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 Excellence in Robotics, IIT Kharagpur, has been pioneering the autonomous ground vehicle technology with the ultimate aim of developing India's first self-driving car. The team has been participating in IGVC since 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 in all aspects i.e. mechanical, electrical and software.

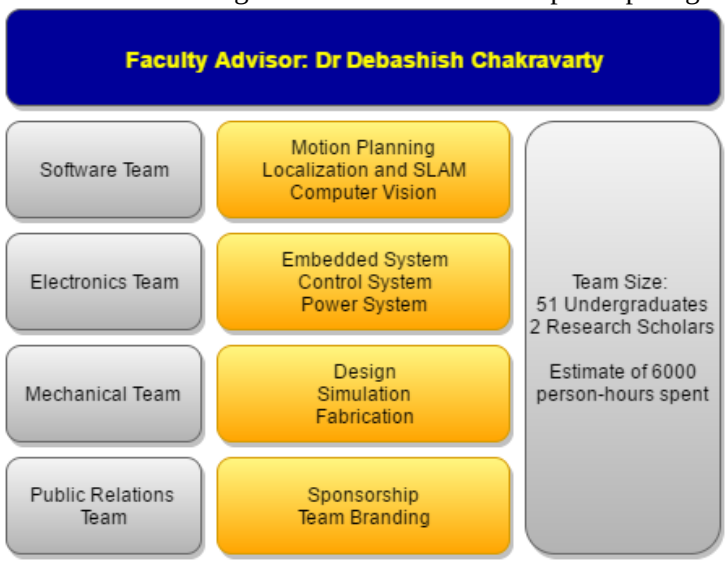


Figure 0: Team Organization

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.

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

DESIGN PROCESS

The failure points of Eklavya 4.0, which participated at IGVC 2015, were thoroughly analyzed Figure (1) describes the major improvements made in Eklavya 5.0.

While designing Eklavya 5.0, some assumptions were taken into account such as - there was no skidding of wheels which meant that the velocities obtained from encoder signals are true, around a centre of curvature, which paved the way for designing the control systems.

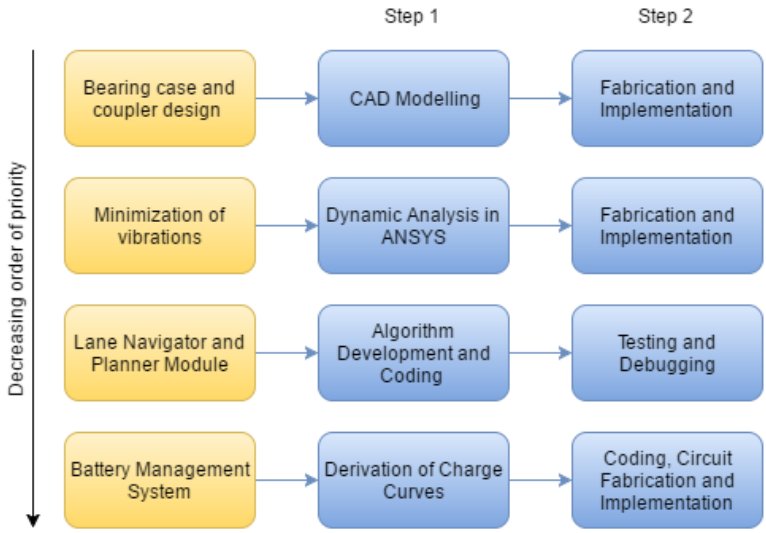


Figure 1: Design process for the vehicle

A new path planning module was built in order to generate kinematically feasible trajectories for our bot. In addition to that, a robust lane navigator was designed after testing on numerous corner case scenarios. The design considerations and the process for Eklavya 5.0 are shown in Figure (1) after incorporating the above mentioned improvements and assumptions.

MECHANICAL DESIGN

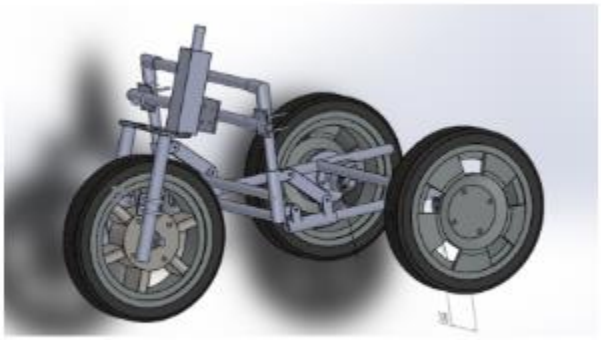


Figure 2. Mechanical Design

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

While designing Eklavya 5.0 (Figure (2)), achieving maximum stability by lowering the centre of gravity and reducing vibrations were the two major concerns. The steering column, when at first connected to the frame through a flat plate, did not produce enough opposing torque to nullify the induced moment of the drive motor. Therefore another link was added to support the other dynamic forces acting on the flat plate joint, and thus reducing the longitudinal vibrations [1]. The height of the camera mount was not sufficient in Eklavya 4.0 for efficient lane navigation. To tackle this, the height of the bot and the caster angle of the front wheel were considered and calculations the optimal height for camera placement came out to be 5.5 ft. Finally, to reduce the transverse vibrations, the design of the bearing case was modified. In order to keep the design simple, compatible and light weight, no suspension system is installed in the robot.

