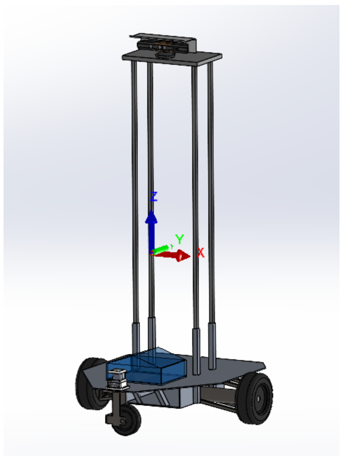
University of Toronto

UTRA Autonomous Rover Team (ART): Caffeine



May 15, 2022

Spencer Teetaert, Team Lead; spencer.teetaert@mail.utoronto.ca

ART IGVC Competition Team

Ammar Vora	ammar.vora@mail.utoronto.ca
Katrina Meng	katrina.meng@mail.utoronto.ca
Kirti Saxena	kirti.saxena@mail.utoronto.ca
Peter Shadrin	p.shadrin@mail.utoronto.ca
Satvick Acharya	satvick.acharya@mail.utoronto.ca
Spencer Teetaert	spencer.teetaert@mail.utoronto.ca
Vicky Huang	vhy.huang@mail.utoronto.ca

ART Contributing Members

Akriti Sharma • Anannay Sood • Andrew Wu • Andy Wang • Anthony Lem • Arthur Xu • Colin Smith • Daniel Li David Chu • Ethan Tang • Fares Soliman • Ghamr Saeed • Himanshu Sharma • Ishika Mittal • Ivy Tan • Jason Yuan Jeremy Bryce • Jimin Kwon • Kajanan Chinniah • Kanav Singla • Lily Kim • Luke Yang • Michael Ruan • Nurul Huda Zuki • Paula Santas • Sara Safadel • Shawn Wang • Tanmay Patel • Tiffany Costas • Yifei Zhou • Yuser Al Kudies

I certify that the design and engineering of the vehicle by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.

[illegible]

Signature

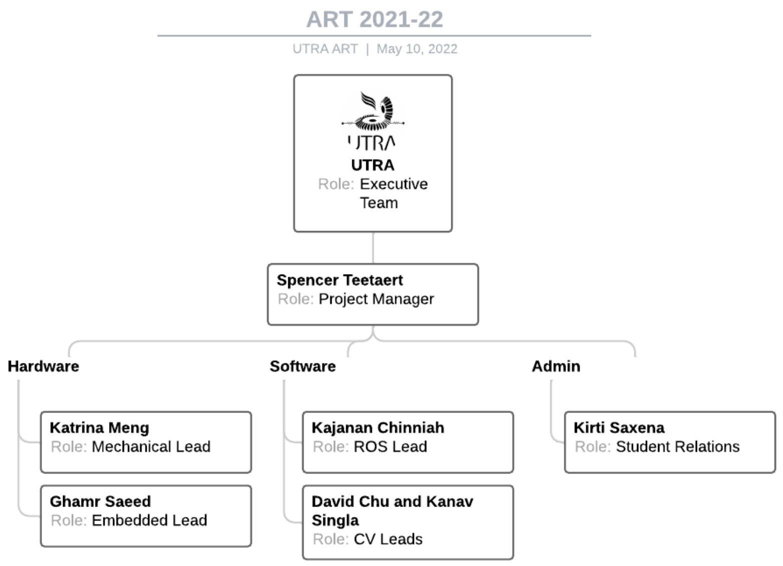
Florian Shkurti
Assistant Professor, Computer Science
Department of Computer Science, University of Toronto
Faculty Member, UofT Robotics Institute
Faculty Affiliate, Vector Institute

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. Students volunteer for the team outside of their coursework. This will be the first time our team has competed since 2012, largely due to a shift in focus brought on by this year's leadership team.

1.2. *Organization*



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 (September) 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, software 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 your vehicle*

2.1.1. *Dashcam height camera mast*

By extending the height of our camera mast to the height of a typical vehicle dash cam, we are able to more effectively use large scale lane detection datasets for training and pre-training our lane segmentation models.

