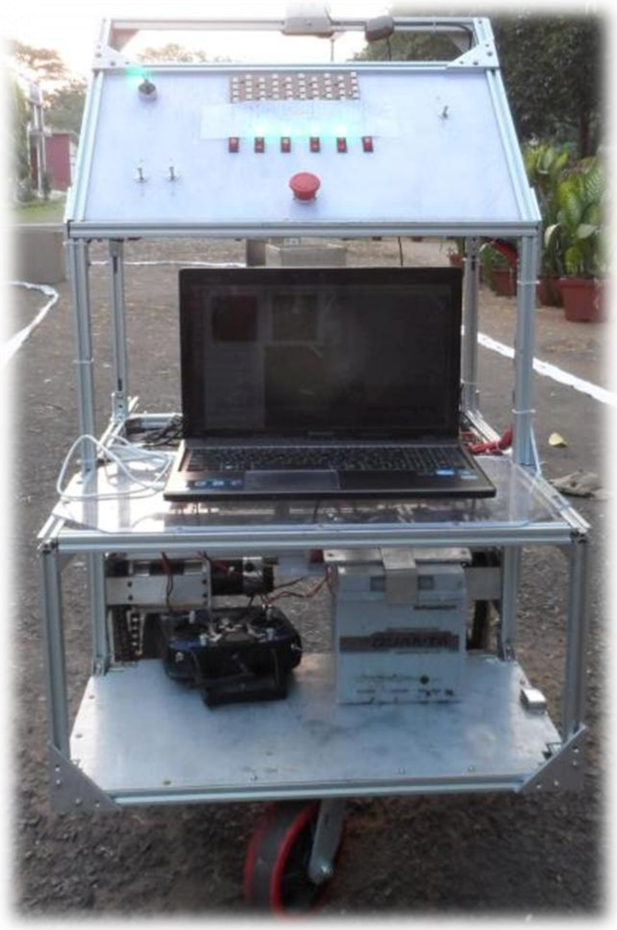
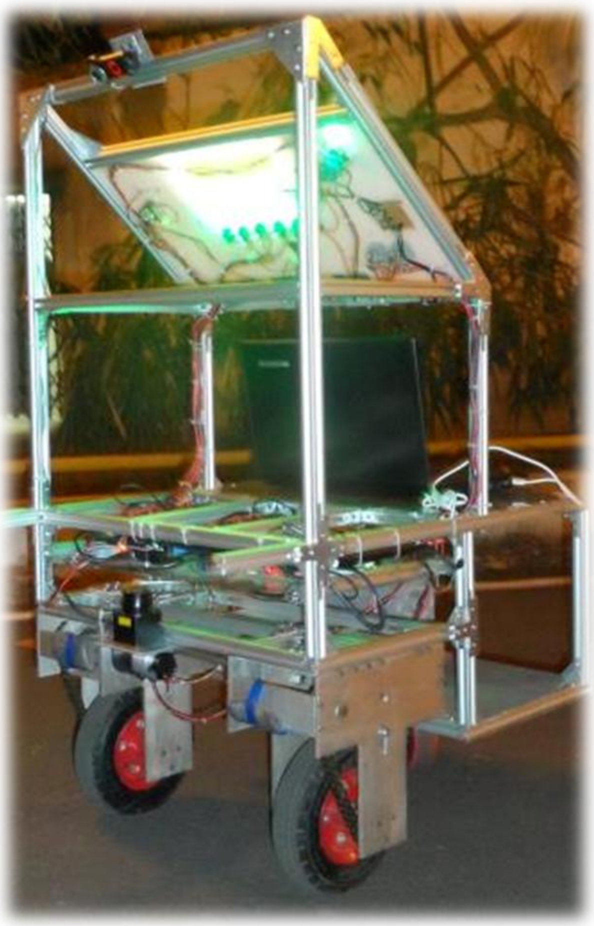
Team Eklavya

Team Members

Software: Bhuvnesh, Samuel, Bhaskar, Tanmay, Sidakpreet, Shiwangi, Satya, Jit, Priyanka, Srinivas, Dibyendu, Eashaan, Atal

Electronics: Anshuman, Bismaya, Ashish, Harit, Diwakar, Apurva, Raunak, Nalin, Nikunj

Mechanical: Naveen, Shubham, Mukil, Parshva, Sayantan, Ravi

Public Relations: Bhargava, Megha, Prashant, Abhilash, Krunal

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 a senior design or graduate project course. Eklavya 2.0 has witnessed significant improvements on the previous bot Eklavya 1.0 in the areas of mechanical design, electrical power distribution, efficient control, system architecture and accuracy of navigation. I wish the team all the success for IGVC-2013.

[illegible]

सह प्राध्यापक
Associate Professor
खनन अभियांत्रिकी विभाग
Deptt. of Mining Engineering
भा.प्रौ.सं. खड़गपुर/I.I.T. Kharagpur

Professor Debashish Chakravarty
Dept. of Mining Engineering
IIT Kharagpur

1 Introduction

Team AGV is a student initiated research group of IIT Kharagpur committed to the long term vision of developing a viable and affordable driverless car. Since its inception in 2011, the team has carried out active research in autonomous technologies and has started projects like Digital City Mapping and All Terrain Mine Rover. The team has participated in IGVC 2012 with the team's landmark vehicle - Eklavya and is currently looking forward to participating in IGVC 2013 with the new and improvised Eklavya 2.0.

1.1 Team Organization

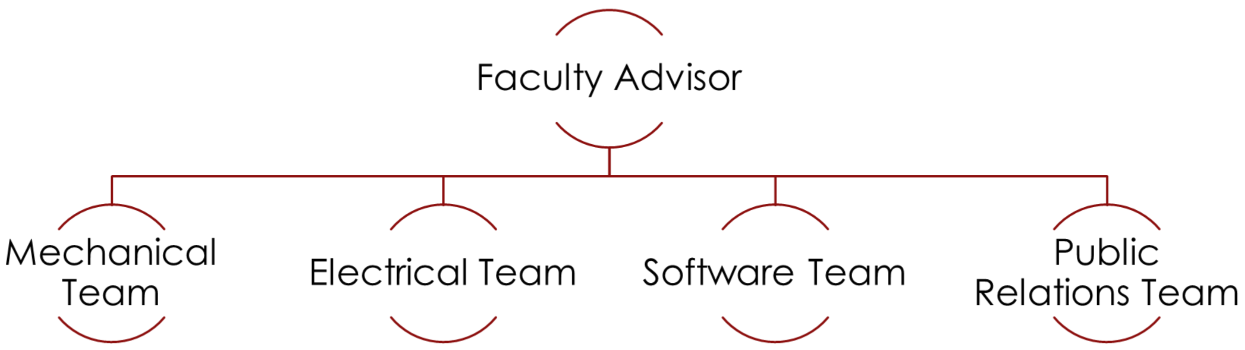


Figure 1: Team Structure

The team consists of undergraduate and postgraduate students belonging to interdisciplinary backgrounds viz., Computer Science, Electrical and Mechanical Engineering and is mentored by Professor Debashish Chakravarty of the Department of Mining Engineering, IIT Kharagpur. The team members are pursuing this project out of their passion for robotics in addition to their academic activities. The team spent an approximately 2400 person hours in design and development of Eklavya 2.0.

1.2 Design Process

The current design of Eklavya 2.0 is a significant improvement in each of the engineering aspects over its predecessor. Learnings from last year's failure have played a key role in developing Eklavya 2.0. CAD modelling and simulation was used extensively for tuning the mechanical design parameters such as the track width ratio for a single castor single differential vehicle. Last year's code base consisted of one single thread which calls all the modules whereas, this year's analysis and design led to a hybrid mechanism which combines a publisher-subscriber mechanism with multi-threading paradigm. The modules were iteratively decoupled to achieve a system with minimal inter-dependency.

