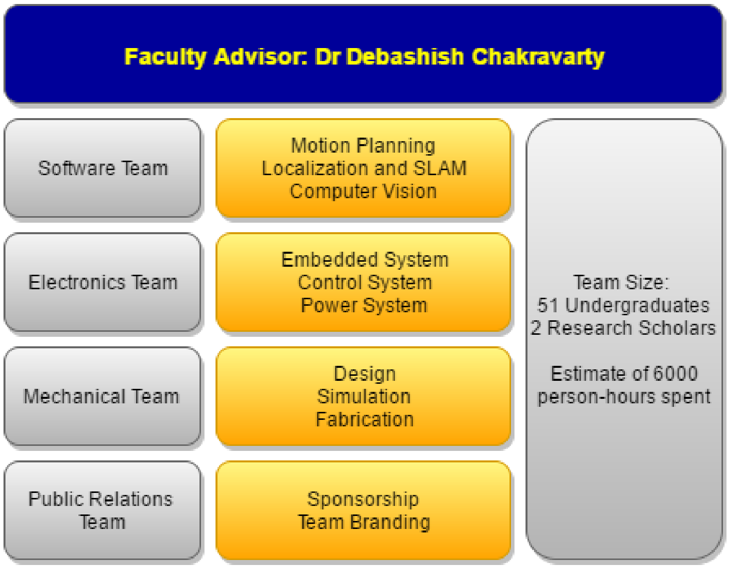
Eklavya 5.0: Autonomous Ground Vehicle Research Group
Indian Institute of Technology, Kharagpur, India
Faculty Advisor: Dr. Debashish Chakravarty[*]

INTRODUCTION

Team Autonomous Ground Vehicle (AGV), under the ambit of Center for Robotics, IIT Kharagpur, has been pioneering the autonomous ground vehicle technology with the ultimate aim of developing the first self-driving car of India. The team has been participating in IGVC since its inception in 2011 with the Eklavya series of vehicles. Eklavya 5.0, another feather in the cap of the Research Group is all set to participate in the 24th Intelligent Ground Vehicle Competition (IGVC), Oakland University. With new robotic innovations, the successor of Eklavya 4.0, is a much more simplified and powerful Eklavya 5.0 in all aspects i.e. mechanical, electrical and software.

TEAM ORGANIZATION

The effort behind this project was put in by a bunch of over fifty enthusiastic and intellectual undergraduate students from various departments of IIT Kharagpur.



Faculty Advisor: Dr Debashish Chakravarty

Software Team — Motion Planning, Localization and SLAM, Computer Vision

Electronics Team — Embedded System, Control System, Power System

Mechanical Team — Design, Simulation, Fabrication

Public Relations Team — Sponsorship, Team Branding

Team Size:
51 Undergraduates
2 Research Scholars

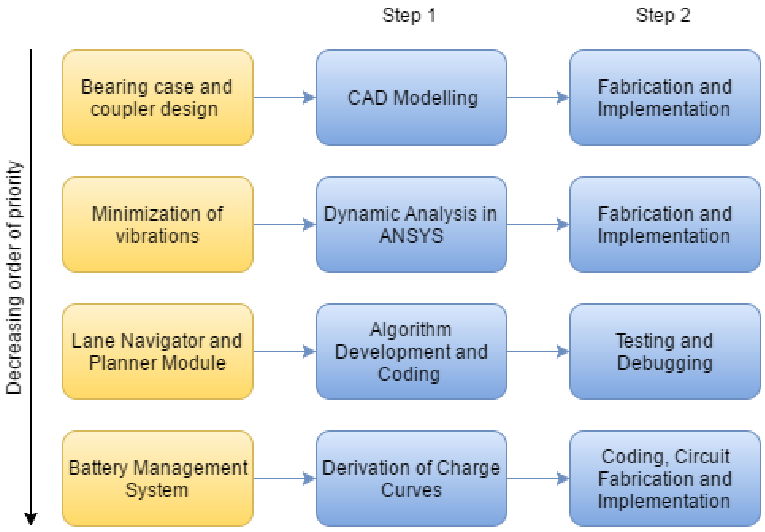
Estimate of 6000 person-hours spent

[*] Associate Professor, Department of Mining Engineering, IIT Kharagpur, C1-100, IIT Campus, Kharagpur 721302.

DESIGN PROCESS

We thoroughly analysed the failure points of Eklavya 4.0 after its performance in IGVC 2015.

The following diagram describes the major improvements made in Eklavya 5.0.


Step 1
Step 2
Decreasing order of priority
Bearing case and coupler design
CAD Modelling
Fabrication and Implementation
Minimization of vibrations
Dynamic Analysis in ANSYS
Fabrication and Implementation
Lane Navigator and Planner Module
Algorithm Development and Coding
Testing and Debugging
Battery Management System
Derivation of Charge Curves
Coding, Circuit Fabrication and Implementation

Assumptions were made such as, no skidding of wheels which meant the velocity obtained by processing signals from encoder were assumed to be true, the bot was assumed to rotate about a centre of curvature which paved the way in designing the control systems.

The path planning module has been changed completely in order to generate kinematically feasible trajectories for our bot. Also, the lane navigator has been made more robust and has been tested to work in a number of corner cases.

Taking into consideration the above improvements and assumptions, the design for Eklavya 5.0 was proposed as shown.

MECHANICAL DESIGN

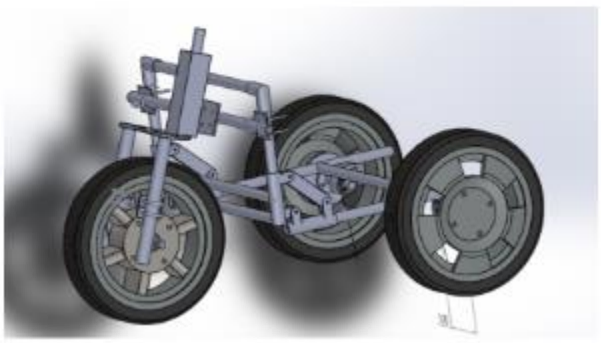


Figure 2. Mechanical Design

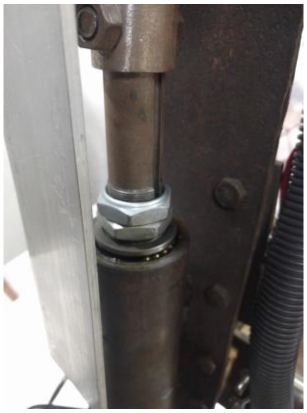
Overview

The Eklavya 4.0 was a front wheel driven and steered vehicle. However, it had many shortcomings. It was vulnerable to undue vibrations. The structure was made up of wood. Hence, it was prone to lateral vibrations as well as longitudinal vibrations