2.2. *Innovative technology applied to your vehicle*

2.2.1. *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.

2.2.2. *Detection of ramp using ultra low-cost map prior*

For ramp detection, we use what is provided about the course, mainly, that there are no physical obstacles between the two waypoints on either side of the ramp. By using a prior of the competition ground, we remove the need to generate depth maps of the environment, saving a large amount of compute time. This also enables us to use a simple planar LiDAR unit instead of an expensive 3D one.

2.2.3. *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.

2.2.4. *Electric circuit layers and zip-ties*

We use perforated hardboard suspended on lead screws to create physical separation between our high and low power circuits. This not only saves on space but reduces the chance of unexpected shorts. It also makes for fast and easy mounting of components.

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. Since we were working with an existing rover chassis, the mechanical team's focus was ensuring the rover, Caffeine, met rules and restrictions outlined in the IGVC 2022 rules. A payload area was installed using brackets to retain the cinderblock sized payload.

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 Jetson to a ROS topic. The topic is then interpreted by the Arduino which in turn sends signals to the motor controllers to execute the command.

Caffeine’s drive-by-wire kit implements a two-wheel skid steering system. Steering is conducted by controlling the rotational direction of two independent east and west mounted motors. Our team selected new motors and designed the retrofitting process to the inherited chassis. The new motors are capable of achieving the competition maximum of 5 mph.

When selecting new motors, the following inputs were considered: weight, minimum and maximum speed, maximum incline, existing battery voltage and system efficiency. The required torque was calculated to be 11.9 N-m with a required power of 209.35W. .35W. Given a wheel size of 5” radius, the rpm required to drive wheels at 5 mph is 168.07 rpm (17.6 rad/sec).

The available space to directly mount a motor to a wheel is approximately 9.25”. Based on the required length, rpm, torque, and power of the motor, a 300W brushless DC motor with a 16:1 planetary gearbox attached was selected. This motor will output a speed of 188 rpm and 14.4 N-m torque. This was higher than was calculated, but it allows for some room for error (safety factor of 1.12) to account for unmodeled factors in our calculations. Mounting plates and shaft extensions for the new motors had to be designed and manufactured to ensure a good fit on Caffeine. The shaft extender incorporates a key and friction fit. The accompanying motor controllers were mounted in a compartment in the underside of the rover.

3.3. Suspension

Suspension was implemented in the form of 4” coil springs. A 4-bar mechanism was constructed, 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 was split into two main categories; the main electronics enclosure, and any peripheral sensors which require their own weather proofing and visibility requirements.

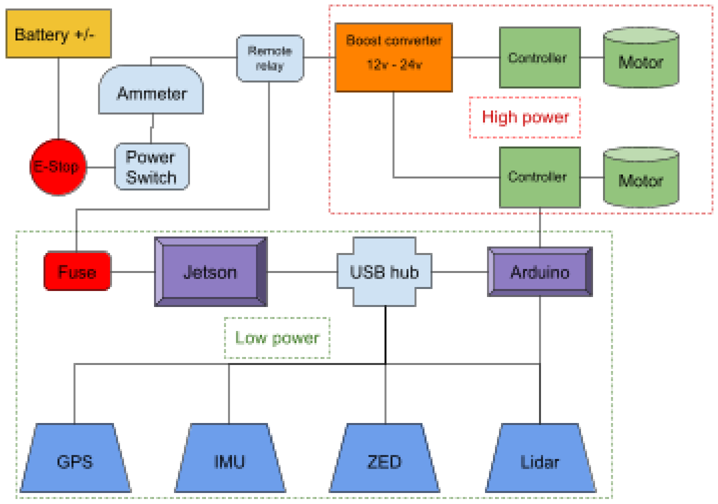
The main electronics enclosure houses the batteries, jetson board, wiring and other electronic components. The team decided to house the electronics in an inverted, clear plastic container. This provides separation from the elements, while maintaining visibility in case of an electronics malfunction and ease of access. Another consideration was the contact between the electronic parts, particularly the jetson boards, and the batteries. In order to create some separation, we have built shelf spaces inside the enclosure out of fiberboard material to avoid direct contact as much as possible.