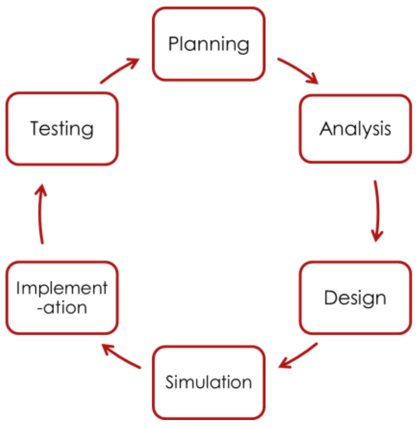


Figure 2: Design Process

2 Innovations

Some of the major innovations implemented on Eklavya 2.0 compared to its previous versions are given here.

2.1 Mechanical

- Chain based transmission for reduced radial loading on the motor shafts.
- Aluminium extrusion rods for improved flexibility during the manufacturing process.
- Machined motor hubs eliminated the problem of motor-wheel misalignment which occurred predominantly in Eklavya 1.0.

2.2 Electronics

- Wheel Odometry has been implemented which estimates the position of the bot based on the observed wheel velocities sensed by wheel encoders.
- Extended Kalman Filter (EKF) has been implemented to improve the accuracy of the localization algorithm beyond the 2.5 m limit set by the GPS/INS system.
- The system is made fault tolerant by adding a backup motor-controller to ensure continuous autonomous operation in case of failure of the primary controller.

2.3 Software

- A distance transform based planning algorithm is implemented to create paths that naturally avoid obstacles and reach the target while ensuring that the planner would not get stuck in local minima.
- A blob based image filter has been implemented on the lidar data as a part of pre-processing to remove the noise that appears due to sunlight in the output of an indoor lidar. The low cost indoor lidar (an outdoor version costs almost 3 times) can now be used in outdoor environments and in strong sunlight.
- An adaptive lane detection algorithm invariant to changes in ambient intensity and surface texture has been implemented.
- Switched to the Robotics Operating System (ROS) for the software architecture.
- The Gazebo simulator has been used in parallel to development on the master platform. This has helped to figure out high level issues with the code flow, planning and strategy.

3 Mechanical Design

3.1 Vehicle Design Specifications

Table 1: Vehicle Specifications

Parameter	Specification
Dimensions	62.3 cm x 59 cm x 109 cm
Weight	45 kg (excluding payload)
Payload Capacity	40 kg
Peak speed possible	2.06 mph
Average running speed	1.5 mph
Drive mechanism	Single caster differential drive
Power Train	Front wheel Powered Chain Transmission
Battery specifications	(12V, 26AH) x2

3.2 Chassis

Eklavya 2.0 is a three-wheeled differential drive mechanism consisting of two motor powered front wheels and a rear lightweight swivel caster. The modular design of the chassis allows one to assemble the robot from scratch in less than 2 hours. Such a modularity helps in easy replacement of parts and make the robot portable over long distances. The payload carrier along with a carrier for the 9 kg battery is built into the rear part of the chassis above the caster. The electronic stack is built on a sliding platform on the front, upper part of the vehicle for easy debugging. The overall chassis is built with galvanized 20 mm x 20 mm extruded aluminium bars, polycarbonate and polypropylene panels.

3.2.1 Skeleton Structure

The skeleton is made up of light weight 20 mm x 20 mm extruded aluminium bars which pack a great strength for their size. They have a linear density close to a hollow bar but the tensile strength of a solid bar of the same dimension. The extruded bars have the flexibility of fixing the fastening elements without the need for drilling as they have inbuilt grooves along the length for placement of the M-5 and M-6 T-nuts. The edges of the grooves are bent inwards by an angle of 2^o. When fastened to the right extent, the groove edges keep pressing against the nuts, thus increasing the grip. The joints which are usually vulnerable to vibrations can now be made robust, thanks to this design. Besides L-shaped bends, aluminium plate of different shapes have been used at several joints, corners and junctions to increase their stress bearing capability.

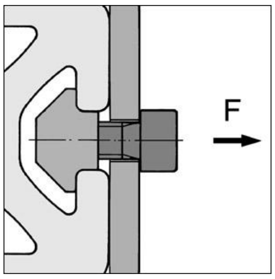


Figure 3: Lock Mechanism

3.2.2 Panels and Waterproofing

Polycarbonate sheets of 3 mm thickness are majorly used for making the panels to reduce the weight of the robot while achieving basic protection against dust and rain. The motor controller panel is situated just above the motor hubs and is made of a 3 mm thick polycarbonate panel. Similar panels are used to cover all the external faces to protect the vehicle from dust and rain. A sensor panel located between the two motor hubs houses the IMU which is strategically placed to be at the CG of the vehicle. This also houses the lidar and the GPS/INS. The switchboard is made out of a 6 mm thick polypropylene panel located on the wedge shaped structure above the laptop. A safety bumper is designed at the front end to protect the sensors and the panel.

3.3 Wheels

Based on several iterations, the team finally arrived at the decision of using pneumatic wheels of 200 mm diameter. A swivel caster of 165 mm diameter with an aluminium core and a load capacity of 250 kg has been used due to its red polyurethane tread which is best suited for grassy, rough terrain and undulated surface.

3.4 Motor Hubs

Instead of the direct coupling of the motor shaft with the wheel axles, a simple chain-sprocket system is used for transferring the load from the chassis directly on to the wheel axles made of solid steel and bypassing the motors. This effectively reduces the radial loading on the motor shafts which are not capable of bearing this load. In each hub, the chain passes through four sprockets out of which one acts as the idler. The idler sprocket slides in a vertical slot such that it causes uniform tension in the chain sprocket mechanism. The idler helps in enlarging the surface of contact between the chain and the sprockets thus increasing the efficiency of load transfer.

3.5 Machining

Jig saw machine is used for cutting the aluminium bars, polycarbonate and polypropylene panels. These plates have been cut to the desired shapes using the jig-saw machine, electric grinder, milling machine, electric drilling machine. Other than the skeleton of the vehicle, the above mentioned machines have also been used for the fabrication of wheel-motor hub which is made of galvanized aluminium plates of 6 mm thickness and consists of complex chain-sprocket mechanism. The individual components of this chain-sprocket system is built by mechanical team members using band-saw machine, lathe machine, electric drilling machine and milling machine.

3.6 Payload Capacity

Eklavya 2.0 is designed while keeping in view of the requirement to carry a decent weight of about 40 kilos in addition to its own including the battery. Several tests have been conducted with a battery load of 9 kilos, a payload of 10 kilos and a 19 kilo 3D Lidar together at various surfaces under different conditions viz., high slopes, moisture rich surfaces and sharp turning areas.

3.7 CAD Model

In the pre-designing phase, after deciding the drive system and some basic mechanical specifications, the team developed a 3D CAD model of Eklavya 2.0 using Solid Works 2012. The model was developed by considering the all measurements which were calculated for the actual vehicle. The design thus conceived had been tested by simulation on the model at various load conditions and surfaces. The final CAD model has been shown in the adjacent figure.