Attainment of maximum stability by lowering the centre of gravity and reducing the vibrations were two major concerns while designing Eklavya 5.0. Initially, the steering column was connected to the frame through a flat plate. But it was not sufficient to counter the induced moments from the drive motor. Hence, it was decided to add another link to support entire dynamic forces acting on the joint. This successfully reduced the longitudinal vibrations [1]. To improve lane navigation, the height of camera mount was not sufficient in Eklavya 4.0. To tackle this, we considered the height of the bot and the caster angle of the front wheel and calculated the optimal height for camera placement to be 5.5 ft. The new camera mount was fabricated and successfully installed on the robot. Finally, to reduce the transverse vibrations, the design of the bearing case was modified. It is to be noted that we have not installed any suspension system in our robot so as to keep the design simple, compatible and light weight.

STEERING COLUMN

Drive motor is attached to the steering column which causes both radial and axial loading. Considering this, we calculated the angle of inclination of the steering column with the horizontal to be 20 deg. This lead to less radial loading which further lowered the torque requirement for steering the vehicle. Additionally, the steering column is designed to be self-centring which helps the bot to move forward easily.



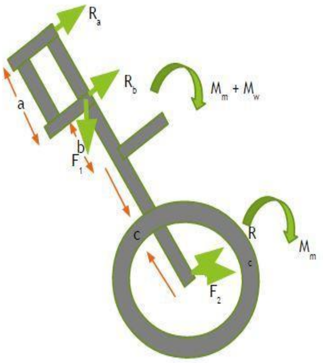


Figure 3. Moment diagram of the steering column.

Figure 4. Manufactured Steering Column

R_a = Reaction due to upper bearing
F_1 = Weight acting on steering stem
M_w = Torque due to weight of motor
R_c = Reaction from tire

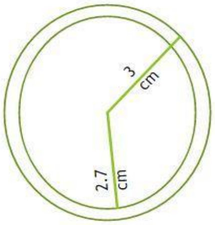
R_b = Reaction due to lower bearing
M_m = Torque provided by motor
F_2 = Force due to acceleration

Table 2.Dynamic Analysis of Steer Column

Scenario	F_1	F_2	Theta	Total length	Shear (Max)	Bending Moment
Dynamic State (Max torque=120Nm)	34.2	99	70	.25	64	54
Stationary State	34.2	0	70	.25	34.202	8.55
During a jerk (5 cm at 10 miles/hour)	34.202	99	70	.25	1050	250

$$\text{Stress} = My/I$$

For our dimensions, we have I = 1.17 x 10^{-8}, Maximum moment = 53.54 Nm, Stress will be maximum at outer face, y = 3 cm, Stress = 58.5 MPa



Conclusion:

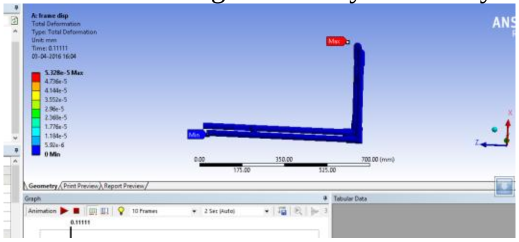
Fork length: 25 cm, Angle with horizontal: 70 degrees, Length of steering

T stem: 20 cm, Diameter of Fork: 3 cm, Thickness of fork: 3 mm

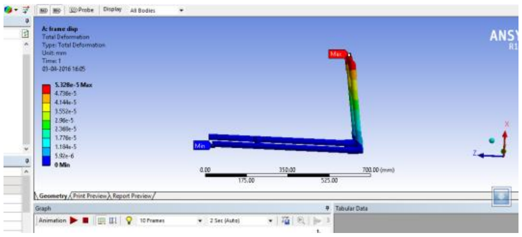
This matches with steering column of Motorbikes steering column. Therefore, steering stem and forks of Hero Honda Aviator scooter were used.

The Reduction of Longitudinal Vibration

The longitudinal vibration [1] was reduced with the introduction of a new rod. This is evident from the following force analysis in Ansys.


A: frame disp
Total Deformation
Type: Total Deformation
Unit: mm
Time: 0.11111
01-04-2016 16:04

Animation | 10 Frames | 2 Sec (Auto)


A: frame disp
Total Deformation
Type: Total Deformation
Unit: mm
Time: 1
01-04-2016 16:05

Probe | Display: All Bodies
Animation | 10 Frames | 2 Sec (Auto)

Figures: When load is applied with the support rod and when load is applied when there is no rod

WHEEL HUB DESIGN

Front Wheel

16 inch wheel with an attached hub motor is used for translation. It is attached to the fork through U-clamps. Load transfer is done effectively through two mild steel couplers.

Rear Wheels

For our design, we have chosen tapered roller bearings because they are capable of carrying loads in both axial and radial directions and discards the need for thrust bearings which creates a problem in disassembling the robot. We can arrange a pair of tapered roller bearings in three ways- "Face to face", "Back to back" and "Tandem (parallel)”. Face to face type has less support width so it does not provide rigid support. This arrangement is less suitable to support tilting moments due to its lower stiffness. In our case we used a pair of tapered roller bearings adjusted in back to back arrangements as it provides enough rigid support to handle the weight transferred on wheel hub.

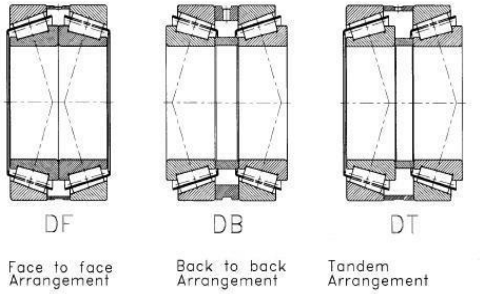
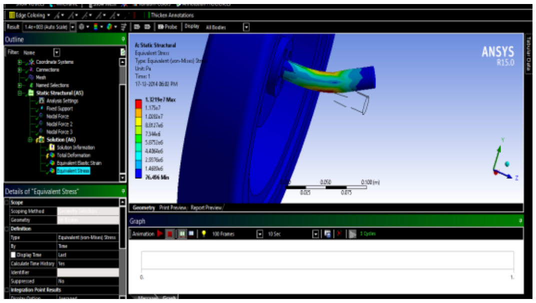


Figure 5. Roller bearing


A: Static Structural
Equivalent Stress
Type: Equivalent (von-Mises) Stress
Unit: Pa
Time: 1
17-12-2014 06:02 PM

Result: 1.4e+003 (Auto Scale) | Probe | Display: All Bodies

Details of "Equivalent Stress"

Scope	
Scoping Method	Geometry Selection
Geometry	All Bodies
Definition	
Type	Equivalent (von-Mises) Stress
By	Time
Display Time	Last
Calculate Time History	Yes
Identifier	
Suppressed	No
Integration Point Results	

Animation | 100 Frames | 10 Sec | 3 Cycles

Figure 6. ANSYS stress analysis of front wheel.

