

EKLAVYA 4.0
Indian Institute of Technology Kharagpur
The Autonomous Ground Vehicle Research Group (Team AGV)
Faculty Advisor: Dr. Debashish Chakravarty
Email: dc@mining.iitkgp.ernet.in

1 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 4.0, another feather in the cap of the Research Group is all set to participate in the 23rd Intelligent Ground Vehicle Competition (IGVC), Oakland University. With new robotic innovations, the successor of Eklavya 3.0, is a much more simplified and powerful Eklavya 4.0 in all aspects i.e mechanical, electrical and software. This report outlines the entire structure of Eklavya 4.0 listing out the innovations and improvements over the previous IGVC versions.

2 Team Organization

The effort behind this project was put in by a bunch of over thirty enthusiastic and intellectual undergraduate students from various departments of IIT Kharagpur. This research group (AGV) works under the able guidance of Prof. Debashish Chakraborty, Department of Mining Engineering, IIT Kharagpur, along with five co-professor-in-charges namely Prof. D K Pratihari, Prof. P P Das, Prof. S K Pal, Prof Manoj Mondal and Prof M Sinha. The team is divided into seven major modules, namely Machine Learning, Computer Vision, Localization & SLAM, Control Systems, Motion Planning, Mechanical and Public-Relations.

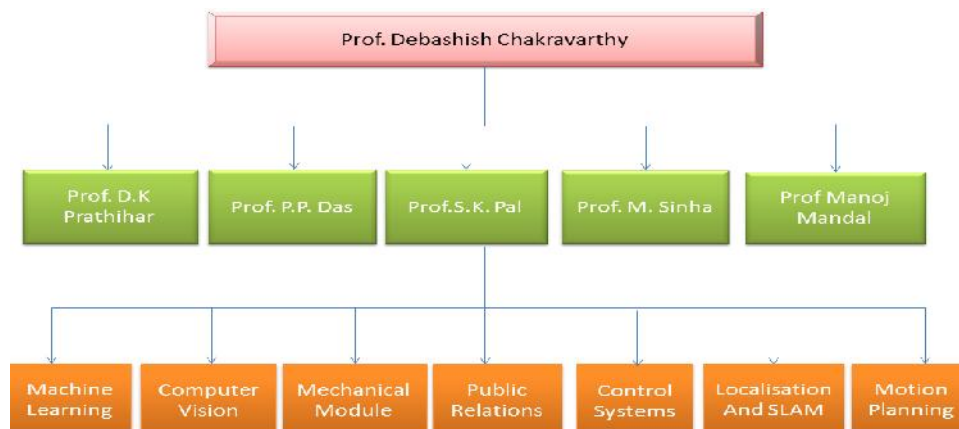


Figure 1. Team Organization

3 Mechanical Design

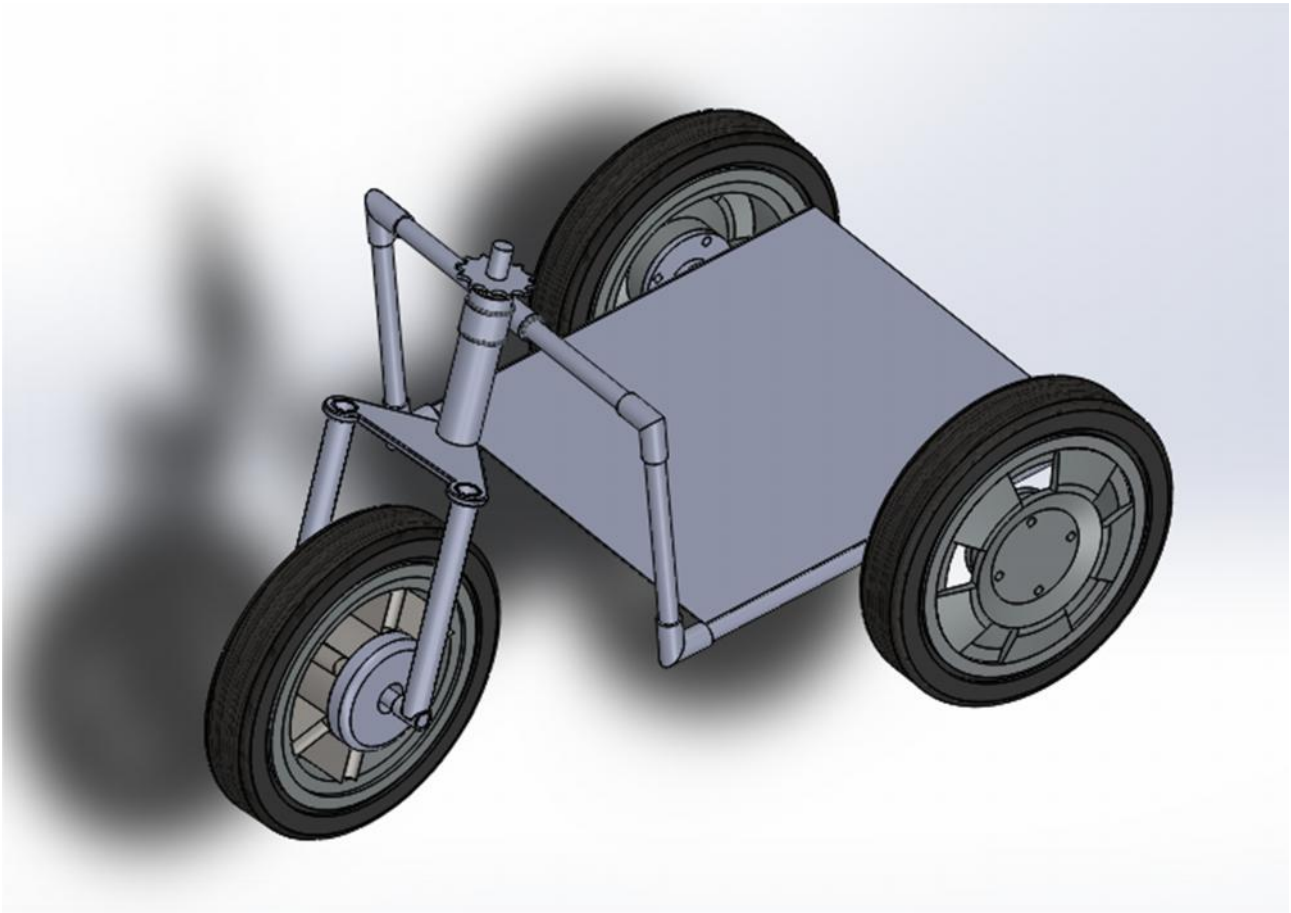


Figure 2. Eklavya 4.0 Chassis

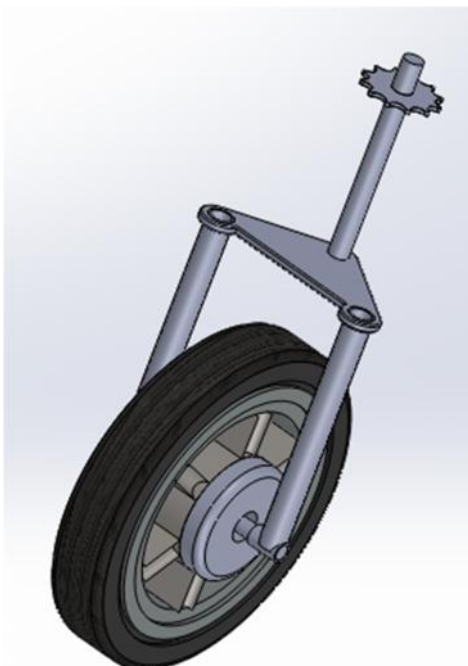


Figure 31. Eklavya 4.0 Steering Column

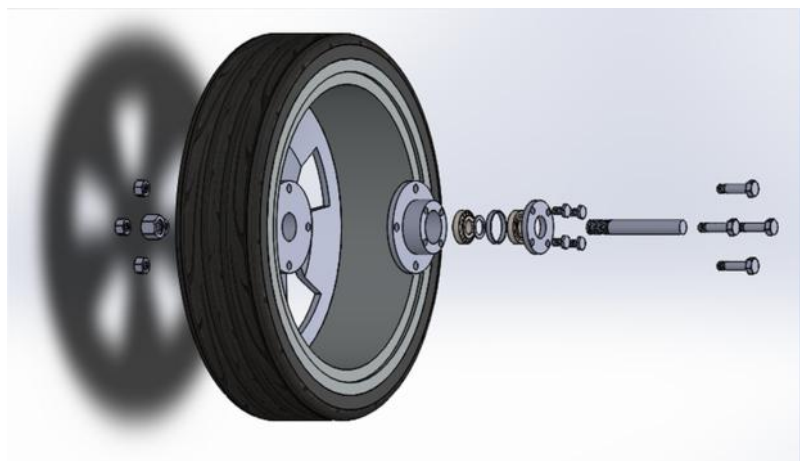


Figure 4. Eklavya 4.0 Rear Wheel Assembly

3.1 The Design Idea

