input signals are fed to the computer using USB and then processed by the computer vision and mapping stacks. The computer interfaces with 2 Raspberry Pis over a network switch. The role of the Pis is to communicate between the motor controller and the ROS stack, where one of them sends commands to the motors and the other sends encoder feedback from the motors to the computers.

Encoders	U.S. Digital E3 optical kit 500cpr

Encoder model: connected to the Raspberry Pis

4.4. *Safety devices and their integration into your system*

The system includes a red switch that will cut the power to the boost converter. The switch is 1.5 inches in diameter and is centered on the rear of the rover. The rover also features a wireless relay switch that will also disconnect the voltage regulator from the battery, finally, a push-and-twist rotary switch is used to physically turn the rover on and off during regular use. In addition to the required safety measures, a 3 Amp fuse is connected to the computer circuit to protect against any unexpected current surges.

In order to communicate between components that send 5 volt signals and the Raspberry Pis' GPIO safely, we use 74hc125 buffer ICs that shift the voltage of outputs up and shift the voltage of the inputs down between 3.3 and 5 volts. Using this IC allows us to safely communicate PWM signals that need relatively fast switching speeds.

5. Description of software strategy and mapping techniques

5.1. Overview

Our software team is broken into two distinct parts: CV and ROS. CV processes camera data to generate a representation for lane markings and potholes. ROS uses sensor data to localize, map, and navigate the environment. Information from all of our sensors are integrated into a single cost map which we use to navigate.

5.2. Obstacle detection and avoidance

5.2.1. Lane Marking Detection

Lane marking detection is a fundamental task which is performed by autonomous vehicles, and there are numerous methods that are outlined in literature suitable for this task. For our system, we developed a classical and deep learning approach, outlined below.

The classical and deep learning method both satisfy performance and accuracy requirements for our case. Since both of these methods are fully integrated with our system, we will choose to run only one at a time, based on the environment conditions present.

5.2.1.1. Classical Approach

We leveraged classical computer vision techniques, such as color spaces and color thresholding to implement a lane marking detector. Once an image is received, it is converted from the red, green, blue (RGB) color space to the hue, saturation, value (HSV) color space. Then, it is shrunk to 25% of its original size which efficiently removes the background noise from the image. The last step is to threshold the pixels in the image – with calibrated values for the lower and upper bounds – which produces a binary mask of the lane markings.

This simple, yet lightweight detector is high-speed, running at > 200 FPS with ~89% accuracy, and contributes to a lower latency for our entire system. Although this method is efficient, it lacks robustness compared to a deep learning approach.

(left) Original image, (right) output of the thresholding method

5.2.1.2. Deep Learning Approach

The purpose of using a deep learning approach to find lane-markings is to make a detector that is robust to environmental changes. Since this task is fundamentally an image segmentation task, the U-Net model, which is a state-of-the-art segmentation model, was used.

Typically, U-Net has a low inference speed (< 20 FPS) due to its large amount of Convolutional Neural Networks and network depth. To make our implementation faster and achieve > 60 FPS, we reduced the depth such that there are only 256

channels in the latent vector. The input to the model includes 4 channels: grayscale, edges, and inverse edges, and gradients. The edges are found using a Canny edge detector. A Sobel filter, applied to the grayscale image, is used to calculate the gradient channel. The hyperparameters (channels, image size, model depth), were thoroughly tuned to find the optimal results. Accuracy of the model was measured with intersection over union (IOU) of the output compared to the ground truth.

The model was trained on a dataset of 5442 images of real-world daytime and nighttime dashcam images, as well as 532 images collected from a previous competition. Approximately 60% of the dataset is composed of synthetic road data. In testing the model, we achieved 97% accuracy (as measured by the IOU) on normal roads and 75% on grass.