ELECTRONIC AND POWER DESIGN

Overview

The electrical system overview is shown in detail in the figure given below.

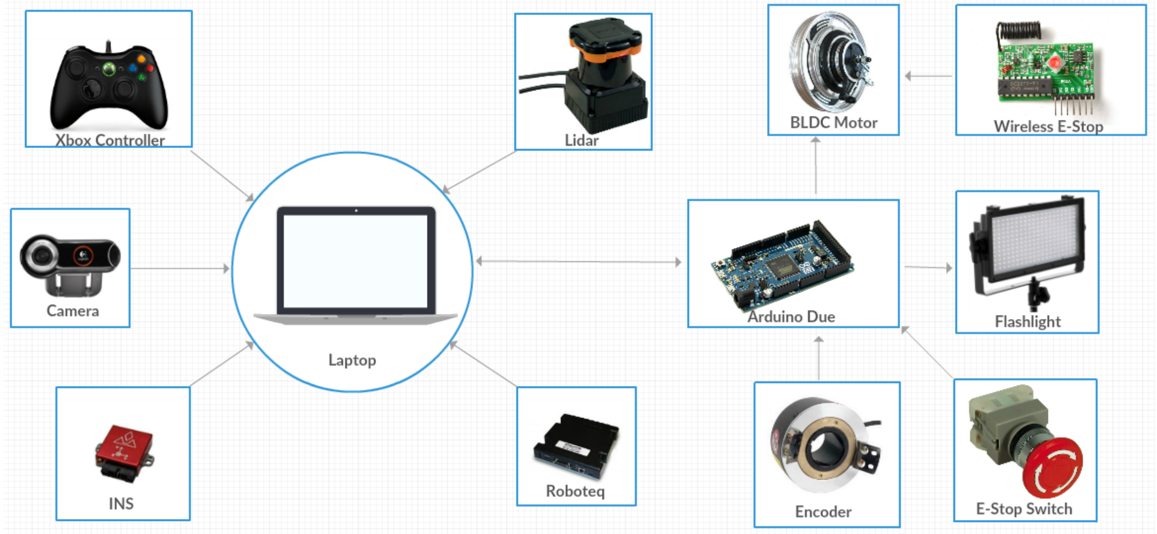
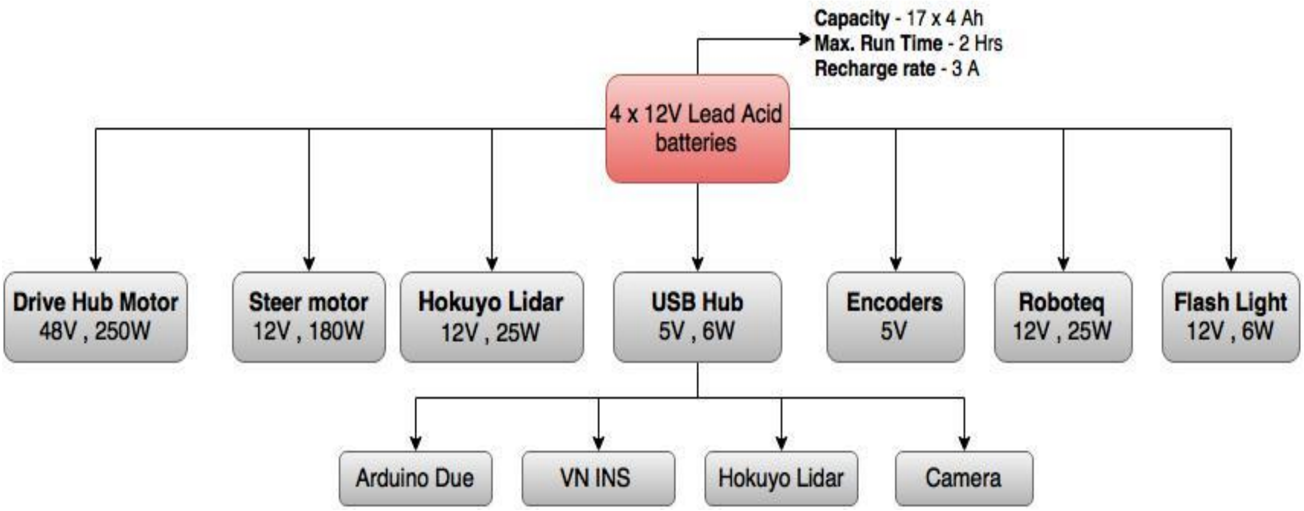


Figure 7. Electronic Architecture

Power Distribution

The power distribution flow is shown below.

Figure: **Power Distribution Flow**



Battery Management System

The previous versions of Eklavya faced problems regarding batteries and their management. State of charge, state of health, estimated time for complete discharge were not monitored and hence there was a possibility of batteries going into deep cycle, further deteriorating their life [2]. The main goal of a battery management system is to monitor above stated parameters of batteries for their safety and take appropriate action for the same.

The battery management system for Eklavya 5.0 continuously monitors the variation of the battery voltage and accordingly displays the state of charge of each battery on 84 mm x 48 mm dot matrix LCD screen which has been installed on the robot. The voltage of each battery is proportionally scaled to logic level using potential dividers and is fed as analogue voltage input to the microcontroller unit, Arduino Nano, which reads the input and displays the state of charge on the screen accordingly. The current sourced by sensors and motors varies in such a way that total charge cannot be obtained with the generic methodology. Hence, we resorted to method of estimating the charge left by deriving the discharge curves. Thus, estimating the charge left by obtaining the battery voltages itself. The discharge curves of the batteries were derived after accounting various discharge cycles [3].