Eklavya 4.0's chassis was designed considering the compactness, easy component accessibility, simplicity to de-assemble and accommodation of all the electric components such that the centre of gravity remains low. The chassis has a width of 55 cm and a 60 cm height (excluding the height of the camera mount). The chassis is made up stainless steel pipes of diameter 1 inch and weighs 10 kg. In the front, the steering column is connected. The links at the top are welded using L-joints while the two bottom joints are the hinged pipe joints, which allow folding of the chassis for packing and transportation. These joints have a locking mechanism which locks them at 90 degrees. The rear part of the chassis includes two small rectangular plates of 6mm at the junction of pipes which connect the wheel hub with the chassis.

3.2 Advantages and Innovations in the new chassis

3.2.1 Steering system

Eklavya 4.0 is driven by a BLDC hub motor mounted on the wheel. It is joined to the chassis by a plate to plate welding to an outer hollow cylindrical tube that supports the entire steering column. It is handled by two ball bearings which allow efficient and smooth rotation. The Steering is rotated by a "MidWest Motion" Brushed DC Motor.

Specifications:

- Length of the T stem: 20 cm
- Fork length: 25 cm
- Fork diameter: 3 cm
- Weight: 6 kg

Experimental Results

- Maximum bending moment: 53.54 Nm
- Maximum stress: 58.5 MPa

Based on these data obtained from the experiments, it was decided to use the steering column of a "Hero Honda Aviator". It is joined to the axle of the hub motor by a coupler. The coupler used is so designed to slide on the fork to change the height of the steering column.