STEERING COLUMN

The drive motor is attached to the steering column which causes both radial and axial loading. The angle of inclination of the steering column with the horizontal was calculated to be 20 degrees. This led to less radial loading which further lowered the torque requirement for steering the vehicle. In addition to that, the steering column is designed to be self-centred which helps the bot to move forward easily. The final moment diagram of the steering column and the final manufactured column are shown in Figures (4) and (3) respectively.

Figure 3: **Manufactured Steering Column**

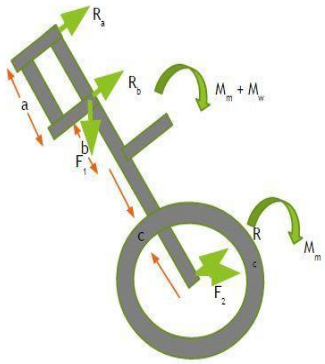


Figure 4: **Moment diagram of the steering column**

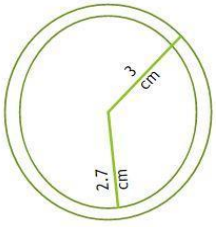
R_a = Reaction due to upper bearing
R_b = Reaction due to lower bearing
F_1 = Weight acting on steering stem
M_m = Torque provided by motor
M_w = Torque due to weight of motor
F_2 = Force due to acceleration
R_c = Reaction from tire

Table 1. 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

Stress = My/I

For the dimensions, 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°, Length of steering T stem: 20 cm, Diameter of Fork: 3 cm, Thickness of fork: 3 mm

These dimensions are similar to that of a motorbike's steering column, so the steering stem and forks of a Hero Honda Aviator scooter were used.

The Reduction of Longitudinal Vibration

The longitudinal vibrations [1] were reduced with the introduction of a new rod. This is evident from the force analysis done using Ansys (Figure (5)).

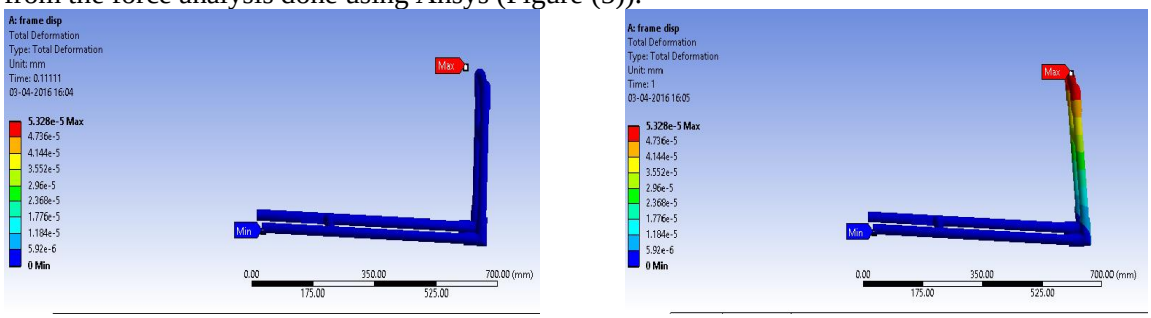


Figure 5: Comparison made when load is applied with the support rod and without support rod

WHEEL HUB DESIGN

Front Wheel

For translation, a 16-inch wheel with an attached hub motor is used. The wheel is attached to the fork with the help of U-clamps and the load is transferred effectively via two mild steel couplers.

Rear Wheels

Tapered roller bearings were chosen because of their capability of carrying loads in both axial and radial directions. This discards the need for thrust bearings which creates a problem in disassembling the robot. A pair of tapered roller bearings can be arranged 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. A pair of tapered roller bearings adjusted in back to back arrangement was used by us, as it provides rigid support to handle the weight transferred to the wheel hub.

The stress analysis of the front wheel performed using Ansys supports the assumption (Figure (7)).

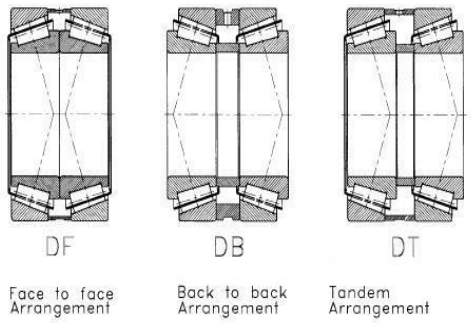


Figure 6: Roller bearing

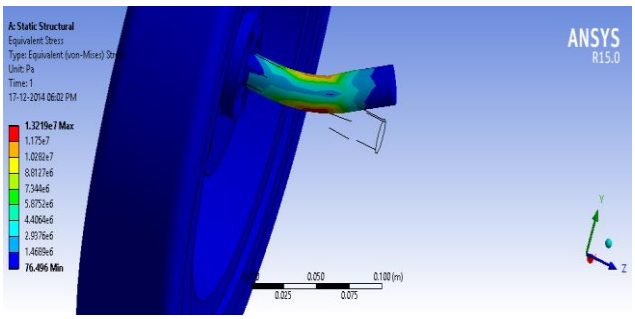


Figure 7: ANSYS stress analysis of front wheel.