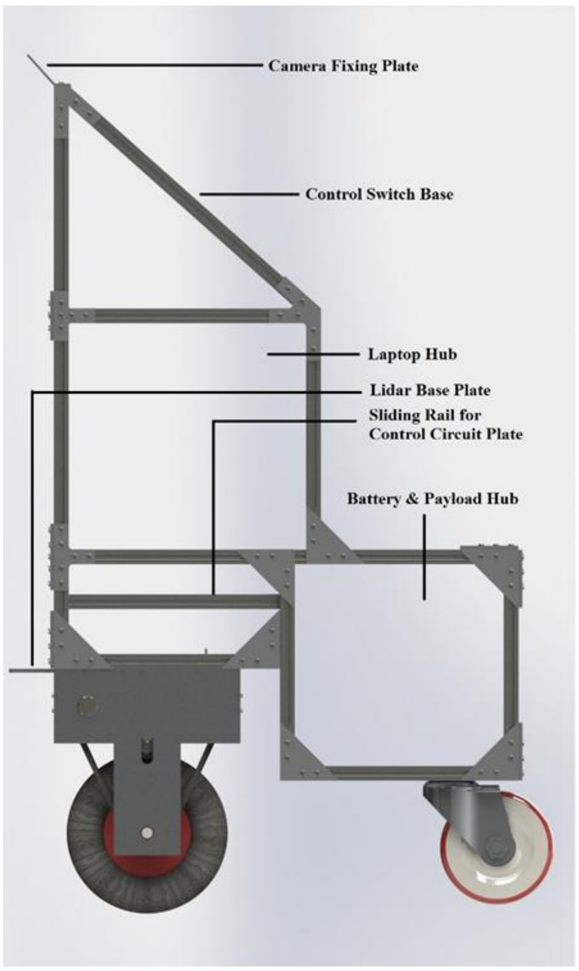


Figure 4: CAD Model

3.8 Drivetrain

Eklavya 2.0 is a front powered wheel differential drive system with a single swivel caster wheel at the rear.

3.8.1 4-Powered Wheels vs. 2-Powered Wheels

A skid steer drive with 4-powered wheels would have a large skid friction. This has to be overcome by the driving wheels and this requires that the motors have enormous torque. Thus, such a model cannot make turns easily. Instead, 2-powered wheels with castors would be a more efficient alternative.

3.8.2 Front Caster vs. Rear Caster

Keeping the caster at the front of the vehicle would make the robot sway in the direction of the caster which is usually controlled by the motion of the powered wheels. However, on rough and undulated surfaces, the caster generates its own direction at every moment which makes the robot deviate from the desired path. So, the caster in Eklavya 2.0 is placed at the rear of the vehicle.

3.9 Centre of Mass

As a three-wheeled robot with a trailing caster, Eklavya 2.0 takes turns around the axis passing through the geometric centre of the frame, halfway between the front powered wheels. As the wheel hubs weigh a little less than that of the rear part which shifts the centre of mass of the vehicle towards rear part, in order to bring the centre of mass close to the centre of the vehicle, a laptop hub and a wedge-shaped structure is designed on top of the front part of the chassis. The wedge houses the switch control board, cameras, E-Stop, safety indicator lights.

4 Electronics

4.1 Motor Controller

Eklavya 2.0 uses the Motor Controller MDC2230 from Roboteq. Having successfully implemented PID control in 8-bit ATmega microcontroller, we decided to make the electronics system robust by introducing a more precise 32-bit commercial motor controller. Roboteq controllers are used in Automated Guided Vehicles and have built in features supporting some common utilities such as Encoder Data Acquisition, Current Sensing, PID, RC Remote Integration, E stop etc. The support for the RC remote enables ease of switching to manual control. The priority of the manual control through wireless remote is set greater than serial data transmission, enabling the Eklavya2.0 to swiftly toggle into manual mode, once the RC controller is activated. This is a significant improvement over Eklavya 1.0, which had the original 8-bit ATmega micro-controller and hercules-16v-16amp motor driver. But the system is made robust by using a two stage controller where the second stage is the original ATmega microcontroller which is activated in case of failure of the primary controller that is Roboteq.

4.2 Control Algorithm

The control algorithm implemented is the Proportional Integrative and Derivative Control (PID) on the speed of the motors. In the Roboteq controller, although a high frequency inbuilt PID controller attempts to achieve the desired set speed, it was required to tune the constants as per our requirements to get the optimum response. On the ATmega controller (secondary), the team scripted the PID algorithm, which runs at a frequency of 100Hz and takes an average of 3-4 cycles (30-40ms) to achieve an almost zero error (stabilization). Any failure of the Roboteq controller, indicated by its status registers, was routed to a digital pin. This pin drives a relay that flips the control to the ATmega Controller.

The tuning was done using the Ziegler-Nichols Rule. It involved setting each gain to zero and increasing K_p till oscillations occur (let it be K). The time period of oscillations were noted and K_d and K_i were calculated as:

$$K_p = \frac{K}{2.2}, \qquad K_i = \frac{1.2 * Kp}{T}, \qquad K_d = 0.0$$

The K_d value was set to zero as the system's response was quick enough to recover from its former speed. The electronics team is currently developing more robust algorithms for PID using fuzzy logic to check for possible improvements.

4.3 Sensors

Eklavya 2.0 is equipped with high-end robust sensors that have been phenomenal in accurate, high-speed and reliable automation.

4.3.1 Laser Scanner

The vehicle uses a laser scanner from Hokuyo with a view angle of 240° and gives 652 readings which are spread equally throughout the range at rate of 10 scans per second. The range of laser scanner is 4m which is sufficient for our planning algorithm.

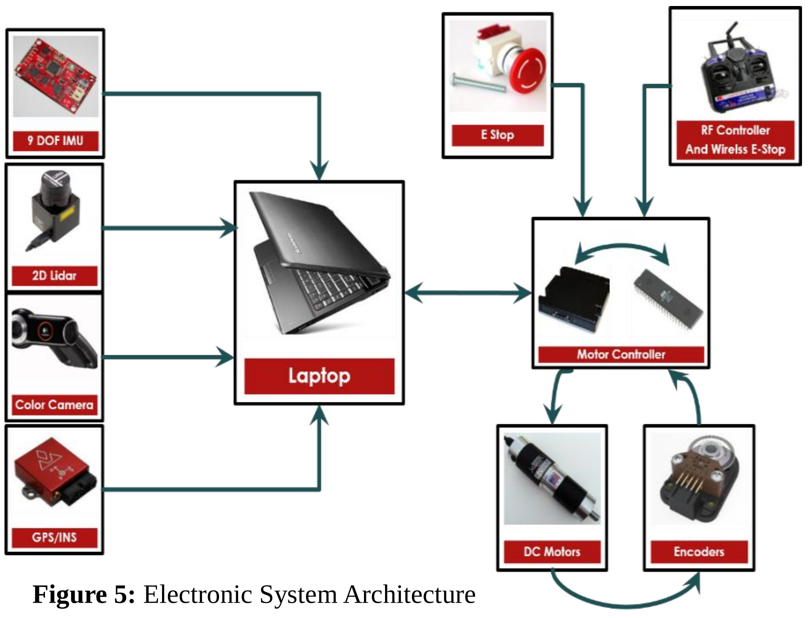


Figure 5: Electronic System Architecture

4.3.2 Logitech Colour Camera

The Logitech QuickCam Pro 9000 colour camera is installed which gives high definition video (1600 × 1200) at 30 frames per second. The camera has inbuilt auto focus and blur removal which maintains the video quality when the vehicle is in motion. It has a high horizontal field of view (60^0) which makes it possible to detect both the lanes simultaneously.

4.3.3 Microsoft Kinect

Kinect for Xbox 360 is used for detection of flags. The Kinect camera gives high definition video at 30 frames per second. This sensor is used as an extra camera in addition to the Logitech camera since the latter camera could miss the obstacles/flags in front because of its orientation w.r.t horizontal which is 45^0.

4.3.4 Wireless Manual Controller and Wireless E-Stop

Eklavya 2.0 is programmed to be controlled by a “Flysky FS-CT6B 2.4GHz 6CH Transmitter and Receiver”. Two of the channels provided by the RC receiver are used to control the directional velocities through the primary Roboteq controller. The secondary ATmega32 controller is configured with a Bluetooth Module. An Android App has been designed that connects to this Bluetooth module and commands the bot when it is being controlled by the ATmega32 controller, during the manual operation.