Advantages

The complex chain and sprocket system is replaced by a single hub motor. The complex differential drive control is replaced by the steering control which is much easier. It carries a BLDC hub motor with best mechanical efficiency. The new design allows better control to the robot compared to differential drive.

3.2.2 Wheel Drive

We had used Differential drive in our last vehicle but now we changed our drive mechanism from Differential to Steered drive method. There are many reasons for this change. We used two different motors and a rear castor wheeled drive for stability but there were many unexpected errors in differential drive.

Disadvantages of Differential Drive:

We had used a castor wheel drive in our old bot. This wheel used to get lock by itself frequently that disturbs the localization of the bot. Because of this locking, skidding takes place which also reduces friction in castor wheel. Sometimes it needed a manual help to go back its original position. It takes more time to make a turn because it slows down its speed while turning. These are the disadvantages which makes us to change the driving mechanism from Differential drive to Steering drive.

Advantages of Steering Drive

The movement of vehicle is more precisely done by front steered control wheel. The two rear wheels are for support of the bot. There is no castor wheel which reduces the risk of locking. Steering of front wheel is done by servo motor which is connected to front wheel. Slipping of wheels is reduced resulting in constant movement of bot.

Specifications:

| | |
|----------------------------------|--------------------------|
| Gear ratio: | 1:1 |
| Max Acceleration: | 2.548 m/s ² . |
| 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. |

4. Electronics System

4.1 Flow of Information

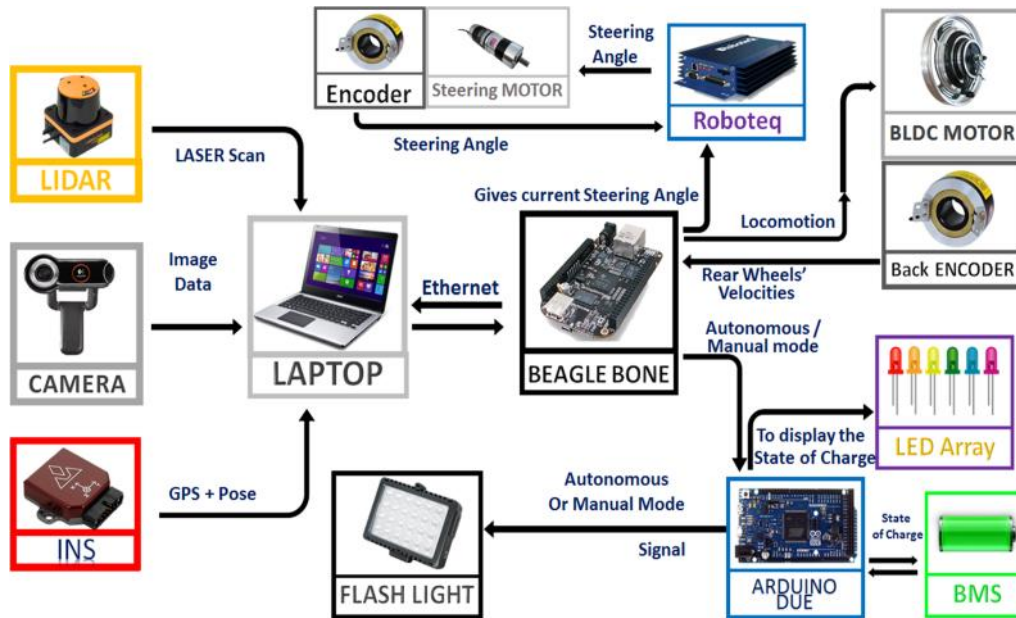


Figure 8. Eklavya 4.0 Flow of Information

4.2 Implementations and Innovations

4.2.1 Working on Beaglebone™

The Beaglebone Black is a development board with an ARM335x 1GHz ARM Cortex- A8 processor from Texas Instruments with 512MB DDR3 RAM. Eklavya 4.0 uses a Beaglebone Black for processing the data from incremental encoder, communicating with the Xbox Wireless controller for manual commands and implementing the controls systems for precise actuation of the actuators. In addition to being an electronics hobby board, the Beaglebone Black is also a fast booting full-fledged Linux running Computer, which is suitable for our vehicle's ROS platform. BeagleBone Black has been effective in precise and fast information exchange between the software and mechanical interfaces.