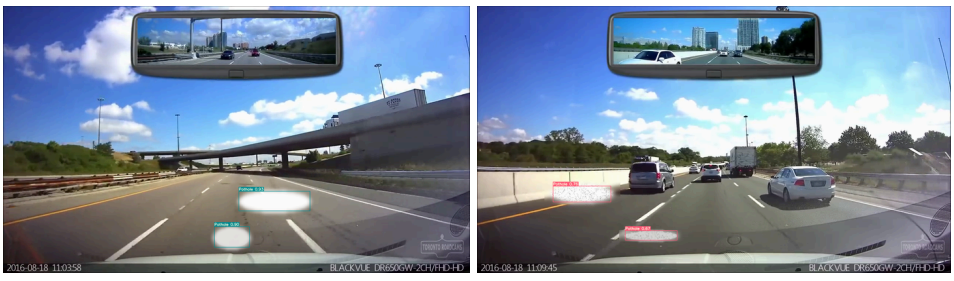
(left) Original image, (right) output of U-Net model

5.2.2. Pothole Detection

5.2.2.1 Deep Learning Approach

We detected potholes with the YOLOv4 Tiny model. This model was selected for its ability to bound objects with high accuracy in a single pass. This makes the YOLO model faster than other object detection algorithms, such as R-CNN, which use multiple stages. YOLO does not utilize a sliding window approach, which allows features to be extracted in context of their backgrounds, decreasing the number of false positives [1]. The YOLO model can generalize to a variety of environments, which increases the robustness of the model.

We used the YOLOv4 Tiny model since it can achieve 92% of the accuracy of the full model but with a 30% speed up [2]. The Tiny model is faster to train, uses less memory, and has higher inference speed, making it ideal for the real-time nature of the competition, where compute is limited. To further boost accuracy and decrease training time, we transfer learned using the weights provided by Darknet [3].



Example of generated pothole data

5.2.3. Physical Obstacle Avoidance

To detect physical obstacles, we rely on a Hokuyo LiDAR (URG-04LX). We chose to do this over a computer vision based approach because LiDARs are known to be highly accurate

and are a reasonably fast sensor. In particular, with a computer vision based approach, we would have to rely on reprojecting the points to 3D, while relying on a LiDAR-based approach would allow us to use the data directly. We also chose to use a 2D planar LiDAR over a 3D LiDAR. This is an inexpensive solution, which fits well given our budget. Due to the relative sparsity of the data, it was also computationally feasible for us to avoid filtering the point cloud. We simply directly add points detected by our lidar to our costmap. We note that this makes the assumption that anything the lidar detects is an obstacle, but we believe this assumption is reasonable as regardless of the obstacle, we do not want our robot to hit anything. This also reduced computation expense, as we didn't have to spend time running point cloud classification. One notable disadvantage of this approach is that we are unable to distinguish between significant inclines and physical obstacles based on the single LiDAR. This is discussed later.

5.3. Software strategy and path planning

For path planning, we rely on 4 key packages, move_base, costmap_2d, navfn and base_local_planner. Move_base serves as our high level interface, and is used to set goals and have our robot reach them. We do this by using the costmap_2d package, which creates a local representation of the robot's environment. This way, we can avoid any obstacles that appear as we navigate. The costmap_2d also integrates information from our camera into obstacle detection. The segmentation mask outputted by lane detection is reprojected into a 3d representation that is then converted to an equivalent LiDAR message, essentially creating virtual walls within the LiDAR. This can then be fed into our costmap and any other mapping packages. We assume that the output is discrete, hence to create an interpolation between potentially missing points, we rely on linear interpolation via the Bresenham algorithm. Afterwards, we use navfn and base_local_planner to generate a global and local plan respectively that we follow.

For the ramp, we are limited by the 2d LiDAR and need a reliable method of distinguishing it from actual obstacles. Using our existing stereo camera we were unable to detect the ramp consistently. We chose to mount a second LiDAR above the primary one and leverage the distance difference resulting from the incline of the ramp. The lower LiDAR would detect an obstacle some distance closer than the upper LiDAR, informing us that the obstacle is a ramp. We then publish a new message with the range detecting the ramp removed and use that for mapping and obstacle detection so the robot does not avoid the ramp. This method was simple but consistent, and cost-efficient as we already had a second LiDAR. We computed the optimal distance threshold to trigger filtering using the fact that the ramp doesn’t exceed a certain incline and that no other obstacle has an irregular shape. During this time, we continue using the CV pipeline to find the lanes on the edges of the ramp to ensure we remain centered.

