MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Institution of MAHE

PROJECT MANAS
BRINGING BITS TO LIFE

IGVC Design Report

15th May, 2018

Team Leader: Sarthak Mittal (ai_head@projectmanas.in)
Team Manager: Sanjeev Nayak (team_manager@projectmanas.in)

I hereby certify that the development of vehicle, **Adam**, described in this report has been equivalent to the work involved in a senior design course. This report has been prepared by the students of Project MANAS under my guidance.

Professor & Head
Dept. of Computer Science & Engineering
Manipal Institute of Technology
Manipal - 576 104

Ashalatha Nayak
Professor and Head of Department
Department of Computer Science & Engineering
E-mail: asha.nayak@manipal.edu

Contents

1 Introduction

Project MANAS, the AI robotics team from Manipal Institute of Technology, has designed its first autonomous bot **Adam** to compete in the 26th Intelligent Ground Vehicle Competition. With Adam, Project MANAS continues to upload its founding vision – To inspire advanced research in Artificial Intelligence and oversee its implementation enabling automated systems to be made available to the general populous. Prior to IGVC, the team was focused on building India's first driverless car for the Mahindra Rize Prize Challenge. Our team consists of students from all branches of engineering and our interdisciplinary nature is the catalyst for our innovation and creativity.

2 Team Organization

The team is widely divided into three separate divisions – Mechanical, Sensing & Automation, and Artificial Intelligence. Each division is further divided into two subdivisons. All subdivision and division heads are solely responsible for his/her division/subdivision. To assist with the managerial tasks of the team, we have a separate non-technical management division. The primary board consisting of the Team Leader, Team Manager and Tech Head take all majors decisions pertaining to the team with the guidance of our faculty advisors. The team is comprised exclusively of undergraduate students, numbering 46 in strength.

The list of members who contributed towards **Adam** are: Dheeraj R. Reddy, Mukesh Kumar, Sanjeev Nayak, Anirudh Kaushik, Pranjal Sinha, Sarthak Mittal, Ansh Verma, Vishnu Menon, Apratim Mukherjee, Shreyas Sharma, Mahima Rao, Damodar Nayak, Johir Suresh, Tanuj Agarwal, Raunaq Kalra, Tanaya Mandke, Arya Karani, Rakshit Jain, Manav Sachdeva, Gaurav Singh Thakur, Varudhini Panuganti, Thomas Alex, Avirat Verma, Dasarath S, Rishab Agarwal, Ansel Dias, Vibhuti Ravi, Sanjay Subramaniam, Shivanshu Agarwal, Omkar Jadhav, Vishnu Dasu, Shrijit Singh, Shivesh Khaitan, Dheeraj Mohan, Sarath Krishnan Ramesh, Chaitanya, Saksham Banga, Shardul Purohit, Sarthak Gupta, Aneet Mandal, Sahil Swaroop, Yudishthar Lokhande. Total person-hours spent on the project is around **650** hours. Total cost estimate without sponsorship: **$12,700**. Total cost to the team: **$9,000**. The cost includes shipping charges from India to Oakland University.

3 Design Assumptions and Design Process

The vehicle and all its constituent components were designed on Autodesk FUSION 360. This software has a shared cloud workspace, which allowed members of the design team easy access to other's designs and made the assembly process very smooth. Further, load bearing capacity studies and other analyses were performed on ANSYS.

After team-wide brainstorming sessions, the following objectives were compiled before starting the design process:

- It must have a low centre of gravity
- The chassis must be modular, preferably assembled and not welded
- There must be a tall sensor pole for mounting cameras

- The vehicle must have high power to weight ratio
- The internal layout must be meticulously planned to save space and optimize wiring

Based on these, three designs were made with varying levels of compromise between maneuverability, compactness and robustness. Of these, the design that most closely fit each division's requirements was chosen, and improved upon with successive design iterations.

We use a multi-step process for all major technical decisions. We have a division or subdivision wise brainstorming session for possible solutions, followed by a period of research or prototyping, if preliminary results are successful, we proceed with full fledged implementation, followed by rigorous testing in simulators and real world track validation.

