PUSHPAK 3 - DESIGN REPORT FOR IGVC 2014

Indian Institute of Technology Bombay
P Maheshwari, K Keshav, P Jain, E Attarwala, R Yadav, A Gupta, K Gupta
Prof. S N Merchant (merchant@ee.iitb.ac.in)
Prof. R K Singh (ramesh@me.iitb.ac.in)

1. INTRODUCTION

The Intelligent Ground Vehicle Challenge provides an excellent opportunity for the student fraternity of the Indian Institute of Technology Bombay (India) to explore the design and implementation of advanced unmanned systems. Pushpak 3 is the third unmanned vehicle developed at IIT Bombay as an entry to 22nd Intelligent Ground Vehicle Competition.

2. TEAM STRUCTURE

The IIT Bombay team consists of students of various engineering disciplines, many of whom have worked on IGVC 2013 and ASME Student Design Competition 2013.

The team has 7 student members from various engineering disciplines. Considering the small size of the team, we opted for a flat team structure. The team has students involved in multiple roles as per requirement of the project. The structure we followed is as follows

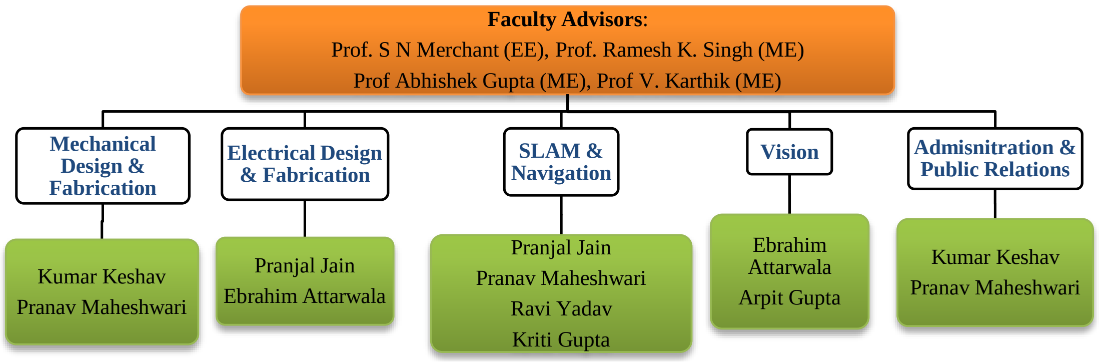

Figure 1. Team Structure of Pushpak 3

3. DESIGN PROCESS

Since this is our third year in the competition, we have taken several learning from our past year experience and hence we have evolved this methodology of working on parallel development. Since we already had a working prototype (Pushpak-2) we started working on the software end, meanwhile we simultaneously developed our new mechanical structure. Even though we completely overhauled the design, we made sure that there were no major changes that had to be implemented while switching to the new vehicle.

4. NOVELTIES

4.1 Software

- Robust and fast lane detection algorithm incorporating illumination correction
- Increase in field of view using two monocular cameras
- Extended Kalman Filter based localization using IMU, GPS and wheel encoders
- Hybrid global and local map based path-planning

4.2 Design

- System highly modular, easy to assemble and transport
- Compact double wishbone suspension system
- Vehicle industrially compliant with IP-65 protocol

5. HARDWARE DESIGN

5.1 Mechanical Design

The maximum load on the machine is during a 30^{O} incline. Assuming a total weight of 58 kg (with payload), and a wheel radius of 19 cm, the load torque on each motor is

$$\zeta = \frac{mgRsin\theta}{2} = \frac{58 \times \sin 30 \times g \times 0.19}{2} = \mathbf{27\ Nm}$$

The machine has to achieve an average speed of 1 mph (0.44 m/s). We designed the machine to have a max speed of four times this value. Hence, the speed of the motor at rated load is

$$\omega = \frac{4v}{r} * \frac{60}{2\pi} = \frac{0.44 \times 4}{0.19} \times \frac{60}{2 \times \pi} = \mathbf{88\ RPM}$$

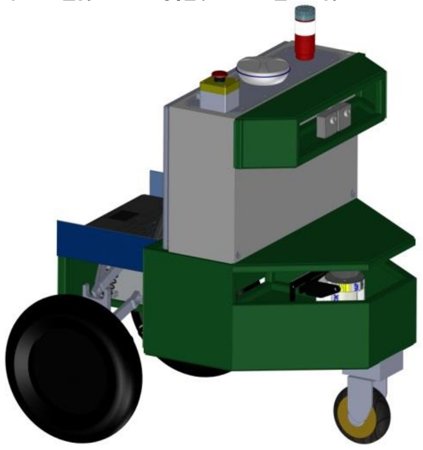

Figure 2. CAD model of Pushpak 3

Pushpak 3 measures 37”x26”x41” (length*width*height). The machine is designed for easy maneuverability in crunch spaces and hence we opted a simple 3 wheel differential concept with two driving motors and a free castor at the front. The height of the machine can be adjusted as per the requirement of the camera. The machine has two alloy wheels, each of diameter 15” and a castor of diameter 6”.