ELECTRONIC AND POWER DESIGN

Overview

The electrical system overview is detailed in Figure (8).

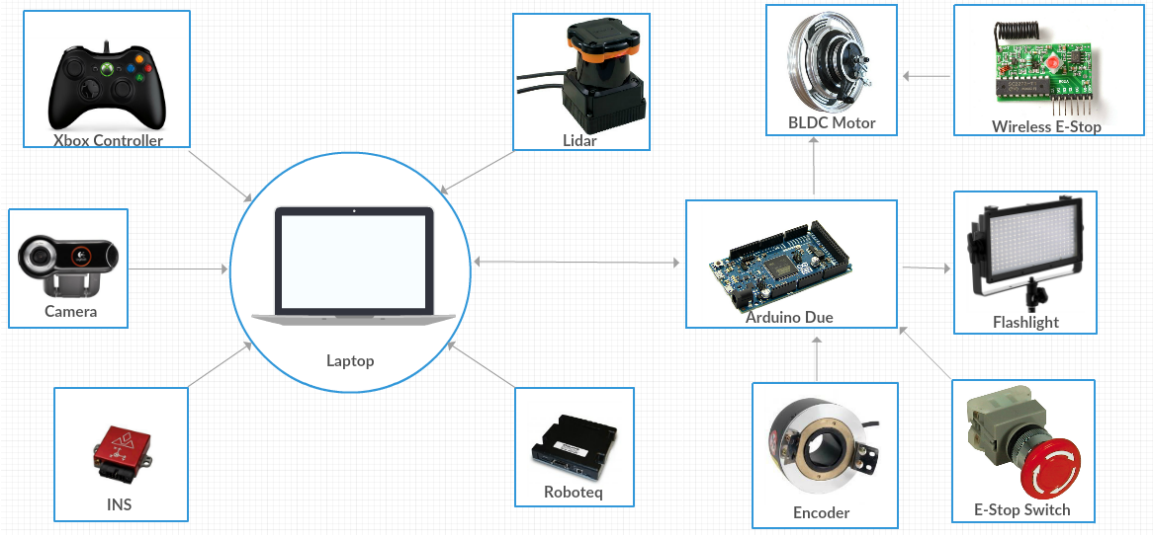


Figure 8: Electronic Architecture

Power Distribution

Figure (9) briefly describes the Power Distribution in Eklavya 5.0. The fabricated circuit developed by the team is shown in Figure (11).

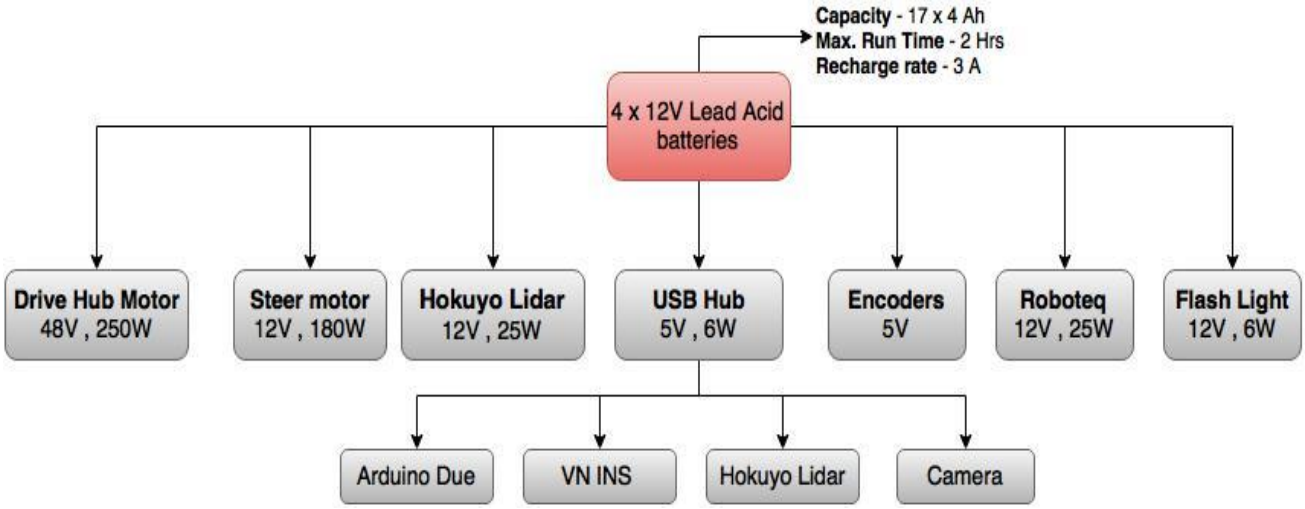


Figure 9: Power Distribution Flow

Battery Management System

The previous versions of Eklavya faced problems with 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 decreasing their longetivity[2]. The main goal of a battery management system is to monitor the 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 for each battery on a 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 5V logic levels using potential dividers and is fed as an analog voltage input to the microcontroller, Arduino Nano, which reads the input and displays the state of charge on the screen accordingly. As the current consumed by sensors and motors varies in such a way that total charge cannot be obtained with a generic methodology, a method of estimating the charge left by deriving the discharge curves was employed. Thus, the charge left by evaluating the battery voltages itself is calculated. The discharge curves of the batteries were experimentally obtained from many discharge cycles observations [3]. The working principle of the battery management system is briefed in Figure (10).

