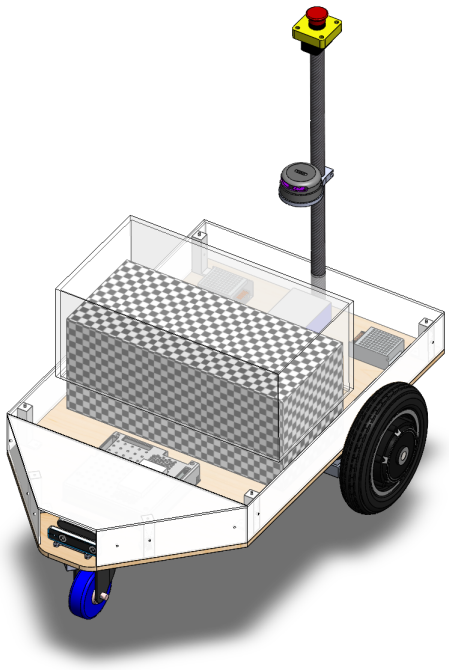
Autonomous Robotics Competition Club AT PENN STATE

Supreme Lion



May 17 2022

Ghanghoon Paik (gip5038@psu.edu)
Skanda Bharadwaj (ssb248@psu.edu)
Vidullan Surendran (vus133@psu.edu)

Faculty Advisor: Dr. Eric Johnson

"I certify that the design and engineering of the vehicle performed by the ARCC Team has been significant and equivalent to what might be awarded credit in a senior design course."

Faculty Advisor Signature: _____ Date: 05/18/2022

1 Introduction

The Autonomous Robotics Competition Club (ARCC) is a student organization at the Pennsylvania State University (University Park, PA, USA) that focuses on participating in autonomous robotics competitions. To date, the club has primarily focused on aerial vehicles and through IGVC we hope to develop expertise in tackling the unique challenges faced by ground vehicles. Our entry is a three-wheeled differential drive vehicle which relies primarily on computer vision to map the course and autonomously navigate it.

1.1 Team Organization

Table 1 lists the name, area of study, contribution, and the estimated number of hours invested by the team all of whom are currently pursuing their PhD. Due to the small team size, members were forced to contribute wherever possible which is reflected by the number of tasks taken on by each member.

Table 1: Team members and roles

Name	Major	Tasks	Hours
Ghanghoon Paik	Aerospace Engineering	Electronics Low-Level Controls Software Structures System Integration	173
Skanda Bharadwaj	Computer Science	Computer Vision Path Planning Localization	87
Vidullan Surendran	Aerospace Engineering	Electronics Localization Path Planning Power System Structures Software System Integration	169

1.2 Design Process

Due to the small team size, our design process was heavily derived from agile management techniques depicted in flowchart form in Figure 1. This meant breaking the project into sub-tasks which were ranked by importance and critical tasks that blocked progress on others if not completed were prioritized first. Quick iterations and a focus on prototyping and producing working solutions over structured management approaches was preferred.

The team heavily focused on software design as the challenges posed by this year's course relied heavily on obstacle detection and path planning rather than innovate vehicle hardware design;

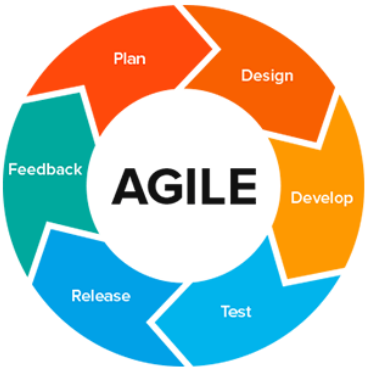


Figure 1: Iterative design process utilized by the team to develop our vehicle in under 2 months

especially when the maximum vehicle speed is capped at an average of 5mph. We utilized a simple differential drive configuration that would serve as our sensor platform which allowed us to leverage our expertise which lies in computer vision, optimization, dynamics, and autonomous robotics. This allowed us to meet the task deadline with limited personnel.

2 Innovations

2.1 Direct drive using brushless hub motors

Inspired by the widespread use of hub motors in personal electric mobility vehicles such as scooters and hoverboards, we drive our wheels directly using hub motors. Using mass produced motors also lowered vehicle cost compared to brushed geared motors used commonly in Robotics competitions which have limited suppliers and consequently much higher cost. In addition to brushless motors having lower friction compared to brushed designs, the lack of a gear box further reduces mechanical losses, vehicle mass, and points of mechanical failure further improving vehicle power efficiency.

2.2 Visual SLAM

Similar to TESLA motor car company, we rely purely on vision for obstacle detection, localization and mapping which eliminates the need for a costly high resolution LIDAR. The LIDAR used on the vehicle is a low cost LIDAR which is used for reactive collision avoidance and provided as a redundant sensor in case of vision pipeline failure.