This year, machine is a little heavier from last year version considering that we have upgraded the machine to industrial grade. The machine weighs around 48 kilograms. Chassis weighs only 6 kg as the main skeleton frame is made using 3mm thick ¾” by ¾” Aluminum 6061 box. Sensors including LIDAR, GPS, IMU and camera cumulatively weigh 2.5 kg. Drive train comprising of motors, couplings, wheels and castor together weigh 7 kg. Suspension assembly weighs 5 kg. We have incorporated the used of suspension and dampers as we felt their need from our past experience to ensure smoother ride and less equipment failure due to shock. Processing unit (laptop along with the docking station) weigh 2.5 kg. Electrical components like Programmable Logic Controller (PLC), Human Machine Interface (HMI), power distribution system, batteries, inverter and motor controller are fitted inside a panel, which together weight 25 kg.

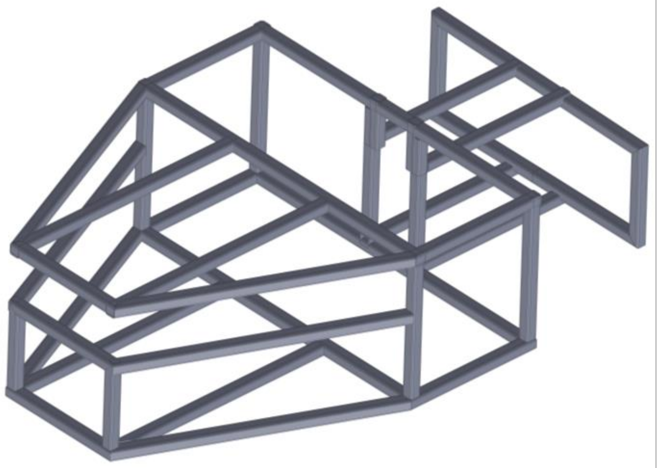

Figure 3. Chassis of Pushpak 3 made from 3mm thick ¾” by ¾” Aluminium 6061 box

Table 1. Design Specifications of Pushpak 3

Parameter	**Value**
Dimension	
Length*Width*Height	37*26*41 inch
Wheel Radius	7.5 inch
Castor Radius	3 inch
Mass	
Chassis	6 kg
Sensors (LIDAR, GPS, IMU & Camera)	2.5 kg
Processing Unit (Laptop & Dock)	2.5 kg
Panel (PLC, HMI, Batteries & Controller)	25 kg
Drive Train (Motors, wheels and castor)	7 kg
Suspension Assembly	5 kg
Total	48 kg
Motor Specifications	
Outer diameter	50 mm
Nominal Speed	5680 RPM
Nominal (Max. continuous) Torque	0.405 Nm
Nominal Power Rating	200 W
Gearbox Specifications	
Outer diameter	52 mm
Reduction	66 : 1
Max. continuous torque	30 Nm
Output	
Nominal (Max. continuous) Shaft Speed	86 RPM
Nominal (Max. continuous) Shaft Torque	26.73 Nm
Max Speed	3.85 mph

5.2 Cost Analysis

Table 2 includes a cost analysis of all the components in Pushpak 3. Some of these sensors and equipment (e.g. LIDAR, GPS, laptop etc.) were bought during the construction of Pushpak & Pushpak 2 hence their team cost is zero. While some of the other sensors and equipment (e.g. PLC, HMI, etc.) are included in this year's design and are included in team cost. There is also a third category of sensors and equipment, on which companies provided discount (e.g. Roboteq, IMU, etc.), and thus their discounted price is included in the team cost. Overall the machine is worth $20,630, while the actual expenditure during the construction was $7563.

Table 2. Cost of Various Components

Item	Capital Cost (USD)[a]	Team Cost (USD)[b]
Maxon Brushed DC Geared Motors	2650	2550
Roboteq HDC2450 Motor Controller	645	258
SICK LMS 111 Laser Range Finder	4300	0
Hemisphere A100 Crescent GPS Antenna	1900	0
VectornNav VN-200 INS	3200	1280
Allied Vision Pro Silica GT-1290	1300	0
Sick UM-18 Ultrasound Sensors	600	0
Spektrum Dx6i 2.4GHz trans-receiver	300	300
ThinkPad w530 laptop with Slice Battery	2300	0
PLC, HMI and Analog I/O module	900	900
FTDI-2232H USB-Serial Modules	35	0
Starcom PCI to dual ethernet express card	200	0
Li Polymer Batteries	300	300
Raw Material for Chassis & Fabrication	1600	1600
Wheels & Castor	100	75
Miscellaneous Electrical Hardware	300	300
Total	**20630**	**7563**

a: it is the market price of the equipment.
b: it is the actual amount paid by team for acquiring new equipment. For old equipment, this cost is zero.