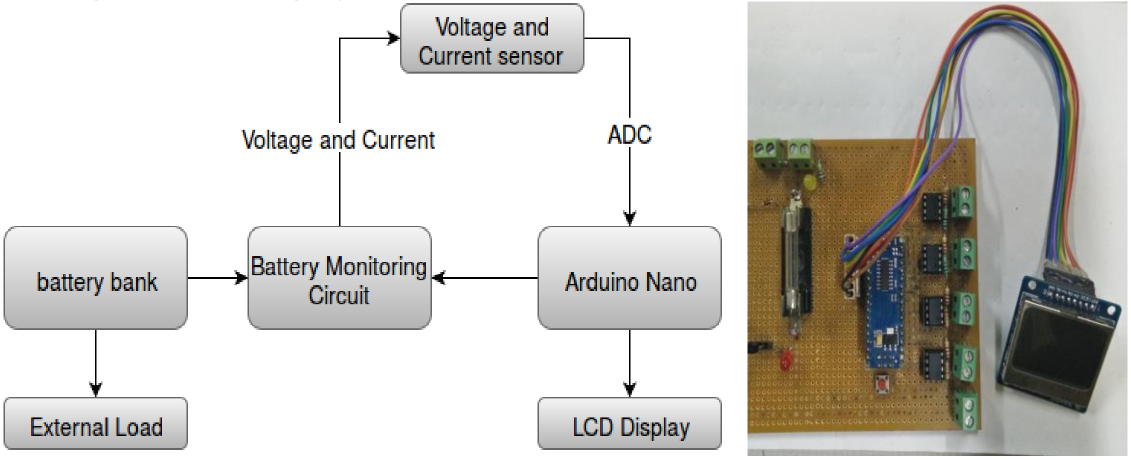


Figure 10. Battery monitoring System

SENSORS and ACTUATORS

Sensors	Specifications
1. Autonics E80H Encoders	• 10 Bit Resolution • hollow shaft Quadrature Type • 6 Channel - 4 Output , 2 for Verification
2. Genius Webcam	• 120 degrees ultra wide angle view at 30 FPS • 12 MP , 1080p Image view • Manual Focus with Glass lens
3. Vectornav VN-200 INS	• 3-axis accelerometer, 3-axis gyrometer, 3-axis magnetometer, barometric pressure sensor. • GPS-aided Inertial Navigation System (INS). • Low power input 0.5 W • Accurate Signal output owing to Internal Kalman Filtering
4. Hokuyo UTM-30LX LIDAR	• Range of 30 m in 270 degree Plane of device • Millimeter resolution in a 270° arc. • Accuracy ±50 mm within a range of 0.1-30 m

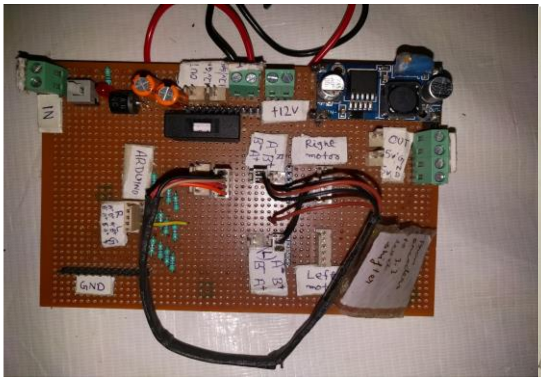


Figure 8. Power circuit for sensors

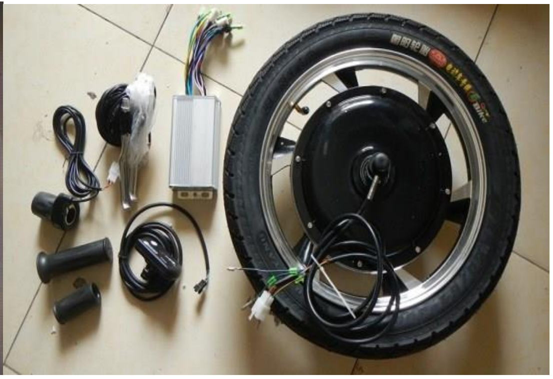


Figure 9. BLDC hub motor

Actuators	Specifications
1. Brushless DC Hub Motor	• Reduces Space consumed by conventional DC motor • **Operating Voltage**: - 48V. • **Current** :- Max - 9 Amp Normal - 7 Amp • 5 Pin hall effect wiring , 3 stator wire • Speed control with specified Analog value
2. DC Steer Motor	• Inline Motor for compatibility with steering Column • **Operating Voltage** :- 12V • **Current** :- Max - 15 A Normal - 10 A • **Torque** :- 100-125 IN-LBS • 12 Bit resolution optical encoder for feedback • Compatible with Roboteq

CONTROL SYSTEM

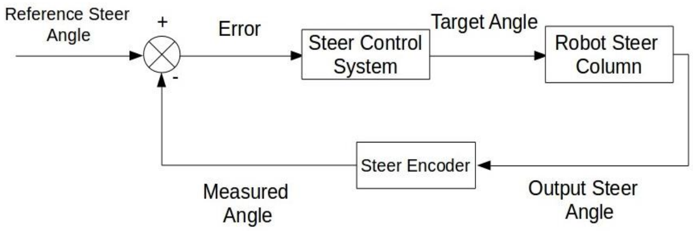
The speed control system, curvature control system and an angle control system are the three main control systems working in Eklavya 5.0. The steering angle control is implemented on a Roboteq motor controller while the other two controllers are implemented in the C++ code running on the main computing platform of the robot.

Speed Control System

The speed control system tries to reject the environmental disturbances and tracks the given speed unit step commands. The control action is actuated using a BLDC hub motor. As such, the controller is a mixed-signal control system as the BLDC motor runs on analogue voltage values while the rest of the control system, viz. the controller, the speed measurements and reference commands are in digital domain. A Digital to Analog Converter (DAC) converts the digital control input signal to analogue voltage command to control the speed of the BLDC motor. The speed control is an experimentally tuned PID controller implemented on the C++ code. PID control scheme is chosen because of its ease of implementation and the degree of freedom of tuning three parameters to achieve better performance. The speed feedback is obtained using the two rear wheel encoders.

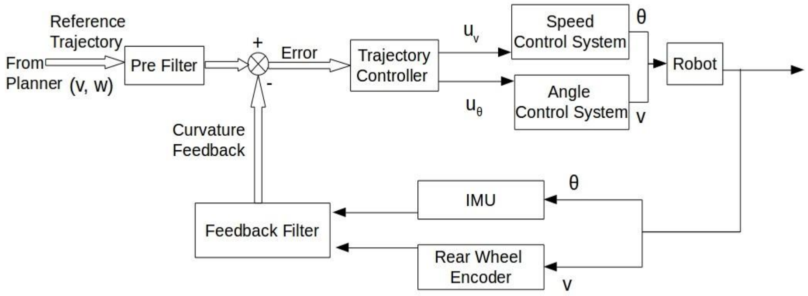
The experimentally tuned PID control scheme was verified by simulations on MATLAB. Using system identification techniques [4], a transfer function model was obtained for the BLDC hub motor. For the obtained transfer function, a PID controller was designed and performance was simulated on MATLAB.