2.3 Safety Through Design Choices

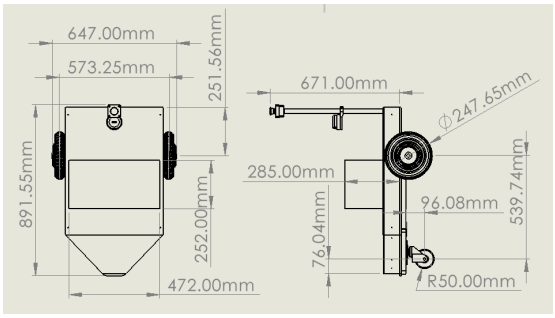
Unlike regular brushed motors which can be accidentally powered on in case of an electrical fault, brushless motors require a controller to constantly switch phases eliminating the motor spinning in case of a short to the power source or controller failure. We use two independent batteries to power the motors and the computational electronics. This ensures that the motors lose power before the computational stack preventing a runaway scenario.

2.4 Low Cost Modular Structure/Weatherproofing

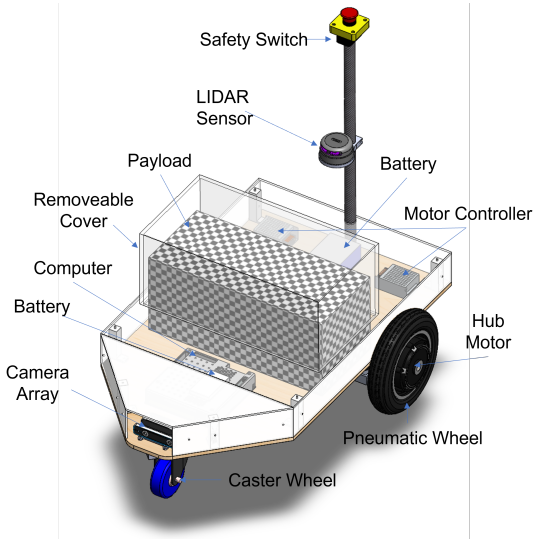
We constructed vehicle frame from commonly available 6061 square tube using a plywood board as the base. Non load bearing members were 3D printed using carbon fiber infused PLA and PETG allowing rapid prototyping and creation of parts that would be difficult to machine using subtractive manufacturing. FEA/CAD allowed us to confirm structural stability while minimizing weight. This platform is easily rebuilt and provides an accessible platform for robotics research.

In order to avoid component damage caused by light rain, our vehicle can be covered with plexiglass or foam-board. While plexiglass can be visually attractive it has a higher price tag. Foam-board, on the other hands is cost effective and easy of manufacture but weak against impact damage. The ability to easily switch between different types of materials allows the vehicle to be customized on the fly based on use case.

3 Mechanical Design



(a) Relevant dimensions of the vehicle. Drawing scale 1:20.



(b) CAD model of vehicle with weather covers made translucent for illustration

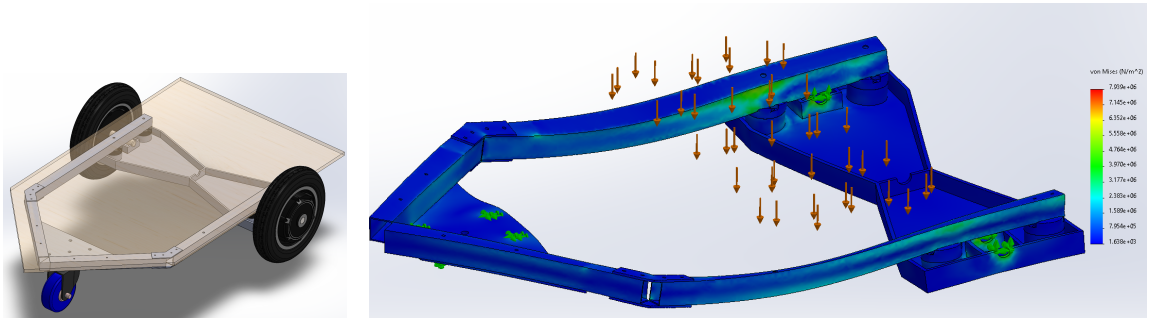
Figure 2

We selected a differential drive configuration as it simplifies path planning due to its ability to virtually turn in place. Having more than 2 driven wheels leads to skid-steering because the inner wheels traverse a smaller arc compared to the outer wheels. This is undesirable due to the loads placed on the drive motors. This year's competition being on asphalt meant much greater tyre grip further reducing the desirability of skid steering. These considerations led us towards a 3 wheeled robot with two independently driven wheels with a passive caster in front. The disadvantages of such a system includes instability during high speed turns and reduced load bearing capability, but these drawbacks are non-issues due to the restricted top speed and small payload capacity required

to complete the course.

We did not implement any suspension mechanism and rely on our pneumatic wheels to provide damping. In addition, a rubber damping pad is placed between the caster wheel and its attachment point to reduce the transmission of vibrations to the frame.