The wireless E-Stop is engaged as soon as the wireless RF controller is turned ON. The priority of this controller has been set higher than Serial transmission, enabling immediate stop as soon it is turned ON with buttons set to zero velocity. The RF Receiver is hardware-hacked and connected to the Reset of the ATmega controller, thus, turning it OFF as soon is the RF Transmitter is turned ON.

4.3.5 VN-200 INS/GPS

The VN-200 is a miniature high-performance GPS-Aided Inertial Navigation System that combines MEMS inertial sensors, a high-sensitivity GPS receiver and advanced Kalman filtering algorithms to provide optimal estimates of position, velocity and orientation for industrial applications.

We used VectorNav’s C/C++ software library to fetch data through the INS solution provided by the sensor. This gives us thermally and magnetically corrected pose with a horizontal dynamic accuracy of 2.5 meters. The latitude, longitude and altitude that were obtained from the sensor were published directly giving us the current location. The static accuracy of the sensor is nearly 5 meters due to which the INS data from the sensor was fused with the odometry data to get higher accuracy. Extended Kalman Filter (EKF) was applied on the odometry data, IMU orientation and latitude longitude from the GPS to accurately localize the vehicle in 3D.

4.3.6 9 DOF IMU from Sparkfun

The yaw data that we need for target location was obtained from a 9 DOF IMU from Sparkfun. The yaw data from the VN-200 which has a static accuracy of +-2^0 was not as accurate as the Sparkfun’s IMU.

4.4 Motors

Eklavya 2.0 uses reversible DC geared Midwest Motion Motors equipped with a 1024 CPR quadrature encoders. The maximum output speed is 86 RPM delivered at an internal gear ratio of 50.89:1. The rated continuous torque provided is 111 in-lbs. The motors run at 12V DC and has a rated continuous current of 15A.

As per the requirements of IGVC, we needed our bot to move at a speed of 1.5 mph. With a wheel diameter of 8 inch, the required angular velocity of wheels was 71 RPM. After the mechanical calculations, the continuous torque requirement came out to be 110 in-lbs.

4.5 Computing Hardware

Eklavya 2.0 uses a Lenovo Z580 laptop with 4GB RAM running Ubuntu 12.04 and ROS on an i5 processor augmented by a 1GB graphics card. The control for the motors are implemented on the Roboteq motor controller, which is connected to the laptop using an USB-RS-232 interface. All of the other sensors viz., the lidar, GPS/INS, camera, IMU, Kinect are all connected via an USB Interface to the laptop.

4.6 Control Hub and Dashboard

The Eklavya 2.0 houses an electronic Control Hub for the motor controller circuitry right above the motor hubs. The Hub houses the Roboteq motor controller and the backup ATmega controller. Connections from the battery enter the panel and those from the panel exit at sensors, motors, switches, flash lights, E-Stop etc. The Hub has a sliding panel that helps in easy debugging of the electrical equipment. A cooling arrangement using fans is also implemented for the control hub. The dashboard is on the top of the bot for easy access to the main power button, switches, E-Stop and several indicator lights.

4.7 Power System

Eklavya 2.0 uses two Amaron Quanta 12V, 26Ah, lead acid batteries as the power source which provide unregulated power through a series of switches to the different modules. An inexpensive DC-DC convertor is used to scale down the voltage to 5V to power the USB Hub to run the individual sensor modules. Fuse wires of appropriate rating has been used with each module to minimize electrical damage.

A proper switching logic for operation of different individual modules has been implemented on the Dash-Board. The failure of the Primary Roboteq controller, indicated by a Digital Pin, switches relays causing transfer of power and motor controls to the secondary ATmega Controller, which consequently undertakes the responsibility. The computing hardware i.e. Laptop with its own battery has a battery life of more than 4 hours.

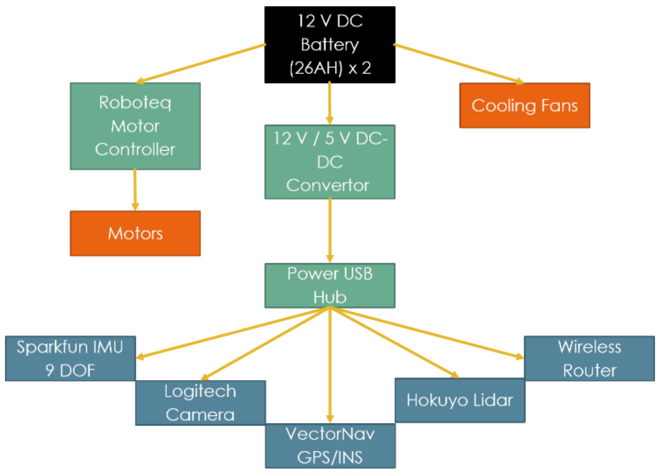


Figure 6: Power Flow Diagram

4.7.1 Power Utilization Table (Worst Case)

Sl. No.	Component	Voltage (V)	Current (A)	Power (W)
1	Motor + Roboteq	12	15(x2)	360
2	Hokuyo Lidar	5	0.7	3.5
3	Camera	5	0.250	1.25
4	GPS/INS	5	0.5	2.5
5	IMU	5	0.1	0.5
6	Kinect	12	1.08	12.96
7	Cooling Fans	12	0.42 (x4)	20.16
8	Flash Lights	12	1.4	16.8
Total			**35.71 A**	**417.67 W**

Table 2: Power Utilization Table

- According to the above calculations, Life Expectancy of the battery was found out to be $\frac{52Ah}{35.71A} \cong \mathbf{1.45}\ \boldsymbol{hour}$.

4.8 Safety Considerations

The safety features include an E-Stop button installed on the dashboard. A wireless E-Stop arrangement has also been provided that can stop the bot from a remote distance (minimum 100 feet) as per IGVC requirements. The E-Stop buttons cut the power to the H-Bridges that control the motors. The Roboteq controller provides a programmable current limit (which is set to 15A) that limits the maximum current flowing though the motors. A wireless, manual-to-autonomous switch toggles between the two modes. As a safety indicator, flashing light has been installed on the panel that keeps blinking when the bot is in autonomous run condition.