5.3 Electrical Design

Unlike last year, Pushpak 3 is using three batteries for its power requirement. First is the main battery which supplies power for driving the vehicle. This is a 6 cell, 6 Ah lithium-polymer battery which last up to 3 hours on full charge. Second is the auxiliary battery which supplies power to all the sensors, controller, PLC, HMI and status light. This is also a 6 cell, 6 Ah lithium-polymer battery which last up to 4 hours on full charge. Third is the tertiary battery which is used to power the laptop and the docking station via an inverter. This is a 12V, 20Ah lead-acid battery which increases the battery backup of laptop from 3 hours to 5 hours.

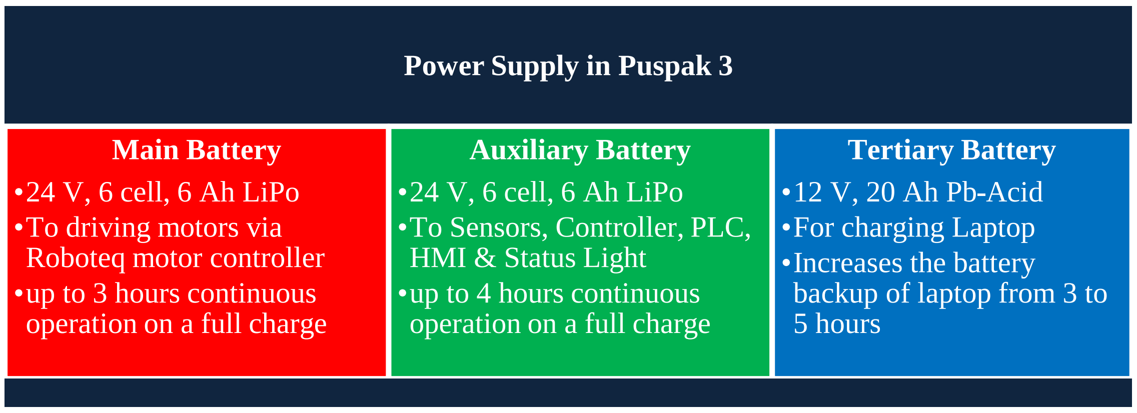
Power Supply in Puspak 3
Main Battery
•24 V, 6 cell, 6 Ah LiPo
•To driving motors via Roboteq motor controller
•up to 3 hours continuous operation on a full charge
Auxiliary Battery
•24 V, 6 cell, 6 Ah LiPo
•To Sensors, Controller, PLC, HMI & Status Light
•up to 4 hours continuous operation on a full charge
Tertiary Battery
•12 V, 20 Ah Pb-Acid
•For charging Laptop
•Increases the battery backup of laptop from 3 to 5 hours

Figure 4. Power Supply from various sources in Pushpak 3

Table 3. Average and Peak Power of various components of Pushpak 3

Average Power Drawn (W)	Peak (W)	Average (W)
Motors	400	150
Sick LMS 111 Lidar	25	20
Hemisphere A100 GPS	2	2
VectorNav VN-200 IMU	0.2	0.2
USB Camera	1	0.4
ThinkPad W530 Laptop	170	120
Status Light	10	6
PLC, HMI, Motor Controller	10	8
Total	**618.2**	**306.6**

5.4 Sensor Interfacing

Pushpak 3 consists of 5 different sensors (LIDAR, IMU, GPS, camera and encoders) which are interfaced using different protocols. The previous version operated on LabVIEW in Windows 7 for computation, but during our algorithm development, we found out that the amount of data generated by interfacing sensors and then using them in the path planning algorithm created a lag, which resulted in poor performance. So, we shifted our platform from LabVIEW to ROS groovy in Ubuntu 12.04. ROS is specifically designed for implementation of SLAM (Simultaneous Localization and Mapping) and Navigation which are most vital in IGV application.

The Hemisphere A100 DGPS antenna is interfaced using a USB to Serial Development Module (FT2232). It operates with an accuracy of 60 cm in its DGPS mode. Two USB cameras are used instead of one to increase the field of view for detecting lanes.