Peripheral sensors such as the camera and lidar had differing mount location and visibility needs. The mounts were designed ensuring the sensors were secure but could be easily removed from the rover for testing. In most cases, weather proofing was done by designing and 3D printing a “canopy” which allowed visibility, but would direct water flow away from the electrical components.

4. Description of electronic and power design

4.1. *Overview*

Caffeine's circuit consists of two main branches. One branch operates at 12V the motors and motor controllers and the other powers the computer and sensors. The entire system is powered by a lead acid battery.



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 use a Jetson TX-2 Development Kit to do most of the computations, it is directly connected to the following list of sensors:

IMU	PhidgetSpatial 3/3/3 Precision 1044_1
GPS	BU-353S4
Lidar	Hokuyo URG-04LX

Stereo Camera	Zed Stereo Camera

Full list of sensor modules (all of which are connected directly to the Jetson.)

The sensor input signals are fed to the Jetson using USB and then processed by the computer vision and mapping stacks. The Jetson also interfaces with an Arduino that is used to send signals to the motor controllers which in turn send back speed signals creating a feedback loop.

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 boost converter 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 Jetson to protect against any unexpected current surges.

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

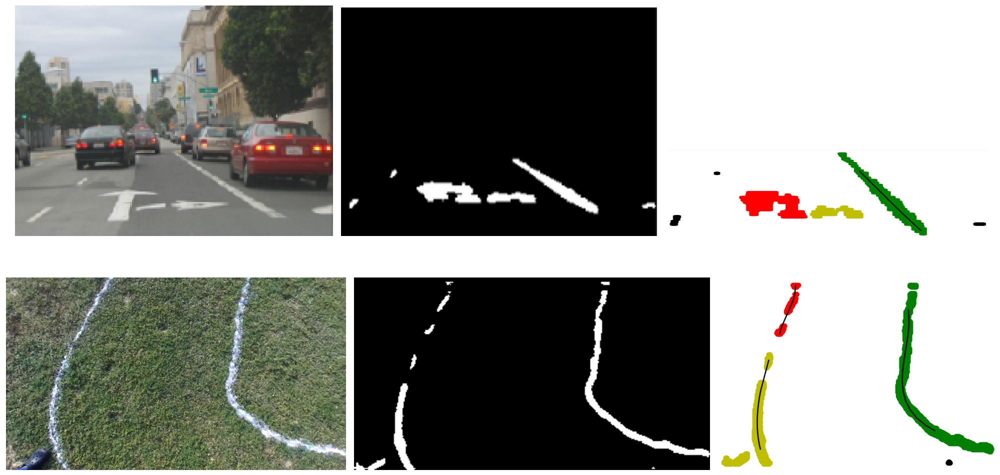
Detecting lane markings is fundamentally an image segmentation task. This led us to use the U-Net model, which is a deep learning architecture that excels at image segmentation tasks.

For our U-Net model, we have five channels of inputs and one channel of output. The first three channels of inputs are the RGB channels of the colour image. 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. The output of our U-Net model is a two-dimensional image of boolean values denoting which pixels are part of a line marking. Accuracy of the model was measured with intersection over union (IOU) of the output compared to the hand-labeled oracle values.

We chose to use a deep learning approach instead of a classical segmentation approach because of its greater robustness to unknown conditions. Weather, cloud coverage, lane position in frame, and camera angle all greatly affected the performance of our team's classical approach from previous years.

The U-Net model beat out other deep learning models of comparable size for accuracy. Having a light weight model is a requirement for us as we are running the full rover software suite on the Jetson board.

We trained our U-Net model on a dataset of 5442 images of real-world daytime and nighttime dashcam images, synthetic road data, and lanes on grass. Approximately 60% of the dataset is composed of synthetic road data. In testing the model, we achieved 90% accuracy (as measured by the IOU) on normal roads and 75% on grass. The performance of U-Net is visually confirmed to work better than our team's existing classical approach.