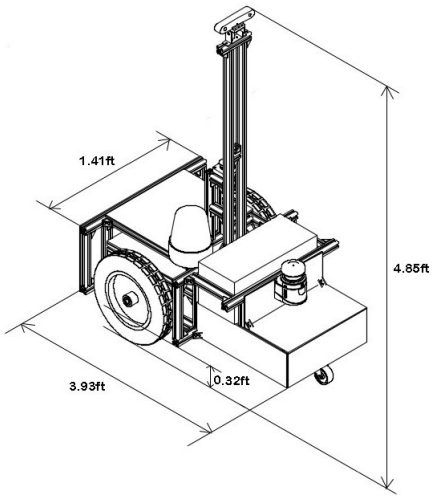
4 Innovations

- **Modular Design**: The major component systems of the vehicle may be detached and reattached with ease, and makes for ease of construction and shipping.
- **Vibration Damping System**: Using passive isolation, a suspension system was created for the motors, that effectively removed vibrations in the structure.
- **PID with Gain Scheduling**: Implemented a PID with Gain Scheduling for better motion planning in varying environments such as in inclines
- **Clustering based Lane Detection**: Using K-Means clustering for finding the lanes is extremely robust to lighting conditions, and movement of the bot
- **CUDA Accelerating the ROS Stack**: To reduce the software overheard, the most time consuming parts of the software stack have been CUDA accelerated, such as the map generation and clearing, as well as the K-Means clustering in lane detection
- **Visual Odometry**: We use optical flow from the stereo camera to calculate odometry with very low error bounds, this can be further combined with more traditional sensor such as IMU and wheel encoders to get excellent odometry

5 Mechanical

5.1 Overview

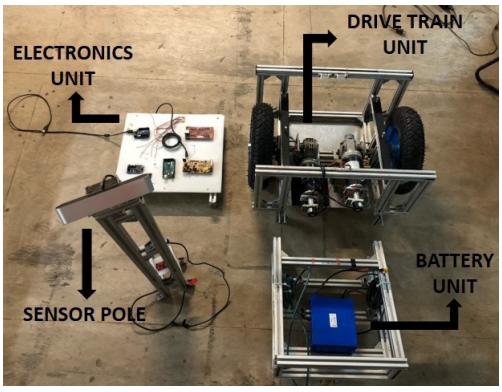
Adam is a two wheel differential-drive vehicle with a caster wheel in the front for balancing.



5.2 Structure

The frame of the vehicle is assembled by fastening aluminum members together. This approach was chosen over conventional welding as it allowed for the modular structure that was envisioned during initial concept design.

The frame of the vehicle has 4 distinct units, namely the drive train unit, the electronics unit, the battery unit and the sensor pole. Any of these units may be detached from the main frame in an instant, as even the wires are joined using detachable wire connector modules. This ease of separation enables the different divisions of the team to work on different components, without interference.



The heavy components are placed close to the base plate and towards the rear of the vehicle to achieve the most desirable location of the centre of gravity. The detachable battery unit is placed in front of the motors and electronics to optimize wiring routes.

5.3 Chassis

The chassis of the vehicle is made from aluminum extrusions with T-slots. The wide range of fastening and connecting elements that are compatible with these T-slots, make them extremely easy to work with. The use of this method also allows for changes in the structure, which is as a huge advantage in the initial stages of production where multiple iterations are tested and analyzed.

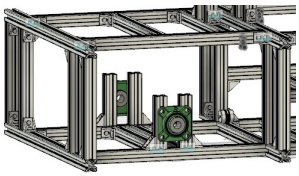
The T-slots are connected by die-cast L-shaped brackets and fasteners. Each of these fasteners are coated with thread-locking fluid to provide an additional layer of security. The slots on the T-slots also help conceal the wires of the various electronic components.

5.4 Drive-train

The team explored various driving systems like tracked vehicles and standard four-wheeled drive. The differential-drive model was chosen as itâĂŹs dynamic framework has been extensively studied and tested.

The motors are placed parallel along the length of the vehicle. A designed, custom manufactured reduction right-angle gearbox is affixed to the end of the motor shaft. Each motor has a rear shaft to which a rotary encoder is attached. The motors are isolated from the base by an anti-vibration pad.

Balancing and 3D wheel alignment are big challenges to any driving system where the wheels are not attached to a solid-axle. To solve this, two square pillow block bearings are placed anti-parallel to each other and aligned in all 3 planes. This ensures that any shaft passing through these bearings are on a straight line.