5.4. Map generation

For map generation, we rely on the gmapping package. We chose to use this because all the information we need can be represented by a 2d plane, discarding the need for more computationally expensive mapping packages. Additionally, the features to be added to the map primarily depend on the LiDAR messages from the physical sensor and the transformed messages from lane detection (as discussed in section 5.3).

5.5. Goal selection and path generation

To set goals, we enter the GPS coordinates given to us into an ordered json file, with additional instructions like heading and laps to complete. Then, we send the transformed pose to a move_base goal. Move_base handles the path generation based on the map and costmap_2d. Afterwards, we query our GPS to see if we reached the point or not. If

we reach it, we look at the next point and set it as our goal. We include heading direction as a parameter in case we are to complete the course backwards.

To track the robot's current position, odometry is generated by fusing data from the wheel encoders, IMU, GPS data and visual odometry using the robot_localization package [4]. Visual odometry is generated using a stereo odometry node from the rtabmap_odom package, using ZED camera data [5]. The ekf_global and ekf_local nodes implement extended Kalman filter state estimation to output global and local odometry, respectively [4].

Prior to fusing GPS data using the ekf_global node, the GPS data is entered into the navsat_transform_node with an initial orientation and heading given by the IMU and its magnetometer to generate a transform from UTM (Universal Transverse Mercator) coordinates into cartesian [4]. This is necessary since our map and odometry is in cartesian coordinates.

5.6. Additional creative concepts

To decrease the computational load on the laptop, we split tasks between the two Raspberry Pis' as well. One handles motor control and sensor processing, while the other computes odometry and localization. Since the laptop and Raspberry Pis are connected with a network switch, information and the same topics are shared.

6. Description of failure modes, failure points and resolutions

6.1. Vehicle failure modes (software, mapping, etc) and resolutions

6.1.1. Incorrect Lane Marking and Pothole Detections

Failure to detect lane lines or detecting phantom line markings can lead to the rover making illegal crossings or becoming stuck or significantly slowed in a box of phantom lane lines.

Failure to detect potholes or detecting phantom potholes leads to failure similar to the aforementioned lane line detection failure. The rover may go into a pothole or become hemmed in by phantom potholes, slowing or stopping the rover's progress.

We mitigate the above failure risk of incorrect lane makings by preserving fit parameters of previous detections. The goal of using U-Net is that it is compact enough to make multiple detections of frames in a second. If one detection of the lanes yields inconclusive results, the last successful detection's fit parameters will be used to approximately extrapolate our present lanes.

6.1.2. Ramp Assumptions Breaking

If the prior assumptions we place on the ramps prove to be false, like another obstacle can trigger LiDAR filtering (see section 5.5), due to our limited sensor stack, the ramp could prove to be a challenge for us. As an emergency backup, we use the fact that the ramp exists between the 2nd and 3rd waypoint and purposefully ignore the entire lidar data for this portion of traversal. This is, however, a last resort as we lose other valuable data. We have been unable to address this failure point completely due to time and human resource constraints.

6.2. *Vehicle failure points (electronic, electrical, mechanical, structural, etc) and resolutions*

6.2.1. *Nuts coming loose with vibrations*

All nuts and bolts are secured with washers to evenly distribute the force over a larger surface area. Nuts and bolts securing critical hardware, such as structural chassis and motor components, are secured with lock washers to reduce the likelihood of loosening. If, during our testing, we find the lock washers are not sufficient to prevent loosening due to vibration, thread-lock will be applied. All attachments will be checked prior to competition.

6.2.2. *Battery drained*

We have redundant charged batteries ready for competition. We will bring our battery chargers as well to charge in between runs.