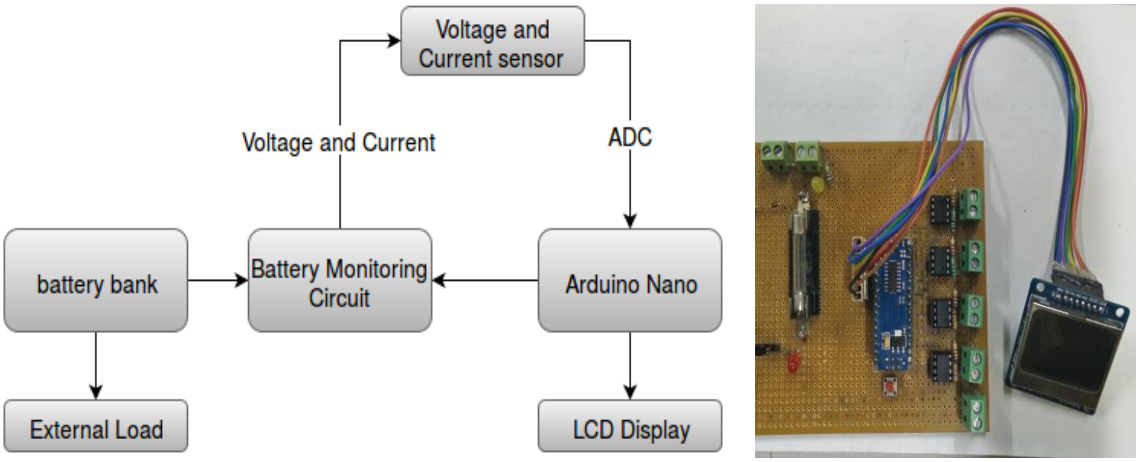


Figure 10: Battery monitoring System

SENSORS and ACTUATORS

Table 2. Specification for sensors.

Sensors	Specifications
1. Autonics E80H Encoders	• 10 Bit Resolution • hollow shaft Quadrature Type • 6 Channel - 4 Output , 2 for Verification
2. Genius Widecam F100 Camera	• 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 gyroscope, 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

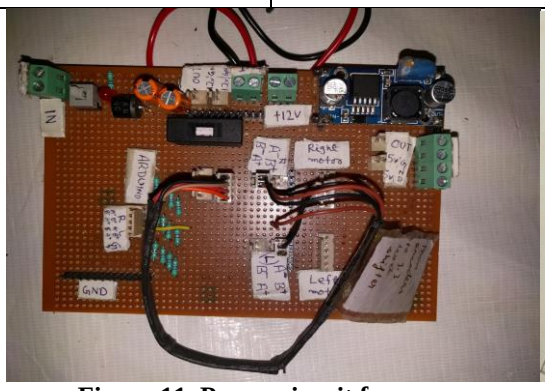


Figure 11. Power circuit for sensors

Figure 12. BLDC hub motor

Table 3. Specification for Actuators

Actuators	Specifications
1. Brushless DC Hub Motor (Figure 12)	• 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

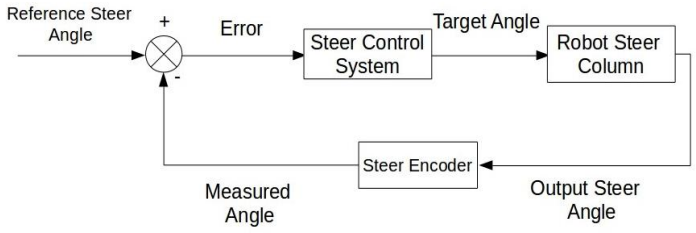


Figure 13: Block Diagram for Angle control

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 block diagram in Figure (13) 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

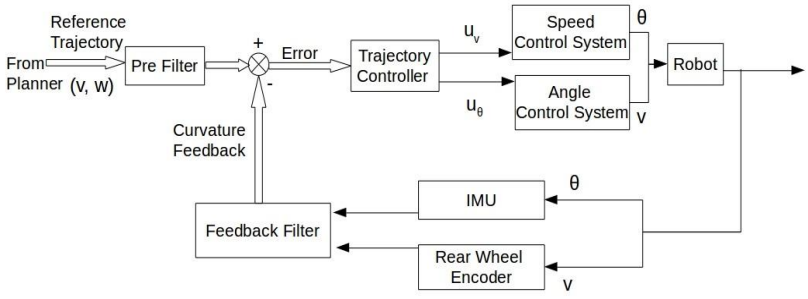


Figure 14: Block Diagram for Curvature Control

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 block diagram in Figure (14) describes the control system in detail. The curvature control system feeds the angle and speed control systems as shown with their respective reference commands. A simplifying assumption is 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

Figure (15) gives an overview of the software architecture of the robot. The details of each of the blocks are presented in detail in the following sections.