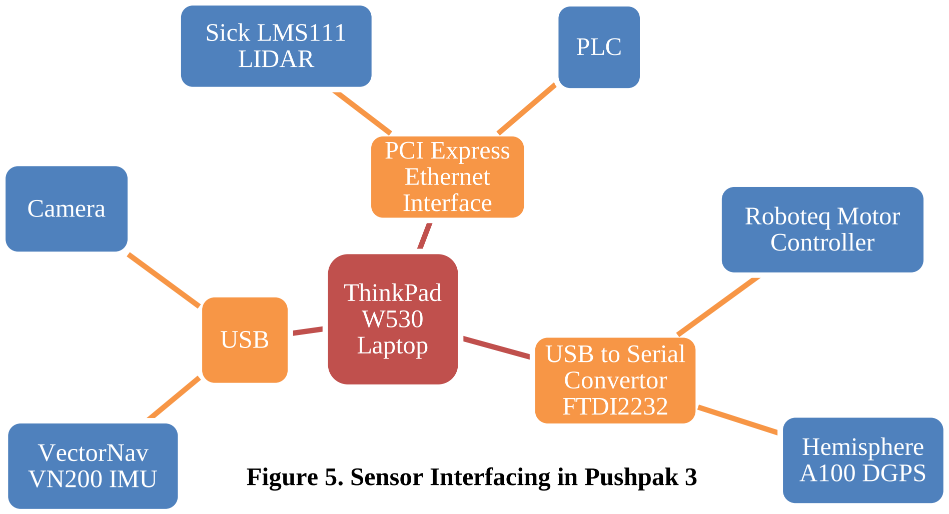

Figure 5. Sensor Interfacing in Pushpak 3

A Sick LMS 111 is fit in the front of the machine to provide a 2-D map of obstacles in front of the machine. It measures any obstacle's distance within 270 degree and 20 m of its position. The LIDAR and PLC both use Ethernet port for passing data via TCP/IP protocol. Since the laptop has only one Ethernet port, we used a PCI to Ethernet Express Card for fulfilling our need. The express card works at 400 Mbps and hence ensures no lag in data acquisition

The IMU is powered by USB port and it also uses the same Port to transfer the data. It can provide the Accelerometer, Gyroscope and Magnetometer reading along all 3 axes along with Yaw, Pitch and Roll information.

The motors are controlled by Roboteq HDC 2450 Motor Controller. The device comes in with an inbuilt 24 bit microcontroller which has a number of analog/digital inputs and outputs. It also has the facility of directly connecting the motor encoders as well as the RC remote inputs to the microcontroller. Status light is also interfaced using the motor controller and relay board. The inbuilt controller can be connected to laptop using USB and programmed using its proprietary software or serial port commands. This very feature eliminated the requirement of an external microcontroller.

5.5 Industrial Grade Automation

The main focus of this year's hardware design is robustness. The machine is fully compliant with industrial standards (IP 65). It has a programmable logic controller (PLC) integrated with a human machine interface (HMI) which enhances accessibility to the sensor data and thus improves debugging. It has two level of safety controls, the first level disconnects only the drive train while the second level protection isolates the machine from all three batteries, thus protecting all the instruments and sensors during any fault.

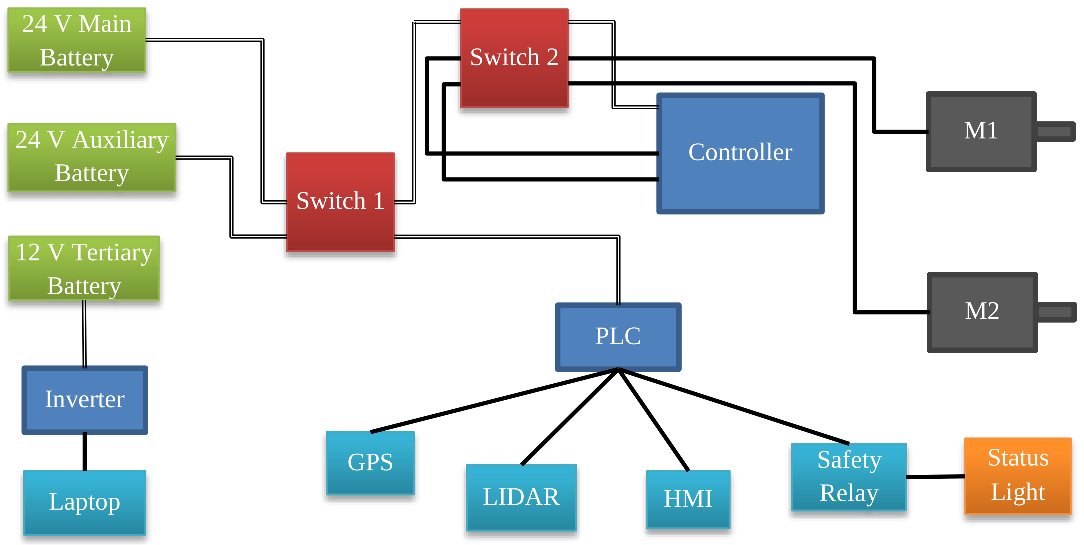

Figure 6. Schematic of Power Flow in Pushpak 3

6. SOFTWARE DESIGN

6.1 ROS based development platform

Unlike our previous vehicles where LabVIEW was used as a development platform, software systems of Pushpak 3 are developed on Robot operating system (ROS). ROS is an open source platform which is a collection of software frameworks designed specifically for robot software development. Using ROS is advantageous in a lot ways. It uses fairly versatile cross-language inter-communication system. Drivers for common sensors like LIDARs, cameras, IMUs, GPS units are readily available. In addition to this availability of numerous built-in software packages for common robotic applications like kinematic transforms and mapping allow us to implement complex algorithms in a very elegant and efficient way.