We weatherproofed the vehicle against a light drizzle by enclosing the components using panels made of Plexiglas/foam-board. The side panel seams are treated with silicon sealant to prevent water from dripping inside whereas the top panel is screwed onto the vehicle in case it needs to be removed to access the electronics. Since the payload is the tallest component, the height of the enclosed compartment was chosen based on the largest electronic component. The payload, which juts out of the top cover is then covered with a removable cover. A labelled CAD model of the vehicle is shown in Figure 2a where this payload cover is labelled. A drawing of the vehicle is shown in Figure 2b which calls out the relevant dimensions confirming that the competition requirement regarding vehicle size and payload dimensions have been met.



(a) Primary load bearing members of the vehicle

(b) FEA showing exaggerated deflection at 3G static loading condition

Figure 3

The load bearing members of the chassis are shown in Figure 3a which was constructed out of a repurposed A356 cast aluminium base and 6061 aluminium alloy 1-inch square tubes with a wall thickness of 1/20 inches. On top of this chassis, a wooden platform was laid to house the payload and electronics. We validated the structure using FEA. Figure 3b plots the Von-Mises stress distribution and visualizes the deflection of the frame under a static loading condition where the payload and the mass of the vehicle itself were applied normal to the vehicle (in the direction of gravity). Fixed constraints were applied at the wheel mounting locations. The maximum Von-Mises stress was $7.9e^6 N/m^2$ which was well below yield stress with a maximum deflection on the order of 0.1mm. This was identified as the worst case loading scenario, and as mass of the vehicle was not identified to be a critical design driver, we did not attempt to optimize the structure any further.

3.1 Mass and Cost Budget

The mass and cost budget is detailed in Table 2.

Table 2: Weight and cost breakdown

Component	Quantity	Weight (g)	Cost ($)	Total Weight (g)	Total Cost ($)
Jetson TX2	1	460	600	460	600
Pi Pico	1	3	6	3	6
LIDAR	1	3	600	3	600
Stereo Camera	1	190	300	190	300
GPS	1	6	70	6	70
Hall Sensor	6	0.1	2.2	0.6	13.2
IMU	1	10	150	10	150
Radio Receiver	1	4.5	26	4.5	26
Voltage Regulator	2	2	4.5	4	9
Hub Motor	2	1500	45	3000	90
ESC	2	170	15	340	30
Level Shifter	2	0.1	6	0.2	12
USB Hub	1	42	20	42	20
E-Stop button	1	45	8	45	8
LiPo Battery	1	805	65	805	65
Li-Ion Battery	1	950	60	950	60
			Subtotal	5863.3	2059.2
Structure	1	6400	80	6400	80
Caster Wheel	1	650	12	650	12
Payload	1	9070	-	9070	-
Miscellaneous	-	500	100	500	500
			Total	22483.3	2651.2

4 Drivetrain

Brushed motors and geared systems have inefficiencies arising from mechanical losses with average efficiency in the range of 75%-80% compared to brushless motors which are typically 85%-90% efficient. One of our innovations is the use of hub motors for direct drive of the wheels which provides a space efficient drive train without the extra weight and mechanical complexity of a gearbox. The hub motor we installed was originally designed for a hoverboard. Each motor is rated to 36V and can generate approximately 45Nm of torque. Since the designed specification of the motor exceeds our needs, we used a dedicated micro-controller to enforce a hard limit on the RPM of motors in order to meet the rules. Our motor controller also operates in low speed mode further reducing and limiting wheel RPM.

5 Electronics and Power System

The architecture of our system is shown in Figure 4 which lists the components, shows how they communicate with each other, and the bus voltage. The sensors use a combination of USB, I2C, and Serial protocols to communicate with the main computer which in turns sends control commands