5 Software Architecture

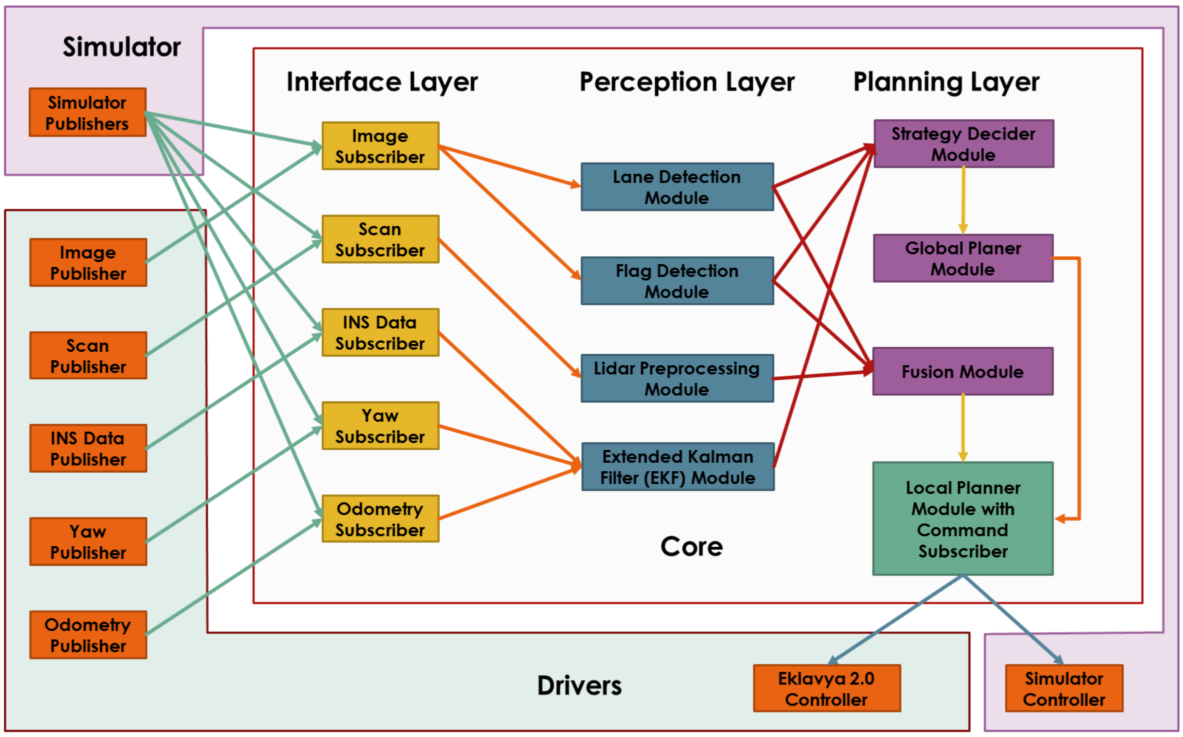


Figure 7: Software Architecture Diagram

A new architecture is used for Eklavya 2.0 which makes use of both the publisher subscriber message passing of ROS and threads of C++. The three main components of the system are the drivers, core and simulator. A purely threaded architecture is notorious for the presence of dead locks. These can be taken care of by using a publisher subscriber message passing paradigm. A pure message passing system is again not advised due to the large network traffic generated by publishing bulky images wrapped up as messages. We combined the merits of both these domains by using threads for accessing all the maps via mutex and messages for accessing all the other data types.

5.1 Drivers

Each sensor has its own ROS node which publishes the data in a standard format. The drivers can be launched independent of the core which allows easy debugging. Also, the novel *rosbag* approach allows one to *record* the messages from the driver publishers and *replaying* them at ease.

5.2 Core

The core is essentially a threaded architecture with the major data structures viz., maps, bot pose etc., shared among them in the form of global variables. The subscriber threads subscribe to the device publishers. On receiving the messages, they update the corresponding shared variables via mutex constructs. The processing threads then act on this shared data and make intelligent decisions. The data processing threads handle image processing and EKF. The strategy thread decides the strategy to select based on whether flags and lanes are visible in the camera view. The strategy can be either to follow the lanes or navigate around the flags. The global planner considers all the GPS waypoints and selects one of them as the next immediate target to follow and in the process, checks out previous ones. The local planner is given a local map and a local target and is entitled with planning the path locally. Once a plan is formed, the controller is responsible for keeping the bot on the path until a new one is decided.

The core has a threaded structure with each thread running a module. The first layer is the subscriber threads that subscribe to the messages published by the device publishers. On receiving the messages, they update them in the shared variables.

5.3 Simulator

Since the messages being used are in the ROS standard format, the simulator is completely compatible with the core. The compatibility ensures a seamless transition between the vehicle and the simulator.

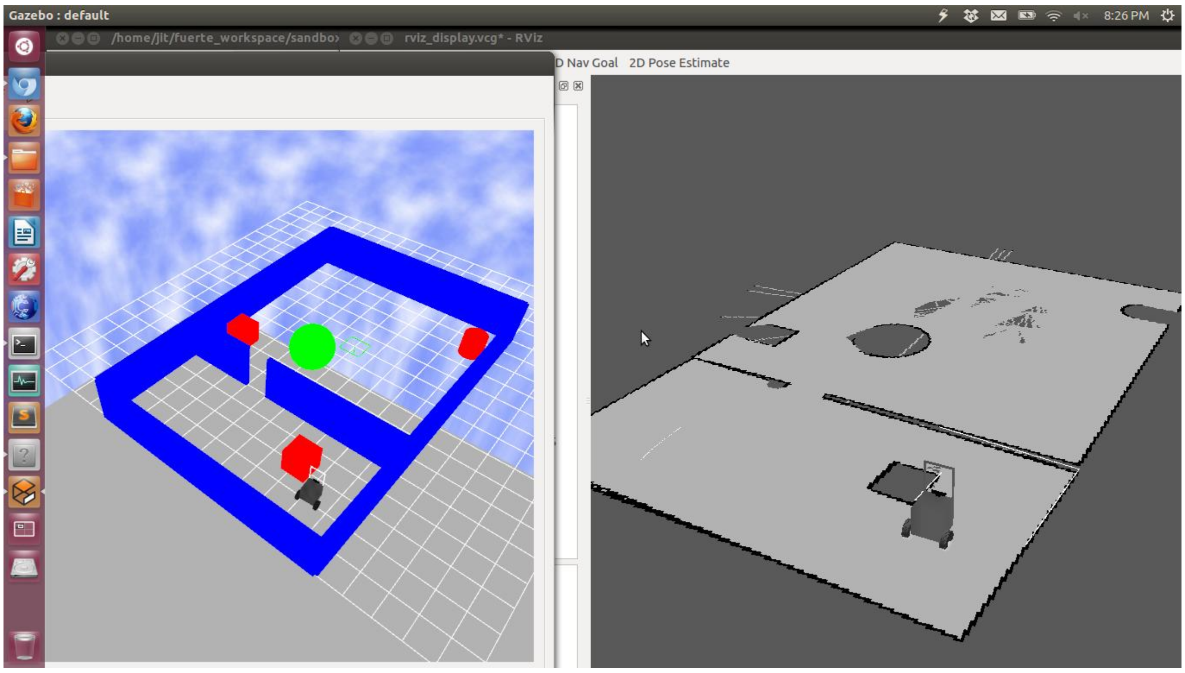


Figure 8: Gazebo Simulator implementing Graph-SLAM

6 Sensor Processing

6.1 Lane Detection

The lane detection module subscribes to image published by the camera publisher. The detection algorithm is then applied followed by inverse perspective transform to re-project lanes into real world coordinates with respect to the vehicle. Lanes are then updated in the shared memory for the local map.

6.1.1 Data Acquisition

The colour camera node published data at 30 frames per second (FPS) while the FPS of the lane detection algorithm is 10. The subscriber thus keeps a buffer of 2 frames removing the older ones as new frames are received.

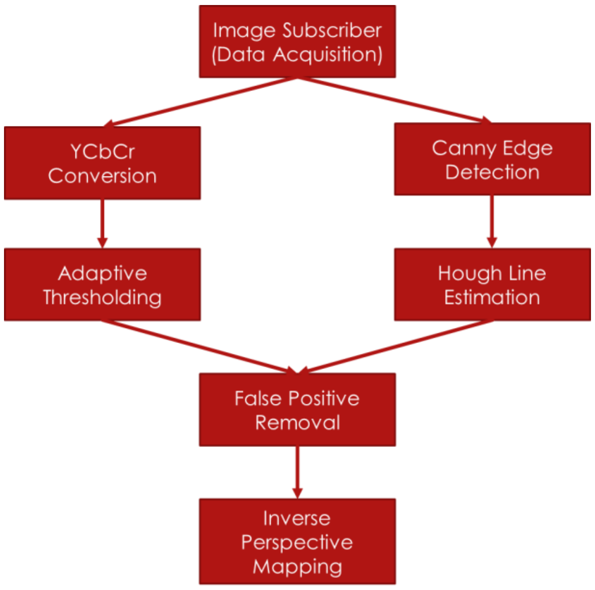


Figure 9: Lane Detection Algorithm

6.1.2 Colour based detection

As the lighting conditions and lane colours keep on varying during run of vehicle, it poses a challenge for standard colour threshold techniques in vision processing to detect lanes correctly. To overcome this problem, we convert the image to the YC_bC_r colour model where Y carries luma information and C_b, C_r are responsible for imparting colour to image. We use only the Y component to segment lanes. Y_{avg} (Average value of Y) and Y_{std} (standard deviation of Y) are computed. Pixels having Y value greater than $Y_{avg} + K * Y\_std$ are classified as probable lane pixels.