- **Composition of functionality:** ROS stack is a collection of nodes which is a process that performs specific computation. They communicate with each other using topics unto which relevant data is published which can be subscribed by any other node. This modularity makes implementation and testing of an algorithm simple

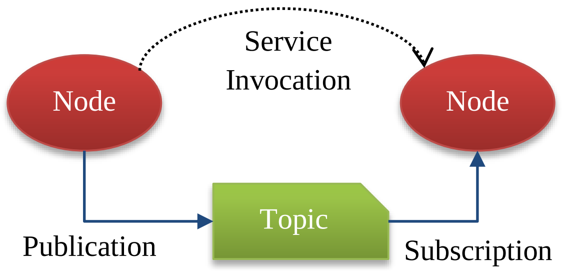

Figure 7. Subscriber and Publisher framework in ROS

- **Visualization and Monitoring:** During runtime it is always important to examine various state of the system. Dynamic and Modular nature of ROS topics can be exploited to observe any message stream publishing on the system. In addition to this ROS has powerful visualization plugin 'rviz' in which large variety of data-types, such as images, point clouds, geometric primitives can be displayed and visualized in an interactive fashion

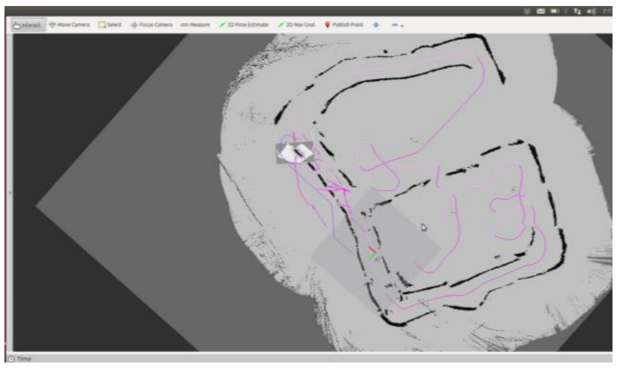

Figure 8. Mapping as seen in rviz

6.2 Image Processing

Image processing algorithm is written in C/C++ using Intel OpenCV libraries. It is used for lane detection and this year we have incorporated flags detection also. The images from the cameras mounted in front of the vehicle are processed in the laptop and then relevant information is passed to construct a map around vehicle.

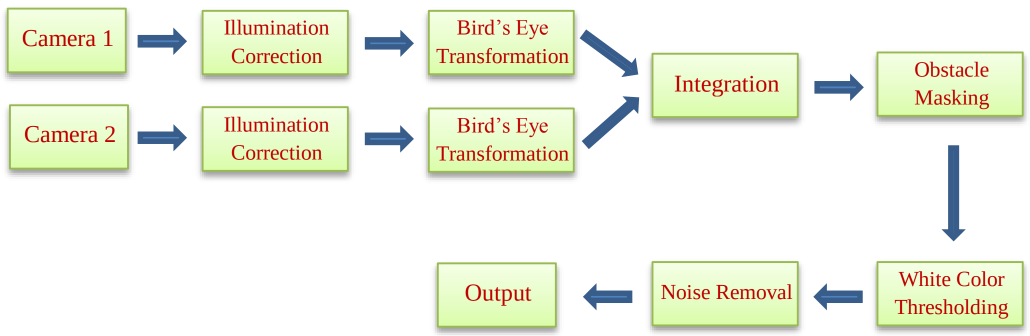

Figure 9. Image Processing Block Diagram

6.2.1 White Color Thresholding

White color thresholding algorithm classifies each pixel of the frame as white or non-white. This is done by converting RGB frame to HSV color space; this is because HSV color space is more related to human color perception. In HSV color space, hue component gives indication of the actual color, saturation indicates the darkness of the color and value indicates the intensity, illumination of the image. Because these three channels are independent, it is preferred to use HSV or YCrCb color space for color thresholding.

Output of the white color thresholding block would be binary image with only lane data as white and everything else as black.

6.2.2 Illumination Correction

The main problem in real time image processing in open environment is effect of sunlight. Sunlight region is mainly detected as white color in the white color thresholding algorithm which in turn gives false results.

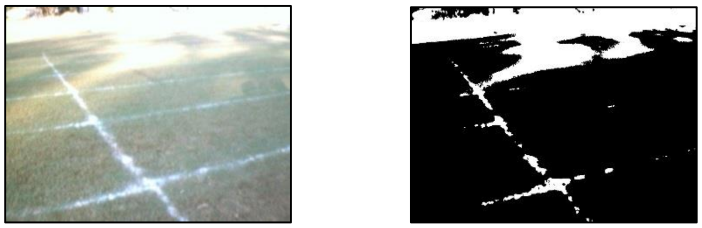

Figure 10. White color thresholding without Illumination Correction

In the previous version of the vehicle, we had used Discrete Cosine Transform (DCT) for illumination correction. The problem in that algorithm was it divides the whole image in finite blocks and then compared all such blocks based on intensity. So in case of no sunlight, lane information is being supressed. This year we have used Homomorphic Filtering for removing unwanted effects of variable illumination conditions.