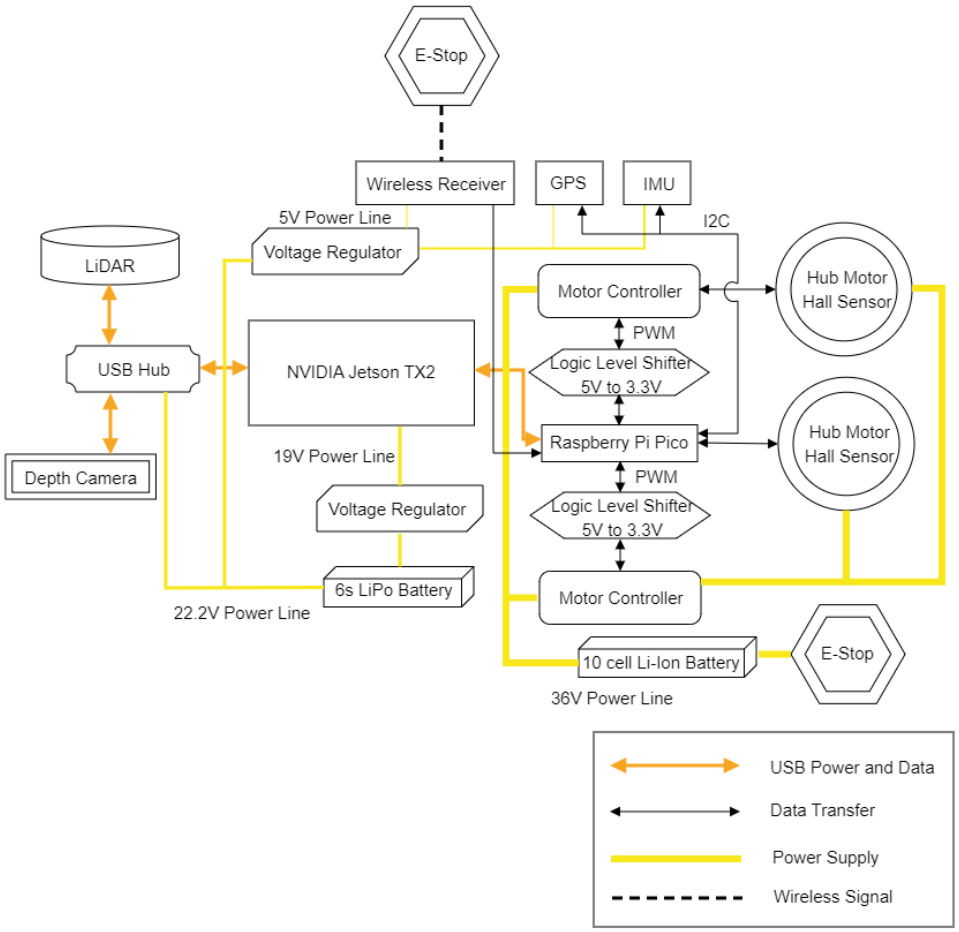


Figure 4: Electrical subsystem architecture showing communication protocols and voltage levels

to the motors through a micro-controller which commands motor controllers. A physical E-Stop that disconnects the motors from the battery, and a wireless e-stop activated by a wireless receiver provide two redundant methods of shutting down the system in case of an emergency.

5.1 Electronics

Our primary compute board is the **NVIDIA Jetson TX2**. It was selected mainly due to its 256 CUDA cores which allowed us to perform object detection using deep learning architectures at a frame rate exceeding 30fps. In addition, its ability to run a Linux based OS allowed us to use the Robot Operating System (ROS) which provides a framework for autonomous robotics greatly reducing development time.

Interfacing with sensors and running PID control loops requires real time compute capabilities. The **Raspberry Pi Pico** micro-controller board was used as it had the required number of hardware

interrupts, I/O pins, serial ports, and its Arm Cortex-M0 processor far exceeded the compute power required for our application. Our sensor package consists of

- **Intel D435** Stereo camera used for lane and object detection, and to generate a depth map
- **RPLIDAR-A3** Laser range scanner for accurate distance measurements to objects
- **u-Blox Neo-M8N** GPS sensor to obtain coarse vehicle position
- **Honeywell SS41** Hall sensors to generate wheel odometry
- **MPU9250** 9 degree of freedom IMU to detect changes in vehicle pose

Other components include

- **Lemon RX DSMX** 2.4GHz 6-channel radio to receive commands from a handheld transmitter
- Step-down voltage regulators to power the electronics which run at 19V, 5V, and 3.3V
- Logic level shifters to enable devices operating at different logic voltage levels to communicate with each other
- Powered USB 3.0 hub

5.2 Sensor Fusion

Estimating the current position and velocity of the robot is a key requirement for autonomous navigation. Physical sensors tend to be noisy and can fail to produce valid outputs given a particular set of environmental conditions. Thus, we rely on a suite of sensors that measure different modalities and combine the readings to produce a robust estimate of the robots state parameters.

5.2.1 Filtering hall sensor readings

The 3 hall sensors installed in each wheel produce detectable pulses as the poles of the brushless motor passes over the sensor allowing one to determine the direction and speed of rotation. But, this instantaneous wheel angular velocity measurement is noisy and a low pass filter was employed to attenuate high frequency noise. The transfer function for a low-pass filter is,

$$H(s) = \frac{\omega_0}{s + \omega_0} \tag{1}$$

This continuous transfer function was converted into its discrete from and then converted from the frequency to the time domain for a filter with a cutoff frequency of 25Hz and a sampling frequency of 1KHz. The difference equation of the low-pass filter employed is,

$$V_n = 0.854\, V_{n-1} + 0.0728\, v_n + 0.0728\, v_{n-1} \tag{2}$$

where, V represents the filtered values and v denotes the raw sensor readings. Using the known wheel radius, this angular velocity can be converted to ground distance traversed by the wheel assuming no wheel slip occurs.

5.2.2 Extended Kalman Filter

The readings from the IMU, GPS sensor, and filtered wheel odometry need to be combined to estimate the current position of the robot and the extended Kalman filter (EKF), which is an extension of the Kalman filter for non-linear systems, is routinely used for this task [4, 8, 11]. The 12 dimensional state vector consists of the 6DoF vehicle pose describing the position and rotation of the vehicle around the X,Y, and Z axes, and the velocity twist which is the linear and angular velocity of the vehicle about the 3 axes. We used the ROS robot_localization package which provides an implementation of the EKF equations [6].