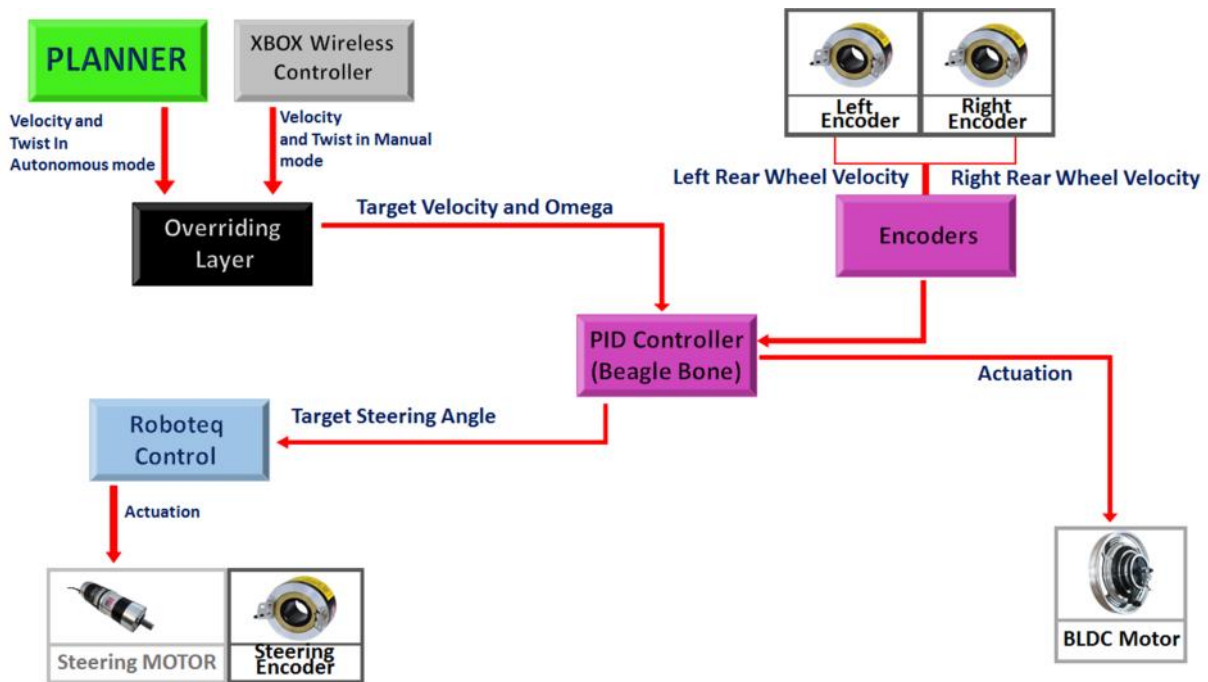


Figure 9. Eklavya 4.0 Flow of Information in controls system

4.2.2 Wireless Remote control

In previous iterations of Eklavya, the car was controlled by a RF transmitter and receiver remote. It led to incubation of lots of noise and led to imperfect control. Also the range of the transmitters was something to be worried about.

To overcome these hardships, Eklavya 4.0 is controlled in the manual mode by a Wireless XBOX 360 controller. The controller sends its data to the Beaglebone via wireless connection. The Controller has buttons for E-Stop Enable, E-Stop Disable, Speed Control and Steering Angle Control; with scope of implementation of cruise control and much more remote useful functionality. The data is sent to an overriding layer built on ROS that decides whether to use Manual or Autonomous Control. The Flashlight Stays On or Blinks Accordingly.

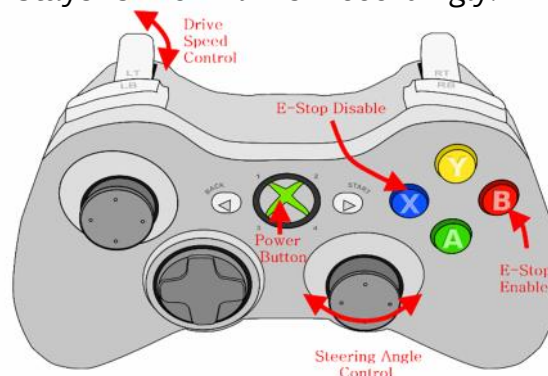


Figure 10. Xbox Wireless controller implementation

4.2.3 Electrical aspect of Motor Drive

Eklavya 4.0 has been upgraded with a brushless dc motor included in the drive mechanism. This upgrade shall cater to the mechanical design requirements and can provide large torque at high RPM. It has even reduced the weight by decreasing the number of motors. Brushless DC motors are DC machines which has a synchronous mechanism of rotating the rotor.

A hub motor of an electric bike was found to achieve our requirement after repeated tests on the motor. Hence we used the hub motor of an “Oreva electric bikes”. The motor runs on an input voltage of 48V and has a power rating of 250W. The speed control is done using Analog Input on the motor, for which the PWM of the Beaglebone is filtered to turn it into a pulsating DC. The E-stop is performed by manipulating the functionality of the Hall Effect sensor used in the BLDC motor. The E-stop is achieved by interchanging the terminals of the hall sensor and the motor stalls suddenly.

4.2.4 Battery Management System (BMS)

The previous versions of Eklavya had faced some problems regarding batteries and their management. That included State of charge, State of health, estimated time for complete discharge not being monitored and hence it arose the possibility of batteries going into deep cycle further deteriorating the life of batteries.