Homomorphic filtering is generalized technique for image enhancement and/or correction. It simultaneously normalizes the brightness across an image and increases contrast. Image can be expressed as the product of illumination and reflectance:

$$f(x,y) = i(x,y)\,.r(x,y)$$

Illumination part is generally characterized by slow spatial variation and reflectance tends to vary abruptly. It means illumination is the low frequency component and reflectance is the high frequency component. To filter the illumination part out of the image, direct filtering is not possible as it is productive noise. So, we have to first take the log of the image and then have to apply the high pass filter to filter out the illumination part (low frequency).

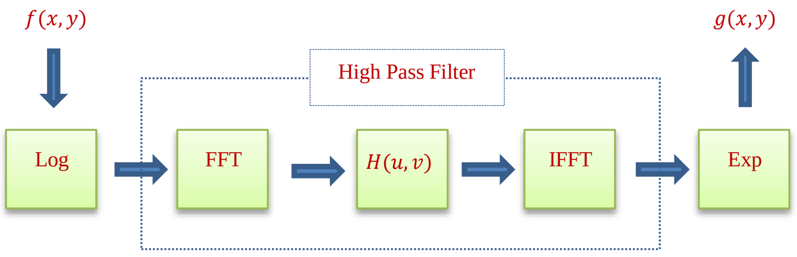

Figure 11. Homomorphic Filtering

$$\mathcal{F}\{g(x,y)\} = \mathcal{F}\{\ln i(x,y)\} + \mathcal{F}\{\ln r(x,y)\}$$
$$G(u,v) = I_l(u,v) + R_l(u,v)$$

Applying filter,

$$S(u,v) = H(u,v)G(u,v) = H(u,v)\,[I_l(u,v) + R_l(u,v)]$$

In spatial domain,

$$\begin{aligned} s(x,y) &= \mathcal{F}^{-1}\{S(u,v)\} \\ &= \mathcal{F}^{-1}\{H(u,v)I_l(u,v)\} + \mathcal{F}^{-1}\{H(u,v)R_l(u,v)\} \\ &= i'(x,y) + r'(x,y) \end{aligned}$$

Taking exponent,

$$\begin{aligned} s'(x,y) &= \exp\{s(x,y)\} \\ &= \exp\{i'(x,y)\}\,.\exp\{r'(x,y)\} \\ &= i''(x,y)\,.\,r''(x,y) \end{aligned}$$

Applying Homomorphic filtering on gray image removes the sunlight portion but there is problem in color thresholding algorithm. As in grayscale image there is only one channel, so only one degree of freedom, only one threshold that we can change. So, we can’t distinguish between the corrected sunlight portion and white region. So it is better to apply filtering on 3 channel image. First convert the image into YCrCb model and apply filtering only on Y channel as it represents the intensity portion of the image. This modified Y channel is then merged with the unchanged Cr, Cb channel to get the final filtered image.

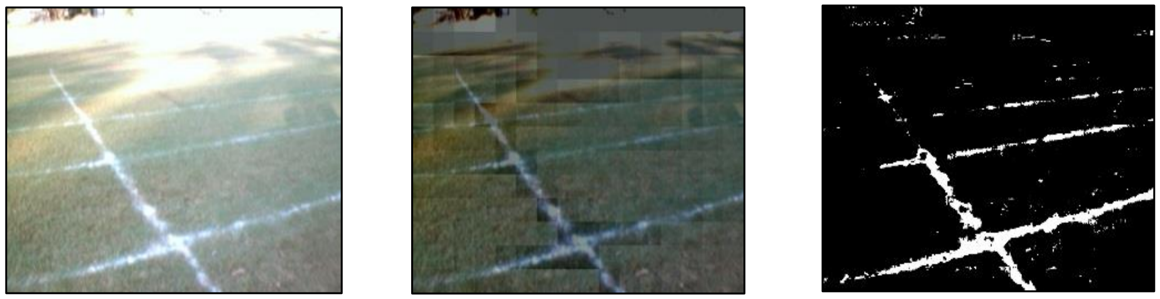
Figure 13. Illumination correction - Discrete Cosine Transform

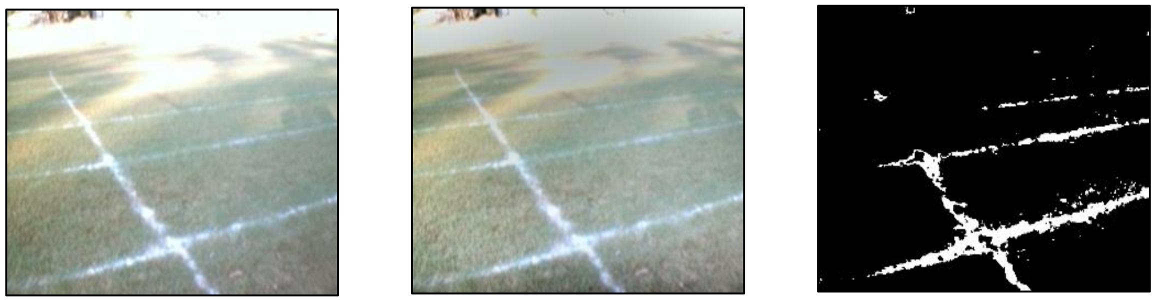
Figure 12. Illumination correction – Homomorphic Filtering

6.2.2 *Obstacle Masking*

Since we are just using white color thresholding algorithm for detection of lanes, obstacles in the frame may also be of white color which would interfere with the lane detection algorithm and gives false results. So for addressing this problem, we have used the data coming from LIDAR into image processing algorithm to mask the obstacles. Image is first converted to bird's eye view using perspective transform. Now distance of each pixel in a plane can be estimated by just accessing the pixel coordinates. So distance data from LIDAR is converted to pixel format and then corresponding pixels in the frame are then masked. Now, white color thresholding algorithm won't give false results because of the white obstacles in the frame.