We also distribute our power consumption over several battery sources where possible: the Raspberry Pis run on independent power banks and the computer runs on its own battery.

6.2.3. *Vehicle tipping on incline*

Caffeine was designed keeping in mind a low and evenly distributed center of mass (COM). While the sensor stand is quite tall, it is made of lightweight materials. The majority of the rover weight is in the motor, batteries, and chassis, which we ensured were mounted as low as possible, resulting in a low COM. Rigorous testing was conducted to ensure the vehicle will not tip on the maximum 8.53° (15% gradient).

6.3. *All failure prevention strategy*

Our team mitigates error by extensively testing all systems in simulation and on validation data. Parts that are known to die (batteries, fuses) we will bring extras of to competition and design their placement on the rover to be easily accessible.

We also will bring a full tool set with all required tool sizes and extra fasteners and raw materials. This will help mitigate the risk of something unforeseen going wrong during transport or qualification having a lasting effect on our ability to compete.

6.4. *Testing*

6.4.1. *Electrical and electronic component testing*

Incremental testing was done for each electronic component and sub-system to make sure that it functions correctly before integrating it into the rover. Most initial testing was done by attempting to operate an individual component. All voltage levels were checked prior to plugging in any component to reduce the risk of accidental overloads. During the testing process, a switch was wired into every circuit to ensure power remained off until the wiring was complete.

To test the speed and the accuracy of movement commands, we used the current wheel encoders as well as an independent tachometer and made sure that the speeds we were getting from both matched the expected speed.

6.5. Vehicle safety design concepts

6.5.1. The vehicle was designed to comply with the safety measures of the competition. Electrically, safety components are used throughout the rover to give easy access to shut down in case of an emergency or other unexpected behaviour. An emergency cutoff switch is accessibly located on the back of the rover that cuts power to the motors. A remote relay achieves the same result from over 800 ft away. We use fuses in our control network to protect our electronics from unwanted power surges. Caffeine is a large and powerful rover, as such if either motor experiences an unexpected change in velocity, or the IMU experiences a rapid acceleration, we cut power to the motors. This is to handle the case of an unseen collision.

Even though the motors are capable of moving the vehicle faster than the competition speed limit of 5 Mph, our boost converter limits its output current to restrict the vehicle to move at 4.5 miles per hour. We also connected zener diodes with a breakdown region starting at ~4.5 volts to the inputs of the motor controllers to ensure that the controllers will never receive a higher voltage than what the competition allows.

7. Simulations employed

7.1. Simulations in virtual environment

7.1.1. Data Augmentation and Generation

To increase robustness, we aim to account for as many scenarios and conditions that can be captured by our cameras as possible. We do this by modifying and expanding our dataset by augmenting existing data and creating artificial data.

For data augmentation, we apply a set of simple transformations to both the image and the labels. We added noise to the image and applied a Gaussian blur. We added transforms to account for the various positions the vehicle may take: these include rotation, shear, reflection, and resizing. We also included transforms to account for various weather and lighting conditions: manipulating the three color channels of our images or the hue-saturation-and-luminosity to simulate a cloudy day. More complex augmentations involved simulating water droplets on the lens by localized, shaped blurring and simulating shadows on the course, by applying local darkening to specific areas of the image. Augmenting existing labels allows us to reuse our labels.

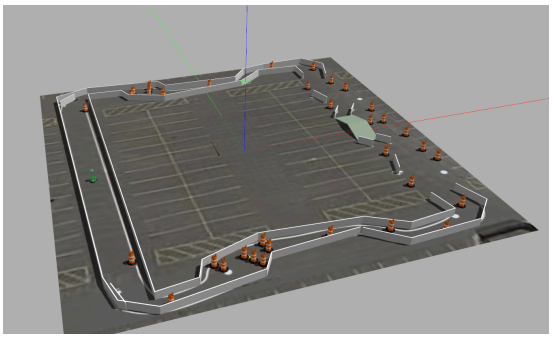
For the pothole data, we generated synthetic pothole data by adding up to four simulated, non-overlapping, randomly generated potholes to our real-world dashcam images. We modeled potholes as white ellipses, similar to what is described in the rules. We add noise to the pothole (such as salt-and-pepper noise, streaks, shadows) to better represent real-world data. We also match the color of the pothole to the asphalt— darker asphalt likely indicates a shadow, so a darker pothole would be required. Generating the data artificially allows us to create the labels automatically.