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

Present Implementation:

The current BMS is capable of monitoring State of charge, Current drawn, Terminal voltage, Total Pack Voltage and systematically displaying them over user LED array display. For such purpose we have used Current sensor and Voltage

4.3 Upgrade

The following Table summarizes the changes incorporated on Eklavya 4.0 in Electronics Subsystem.

| Ideas | EKLAVYA 3.0 | EKLAVYA 4.0 | Problems resolved |
|--------------------------------------|--|--|--|
| Microcontrollers and Microprocessors | Arduino Mega. | Beaglebone, Arduino Due. | Beaglebone has more number of I/O pins and a faster processor. Arduino Due also has a faster ARM microcontroller. |
| Power supply and management | Lack of advanced Battery management system and supply distribution | Equipped with Advanced Battery management system and Power supply units with fault detection | Prolonging the life of onboard batteries and estimation of power parameters |
| Drive | Differential drive using brushed DC motors. | Single front wheel drive by BLDC Motor steered by a Brushed DC Motor. | Efficient speed and direction control, less skidding, easy manipulation of data for odometry. |
| Remote control | Radio 2.4Ghz ,4 channel controller | XBox 360 wireless controller which has more than 15 channels. | Clean and systematic control over various functionalities of vehicles. Moreover, Xbox provides large number of user button and interfaces. |
| Control Systems | Inbuilt control System. | Programmable Control System which is controlled through ROS. | Tuned and implemented as per the custom requirement. |

Table 1 : Upgrade in electronic system

5. Perception

5.1 Lane Detection

Eklavya 4.0 continued what Eklavya 3.0 featured, a machine learning based algorithm adaptive to the fluctuations characteristic of the ambient scenery. Grassy portions of the image were removed with a SVM classifier where features for learning were taken as a kernel of an NxN ROI of the image. This kernel was

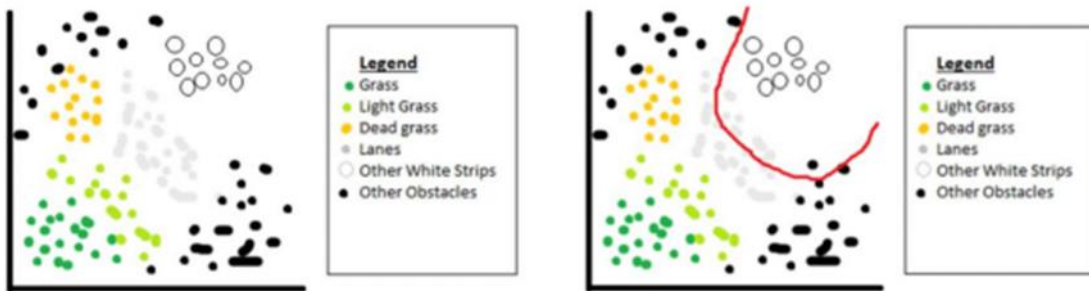


Figure 12. Image Clustering

classified as grass or non-grass based on a polynomial SVM classifier. This classifier test was used at each pixel of the image to remove the grassy portions.

A major change implemented in Eklavya 4.0 is the shadow removal technique. As shadows change the HSV values of regions slightly, when the effects became more prominent the classifier was unable to produce satisfactory results. The image was first converted to the YCrCb colour space. The standard deviation was then calculated of the Y channel. All pixels with intensity less than 1.5 times the standard deviation were classified as shadow pixels and the image was converted into a binary one. Window based thresholding was done to improve the accuracy and then the shadow was removed with colour correction and Y channel adjustment of the image.



Figure 13. Original Image

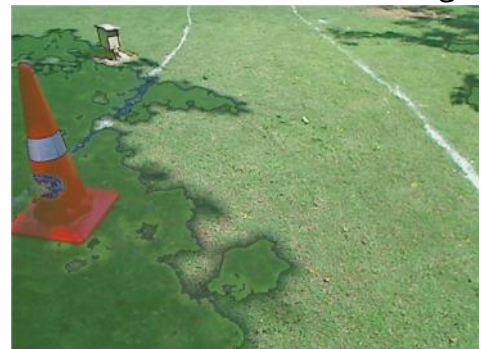


Figure 14. Image After Shadow Removal

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. The next task was determining the number N , size of the kernel. It was observed that the code's FPS increased with N and the classification accuracy decreased with N . By experimentation the value of N was kept between 7 and 12 when tested on a 640×480 image with a first generation i5 processor.



Figure 15. Original Image

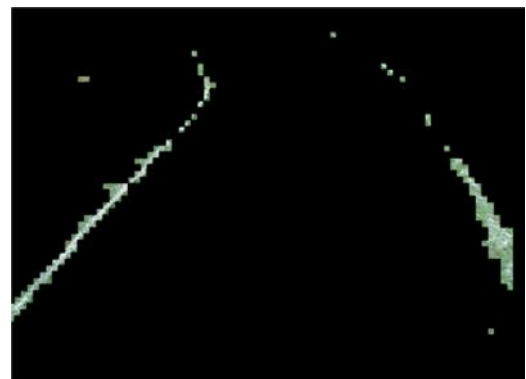


Figure 16. Image After Grass Removal

Finally the image was transformed to a top down view using inverse perspective transform (IPT). This helped the planner work with ease.



Figure 17. Original Image



Figure 18. Image After IPT

5.2 Obstacle Detection and Fusion

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. Initially a color based algorithm was applied to extract the non-white portions of the barrels and dilate them so that they cover the white

strips lying in between them. Still, the problem prevailed in certain cases, especially with the white ladders. Thus, instead of a color based filter, the LIDAR data was used to erase parts of the image which coincided with the LIDAR readings. After this step, the image contained only the lanes and some random noise. The lane was further filtered by a color based threshold algorithm followed by an edge detection algorithm which resulted in high lane detection probabilities with very few false positives.