Clearly this thresholding method can give erroneous results depending on the values of K. This is taken care of by choosing a relatively soft threshold value for K so that none of the lanes pixel are missing. The other false positives are taken care of subsequently in the algorithm. Generally K value is chosen to be 1 for grass and 2 for roads.

6.1.3 Edge based detection

A distinctive property of the lane markings in image is sharp the change in intensities and hence presence of strong edges along them. Therefore we apply canny edge detection to detect strong edges in image. The lower threshold value in canny algorithm are kept relatively small so as to preserve continuity in edges. The obtained edges are then subjected to probabilistic Hough transform to get lane edges as line segments. The advantage of using Hough transform is that it is able to approximate even curved lanes as set of line segments which otherwise has to done using cubic spline fitting which is computationally very heavy. The length value of line segments is set relatively on higher side to ignore small edges which may be present due to some other objects in frame.

6.1.4 Removal of false positives

Both the thresholding and Hough line method can output false lanes, especially in thresholding as it is very difficult to find fine threshold values. To overcome this problem, we take intersection of both the results so that only the points satisfying the colour, edge and line property are selected as possible lane markings. This automatically removes all the false positives from both methods.

6.1.5 Inverse Perspective transform

The angle of view of camera and distance of objects from camera contribute to associate different information to each pixel (perspective effect). The detected lanes need to be plotted on local map of the vehicle. For this, we need to first take care of the perspective effect on the image. To cope up with it, inverse perspective mapping is used which removes the perspective effect from the acquired lanes, thus remapping them to a new 2D domain where information content is homogeneously distributed among all pixels. Now the points from this new domain are mapped to real world coordinates by multiplying them with constant matrix obtained by calibration.

The calibration is very time consuming as it needs to manually choose landmarks and measure distance between them. To overcome this problem we developed automated calibration system using a rectangular checkerboard. It automatically detects corner of the checkerboard and calculates warp matrix for IPM using dimension of the board known *apriori*.

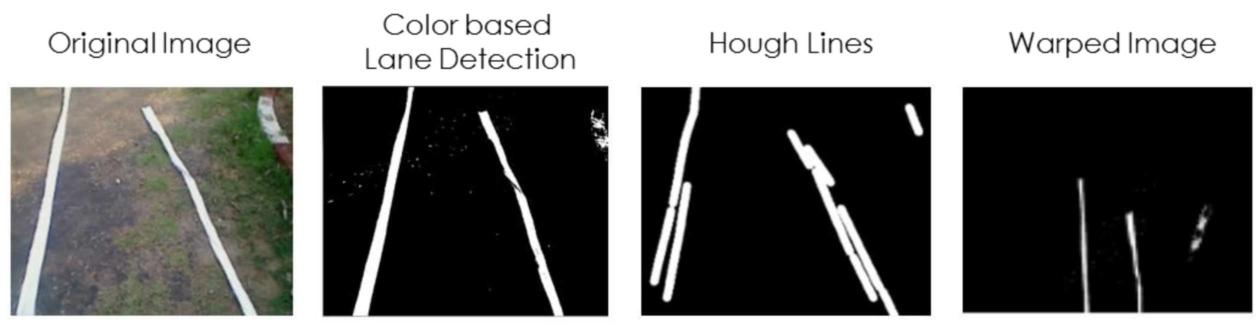


Figure 10: Image Processing Pipeline

6.2 Obstacle Detection

The obstacle map is built using only the 2D-LIDAR. The ranges obtained from the sensor are initially plotted as it is in an image map. Then a threshold is applied on area of blobs detected from the image which acts as a filter for noise. The reason for applying this filter lies in the fact that the noise which occurs in the sensor data is irregular. The abnormal peaks or trenches that are formed are of variable thickness but they are not very thick i.e. their thickness will be less. So the cluster of noise can be filtered according to the threshold applied on the size of the obstacle. Once calibrated for a certain environment this filter removes nearly all of the noise. But this filter also poses some drawbacks such as the following.

- It has to be calibrated for each different environment. For indoors, the minimum threshold works at any value less than the obstacle size but in presence of very bright sun we keep a threshold value for nearly 5-15 cm thickness (blobs contain 100-200 pixels) to avoid the random noise.
- The calibration is to be done so that it does not exclude the fence since the thickness of fence is very less.

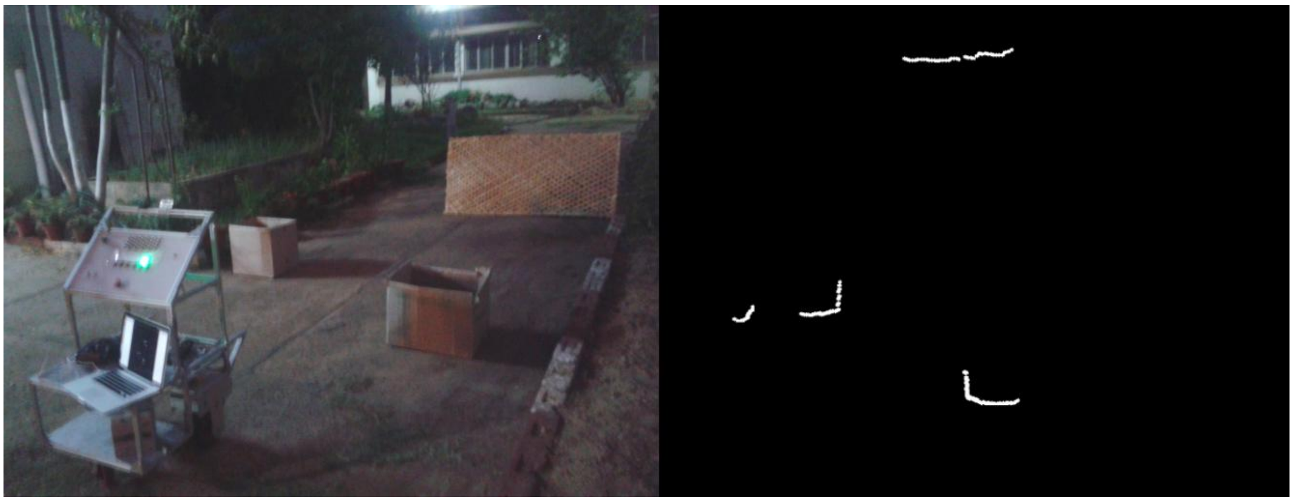


Figure 11: Obstacle Detection using 2D Lidar

7 Data Processing

7.1 Sensor Data Fusion

The local maps obtained individually from the lidar and the camera have origins different from that of the vehicle. Thus, these maps should be transformed independently to fuse the correct set of pixel values. Confidence of each sensor is calculated by taking 100 data points and the resulting fused image is obtained by taking a weighted mean of the individual observations with confidence measures as the weights.

7.2 GPS Way Point Tracking

The latitude, longitude and altitude, once obtained from the GPS/INS publisher, give the current location of the bot. An additional record is kept for the target points. Given the bot's current location, the distance from the target location can be obtained. Using the yaw obtained from the yaw publisher, the orientation of the target with respect to the local frame is calculated. These coordinates are first converted into the East-North-Up (ENU) co-ordinates. By considering them as the origin, we can calculate the global target.

When the bot closes on the current target within a given radius (say within 20cm), the target in the Navigation is modified to the next target. Once the final target is reached, the initial position is set as the current target so that the bot can complete the course.