Input, corresponding output of UNET segmentation model, and fitting of output (the lanes with an appropriate fit are drawn in black)

5.2.2. *Pothole Detection*

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. 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 attempt to have our robot reach those goals. During navigation, naturally, we must update the obstacles that the robot views. To do this, we use the costmap_2d package that has a representation of the robot's environment. This way, we can avoid any obstacles that appear as we navigate. The costmap_2d also serves as our method of integrating information from our camera onto our costmap. We do this by reprojecting the segmentation mask that is output into a 3d representation that is then added to our costmap. We assume that the output is discrete, hence to create an interpolate 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.

5.4. Map generation

For map generation, we rely on the rtabmap package. We chose to use this over using a 2d method, because despite the higher computational expense, we note that we can extract more information, such as potentially the ramp location

5.5. Goal selection and path generation

During no man's land, we no longer can rely on our sensors. Instead, what we do is represent the GPS coordinates given to us beforehand in an ordered json file. Then, we send a move_base goal. 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.

For the ramp, due to the limitation of using a 2d lidar, we need a reliable method of detecting it. Using our existing stereo camera we were unable to detect the ramp consistently. We chose to use the fact that the ramp is between the 2nd and 3rd waypoints as a prior. We note that between the 2nd and 3rd waypoint, all that exists is the lane and the ramp. Therefore, we chose to ignore the lidar data for this portion of traversal, and just drive towards the 3rd waypoint, treating the lidar as if it was non-existent. During this time, we continue using the CV pipeline to find the lanes on the edges of the ramp to ensure we remain centered.

5.6. Additional creative concepts

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 (more precisely “false positives”) 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. Manual Override

There may be situations during autonomous control of the rover that are unrealistic for us to prepare for autonomously. In situations where an emergency power cutoff may not be the most appropriate reaction, a human in the loop override is used. For example, if while on the course another rover has lost control and is unable to be stopped, causing damage to other vehicles.

We use a twist mux that assigns priorities to different velocity sources. The highest priority is assigned to our teleop topic. This acts as a manual override for the rover control. Implementing a manual override allows us to protect our rover by moving it off the course (something that in autonomous mode it should not be doing). This manual control is also useful for transporting the rover on and off the course.

6.1.3. Failure to Localize

When the robot fails to reach its goal, or fails to localize correctly, our robot will start to turn in place to try relocalization. This is the default navigation recovery behavior and what we rely on to try relocalizing. We note that sometimes it's not possible for the robot to relocalize, and this proves to be a potential failure point.

6.1.4. Ramp Assumptions Breaking

If the prior assumptions we place on the ramps prove to be false, due to our limited sensor stack, the ramp could prove to be a challenge for us. We have been unable to address this failure point due to time and human resource constraints.

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

6.2.1. Electronic failure from water short

All wire connections are sealed using heat shrink and housed inside our waterproof electronics enclosure where possible. All connections coming into our waterproof enclosure come from beneath, with a water break point before any electronics.

6.2.2. 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.

6.2.3. Battery drained

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

6.2.4. Wheel wobble from loose shaft extension

The shaft extension uses a friction fit, keyed design. In order to minimize wobble, there is sufficient overlap between the existing motor shaft, and the extension piece. The extension shaft is machined on a lathe to ensure rotational symmetry. The friction fit nature should ensure both parts are concentric. Further testing is required.

6.2.5. 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. Due to the north-south asymmetry, the placement of the electronics and payload bay are such that the imbalance is minimized. 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. Where possible, we use low fidelity prototyping of physical conditions to test systems after being integrated. 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 (mechanical, electronic, simulations, in lab, real world, etc.)

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.

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 behavior. 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.

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.