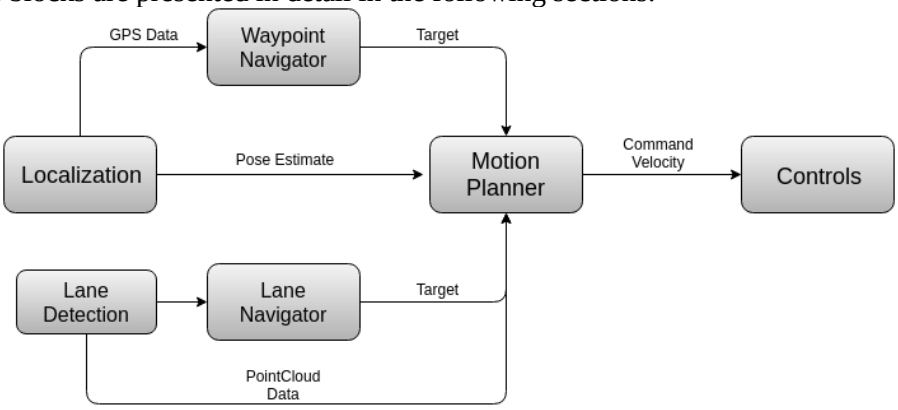


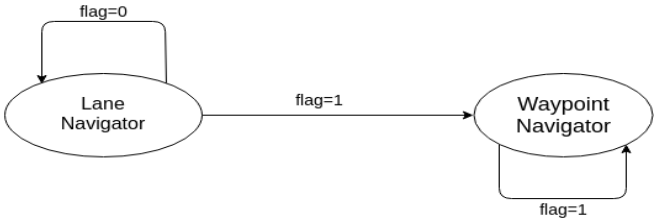
Figure 15: Overview for Software Architecture

Obstacle Detection and Avoidance

The white strips in the obstacles 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 was successfully solved in this new version. First, a median filter was applied. Next, using a Canny edge detection technique, the remainder few obstacle points do not interfere in the lane navigation algorithm. Hence, with this new approach the obstacle interference was bypassed in a very novel and simple way. In conjunction to this, circular Hough transforms were used to detect and remove potholes.

Software Strategy and Path Planning

High Level Planner



The high level planner of Eklavya 5.0 was implemented as an FSM (finite state machine). The two states of the FSM are - the lane navigator state and the waypoint navigator state. The transition between these states is governed by the

following observations.

Figure 16: High level planner FSM

1. If the bot is outside the no man's land and can see lanes, the lane navigator state of the FSM is switched on.

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, the inbuilt move base node of ROS was used for path planning. However, the planner didn't work well as it didn't generate kinematically feasible trajectories at all times for the front steered bot.

To solve this problem, the TP-RRT planner was implemented on Eklavya 5.0. Compared to many other planners, it has an advantage of planning a kinematically feasible path for the robot. Also, it is relatively faster compared to the planners which employ algorithms like Dijkstra's and A*. In this case, there is an acceptable trade-off between speed and optimality. Figure (17) and (18) show the result of TP-RRT planner implemented in Eklavya 5.0.

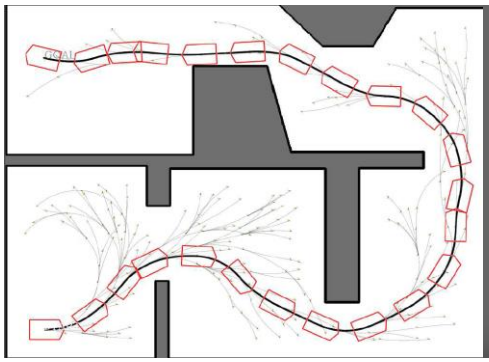


Figure 17. TP-RRT- an overview

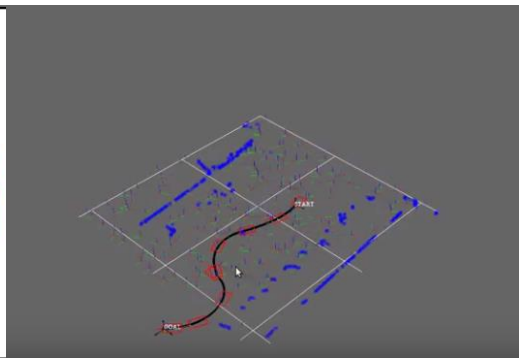


Figure 18. Path planned by TP-RRT

The TP-RRT planner implements the TP Space-RRT algorithm [5]. The planner first converts the entire frame into TP (trajectory parameter) space [6] wherein the RRT (rapidly exploring random tree) algorithm is used to plan the path to the target. 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 without any 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 the code, the RRT algorithm isn't directly applied to the free-object space. It is further filtered 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

The robot was localized 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]. In the last iteration, there were problems while integrating GPS data into the filter, especially when the satellite data was inaccurate at some places. In this iteration, the covariance matrices were tuned and an average of 100 iterations of GPS data was used to set the origin in the GPS frame. With this, errors as low as 0.2m (in x and y directions) were achieved after following a closed loop path of perimeter 400m. For localization, two frames were used. The bot was localized in the 'odom' frame (starting point taken as the origin and the frame drifting over time due to odometry errors). The bot frame was assumed to be 'base_link' (i.e. what the bot sees at a particular instant). Figures (19)-(21) show the error being eliminated using the filtering.