5.5 Weatherproofing

The vehicle is covered all over with High Density Polyethylene (HDPE) sheets. HDPE is a low cost, lightweight thermoplastic that is easy to machine and has strong chemical resistance and high tensile strength. Vinyl sheets were coated over these HDPE sheets for dust-proofing and aesthetic purposes. Gaps and crevices between adjacent sheets are sealed by caulking.

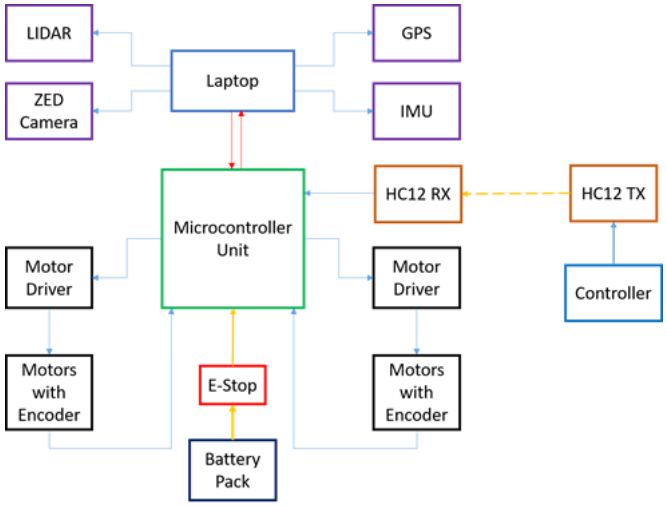
5.6 Vibration Damping

At high RPMs, the motor-gearbox unit vibrates at a high frequency and sends tremors throughout the chassis. To resolve this the motors are raised from the base plate and rested on a mount that is made from a combination of wood and rubber padding. This mount acts as a passive isolation system, reducing the vibrations to a minimum. Further, spring washers are attached to the bolts in the drive-train.

6 Electronic and Power Desgin

6.1 Overview

The electrical system of Adam includes two Cryton MD10C motor drivers, a microcontroller unit, an HC12 wireless transceiver, encoders and Predator 15 laptop for computation. Adam is also equipped with a multitude of sensors like SICK LMS133 LiDAR, Razor 9DOF IMU, ZED Stereo camera and a GPS module.



6.2 Power System

The motors and the LiDAR are powered by a dual 12V Lead Acid connected in series with a combined voltage of 24V and a capacity of 20Ah while the microcontroller unit is powered by the Laptop itself.

Component	Max Power Consumption	Operating Voltage	Source
Motors (x2)	120W (x2)	24V	Battery Pack
LiDAR	7.2W	24V	Battery Pack
Microcontroller Unit	0.5W	5V	Laptop
ZED Camera	1.9W	5V	Laptop

The Battery pack powers the motors through the Cryton motor drivers, and also the SICK LiDAR. The rest of the sensors and microcontroller units are powered by the laptopâĂŹs internal battery itself. The battery pack takes about 4 hours to fully charge while the laptop takes about 2-3 hours to charge. The maximum power consumption of the components running on battery 247.2 W which translates into about 1.5 hours of run time. The laptop on the other hand can run at its maximum power for about 0.5 hours. But the operating power consumption is generally half the maximum power and hence the battery pack can last about 3 hours and the laptop can last about 1 hour. This was verified during testing. All the electrical connections are done on Veroboard for cleaner aesthetic and better reliability.

6.3 Electronic Suite Description

6.3.1 Computation

The onboard laptop is used to process and compute all the sensor data and camera feeds, giving an input to the motor controllers with encoders to obtain a closed loop operation.

Adam carries an Acer Predator G9 793 with 7th gen Intel core i7 clocked at 2.8 Ghz, Nvidia GeForce GTX 1070 6GB graphics card and 16 GB RAM. This computer is responsible for most sensor data processing and all path planning and control algorithms.

6.3.2 Microcontroller Unit

Communication and control of much of the hardware is handled using the Arduino UNO and the Mega microcontrollers. Adam has three of these on board. The two Megas handle the motor control and the encoder interfaces. The UNO handles the E-stop, the battery monitoring functionality and the lighting controls, while acting as the master for the two Megas. A custom board has been designed and built by our team to match the layout of the entire circuitry to help to organize the connection routing.