5.3 Map Fusion

The output of the lane detector comes after undergoing Inverse perspective transform. Lidar data is already in a similar world frame. SVM Classification Results for Grass Removal comes with some offset $(x, y,)$ compared to the lane map. Both these maps are corrected for the relative offset and their union is taken as the final fused map. Note that the lanes are being concerned as non-walkable for all future purposes.



Figure 19. Output at different steps, from original image (left) to grass Removal (centre) and obstacle removal with lidar data (right)

A point cloud is created storing the final lane pixels which is then used to generate a costmap for the planner. This new approach removed much of the noise as the point cloud can be analysed to find the lanes and thus remove any arbitrary pixels falling in between them and also increased the efficiency of the whole setup. The point cloud system approach ensured that the whole image need not be traversed multiple times as only the points of interest were stored in the cloud. This cut down on the time complexity and also minimised the lag between the different subsystems running.

6. Planners

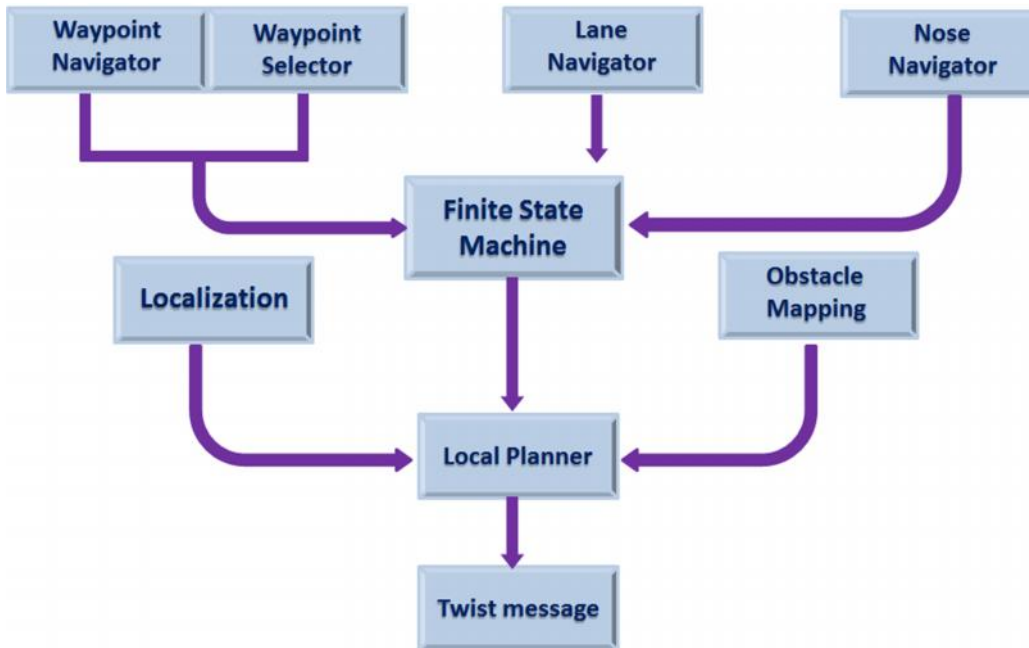


Figure 20: Planner Data Flow

6.1 High Level Planner

In the Navigation System of Eklavya 4.0, the lane-navigator and the waypoint-navigator node will be continuously publishing targets on their respective ros topics. But, there can be only one target depending on the situation in which our bot is in. So, there has to be a system which selects between the two targets. This job is achieved by the High Level Planner. In the next few lines, we will describe its basic structure.

The High Level Planner in Eklavya 4.0 has been implemented using the concept of Finite State Machine (**FSM**). An FSM is basically an abstract machine consisting of a finite number of states. The machine can only be in a single state at one particular instant.

The FSM implemented in our high level planner consists of two **major** states:

1. Lane-Navigator state.
2. Waypoint-Navigator state.

The transition between these two states is defined on the following line:

1. If the bot is in the waypoint zone(no man's land), then the FSM is in its Waypoint-Navigator state.

2. If the bot is not in no man's land and **can** see the lanes, we make a transition to the Lane-Navigator state of the FSM.

Along with these states we have also incorporated the states for test and autonomous modes in our FSM implementation so as to take testing under consideration.

The most important advantage of implementing FSM in Eklavya 4.0 is that it is possible to incorporate certain other special cases in the high level planner without disturbing the earlier structure.

6.2 Global Planner

The Global planner is responsible for generating a high level plan for the navigation stack to follow. Given a goal that is arbitrarily far away from the robot, the Global planner will create a series of waypoints for the local planner to achieve.

The current implementation of the navigation stack uses a grid-based Global planner that assumes the robot is circular in shape and it produces waypoints for the robot that are optimistic for the actual robot footprint, and may in-fact be infeasible.

The Global planner takes in sensor data in the form of point-cloud messages and LIDAR data in the form of laser-scan messages it uses these information to create the Global costmap. The static map layer represents a largely unchanging portion of the costmap, like those generated by SLAM.

If the bot is in Waypoint navigation mode then our Global path planner plans a sequence of the waypoints in two ways:-

1. It first goes to the nearest waypoint and then continues traversing all the waypoints in the least distance order.
2. It first plans a path so that it can traverse all the waypoints by covering the minimum distance by using the greedy algorithm.