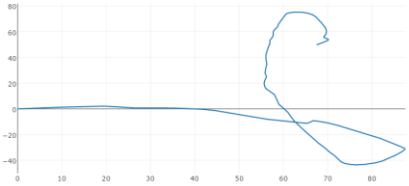


Figure 19. Data from encoders (drift error being integrated over time)

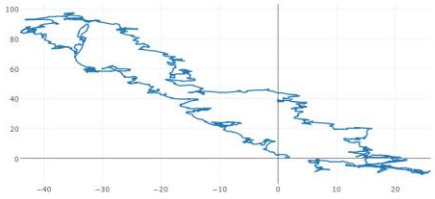


Figure 20. Data from GPS (axis rotated 90°)

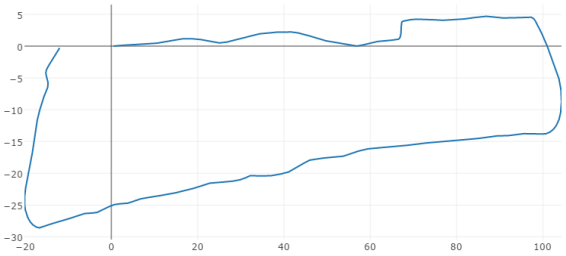


Figure 21: Filtered data using EKF

Mapping

For both lane and waypoint navigation, LIDAR data was used to find out the obstacles around the vehicle space. First the LIDAR data was converted to a point cloud in the 'base_link' frame (the body axis). To generate the cost map for the lane navigator and the waypoint navigator, the point cloud of the lanes was fused with the LIDAR data. Finally, resulting point cloud data was converted to 'odom' frame for navigation.

Goal Selection and Path Generation

Lane Detection

The grassy portions of the image were removed with an SVM (Support Vector Machine) 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 type using a polynomial SVM classifier.

The classifier was unable to generate satisfactory results due to the shadows which perturb the HSV values of the regions. Hence, a shadow removal technique was used. To that end, the image was first converted to the YCrCb colour space. Then, all the 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 binary [9].

Curves were generated by the classifier based on results over the shadow removed images. Although, this was prone to false positives, most of the lanes were classified as non-grass. Also, grass offers a more uniform patch as compared to lanes as the lane portions in the image vary in brightness and lightning conditions. Lanes also exhibit non-uniform thickness. Hence, both the thresholding and Hough line method would still output false lanes. This would even more be the case in thresholding, as it is very difficult to find fine threshold values. So, Random Sample Consensus (RANSAC) was incorporated 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 by using an inverse perspective transform (IPT). The output of the lane detection algorithm is shown in Figure (22).

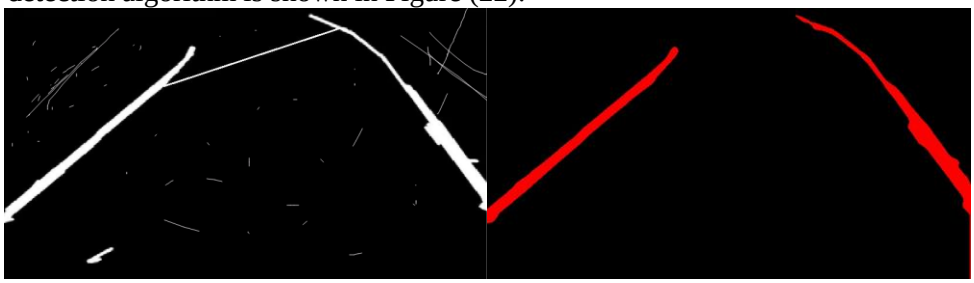


Figure 22: Detection of lanes after removing noise

It was further observed that the height of the camera had 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 favorable than the double lane case a 120° FOV (Field of View) camera was used instead of the 75° FOV camera used last year Figures (24) and (23) clearly show the improved performance with the higher FOV camera.