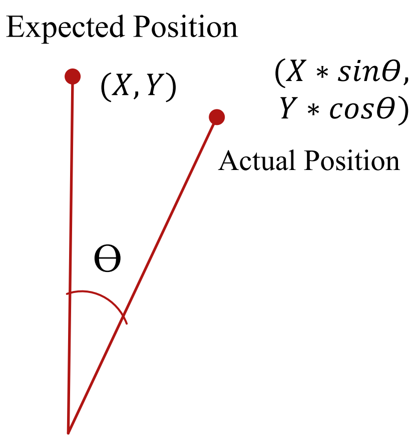
7.3 Wheel Odometry & Localization

Wheel odometry is concerned with localization using encoder counts. Conventionally, calculating the position with wheel odometry includes counting the left and right encoder counts and taking the average. A slight improvement in this method will be to include the deviation from the original path. The following are some calculations:

$$D = (left\_counts + right\_counts) / 2$$

$$\theta = (left\_counts - right\_counts) / wheel\_base$$

$$(X, Y) = (D * \sin(\theta), D * \cos(\theta)).$$



Wheel Odometry Calculation

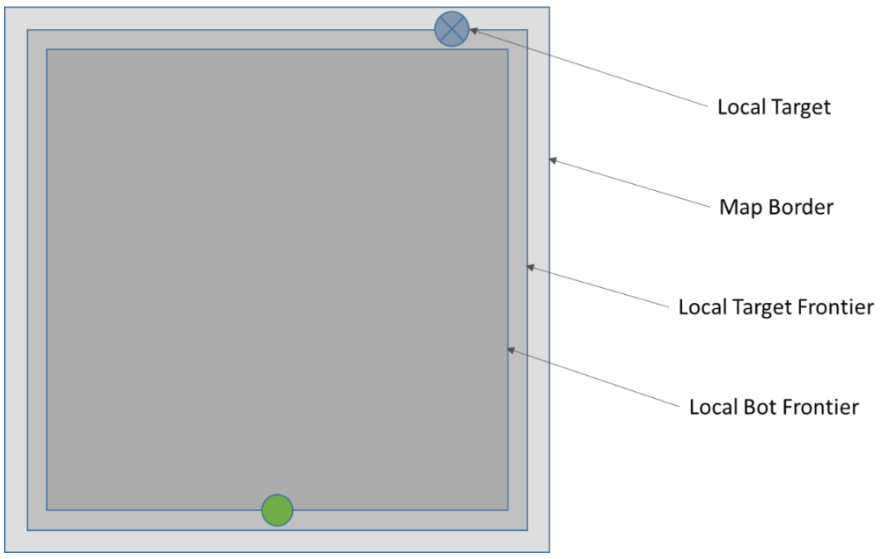
Although this is theoretically simple, consistent values result only after calibrating this with the actual values. After consistent trials, a suitable constant of calibration was found out. It was also found that this constant K depended heavily on terrain. An intelligent algorithm was developed which automatically averaged out the guessed constant K, from the data of another intelligent sensor as soon as the terrain was changed. The change of terrain control is so far manual. Whenever there is a change in the terrain, auto calibration mode is turned on which computes the constant K by averaging in a window of 50 data samples.

This data from the odometry is integrated with the pose and latitude longitude values obtained from the inertial navigation system using Extended Kalman filter. The EKF package present in ROS is used to localize the robot with these values. The localization error after applying EKF is reduced to less than 1m which is required for precisely reaching GPS waypoint in IGVC.

7.4 Navigation

The navigation module is entitled with the responsibility of calculating the correct target location to be used by the planner module. It subscribes to the various sensor publishers and depending on the strategy, chooses the right method for calculating the target. The planner works with a local map of 10m X 10m dimension. It can take a target only within this area. Any targets outside have to be appropriately truncated by the navigation module. The truncation method involves cutting the line joining the vehicle's location and the actual target at a point which is at 50 centimetres from the boundary of the local map. This distance serves as a buffer for the planner while it tries to plan near the goal.

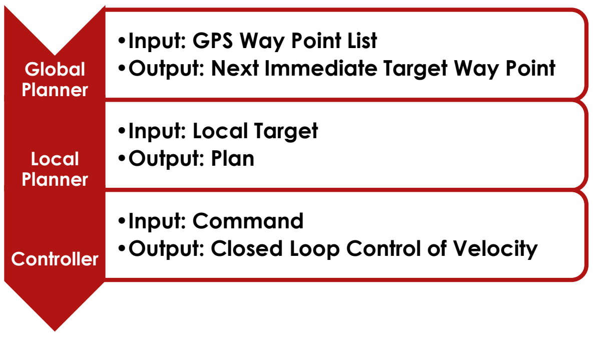
The strategy used in IGVC test runs will be the IGVC_BASIC. In this case, all the sensors are active – both camera and lidar are plotting the obstacles into the local map. The target is set solely based on the next target location given by the GPS module. Heading information is used to find the pose of the target w.r.t the bot by transforming the raw target pose which is in the ENU (East - North - Up) coordinate frame.



7.5 Planner

The planning module is spread over two levels – Global Planner and Local Planner. The Global Planner takes the 8 GPS Way Point coordinates and is entitled with the task of finding the next best way point to go to immediately. Once the immediate target is set, the local planner plans paths up to the target, till the vehicle reaches the immediate target way point. Once the path is planned, the controller is responsible for sending the right command to the bot and making sure that the vehicle does follow this path.

7.5.1 Global Planner

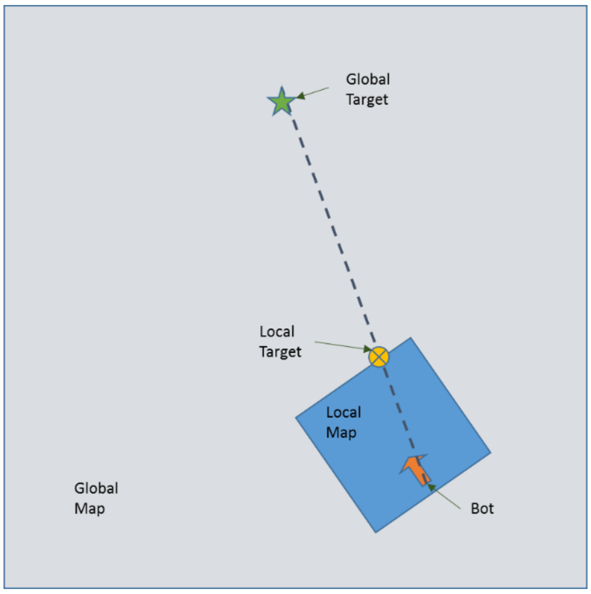

Global Planner
•Input: GPS Way Point List
•Output: Next Immediate Target Way Point
Local Planner
•Input: Local Target
•Output: Plan
Controller
•Input: Command
•Output: Closed Loop Control of Velocity

The global planner module maintains a list of targets in the form of latitude and longitude of the target way points. It maintains a variable to store the current target location and in each iteration, it checks the distance remaining between the current location of the vehicle and the current target. If the remaining distance goes below a certain threshold, another target is chosen and the variable is set to the new target. The target selection policy used for IGVC is the next closest target – pick the target whose distance to the current location is the minimum as the next target. Note that this module runs on a higher level than Navigation module and gives input to the Navigation Module.

7.5.2 Local Planner

In a nutshell, the local path planner implements the A Star (A*) algorithm to find the path between the vehicle's pose and the target's pose. One important deviation from the standard algorithm is that the grids have been replaced with a fixed set of splines (called *seeds*).