6.3.3 LiDAR

Adam employs a LMS133 LIDAR to provide convenient and straightforward obstacle detection. The LIDAR operates at 25 Hz, has a 270Âř field of view and a 20m range.

6.3.4 Stereo Camera

The stereo camera utilized is a ZED from stereo labs with 2k resolution, range of 15m, 6 axis positional tracking, depth perception at large distance and a USB interface with laptop.

6.3.5 Global Positioning System (GPS)

A GPS is used to provide world position to the robot, helping in the localization of the bot and allowing obstacles to be placed in world space and thereby waypoints to be followed. Adam uses a Garmin 18x with a USB interface.

6.3.6 Inertial Measurement Unit (IMU)

The Razor 9DOF IMU is which is a combination of triple axis gyroscope, triple axis accelerometer and a triple axis magnetometer, which the combination of all nine gives us the accurate orientation and acceleration of the vehicle for accurate localization in the given environment.

6.3.7 Encoders

Each gearbox is connected to a quadrature wheel encoder, allowing velocity and absolute distance to be calculated. It has a resolution of 600 PPR and 2400 CPR. The data from the encoders is used for localization as well as feedback for a closed loop control.

6.3.8 Battery Monitoring System

We monitor our battery level using a special circuit that is designed to light up LEDs as an indicator for the remaining charge left in the battery. Each of the four LEDs correspond to different voltage levels and if all LEDs are lit up, it means the voltage is at 24V and if all LEDs are turned off, it means the voltage is below 21V.

6.4 Safety Devices

There are two separate E-stops implemented, one on the hardware level and one on the software. The hardware one is a red colored push button mounted on the vehicle that directly cuts the power from the battery. The software implanted one can be engaged from the remote controller which with the HC12 Transceiver has a range of about 1Km. The mounted LED Light signifies the state at which the vehicle is in, blinking light meaning the vehicle is in autonomous mode and the solid indicating the vehicle is in manual mode.

7 Software System

7.1 Overview

The software has been divided into two modules, namely perception and planning. The modules and different nodes of each module are integrated using the Robotics Operating System (ROS). The perception module is responsible for the detection of lanes, potholes and obstacles in the track using the raw stereo camera images, point clouds and the LiDAR scans. The planning module takes in the information produced by the perception module and is then responsible for the generation of map, localization of the bot, obstacle avoidance, and path planning.

7.2 Perception

7.2.1 Ground Plane Segmentation

The ground plane segmentation node uses RANSAC (Random Sample Consensus) to identify the ground plane in a given point cloud. To estimate the ground plane, three random points from the point cloud are selected and the parameters of the corresponding plane are calculated. Then, all points of the original cloud belonging to the calculated plane according to a given threshold are detected. This procedure is repeated N times; in each one, it compares the obtained result with the last saved one. If the new result is a better planar model, the previously saved planar model is replaced. Since, there is one-to-one correspondence between the point cloud points and the image pixels, the pixels corresponding to the points belonging to the estimated planar model are kept and all other pixels are filtered.

7.2.2 Lane Detection

The lane detection node takes in an image after ground plane segmentation from ZED's camera feed as input and detects both lanes and potholes. Smoothening and dilation operations are performed on this image to increase the intensity of the white regions in the image and make the clusters more apparent.

(a) Original **(b)** Ground plane **(c)** Lanes

K-Means clustering is then applied to cluster pixels on the basis of five features, namely the x and y pixel coordinates and the RGB values of the pixels. From the clusters obtained after applying K-Means, only the clusters with an average RGB value falling in a threshold set for white color are taken and the rest are filtered out. Once the white clusters are found in a frame, the corresponding 3D coordinates of the pixels belonging to these clusters are found from the corresponding point cloud from ZED and added to a new point cloud. This point cloud thus contains only the points belonging to either a lane or a pothole and is sent to the planning module.

The lane detection system is also responsible to send waypoints to the navigation system in the planning module. If in a frame both lanes are successfully detected, the points lying furthest on each lane are identified and the waypoint is calculated as the median of these two points. If only one lane is detected, the point lying furthest on this lane is taken as the waypoint. If no lanes are detected, no waypoint is generated. When a waypoint is found, it is published to the waypoint handler system in the planning module.