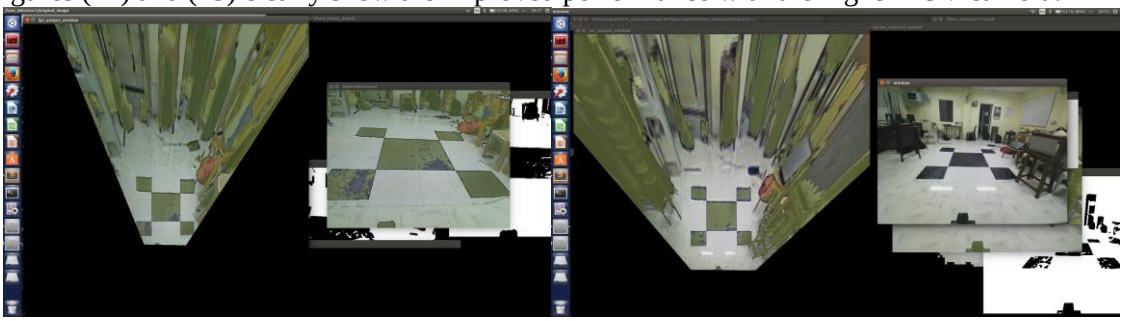
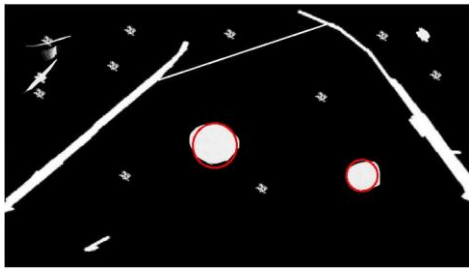


Figure 23: Results from 75° FOV camera

Figure 24: Results from 120° FOV camera

Flag detection

The flags were detected using HSV thresholding for red and blue colors. The algorithm was provided with



parameters that can be modified dynamically. This helped us to adapt to the external environment quickly.

Figure 25: Result for Potholes detection

Potholes detection

The potholes were detected using circular Hough transform, which identifies circles from points on the circumference and selects the maxima from the accumulator matrix as shown in Figure (25).

Lane Navigation

The lane navigation algorithm has been explained with the help of flow chart in Figure (26).

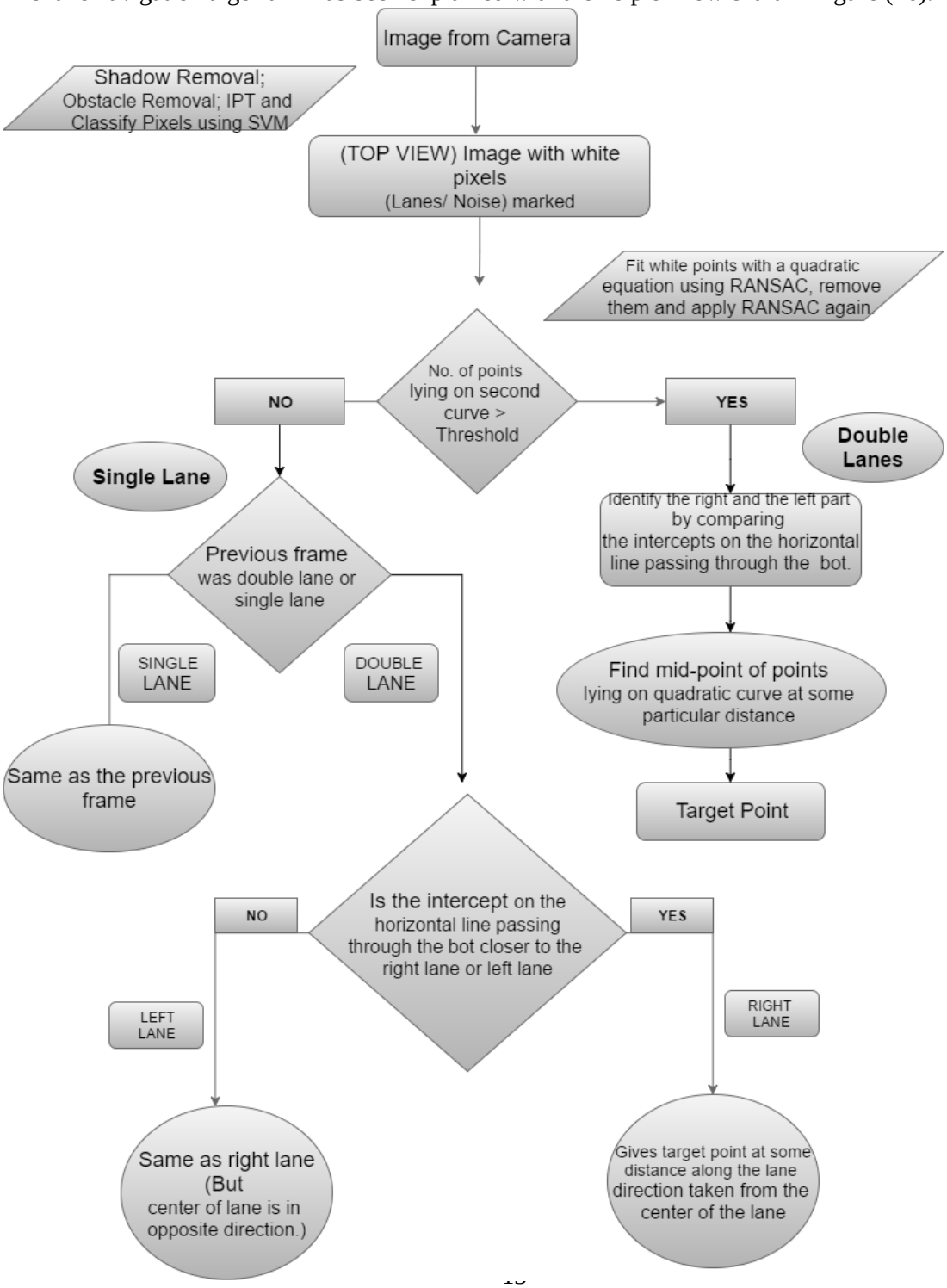
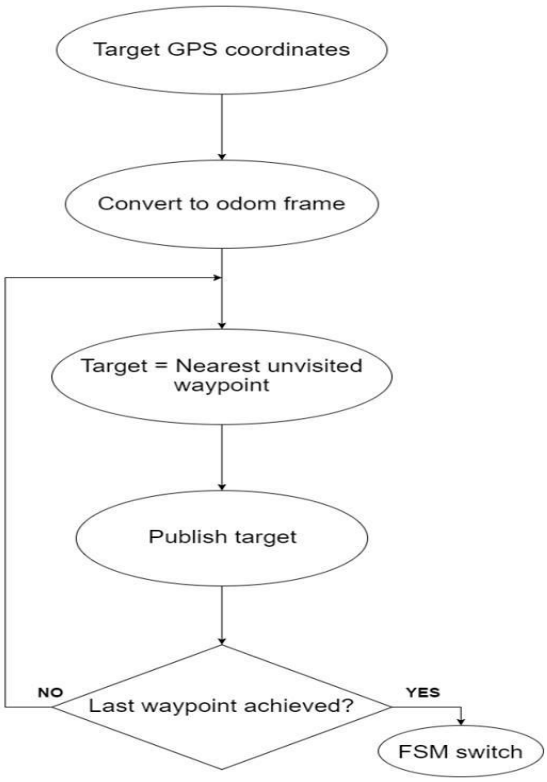