7.1.2. Simulated World

For robot testing, we leveraged the Gazebo simulation,a standard practice employed by the ROS community. In order to create an accurate simulation, we first created a URDF (Unified Robot Description Format) model of our robot based on the final mechanical CAD assembly. After this was done, we simulated all of our sensors using Gazebo plugins with gaussian noise to simulate sensor noise.

For the world, we created a model of the competition field in Gazebo. Our world allows us to test individual features without being blocked on other features being added. For

example, we could test our navigation without the deep learning model since we had the walls instead. A good simulation also allowed us to test software updates independently from robot hardware which enabled our mechanical and embedded teams to work simultaneously on the robot.



Our simulated course for IGVC

7.2. Theoretical concepts in simulations

Our simulation stack simulates all aspects of the auto-nav challenge. Sensor readings, obstacles, inclines, sensor noise, etc. We simulate all aspects of our mapping and navigation stacks.

To assess the accuracy of the local, global, and visual odometry, a ground truth plugin (p3d_base_controller) was added this year to the Gazebo simulation [6]. This plugin gives the current coordinates of the robot in the world frame and were then transformed into the same frame that the robot odometry uses.

8. Performance Testing to Date

8.1. Component testing, system and subsystem testing, etc.

8.1.1. Mechanical verification

Since much of the design is inherited, we mainly relied on the design decisions made by previous members of ART. Thus far, the mechanical design has been verified on SolidWorks CAD models and using FEA analysis'. Caffeine has also proven to be structurally sound and stable in our own weight bearing and tipping tests. The main enclosure was verified by spraying water at vulnerable areas (seams between panels, wiring openings, etc) with the electronic internals temporarily removed, and checking for leaks.

8.1.2. Electrical stress testing

Although the entire system is not yet ready to be stress tested, individual components were tested gradually and upgraded as needed. After a circuit redesign to facilitate competition requirements and new motors, several of the previous year's components were re-specced and replaced. These include the remote relay, motors, motor controllers, boost converter, and battery.

All switches operating were tested on the highest motor speed for around five minutes at a time between 2 to 3 times in a row with no issues detected. The motors, motor controllers and boost converters were tested in the same way. The controllers were tested for low voltage protection at 5 volts and 12 volts. The test concluded that they would not operate at such low voltages, but even at a sustained load for 5 to 10 minutes,

the controllers were not damaged. This test was repeated several times with similar results.

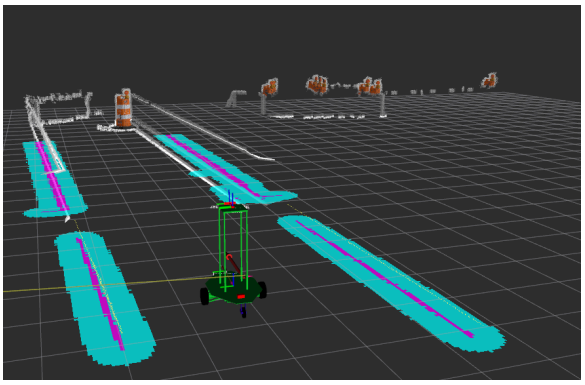
8.1.3. Lane and pothole detection

We collected a separate data set of simulated “competition” data to see how effective the model is. This is kept separate from the training set.We also verified our model on competition video from a previous year, yielding good results.

Model performance on previous year IGVC video. Trained on concrete

8.1.4. Mapping

To test mapping, we relied on our simulated world to gauge performance. We specifically looked at how long it took to generate a map, accuracy compared to the gazebo world, and loop closing ability. We haven't performed accuracy tests over a long period of time yet, but we are planning on doing this with the time leading to the competition. We also haven’t tested the viability of using LiDAR converted lane detection messages with other mapping packages. It is possible that we will change the package we use from the results.