Angle Control System



Similar to the speed control system, the steer angle is controlled using a PID controller implemented on a Roboteq motor controller. The angle feedback is obtained using an optical encoder placed on the shaft of the motor. The Roborun utility of Roboteq helps in tuning the performance of the steering angle control system. The following block diagram explains the implemented control scheme. Verification of the results was done using simulations on MATLAB by identifying the parameters of a second order transfer function. The controlled responses were plotted and hence the experimental tuning was verified using simulations on MATLAB.

Curvature Control System



This is the most important part of the control system of Eklavya 5.0 as it tries to follow the trajectories, the motion planning algorithm generates. The radius of curvature of the instantaneous axis of rotation is calculated using the translation speed (calculated as the average of the two rear wheel speeds measured by the encoders) and the angular velocity data given by the Inertial Measurement Unit (IMU). This feedback is compared with the desired radius of curvature given by the planner and an experimentally tuned PID controller is implemented on the C++ code. The following block diagram describes the control system in detail. The curvature control system feeds the angle and speed control systems as shown with their respective reference commands. We have assumed that there is either no or negligible coupling between the three control systems.

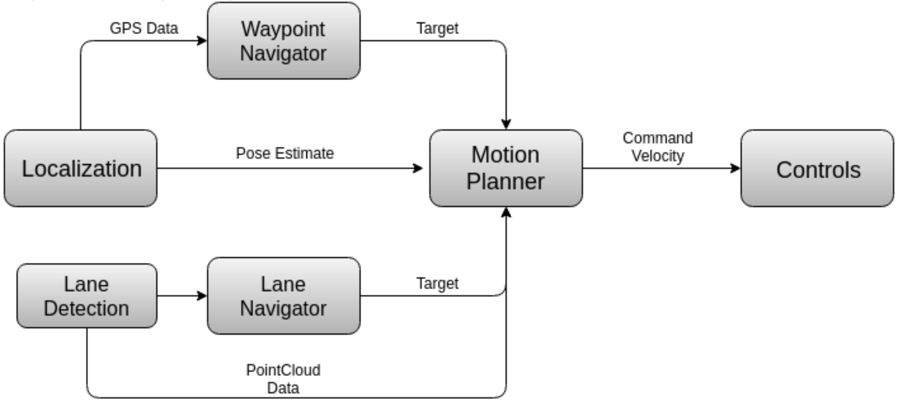
Safety systems and their integration

In order to ensure that the sensors sensitive to the sudden voltage change are always electrically safe, the power circuit of all the components are designed in such a way by using proper voltage regulators, Buck converters, capacitors, diodes and fuses that always clean dc voltage is supplied. The fuses of proper rating are used, along with it LED indicators, which indicate any power cut.

Battery Management System ensures that the batteries never enter deep discharge mode by alarming the user at lower voltages.

Overview of Software

The following block diagram gives an overview of the software architecture of the robot.

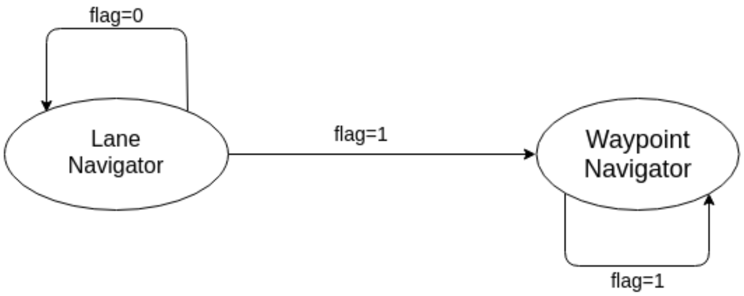


Obstacle Detection and Avoidance

The white strips in the obstacles and the white ladders interfere with the lane detection algorithm as they occur as false positives and thus have to be removed before lane detection. This problem was not dealt with in Eklavya 4.0 and has been successfully solved in the new version as follows. First, median filter is applied. Then we apply Canny edge detection on this image. As, after edge detection, very few obstacle points will be left, they won’t interfere in the lane navigation algorithm. Hence, by this new approach we have bypassed the obstacle interference in a very novel and easy way. Then erosion and dilation is applied on the image to filter out random noises. Along with this, Circular Hough Transforms are used to detect and remove potholes.

Software Strategy and Path Planning

High Level Planner



The high level planner of Eklavya 5.0 has been implemented using the concept of FSM (finite state machine). The two most important states of our FSM are - lane navigator state and waypoint navigator state. The transition between states is governed by the following observations of the bot

1. If the bot is not in no man’s land and can see the lanes, we switch to the lane navigator state of the FSM.
2. When the FSM is in its lane navigator state and distance of a waypoint is less than a predefined threshold, then the FSM switches to waypoint navigator state.

Motion Planning

In Eklavya 4.0, we had used the ROS move_base node for the purpose of path planning. However, that planner didn’t work well in our case as it didn’t always generate kinematically feasible trajectories.

One of the most important advantages of this planer compared to many other planners is that plans a kinematically feasible path which out bot can achieve. Also, it is a lot faster compared to planners which employ algorithms like Dijkstra’s and A*. Our planner has compromised on optimality for speed which is acceptable for this purpose.