Then the Global planner checks if the robot has reached the target or not. It gets this information from two ways:-

1. It checks the local planner status if it has reached the goal or not.
2. It checks if the GPS coordinates of the robot is within a threshold distance from the goal or not.

If the robot is unable to reach the target then the Global planner switches to recovery mode and does either of the following:-

clear_costmap_recovery - A recovery behavior that reverts the costmaps used by move_base to the static map outside of a user-specified range

rotate_recovery - a recovery behavior that performs a 360 degree rotation of the robot to attempt to clear out space.

Then after this Global planner informs the High-level planner if it has reached the goal or not and waits till the High-level planner gives the next target.

6.3 Local Planner

- Base local planner provides a controller connecting the robot with the planner.
- Like the global planner, local planner creates a value function in the form of a grid called cost-map calculating the costs of traversing the cells of the grids.
- The controller using the cost-map of the planner sends the direction of traversal and velocity to the bot.
- The principle of local planning is the search for a suitable local plan in every control cycle. For that purpose, a number of candidate trajectories are generated. For a generated trajectory, it is checked whether it collides with an obstacle. If not, a rating is given to compare several trajectories picking the best.

Two types

1. Trajectory Roller
2. Dynamic Window Approach

The basic idea of both the Trajectory Rollout and Dynamic Window Approach (DWA) algorithms is as follows:

1. Discrete values of the bot's movement parameters are taken and it is simulated to check the trajectory possibilities that may arise.
2. Then each such possibility is evaluated on the basis of :- proximity to obstacles, proximity to the goal, proximity to the global path, and speed and a score is assigned to each trajectory possibility.
3. The highest scoring trajectory is chosen and is passed to the bot for execution and this is repeated.

We have preferred DWA over TR since it samples from stepwise achievable velocities (given max acc.) whereas TR samples over the set of achievable velocities (given max acc.) over the entire forward simulation.

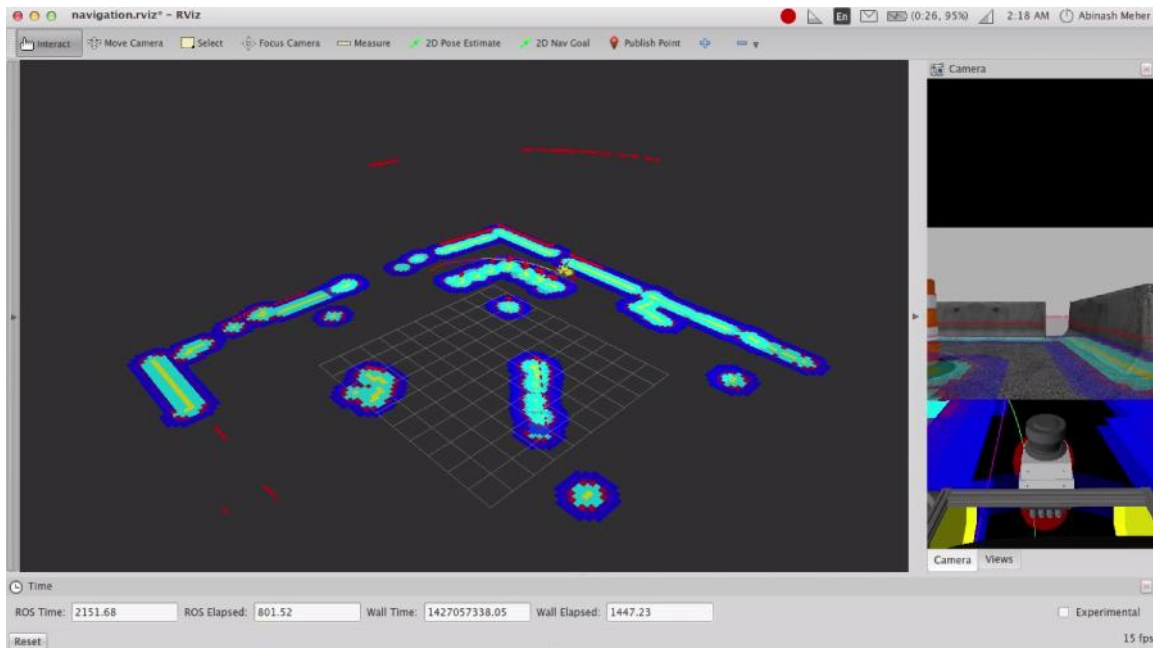


Figure 21: Occupancy Grid from LIDAR

Costmap

- In a particular situation, the decision of path to be chosen is taken by first forming the intersection of global and local cost map.
- The move_base node maintains the common cost map and implements the global navigation tasks.
- Costmap is made using a decay function by allocating values to the points following the rule:- the region closest to the obstacle will be allotted the greatest value and the area where there is practically no effect of the obstacle will be allotted the lowest value.
- Since the bot is considered a point object, the region enclosing the object with a radius of the bot is allocated substantially high values than the rest of the costmap.
- The points that do not qualify in either of the above cases are decided on the basis of other factors.

7. Localization

One of the most important requirement for automating a robot is its localization. By localization, we mean that the robot must, all times, have a fairly accurate idea about its position in a given map or environment. If a robot does not know where it is, it cannot decide what to do next. In order to localize itself, a robot has to have access to relative and absolute measurements that contain information related to its position. The robot gets this information from its

sensors. The sensors give it feedback about its movement and environment around the robot. Given this information, the robot has to determine its location as accurately as possible. What makes this difficult is the existence of uncertainty in both the movement and the sensing of the robot. The uncertain information needs to be combined in an optimal way. We have done so using the Extended Kalman filter algorithm, which is a kind of Bayesian filter.