Running K-Means clustering on about a million points (720p image) in every frame on the CPU proved to be extremely inefficient. On the CPU, it took K-Means about 0.5 seconds to find 4 clusters per frame. To overcome this inefficiency, K-Means clustering algorithm was implemented using CUDA to make it GPU accelerated. The GPU version of K-Means was able to find 20 clusters in about 0.1 seconds per frame. The high frequency that the GPU accelerated version of K-Means achieves, helped us detect potholes and lanes entirely, since even if some portion of a lane is missed in one frame, it is easily detected in the next.

7.2.3 Object Detection

The onboard SICK LMS133 is used for the detection of obstacles on the track. The LiDAR sensor has a range of 0.5-20 metres and a field of view of 270° with an angular resolution of 0.25° giving us an extremely accurate positional estimate of each obstacle. Every obstacle detected by the LiDAR, along with the lane and pothole information from the perception module, is marked onto a 2D occupancy grid of size 100x100. The points occupied by obstacles, lanes and potholes are marked with an extremely high cost and the area surrounding these obstacles is marked with a cost that is exponentially decayed outwards. The exponential decay of cost around the obstacles helps our path planning algorithm to find the smoothest path to get from point A to point B while avoiding obstacles and keeping as far away from lanes and potholes as possible.

7.3 Planning

7.3.1 Map Generation

SLAM is a process by which a mobile robot can build a map of an environment and at the same time use this map to deduce its location. In SLAM, both the trajectory of the platform and the location of all landmarks are estimated online without the need for any a priori knowledge of location.

The Google Cartographer library achieves its outstanding performance by grouping scans into probability grids that they call submaps and by using a branch and bound approach for loop closure optimization. While new scans are matched with and subsequently entered into the current submap in the foreground, in the background, the library matches scans to nearby submaps to create loop closure constraints and to continually optimize the constraint graph of submaps and scan poses. We differentiate between local scan matching which inserts new scans into the current submap and which will accumulate errors over time and global SLAM which includes loop closing and which removes the errors that have accumulated in each submap that are part of a loop. Both local and global matching are run at the same time. During local scan matching, the Cartographer library matches each new scan against the current submap using a Ceres-based scan matcher.

7.3.2 Path Planning

We use the global map to generate an optimal path from one point to another. We experimented with various graph search algorithms and decided to use A* for its speed and reliability.

The generated path takes into account the obstacles and lanes of the course, and gives us a set of points marking the path. The path planner continually updates while Adam moves for finding the best path to the destination at all points during the course, or if Adam veers off path for unforeseeable reasons, and to ensure the waypoint handler can use intermediate goals as opposed to one final goal.

We use the GPS sensor on board to find the destination point within some degree of accuracy.

If Adam is ever stuck in a suboptimal position (too close to lanes, dead ends, centre islands, etc.), we have a set of robust recovery maneuvers coded in. It first clears the map to get rid of any possibly erroneous mapping data, followed by a check of obstacle density in the near vicinity of Adam. It will then choose the area with least obstacles, i.e., "safety" neighbourhood, with some heading towards the destination goal. After recovering from the suboptimal position, we continue to use the path planner.

7.3.3 Waypoint Handler

Giving just the final destination point to the path planner does not achieve desirable results, especially if only partial map has been generated. To overcome this limitation, we use a system of intermediate waypoints to successfully reach the target destination.

As previously discussed, the lane detection algorithm also gives us the intermediate waypoint. However, this waypoint intends to usually be close (1m), adhering to this near waypoint causes the motion planner to accelerate and decelerate more than needed. To overcome this, all waypoints are linearly extrapolated to a much further distance of around 3 metres. This caused no anomalies in our testing, since new waypoints are constantly calculated by the lane detection.

If no waypoint is given in a time interval, the destination is set to the final destination in that interval with no intermediate waypoints, this escape condition condition occurs for only intervals before the lane

detection gives us a newer point.

Therefore, the actual path to follow, is a combination of various path segments generated using lane information, and direction of the final goal.

7.3.4 Motion Planner

Once we have a path, we use simulation based motion planner to find the exact velocity needed to execute the optimal trajectory.

The motion planner uses the faster local map as opposed to the slower global map, since we need almost instantaneous information about the immediate surrounding on the bot for motion planning.