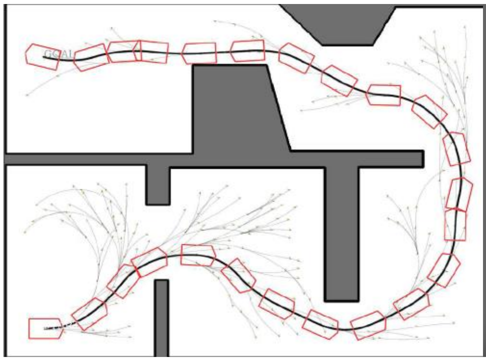


Figure 13. TP-RRT- an overview

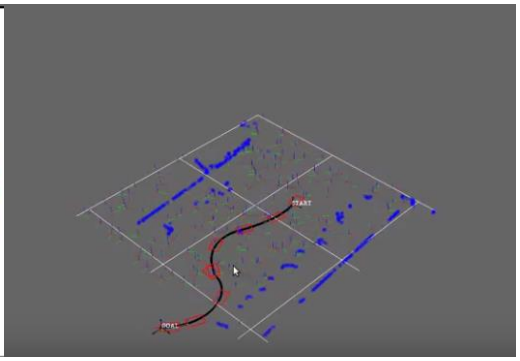


Figure 14. Path planned by TP-RRT

In Eklavya 5.0, the team has used the TP Space-RRT algorithm [5] in the planner. The TP-RRT planner first converts the entire frame into TP (trajectory parameter) space [6] Wherein the RRT (rapidly exploring random tree) algorithm is used. The algorithm incrementally builds a tree of collision-free trajectories rooted at the initial condition. Hence, RRT is initialized as a tree, including the initial state as its unique vertex and no edges. Next, several families of trajectories (PTGs-Parameterized Trajectory Generators) are employed while attempting to grow the tree using random intermediate targets .The most suitable path is chosen after the tree reaches the target node along the expanded tree keeping in mind the kinematic constraints of the bot. In our code, we don’t directly apply the RRT algorithm to the free-object space. We further filter it to a space in which the states of RRT are such that each one of them can be achieved by the bot and this is how the bot gets its holonomic nature.

Map Generation

Localization

We have localized our bot using an extended Kalman filter algorithm (same as previous year) by estimating x, y, θ (yaw) and their differentials from IMU, GPS and encoder data [7]. Last year we were facing problems while integrating GPS data into the filter, especially when the data was inaccurate in areas like Kharagpur, India. This time we tuned the covariance matrices and used an average of 100 iterations GPS data to set the origin in the GPS frame. With this we were able to achieve errors as low as 0.2m (in x and y directions) after following a closed loop path of perimeter 400m. For our purpose we have used two frames. The bot is localized in the ‘odom’ frame (starting point is taken as the origin and the frame drifts over time due to odometry errors). The bot frame is assumed to be ‘base_link’ in our case (i.e. what the bot sees at a particular instant).

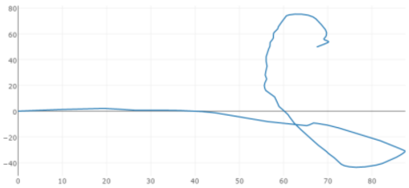


Figure 14. Data from encoders
(drift error being integrated over time)

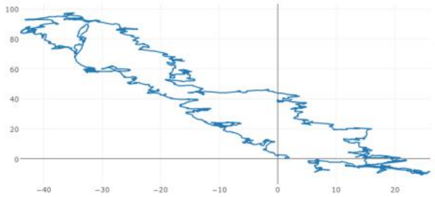


Figure 15. Data from GPS
(axis rotated 90^o)

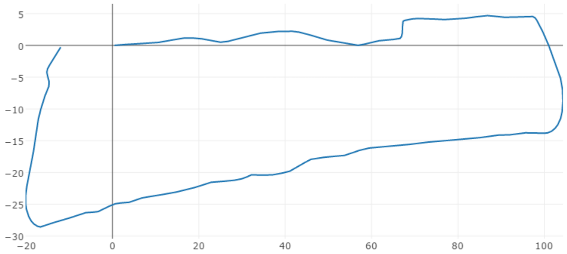


Figure 16. Filtered data using EKF

Mapping

For both lane and waypoint, we use LIDAR data to find out the obstacle around the vehicle space. First we convert the LIDAR data to a point cloud in ‘base_link’ frame. For the cost map of the lane navigator and waypoint navigator, we fuse the Point Cloud of the lanes with the LIDAR data and finally convert the resulting point cloud to ‘odom’ frame.

Goal Selection and Path Generation

Lane Detection

We have removed the grassy portions of the image with a SVM classifier [8] where features for learning were taken as a kernel of an 8×8 ROI of the image. This kernel was classified as grass or non-grass using a polynomial SVM classifier.

As shadows change the HSV values of regions slightly, when the effects became more prominent the classifier was unable to produce satisfactory results. So, a shadow removal technique was used. The image was first converted to the YCrCb colour space all pixels with intensity less than 1.5 times the standard deviation of Y channel were classified as shadow pixels and the image was converted into a binary one [9].

Curves were generated by the classifier based on results over shadow removed images. Although this gives a few false positives, most of the lanes are classified as non-grass. Also, grass offered a more uniform patch compared to lanes as the lane portions in the image varied with variations in brightness and lightning conditions. Lanes also exhibit non-uniform thickness. Both the thresholding and Hough line method could still output false lanes, especially in thresholding as it is very difficult to find fine threshold values. So we incorporated Random Sample Consensus (RANSAC) to detect lanes. On rigorous testing, RANSAC was found to be a reliable technique for curve-fitting. Finally the image was transformed to a top down view using inverse perspective transform (IPT).

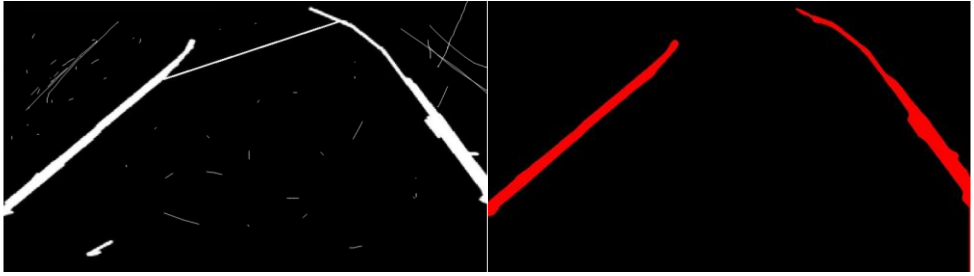
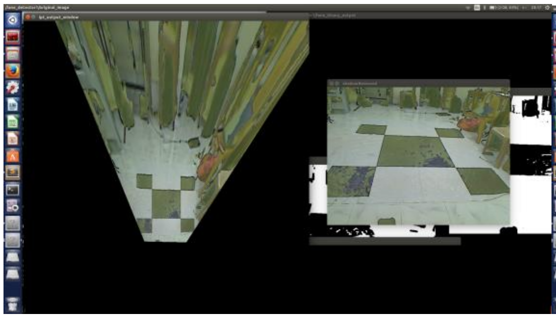


Figure 17. Detection of lanes after removing noise

We further observed that the height of the camera has to be increased as compared to Eklavya 4.0 to account for the fact that obstacles blocked the view of lanes behind it. Also, since classifying single lanes as right or left and giving a target is less favourable than the double lane case we have used a 120° FOV camera instead of the 75° FOV camera used last year.