Caffeine mapping our simulation world

8.1.5. Navigation

To test navigation, we mainly changed the spawn point of our robot and tested to see if we could complete a particular task. For example, to test no man's land navigation, we changed the spawn point to be at the first waypoint, then we tried seeing if we could reach the last waypoint using our GPS navigation pipeline. For navigation, as we didn't have CV integrated for the majority of our testing, we relied on having physical walls in the place of the lanes. This lets us test our navigation performance independent of CV's performance. We tested consistency by running the auto-navigation repeatedly to identify problem spots that stalled or resulted in failure.

8.1.6. Ramp tests

To test the ramp in simulation, we simply changed the spawn point of our robot in front of the ramp. Doing so, we can do rapid testing of our algorithm's viability and make adjustments. We then test our algorithms to see if we can remove the portion of the LiDAR reading containing the ramp, which is reflected in the sensor visualization and the costmap. Lastly, we test if the robot can navigate over the ramp without placing additional obstacles starting from the beginning of the course.

9. Initial Performance Assessments

9.1. How our vehicle is performing to date

At the time of writing this report nearly all of Caffeine's individual components have been demonstrated to work. Our CV lane detection and pothole detection works on validation data. The ROS navigation stacks work in our simulated gazebo world and local, global and visual odometry has been checked against ground truth to validate their accuracies. The mechanical design has been validated as far as we can without full integration. The electrical systems have all been powered on and tested individually, with subsystems having been tested as a whole. As of currently, we have confirmed the viability of the dual Raspberry Pi setup and tested the rover using the teleoperation stack. We do not yet know the viability of the LiDAR transform from lane detections and will test it soon.

We are working overtime integrating our systems and realize we are cutting it tight to competition. We hope to get all systems integrated two weeks before competition so we can test in real world conditions. This will be a lesson learned for us to pass on to future teams, to start systems integration earlier.

10. Closing Thoughts

The objective of UTRA ART was first and foremost to provide a hands-on learning environment for our members, providing robotics experience by working as a team to build upon the foundation of previous years' work. This year we have significantly overhauled the existing design of Caffeine, adding a custom waterproof electronics enclosure, external screen, and replacing the TX2 Jetson and Arduino with a laptop computer and two Raspberry Pi SBCs. On the software end, we added SLAM, additional sources of odometry, sped up the CV pipeline, split computation tasks, improved integration of lane segmentation with ROS mapping packages, and implemented more reliable methods of ramp detection.

Our focus this year has been getting all systems working and integrated into a working rover platform so that future years can devote more time implementing new innovations and optimizations.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *arXiv.org*, 09-May-2016. [Online]. Available: https://arxiv.org/abs/1506.02640. [Accessed: 09-May-2022].

[2] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal Speed and accuracy of object detection,” *arXiv.org*, 23-Apr-2020. [Online]. Available: https://arxiv.org/abs/2004.10934. [Accessed: 09-May-2022].

[3] A. Bochkovskiy, “Yolo v4, v3 and v2 for Windows and Linux,” *darknet*. [Online]. Available: https://github.com/AlexeyAB/darknet. [Accessed: 09-May-2022].

[4] T. Moore, "robot_localization 2.7.4 documentation," 2016. [Online]. Available: http://docs.ros.org/en/noetic/api/robot_localization/html/index.html. [Accessed 15 May 2023].

[5] M. Labbe, "rtabmap_odom," Open Robotics, 19 April 2023. [Online]. Available: http://wiki.ros.org/rtabmap_odom#stereo_odometry. [Accessed 15 May 2023].

[6] Open Source Robotics Foundation, "Gazebo plugins in ROS," Open Source Robotics Foundation, 2014. [Online]. Available: https://classic.gazebosim.org/tutorials?tut=ros_gzplugins. [Accessed 15 May 2023].