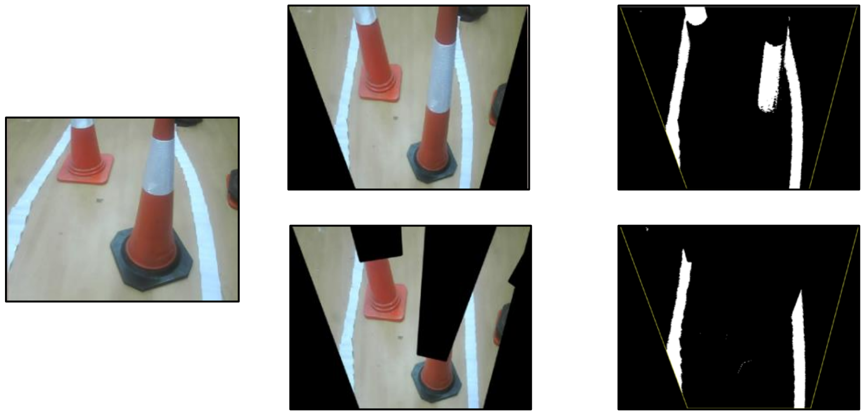

Figure 14. Obstacle masking using data from LIDAR

6.2.3 *Noise Removal*

Illumination correction followed white color thresholding algorithm gives the information of white line, but along with that some noise also gets detected due to uneven grass that needs to be filtered. For removal of that noise, contour filtering is been implemented. Contour detection is an important technique for object recognition and shape analysis. Here, it is used for noise removal. Contour detection algorithm gives the information of all the contours in the binary image. Noise patches also forms such contours but area of such noises would be much smaller than the white lane patches. So by just having a check on detected contour areas, noise can be filtered based on area thresholding. If area of a particular patch is less than some threshold, than it considered as noise and that contour is discarded.

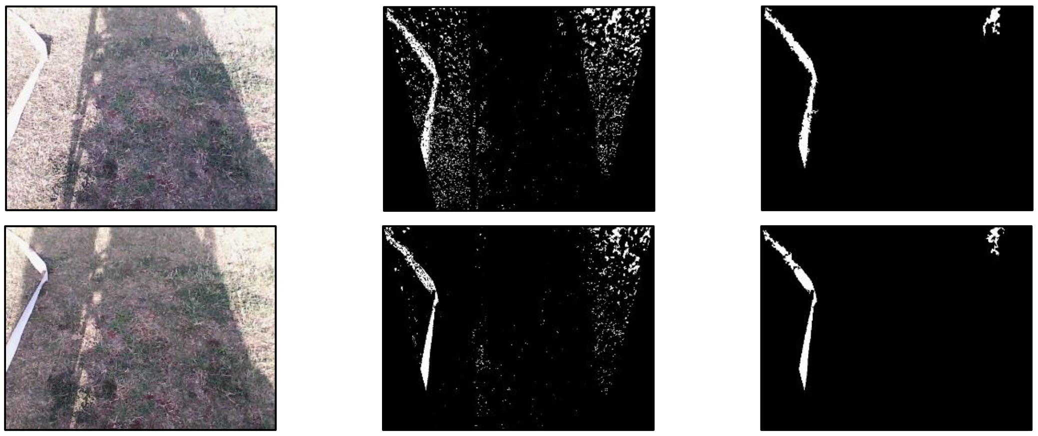

Figure 15. Contour based filtering of noise

6.2.4 Stitching two camera frames

Previous version of the vehicle was having one monocular camera with a field of view 60 degrees which results in having only single line at a time in one frame. This year, we have integrated two monocular cameras two give wider field of view so that environment is perceived with more precision.

Two separate frames, after applying bird eye transformation individually, are rotated, zoomed out and placed adjacently in such a way that the total field of view of the machine gets enhanced.