7.5.2.1 Seeds

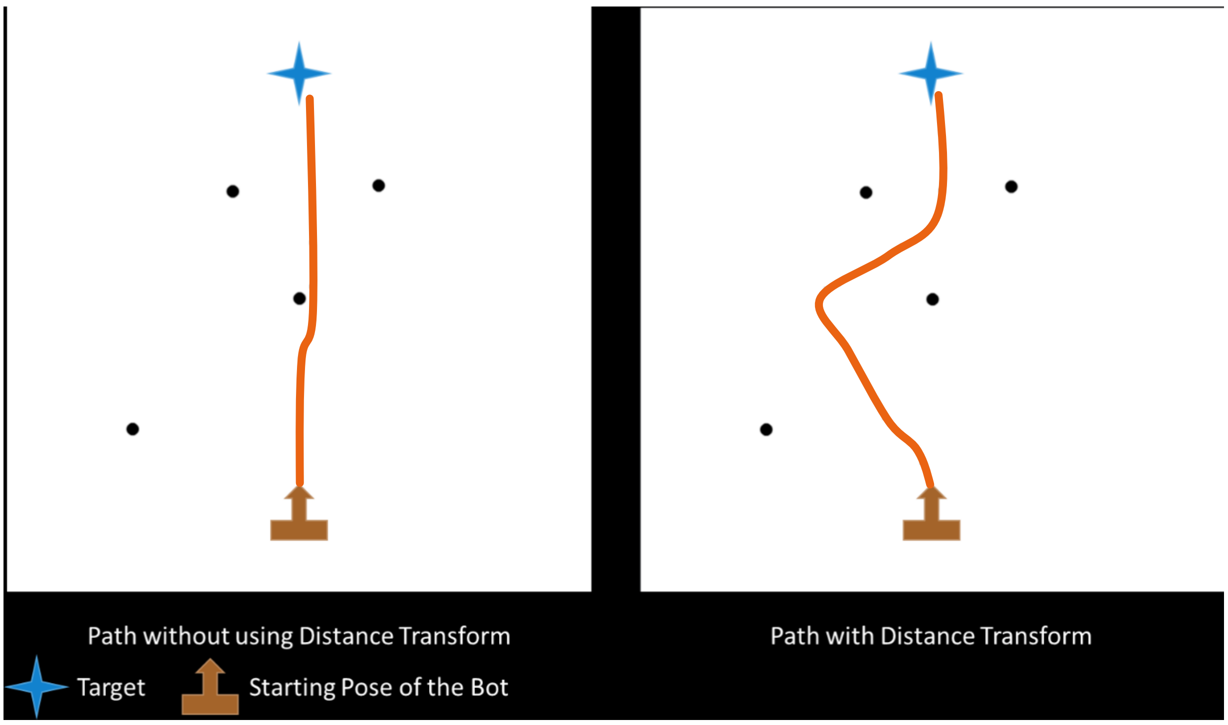


The seeds are the idealization of the trajectory followed by the vehicle under the influence of a specific set of commands when simulated for a fixed amount of time. When constant velocities are used, these trajectories are circular arcs. These seeds are selected from an infinite set of possible seeds based on limitations like the minimum radius of curvature and the time taken by the planner when the seed set is used for planning which can be minimized at the cost of the optimality of the path being formed. The motivation behind replacing the grid based model with a fixed set of splines lies in providing a continuous smooth path avoiding abrupt locomotion commands which otherwise are capable of causing random fluctuations in the sensor data.

7.5.2.2 Modified A* Algorithm

The obstacles detected by lidar and the lanes detected by camera are both mapped as obstacles in the local map. During planning, we start at the current state which stores the position of the bot and explore all the neighbouring seeds and check if their states are already visited. If not, we add them to open list where they are stored in a priority queue in the non-decreasing order of their total cost. From the open list the best state (i.e. state with minimum cost which is available readily at the beginning of the open list due to the priority queue property) is chosen for consideration. A walkability check is then applied which checks if there are no obstacles in its path. Once a neighbour state is cleared of all checks, it is added to the open list. This is continued till the target is reached. Once the target is reached the path can be reconstructed by backtracking the parent states starting from the goal state.

7.5.2.3 Distance Transform


Path without using Distance Transform
Path with Distance Transform
Target
Starting Pose of the Bot

One problem with a distance optimizing search (like A* search) is that it plans paths very close to the obstacles. As the locomotion errors are not taken into consideration while planning, there is definite possibility for the bot to collide with the obstacles. A safe path would take the bot away from the obstacles as much as possible – probably along the voronoi edges created by assuming the obstacles as the voronoi centres. This can be achieved by adding an extra cost to the cost function which represents the overall distance of the path planned so far to the nearby voronoi edges.

7.5.2.4 Implementation

Efforts are made to implement the data structures using the standard C++ Standard Template Library (STL). In order to avoid customization of these inbuilt functions, a clever workaround has been implemented to boost the speed of the planning algorithm. A Hash-Map is used to store membership of each state – whether it belongs to the open list / closed list. In the standard algorithm, in the event there are duplicate states with different costs, the higher cost ones are to be removed from the open list prior to inserting the least cost state. This step however requires sweeping the off the shelf priority queue supplied by STL. In our approach, all the duplicates are kept in the open list under the assumption that the least cost duplicate state is removed first. Then, on future removals, we can check for the membership of the state – if it belongs to the closed list, remove it from the open list and continue.

7.5.2.5 Complexity

Every access to the STL structures used in the module are of time complexity – $O(\ln n)$, where, n is the size of the open list. This leads to a total complexity – $O(\mathrm{n} \times \ln n)$. This was a considerable improvement over the planning algorithm used last year whose complexity was – $O(n^2)$.

8 Performance Analysis

Parameter	Expected Values
Maximum Speed	0.75 m/s
Ramp Climbing Ability	30°
Response Time	0.1 seconds
Battery Life	1.6 hours
Obstacle Detection Range	4 m – 5 m
Localization Accuracy	2.5 m

9 Vehicle Cost

Equipment Used	Specification	Actual Cost (US $)	Cost to Team (US $)
VectorNav GPS/INS *	2.5 m Horizontal Accuracy	3000	75
9 DOF IMU **	5° Dynamic Accuracy	110	110
Kinect	4 m – 5 m Range	240	0
Aluminium Bars	20 mm × 20 mm	370	370
Lead Battery	2 hour backup	90	90
Chassis Material		465	465
Colour Camera **	30 Frames per Second	110	110
Roboteq Motor Controller	2x60 Amp	425	425
Motors	111 in-lb., 86 RPM	1700	1700
Lidar	4 m Range	1700	1700
Motor Driver + ATmega32 Controller **	20 Amp	150	0
Powered USB Hub	7 USB Ports, 1 Amp	40	40
Wireless RC Remote	1 km, 6 Channels	75	75
Machining		930	930
Wheels	8 in Diameter	20	20
Laptop	i5 processor, 4 GB RAM, 1 GB Graphics Card	930	0
Wi-Fi Router	150 Mbps	30	0
Total		**9810 (USD)**	**6100 (USD)**

* - sponsored
** - reused from last year

10 Conclusion

Team Eklavya designed the Eklavya 2.0 with an aim to excel the challenges put forward by the IGVC-2013 and to provide a platform for research in the field of Autonomous Ground Vehicles. The team's effort and innovation is neatly reflected in the improvements on Eklavya 1.0 which reflect the strength of the Autonomous Ground Vehicle Research Group (Team Eklavya), IIT Kharagpur. Having achieved the goals set at the beginning of the year, the team is looking forward to contributing the ongoing research in the field of Autonomous Ground Vehicles and building a driverless car.

11 Acknowledgements

*The Team Eklavya is thankful to the Department of Mining Engineering, IIT Kharagpur for his constant support and valuable guidance. We would also like to express our thankfulness to our sponsors – **VectorNAV** and **MS2 Data Solutions** for their gracious contribution.*