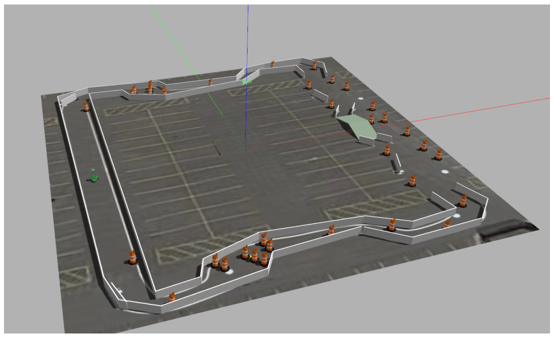
For data augmentation, we apply a set of simple transformations to both the image and the labels. We added noise to the image and applied a Gaussian blur. We added transforms to account for the various positions the vehicle may take: these include rotation, shear, reflection, and resizing. We also included transforms to account for various weather and lighting conditions: manipulating the three colour 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 colour 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 leverage the Gazebo simulation, which is one of the standards employed by the ROS community. In order to create an accurate simulation, we first built a URDF (Unified Robot Description Format) model of our robot. After the mechanical team had an initial CAD model, we converted the CAD model into a URDF model. After this was done, we simulated our sensors using Gazebo plugins. To simulate sensor noise, we simply used gaussian noise. We simulate all sensors that our robot has, including GPS. This allowed us to prototype our robot's various algorithms without having to set up a real robot or real obstacle course.

For the world, we used the png that was supplied from the IGVC manual for the basis of our map, and created a model of the world in Gazebo. Our world has the ability of turning off the ramp, adding walls in place of lanes, and more. We chose to allow for all these features to allow 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. Additionally, if ramp traversal wasn't working well, we could turn it off to test other parts of navigation. Adding on, due to constraints with COVID, it was important that we had a good simulation, as our software teams were unable to meet in person for the majority of the season.



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.

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 rely on the design decisions made by previous students. Thus far, the mechanical design has been verified on SolidWorks CAD models and using FEA analysis'. Caffeine has also proven to be structurally sound in our own weight bearing and tipping tests.

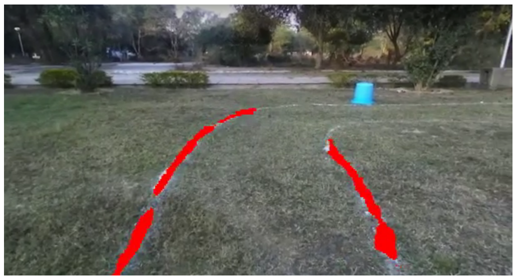
8.1.2. *Electrical stress testing*

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

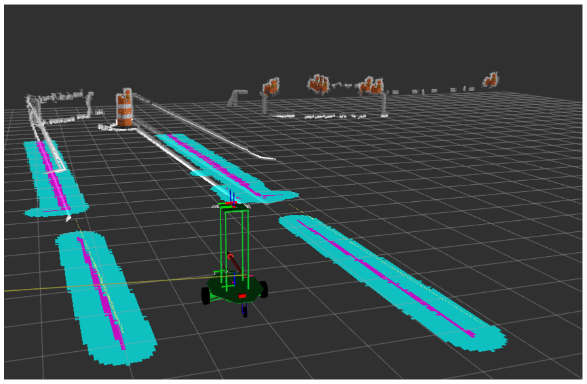
We collected a separate data set of simulated “competition” data to see how effective the model is. This is kept separate from the training set.We also verified our model on competition video from a previous year, yielding good results.



Model performance on previous year IGVC video. Trained on concrete

8.1.4. Mapping

To test mapping, we relied on our simulated world to see the performance. We specifically looked at how long it took to generate a map (e.g. the hertz updates would occur). We haven't performed any accuracy tests over a long period of time yet, but we are planning on doing this with the time leading to the competition. We note that we are using a well established ros package (rtabmap) to perform mapping, and so we anticipate that the algorithm will work well, however this remains to be an open question.



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.

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. Afterwards, we tested our algorithms to see if we can detect the ramp and get the robot to navigate over the ramp. By spawning in front of the robot, we could do rapid testing of whether we can go over the ramp or not with what we wanted to try.

9. Initial Performance Assessments

9.1. *How is your vehicle 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 mapping and navigation stacks work in our simulated gazebo world. The mechanical design has been validated as far as we can without full integration. The electrical systems (motor and sensor power, computer power, power readings, and required safety features) have all been powered on and tested individually, with subsystems having been tested as a whole.

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

Caffeine is at the end of the day a learning project for the members of ART. It is designed using a mix of existing design decisions from previous years and new improvements from our team. We integrated a brand new computer vision pipeline and redesigned mechanical, embedded, and ROS systems. 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.

Reference

[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].