Figure 26: Flow diagram to determine target for Lane navigation



Waypoint Navigation

The Waypoint navigator first selects the target as the nearest waypoint and then traverses all the waypoints by visiting the nearest one at each step. The entire logic along with FSM switching is explained using the flowchart in Figure (27).

Additional Creative Concepts

For lane navigation, the concept of "Tracking" was used to distinguish between single and double lanes and to further distinguish between right and left lanes. A track of the previous frame at every instant was kept and on the point of transition from double lanes to single lane, the distances of the single lane from both the lanes of the previous frame were compared. This comparison yields left and right lane in this case.

Figure 27: Waypoint navigator Flowchart

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

Simulation

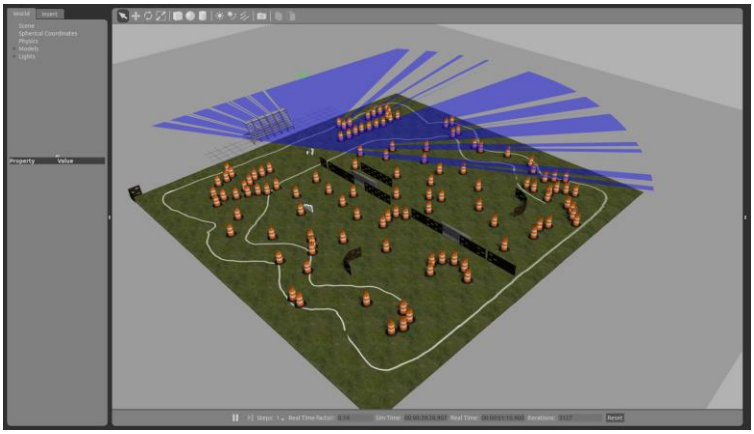


Figure 28. Simulation Arena - Gazebo

Gazebo was used as the simulation software for the vehicle. A close to real representation of the robot as well as the IGVC course was constructed. To analyze the real life robustness of the code, noise was added to the readings of the sensors. The IGVC course was realistically portrayed so as to simulate the code on the actual course.

The SolidWorks model of the bot was imported as a mesh in Gazebo and the sensors used in the bot were simulated with errors

as per specification when available or with experimental data. A view of implementation can be seen in Figure 28. The controller plugin was written specifically for the bot to convert command velocity published by the planner into steer angle and rpm of each of the wheels. \

Failure Modes and Resolutions

- **Lane Detection:** In lane detection, the code fails in the case where the proposed target lies on an obstacle. The issue was resolved by taking input from the LIDAR and checking whether the goal lies on an obstacle or not. The final goal is adjusted accordingly.
- **Localization**: The bot experiences a drift in its odometry in case of wheel slippage. For correct localization of the bot using GPS data, there should be an 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 for Eklavya 5.0 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 which light up on occurrence of fuse blow, low battery and/or short circuit.
- **Control System:** If the tuned PID fails, the PID can be re-tuned easily by changing the parameters in a launch file.
- **Plate coupler failure -** Steer column would break from the main frame if the normal stress in the bolts exceeds 19.4 MPa.
- The bearing would fail in case of rusting, high spots in cup seats, corrosion, etc.

Performance Testing

- Max torque without skidding: 51 Nm
- Max Acceleration: 2.548 m/s2
- 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.