5.2.3 PID Controller

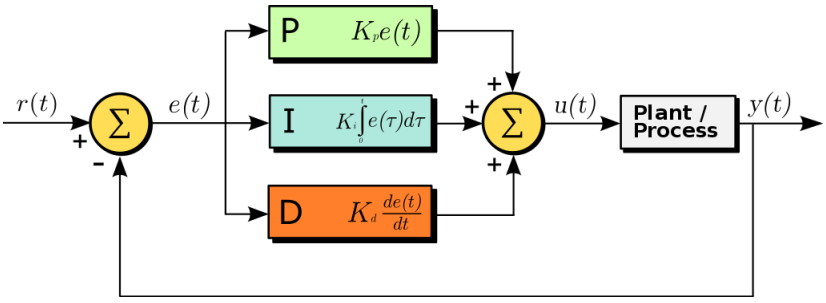


Figure 5: PID controller loop

Control of the actuators, in this case the wheel motors requires the generation of a suitable control signal. This can be achieved using open loop control where controls signals are generated based on a known mapping between the signal and the system output, or closed loop control where feedback in the form of measured system state is used. More precise control is possible using feedback as the system output can be closely matched to the desired output even in the presence of external disturbances and system model inaccuracies. For example, we employed a PID controller to set the motor rotation rate based on the error between the desired output (Motor RPM) and the current state of the system (Measured Motor RPM). A block diagram of the controller is shown in Figure 5 illustrating how feedback from the different control actions is incorporated. While PID controllers are fairly robust to poor coefficient tuning, a well tuned system is able to achieve a faster response to the steady state value, reduce overshoot, and eliminate steady state errors. We manually tuned the parameters to achieve an acceptable response although various analytical techniques exist to perform tuning [10].

5.3 Power System

The system is powered using two batteries, a 36V 4000mAh Li-Ion and a 22.2V 3300mAh Li-Po. The 36V battery is used to power the motors whereas the 22.2V battery is used to power the computational electronics. This separation adds an extra layer of safety as the control electronics will be powered in the event the vehicle runs out of power during motor operation. Table 3 lists the maximum rated power draw per component. The computational electronics consume ∼24W whereas the propulsion system draws ∼502W watts.

Table 3: Max rated power consumption for the components.

Component	Voltage (V)	Max Power Draw (W)	Quantity	Total (W)
Jetson TX2	19	15	1	15
Pi Pico	5	0.45	1	0.45
LIDAR	5	3	1	3
Stereo Camera	5	3.5	1	3.5
GPS	5	0.25	1	0.25
Hall Sensor	5	0.05	6	0.3
IMU	5	0.05	1	0.05
Radio Receiver	5	0.1	1	0.1
Voltage Regulator	6-42	0.25	2	0.5
			Sub-Total	23.15
Hub Motor	36	250	2	500
ESC	36	1	2	2
			Total	525.15

In practice, we measured the average powertrain (Motors+ESC) power draw to be ∼135 watts and the computational power draw to be ∼12 watts when the vehicle was carrying a 20lb payload on a level surface. Assuming a maximum discharge of 70% of pack capacity, which is recommended by manufactures to promote battery health, the 36V battery can provide 100.8W whereas the 22.2V battery can provide 51.3W. Table 4 lists the system runtime for the worst case scenario of maximal power draw and the observed power draw when the vehicle is moving at a constant velocity of 2.2 m/s (5 mph) on a level surface. The maximum runtime of the vehicle is limited by the propulsive sub-system even though the computational systems would continue operation since the vehicle will not be able to move once the 36V battery has been discharged.

Table 4: Runtime in case of maximum and average power draw

Components	Power Draw (W)	Power Available (W)	Runtime
Maximal Load			
Computation	23.15	51.3	2.2 Hours
Propulsion	504	100.8	12 Minutes
Vehicle	525.14	-	12 Minutes
Average Load			
Computation	12	51.3	4.3 Hours
Propulsion	135	100.8	45 Minutes
Vehicle	147	-	45 Minutes

5.4 Safety Devices

One of the key advantages of using a brushless motor is that it requires switching of the powered phase to produce continuous rotation. Unlike a brushed motor where an electrical fault could lead to the motor being erroneously connected to the power source leading to uncontrolled rotation, any faults in the brushless motor controller circuitry, or electrical shorts such as directly connecting the motor to the power source will result in no rotation. This adds implicit safety into our system. Our digital motor controller circuit has built in over-temperature protection, over-voltage protection, and over-current protection. The micro-controller board senses the motor rotation rate at 1000 Hz and prevents the motor from rotating at greater than 170RPM limiting the top speed to 5.06mph.