It searches over the space of possible linear and angular velocities with simulations using a differential drive model.

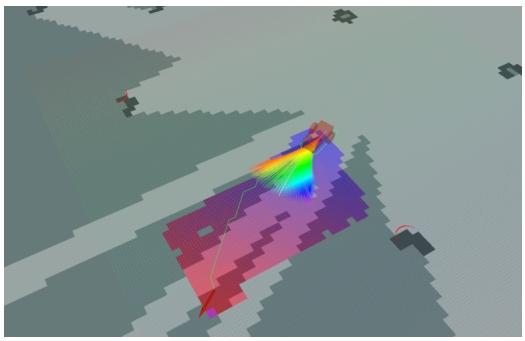


Figure 2: DWA Planner

Our algorithm works in the following way:

1. We discretize the robot's action space into known intervals.
2. For each sample in the discretized action space, we forward simulate the action for a short period of time (2s)
3. We assign each trajectory from each sample a score based on a hand defined heuristic that takes the following into consideration:
   - Proximity to goal
   - Proximity to global path
   - Proximity to obstacles
   - Speed

   If a trajectory collides with an obstacle, it is immediately discarded.
4. We act greedily with respect to the scores, to find the best trajectory and it is executed.

To ensure the motors are able to achieve the velocities required, we use a PID algorithm that has been hand tuned for grassy terrains. We also use gain scheduling for the PID during inclines or declines to ensure optimal behaviour in special conditions.

7.3.5 Localization

We have multiple source of odometry available on Adam such as an IMU, wheel encoders and visual odometry. The various readings from the sensors are combined using a non-linear Kalman filter that has been tuned using a genetic algorithm, evaluated on real recorded data. The combined odometry has demonstrably lower variance and drift in the readings.

Along with the combined odometry, we also use LiDAR based odometry by scan matching the current laser scan reading with the stored map to effectively localize Adam in the precomputed map. This greatly reduces any drift that may arise out of using just the combined odometry.

7.4 Additional Creative Concepts

The fusion of various sensors as described above using an Unscented Kalman Filter requires information about the initial covariance matrices of process as well as sensor noise. These matrices are usually hand tuned by domain experts. Instead, we used a genetic algorithm that tried to minimize the error between the true odometry reading and the fused sensor reading.

We deviated from the general norm of using a purely image processing based lane detection, and instead opted for one that uses clustering, and we used CUDA for its real time performance.

We used CUDA to accelerate some of the bottlenecks in our mapping suite, specifically the raytracer used for clearing certain regions of the map that no longer have any obstacles.

7.5 Interoperability Profiles

7.5.1 Overview

With regards to the IOP challenge, we successfully implemented a number of IOP attributes in order to make Adam's system interoperable. Since, Adam's entire software stack is implemented and integrated using ROS, we implemented a ROS/JAUS bridge to achieve JAUS compliance using the JAUS Toolkit (JTS).

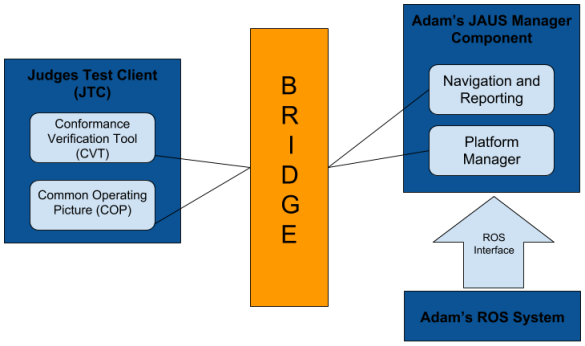


Figure 3: ROS/JAUS Bridge

7.5.2 System Integration

The ROS/JAUS bridge essentially acts as a communication interface between the Conformance Verification Tool (CVT) running on the Judges Test Client (JTC) and Adam's ROS system. The Navigation and Reporting and the Platform Management JAUS attributes are implemented using a single component which provides all of the services defined by these attributes. To the CVT, this appears as a JAUS component when in fact this component runs as a ROS node and does the interfacing between ROS and JAUS using the ROS/JAUS bridge.