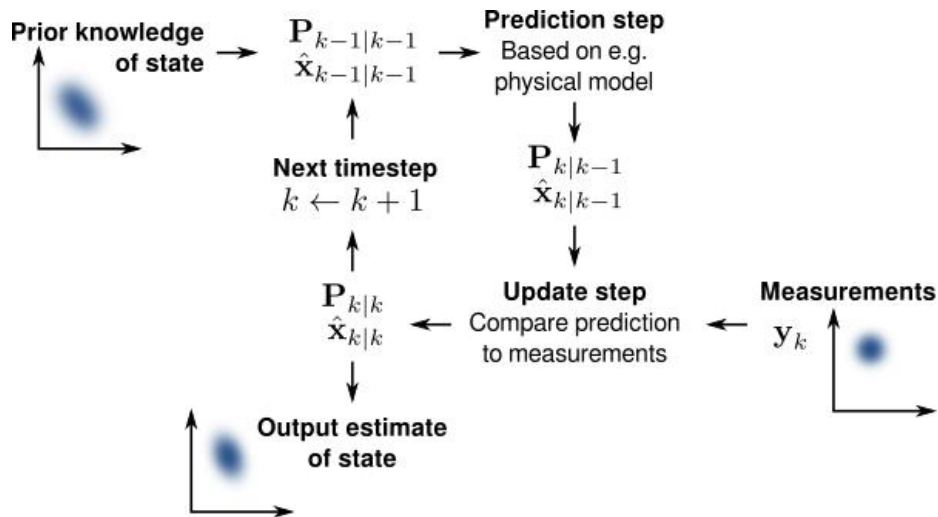


Figure 22: The Kalman Filter Algorithm

To localize the robot, we use three inputs - odometry data (from encoder), IMU data and GPS data (from Vectornav INS). Since, the data received from each of these sensors is inherently noisy, we use the Extended Kalman Filter algorithm. The Extended Kalman Filter is the non-linear version of Kalman Filter which is used widely to produce estimates of unknown variables (linear Gaussian systems) using a Bayes Filter algorithm that uses a series of measurements observed over time (containing noise and other inaccuracies). These measurements tend to be more precise than those based on a single measurement alone. The EKF employs a local linearization using Taylor’s theorem, to employ the Kalman filter algorithm (which uses a linear model) to nonlinear problems.

7.1 Mapping

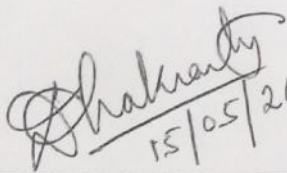
The filtered odometry data is then fed into the Gmapping algorithm along with the input from the LIDAR (Laser Scan). GMapping is a highly efficient Rao-Blackwellized particle filter to learn grid maps from laser range data and form a map showing the obstacles around the robot. It uses the filtered odometry data to get the pose of the robot and merges the data with the laser scans received from the LIDAR to produce a map of the robot’s environment. The map of the environment, thus produced is used by our navigation and path planning module.

8. Cost Report

| Item | Quantity | Cost (USD) | Cost To Team (USD) |
|----------------------------------|----------|--------------|--------------------|
| Hokuyo UTM-30LX LIDAR | 1 | 4974 | 4974 |
| VectorNav VN-200 Rugged | 1 | 2600 | 0 (Sponsored) |
| OREVA BLDC Motor | 1 | 100 | 0 (Sponsored) |
| Logitech QuickCam Pro 9000 | 1 | 110 | 110 |
| BeagleBone Black | 1 | 60 | 60 |
| Arduino Due | 1 | 35 | 35 |
| Midwest Geared DC Motor | 1 | 300 | 300 |
| Lenovo Z510 | 1 | 900 | 900 |
| Xbox 360 Wireless Controller | 1 | 35 | 35 |
| Autonics E50S8 Encoders | 2 | 350 | 350 |
| Roboteq MDC2230 | 1 | 275 | 275 |
| Miscellaneous Circuit Elements | N.A. | 50 | 50 |
| Lead Acid Batteries (Tentative) | 5 | 175 | 175 |
| Bearings (28mm) | 2 | 70 | 70 |
| Rubber Wheels | 2 | 30 | 30 |
| Steering Column | 1 | 75 | 75 |
| Building Materials & Fabrication | N.A. | 110 | 110 |
| Total | | 10249 | 7549 |

FACULTY ADVISOR STATEMENT

This is to certify that the engineering design present in this vehicle is significant and equivalent to the work that would satisfy the requirements of the senior design or graduate project course. Eklavya 4.0 has witnessed significant improvements on the previous bot Eklavya 3.0 in the areas of mechanical design, electrical power distribution, efficient control, system architecture and intelligent navigation. I wish the team all the success for IGVC 2015.


15/05/2015

Professor
: Department of Mining Engineering
Indian Institute of Technology
Kharagpur-72 302

Professor Debashish Chakravarty,
Dept. Of Mining Engineering,
IIT KHARAGPUR