As per the competition rules, we have a safety light that shows the current operation mode of the robot. The physical E-Stop switch located at the rear of the vehicle disconnects the battery from the motor controllers disabling the propulsion sub-system stopping the vehicle. In addition, we have a wireless E-Stop using a dedicated 2.4GHz radio receiver which can be activated by a handheld radio transmitter. Activation of this E-Stop powers down the motor controllers, and this disables the brushless motors as they are unable to operate without digital switching circuitry. In contrast to systems that use Wi-Fi and the main compute board to receive the E-Stop signal, our system is unaffected by computational bugs that can cause the main computer to hang leading to a dangerous situation where the wireless E-Stop is unresponsive.

Finally, we will install fuses on the battery before the competition to add another layer of protection which would physically discount the battery from the vehicle if a fault causes increased current draw.

6 Software Stack

The software stack heavily relies on the Robot Operating System (ROS) as it has implementations of algorithms used for autonomous robotics and a framework for creating custom modules in either C++ or Python. It also allowed the team to use the Gazebo simulation environment and seamlessly deploy the stack on the physical vehicle.

6.1 Computer Vision

We use classical vision techniques to detect the lane markings, employ a deep learning approach to detect obstacles, and use visual simultaneous localization and mapping (V-SLAM) to determine the vehicle's position in the unknown environment.

6.2 Free Space Detection/Lane Detection

Since the boundary lanes for the tracks are white in color, we employ HSV color transformation followed by thresholding in order to extract the lanes. The HSV color space is more robust to variations in ambient brightness levels which can affect the RGB values of the pixels in an image. Therefore, it is ideal to work in this color space in order to extract a particular color in any given scene. Once we transform the image into HSV space, we perform thresholding to detect lanes based on a predefined range of values which could represent the lanes of known colour. We use morphological operations to filter noise in the resultant binary image. The binary image might contain disconnected regions that represent a single lane. We use the progressive probabilistic

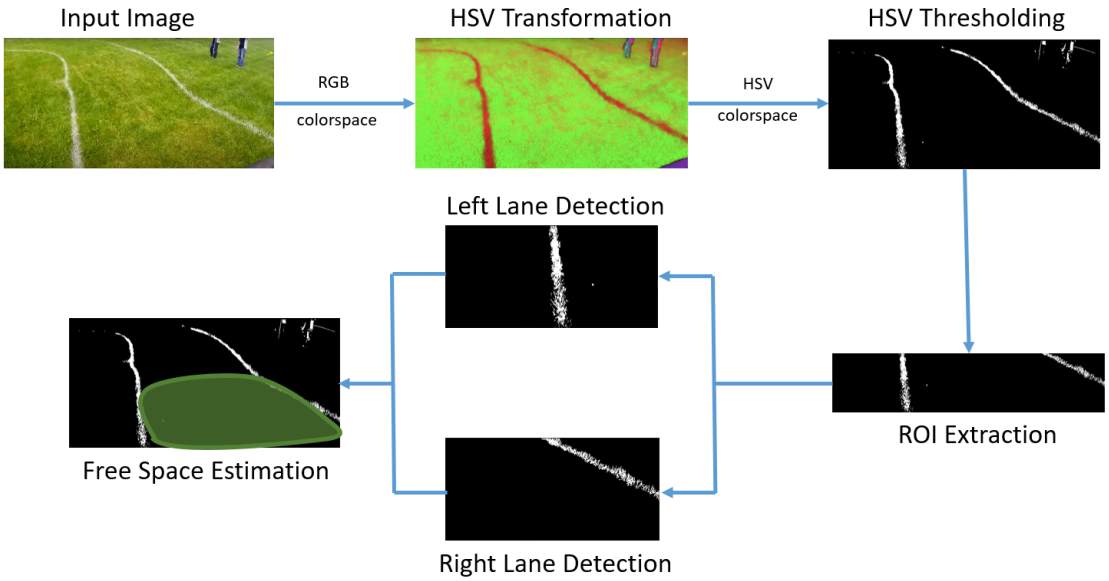


Figure 6: Free Space Detection

Hough transform (HT) which minimizes the amount of computation compared to the probabilistic HT to detect lines in the image [5].

The next step is to perform free space detection between the two detected lanes. Assuming that the vehicle is between the two lane lines at any given instant (we address the boundary condition where the vehicle is not between the lines later in the pipeline), and that we are only concerned with the immediate space in front of the vehicle rather than far ahead. This allows us to process only the lower half of the image reducing computational load and increasing pipeline throughput. In this half of the image, we split the image into two halves each containing one lane line based on the output from our progressive probabilistic HT. All the space to the right of the left-lane marking and the space to the left of the right-lane marking is then classified as free space available for the vehicle to move in. Output generated at each stage of this pipeline is shown in Figure 6.

6.3 Obstacle Detection

We fine tuned the YOLOv4 [1] deep learning architecture on a dataset consisting of images of construction barrels, trees, and traffic signs to be able to detect and classify when they are present in a RGB image. Given that we have detected free space, it is enough for us to simply detect any obstacle that is within this region of interest. The free space obtained is then updated to exclude all regions containing an object. The convex hull of the updated free space is computed to obtain a polygon. These pixel points are projected into world space using the known camera extrinsic and intrinsic matrices providing a list of legal positions the vehicle can occupy.