8 Failure Modes, Failure Points and Resolutions

Potential Failures	Resolution
T-slot chassis were loosened over repeated use	Added additional connection elements and supports, reinforced existing connections with thread-locking fluid
Wheel was tilted at small angles	3D wheel alignment using precision placement of Square pillow block bearings
Motor gearbox started leaking oil	Replaced the gasket, and added spring washers to gearbox casing to reduce vibrations
Motor driver malfunction	Ready availability of extra motor drivers
Battery malfunction or discharge too fast	We are carrying a spare set of batteries
The software stack crashes unexpectedly	All communications between and within each node is logged for later troubleshooting
Sensor are not working as expected	We have replacements for: IMU and wheel encoders. The remaining sensor have software recalibration software
Loose or damaged connections in the electrical system	Due to the modular nature each individual segment can be tested in parallel to due the point of error. Spares are available for quick replacement
Kalman filter does not fuse the various sensors properly while testing in the track terrain	Run the genetic algorithm on some recorded data in said terrian and recompute the process covariance matrix

9 Simulation

Adam was first built and tested exhaustively in the simulator before running any real world tests. The simulator of choice was Gazebo due to its great physics model and readily available ROS support. An accurate physics model of Adam was designed in the simulator by the AI and mechanical divisions.

A life-sized replica of the proposed IGVC map (given in the 2018 rules) was created using super high resolution grass textures, and exactly measured barrels. White lanes were introduced and extra effort was put into making them look like they were made out of paint or chalk powder. All the sensor were calibrated in the simulation with artificial Gaussian noise added to mimic real world readings as much as possible.

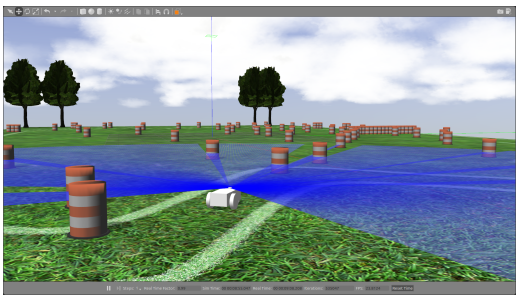


Figure 4: Simulation

The differential drive mechanics of Adam retrieved from the ANSYS model, and use in the simulator, since the model planner uses forward simulation to find the optimal path, having an accurate model of the bot is crucial. The velocity in the simulator did not go through a PID, but instead directly controlled the bot. Using the PID in real life allowed us to benchmark each system in the motion planning pipeline separately.

10 Performance Testing

Adam was tested in an open grass field with a miniature replica of the track, with similar lanes, obstacles and complexity as well as other open areas around our university.

- **Software**: All the algorithms were first implemented and tested in the simulator. For tuning the various parameters, we used ROS bag files recorded while it was being manually driven. Each algorithm was then tested on the track individually, followed by integration tests.
- **Mechanical**: A number of tests were conducted to determine the best position of the balancing caster. Motor's speed, torque, and efficiency curves were calculated repeatedly using tachometer and encoders. This data was used to model the PMDC Motors.
- **Electrical**: All the circuits and sensors were first calibrated in the workshop to ensure their working before being tested. Regular checks were conducted at the end of each day, and damaged components and wires were continually replaced.

11 Initial Performance Assessments

Adam was designed from the ground up with the rules of IGVC 2018 in mind. To that end, from our preliminary testing it checks off all major marks.

- The various mounts of the bot have withstood rocky terrain with very little loosening
- The motors are able to meet the minimum speed and incline requirements specified in the rules
- Max speed of 1.5 m/s, verified during testing
- All the sensor readings are within error bounds during testing
- Our max test runs lasted 1.5hrs and the batteries lasted till the end of testing
- On the software front, all algorithm are working and testing: lane detection, lane driving, mapping, localization, path planning, motion planning and waypoint handling
- Adam can handle complex navigation around obstacles and can navigate out of No Man's Land
- Object detection range is close to the max range of the SICK LMS133 LiDAR which is about 20m.
- Recovery maneuvers have been testing in a few conditions: too close to obstacles, too close to lanes and dead ends
- With our current localization and GPS accuracy, Adam arrives near a waypoint with an accuracy between 0.15m to 0.8m

The next few days will be spent better tuning our localization, mapping and waypoint generation algorithms. We are currently working on redundant backup systems for as many aspects of the bot, in case of damage during or after shipping.