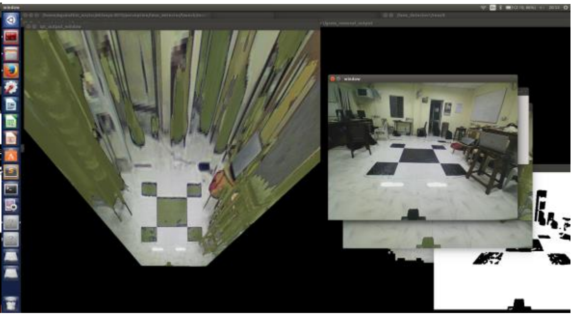


Figure 18. Results from 75° FOV camera **Figure 19. Results from 120° FOV camera**

Flag detection

The flags are detected using HSV thresholding for red and blue colours. The algorithm is provided with parameters that can be modified dynamically. This helps us to calibrate to the external environment quickly.

Potholes detection

This module is being planned to be implemented using circular Hough transform, which detects circles from points on the circumference and selects the maxima from the accumulator matrix.

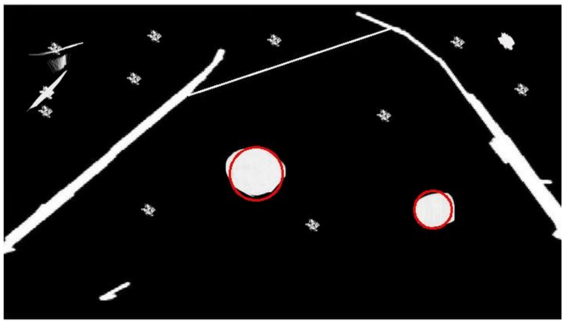


Figure 20. Result for Potholes detection

Lane Navigation

The lane navigation algorithm has been explained with the help of following flow chart.

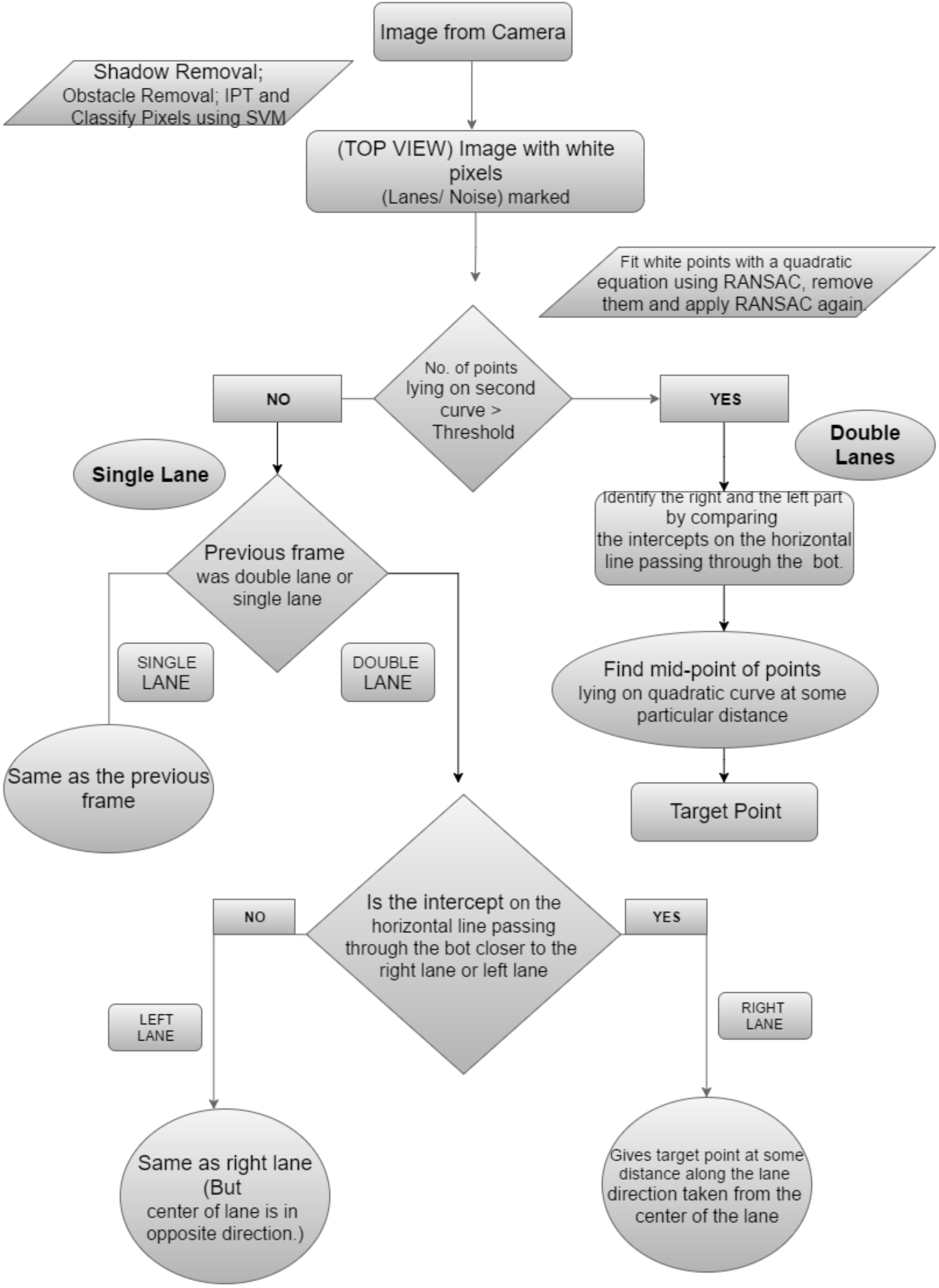


Image from Camera
Shadow Removal; Obstacle Removal; IPT and Classify Pixels using SVM
(TOP VIEW) Image with white pixels (Lanes/ Noise) marked
Fit white points with a quadratic equation using RANSAC, remove them and apply RANSAC again.
No. of points lying on second curve > Threshold
NO
YES
Single Lane
Double Lanes
Previous frame was double lane or single lane
SINGLE LANE
DOUBLE LANE
Same as the previous frame
Identify the right and the left part by comparing the intercepts on the horizontal line passing through the bot.
Find mid-point of points lying on quadratic curve at some particular distance
Target Point
Is the intercept on the horizontal line passing through the bot closer to the right lane or left lane
NO
YES
LEFT LANE
RIGHT LANE
Same as right lane (But center of lane is in opposite direction.)
Gives target point at some distance along the lane direction taken from the center of the lane