6.4 Visual SLAM

The images from the D435 depth camera are used to localize the vehicle through visual odometry. This involves detecting suitable features in the image taken at time t and correlating these features to those identified in the image taken at time $t-1$ to track the motion of these feature points over time. This estimate of motion in the pixel frame can be converted to the world frame through distance measurements in the world space using a ranging sensor such as a Sonars and LIDARs, or using stereo image pairs in conjunction with IMU data from which depth can be estimated. Using this information, we can not only estimate the pose of the vehicle in the world space but also create a 3D occupancy map (Octomap) of the world without the use of a scanning LIDAR which is suitable when high resolution maps are required. Many visual SLAM algorithms exists in the literature [3,7,9] and we use the open source implementation, RTAB-Map [2], to perform V-SLAM.

6.5 Path Planning

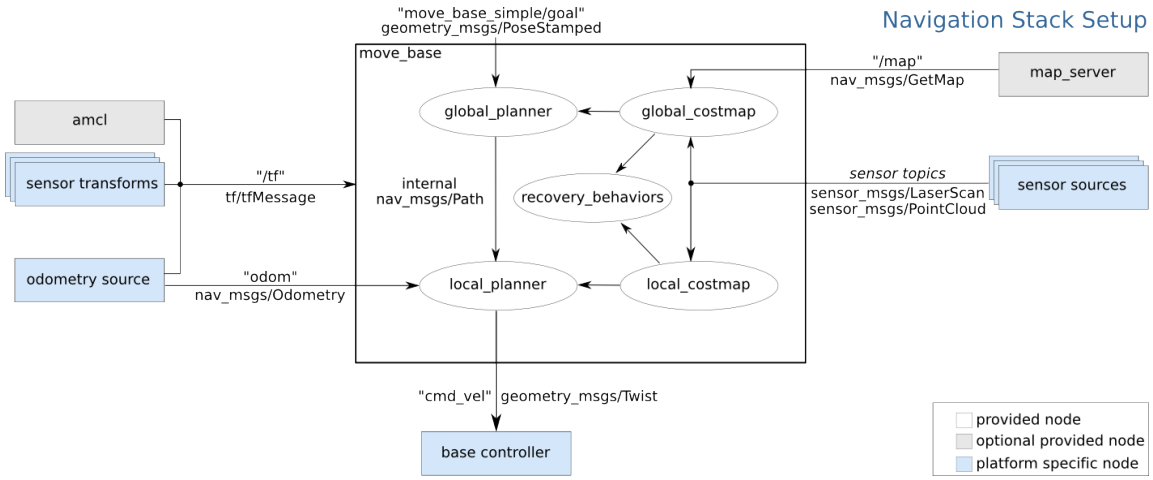


Figure 7: Architecture of the ROS navigation stack which is used for path planning.

We use the ROS navigation stack whose architecture is shown in Figure 7 to navigate the vehicle towards specified waypoints while avoiding obstacles. The navigation stack is fed sensor data labelled as 'sensor sources', and the vehicle pose estimate from the EKF discussed in Section 5.2.2 labelled as 'odometry source'. 'sensor transforms' refers to the transformation matrices used to move between the local coordinate frame of the individual sensors and the global coordinate frame. Finally, 'base controller' is the output of the stack that provides a velocity twist message in response to a set waypoint, which is converted to equivalent PWM signal values sent to the left and right motor controllers. The navigation stack has many different planners available, and we use a dynamic window approach for the local planner and D* algorithm for our global planner.

6.6 Exploration strategy

The first waypoint is manually defined to be a random point in the correct direction the vehicle must traverse the course. As the vehicle moves towards this waypoint, a map of the environment

will be initialized and updated. This allows a greedy exploration strategy that highly penalizes the vehicle from exploring previously mapped regions forcing the robot to move around the track in the correct direction. Without this cost heuristic, the vehicle could explore the track backwards. When the vehicle reaches the set waypoint, the furthest waypoint identified through free space estimation discussed in Section 6.2 is selected as the next waypoint. If the vehicle is in the area with no lane markings, the GPS waypoints are used instead.

7 Failure and Prevention

7.1 Vehicle Failure modes

Our vehicle is operating on a NVIDIA Jetson TX2 which is running real time image processing used for mapping and planning. There are several failure scenarios including algorithmic failure and software crashes.

We employ an operating system level scheduler that constantly checks the status of our ROS stack. In case of total ROS failure, it will trigger a restart of the program. Our program will log data to try and resume operation if possible in case of a crash. If a software crash is detected and no signal is received from the Jetson TX2 for a preset time, the Pico micro-controller will command the motors to stop.

7.2 Vehicle failure points

We may encounter hardware malfunctions and the possible points of failures include all the electronic components. The most likely components to fail are the hub motors, ESC, computational board, sensors, and auxiliary electronics such as the voltage regulators. Motor and/or ESC failure is critical since they cannot be replaced during the competition run and we are not planning to add redundancies due to the short run time of each attempt. Sensor failures may be recoverable depending on the particular component. If either the or GPS or IMU fails, the vehicle can still rely on vision and possibly could complete the mission. If vision fails, we can fall back to LIDAR based mapping. A battery failure is highly unlikely and thus do not account for a sudden battery failure during an attempt.

7.3 Failure Prevention

We designed our vehicle to be highly serviceable by using 3D printed parts yet ensured sufficient safety factor to eliminate the possibility of a structural failure. Our electronic components are encased to minimize exposure to the elements. We have redundant methods of localization and mapping which can be switched to in case of sensor failures. Batteries are monitored for health and balanced every time they are charged.

7.4 Testing and Design Concepts

We have conducted failure tests under multiple conditions. We simulated software failure from killing certain processes and hardware failure by covering sensors and disconnecting them during test phases. During controlled lab tests, we tested each component failure and the system responses from the failure. For example, when the Jetson was disconnected, the motors stopped successfully.

We used open source software in the ROS stack to minimize the introduction of software bugs as these implementations have been tested continuously by the robotics community.

8 Simulation

Using ROS allowed us to use the Gazebo simulation environment to test our path planning and navigation stack. We did not test our computer vision algorithms in simulation due to the difficulty in sim-to-real world transfer.

9 Performance Testing

Our team conducted multiple performance tests of the vehicle to make sure it passes all the competition requirements. List of performance tests and results are following:

- Battery performance test (5mph cruising with 20 lbs load)
  - Vehicle was successfully achieved 62 minutes of continuous run.
- Lane keeping ability
  - Vehicle was able to detect the lanes from the vision data and drive within the lanes.
- Obstacle detection and avoidance
  - LiDAR and distance camera were able to reliably detect the cylindrical obstacles from up to 15 meters away as the RPLIDAR we use has a maximum range of 20m outdoors. The vehicle was able to perform collision avoidance maneuvers by driving around them.
- Driving between multiple GPS coordinates
  - Vehicle was able to travel between given coordinates in order. Also, combined with obstacle detection and avoidance capability, it was able to drive between GPS coordinates without hitting any obstacles.
- Ramp climbing up and down
  - Vehicle was successfully move along the 15 degree ramp with 5 mph speed.

10 Performance Assessment

We have run all the tests multiple times to improve the performance of our vehicle. During the initial tests, we have found some issue with weight balance while climbing the ramp. The vehicle was showing noticeable instability. The front section of the vehicle was too light and started lifting as it climbed up the ramp. We have solved the issue by moving the center of mass and slowing down the vehicle on ramp. Besides of minor instability issues, all the sensors and components were fully functioning as expected. Before the competition, our team will run few more tests to fully assess our vehicle's performance and system stability. If possible we would like to run a full scale test space permitting.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv:2004.10934 [cs, eess]*, April 2020. arXiv: 2004.10934.

[2] Mathieu Labbé and François Michaud. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation: LABBÉ and MICHAUD. *Journal of Field Robotics*, 36(2):416–446, March 2019.

[3] Henning Lategahn, Andreas Geiger, and Bernd Kitt. Visual SLAM for autonomous ground vehicles. In *2011 IEEE International Conference on Robotics and Automation*, pages 1732–1737, May 2011. ISSN: 1050-4729.

[4] Chunbo Luo, Sally I. McClean, Gerard Parr, Luke Teacy, and Renzo De Nardi. UAV Position Estimation and Collision Avoidance Using the Extended Kalman Filter. *IEEE Transactions on Vehicular Technology*, 62(6):2749–2762, July 2013.

[5] J. Matas, C. Galambos, and J. Kittler. Robust Detection of Lines Using the Progressive Probabilistic Hough Transform. *Computer Vision and Image Understanding*, 78(1):119–137, April 2000.

[6] Thomas Moore and Daniel Stouch. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In Emanuele Menegatti, Nathan Michael, Karsten Berns, and Hiroaki Yamaguchi, editors, *Intelligent Autonomous Systems 13*, pages 335–348, Cham, 2016. Springer International Publishing.

[7] Geraldo Silveira, Ezio Malis, and Patrick Rives. An Efficient Direct Approach to Visual SLAM. *IEEE Transactions on Robotics*, 24(5):969–979, October 2008.

[8] M. St-Pierre and D. Gingras. Comparison between the unscented Kalman filter and the extended Kalman filter for the position estimation module of an integrated navigation information system. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 831–835, June 2004.

[9] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):16, June 2017.

[10] Antonio Visioli. *Practical PID Control*. Springer Science & Business Media, November 2006. Google-Books-ID: ymyAY01bEe0C.

[11] Yang Wang. Position estimation using extended Kalman Filter and RTS-smoother in a GPS receiver. In *2012 5th International Congress on Image and Signal Processing*, pages 1718–1721, October 2012.