Figure 21. Flow diagram to determine target for Lane navigation

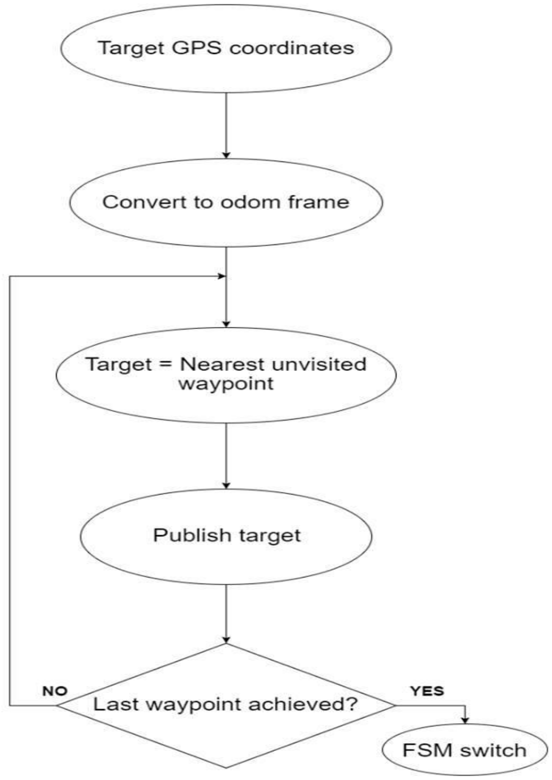


Figure 22. Waypoint navigator Flowchart

Waypoint Navigation

The Waypoint navigator first goes to the nearest waypoint and then traverses all the waypoints by visiting the nearest one at each step. When the last waypoint is reached and the lanes are detected, FSM switches the state to Lane navigator.

Additional Creative Concepts

For lane navigation, we used the concept of "Tracking" to distinguish between single and double lanes and to further distinguish between right and left lane. We keep a track of the previous frame at every instant and on the point of transition from double lanes to single lane, we compare the distance of the single lane from both the lanes of the previous frame and check whether it is right or left.

We have applied Canny edge detection on the image before applying quadratic curve fitting. This makes sure that the white portion in the obstacles doesn't interfere in the curve fitting part. To minimize the errors due to GPS, instead of calculating the target at every step using the fluctuating GPS data, we have converted all the waypoint targets into odom frame in the first iteration itself by using the GPS coordinate of the origin of the odom frame.

Simulation

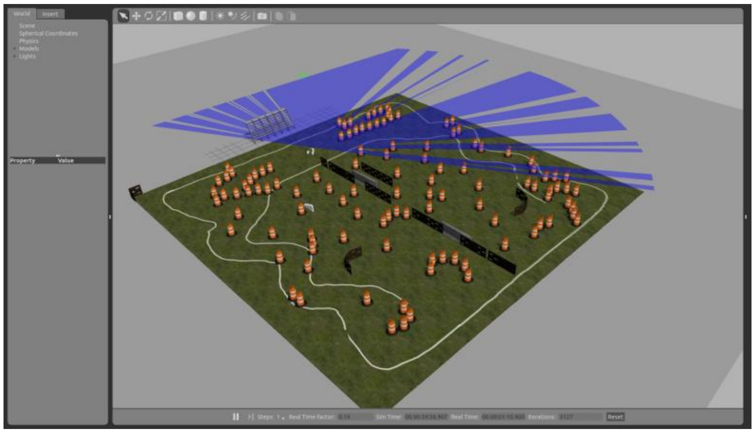


Figure 20. Simulation Arena - Gazebo

We use Gazebo as the simulation software for our vehicle. We have constructed a close to real representation of the robot as well as the IGVC course. To simulate real life robustness of our code, we have added noise to the readings of the sensors. The IGVC course has been realistically portrayed so as to test our code on the actual course.

Constructional Features of the Simulation

The SolidWorks model of our bot has been imported as a mesh in Gazebo and the sensors used in our bot have been simulated with errors as per specification when available or with experimental data. We wrote the controller plugin specifically for our front steered bot to convert command velocity into steer angle and rpm of the wheels.

Failure Modes and Resolutions

- **Lane Detection:** In lane detection, the code fails in the case where proposed target lies on an obstacle. We have resolved the issue by taking input from the LIDAR and checking whether the goal lies on an obstacles or not and adjust the final goal accordingly.
- **Localization**: The bot experiences a drift in its odometry in case of wheel slippage. For the correct localization of the bot using GPS data, there should be adequate number of satellites present (i.e. greater than 4). Also, the IMU unit should be at the centre of the bot in 'base_link' frame, which in our case is the centre of back wheels.
- **TP-RRT Planner:** The planner does not alter the path of bot in presence of dynamic obstacles.
- **Power Management:** Failure mode LED indicators are placed at the power source of BLDC motor, Encoder channels and Steer Motor corresponding to fuse blow, low battery and short circuit.
- **Control System:** If the tuned PID fails, PID can be re-tuned easily by changing the parameters in a launch file.
- **Plate coupler failure-**Steer column will break from the main frame if the normal stress in the bolts exceed 19.4 MPa.
- The bearing will fail in case of rusting, high spots in cup seats, corrosion, etc.

Performance Testing

- Max Acceleration: 2.548 m/s2
- Max torque without skidding: 51 Nm
- Average driving force on the bot: 255 N
- Average Motor torque: 18.53 Nm
- Average speed: 5.6 mph.
- Ramp climbing ability at 30 degrees -1.56 m/s^2

References

[1] Y. Karim and C. Blanzé. "Vibration reduction of a structure by design and control of a bolted joint" *LMSSC, CNAM, 2 rue Conté, 75003 Paris, France*

[2] C. Chen, et.al. "Design and Realization of Smart battery management"

[3] Gerald P. Arada and Elmer R. Magsino "Development of a Power Monitoring System for Backup Lead-Acid Batteries"

[4] System Identification Toolbox MATLAB

[5] Jose Luis Blanco, Mauro Bellone and Antonio Gimenez-Fernandez. "TP-Space RRT – Kinematic Path Planning of Nonholonomic Any-Shape Vehicles". *Int J Adv Robot Syst*, 2015

[6] Jose-Luis Blanco, Javier González, and Juan Antonio Fernández-Madrigal. "Extending obstacle avoidance methods through multiple parameter space transformations. Autonomous Robots" 2008

[7] Sebastian Thrun, "Probabilistic Robotics"

[8] Zhou, Shengyan, et al. "Road detection using support vector machine based on online learning and evaluation."*Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010.

[9] Deb, Kaushik, and Ashraful Huq Suny. "Shadow Detection and Removal Based on YCbCr Color Space." *Smart CR* 4 (2014): 23-33.