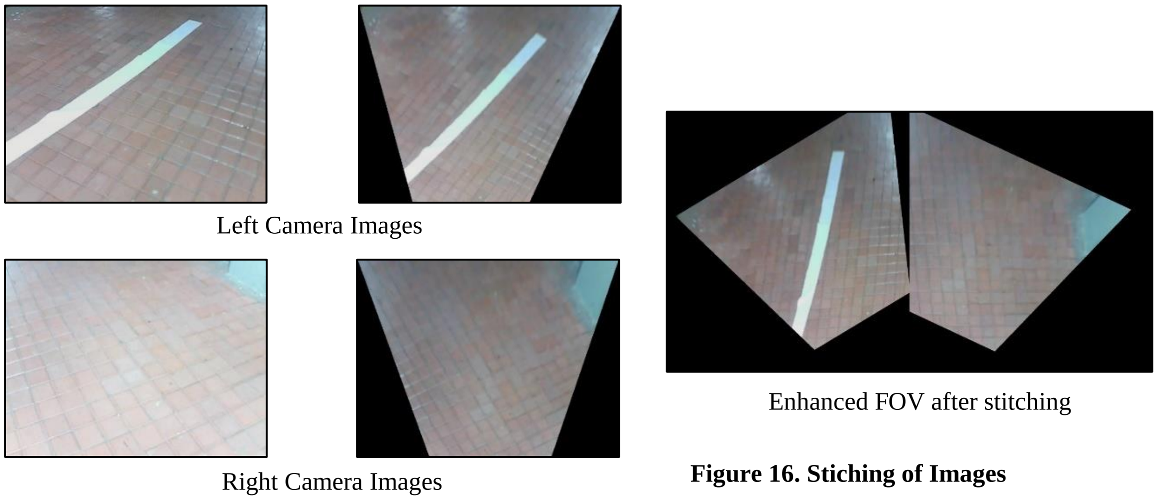

Figure 16. Stiching of Images

6.2.5 Flags Detection

HSV color model is used for red and blue detection. After color detection, filtered image is split into 3 channels, and then red and blue channel image is processed individually. Flag detection based on color channel gives small discontinuous contours which are then joined point to point by their respective centroids. This gives us red and blue lines, based on which allowed path to be followed is decided.

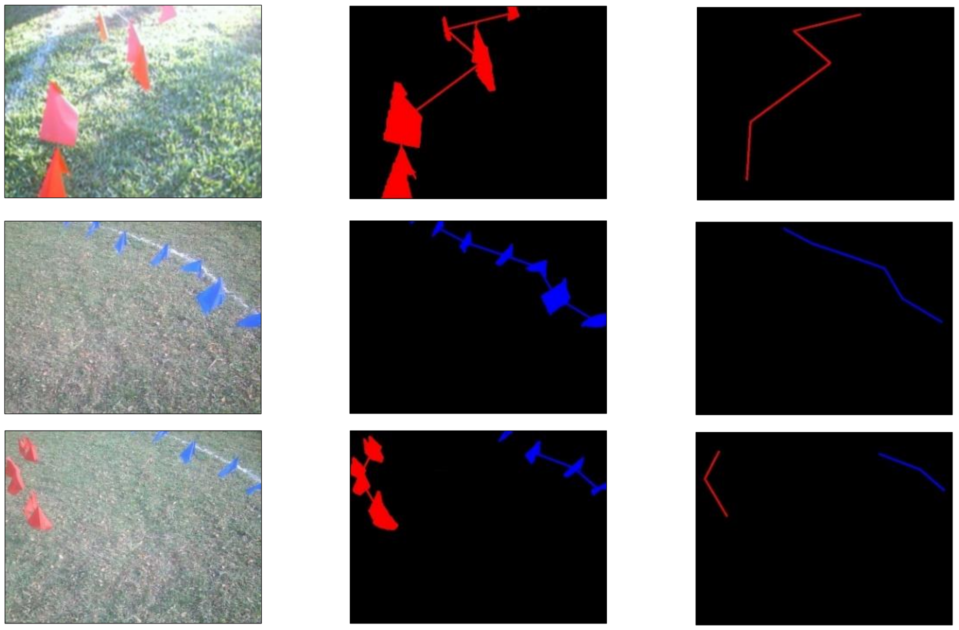

Figure 17. Flags detection based on contour joining

6.3 Environment Mapping

Environment sensing of Pushpak 3 is primarily based on LMS100 laser range finder and dual monocular camera. Occupancy grid mapping approach is used for generating 2d occupancy grid map with a resolution of 5cm X 5cm. Two different maps each for obstacles and lanes are generated using LIDAR and cameras respectively. Individual grid map cell contains floating point value from 0 to 1 which is directly related to the probability of existence of obstruction on that cell. Be it be line or obstacle.

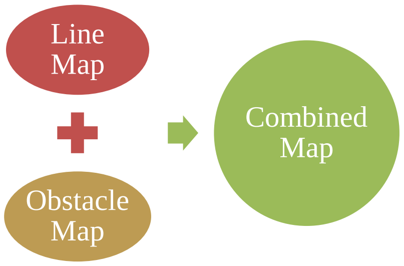

Figure 18. Environment Mapping

6.3.1 Line Map

A scaled image of the ground plane in black and white format (white represents obstructions) is obtained from the image processing unit. The information from this scaled map along with the localisation estimate is updated on an occupancy grid map. The resolution of the map is about 5 cm/pixel but varies slightly according to the calibration value of the cameras. The probability distribution and thresholding provides a low pass frequency filter which removes noise from the data to a large extent.

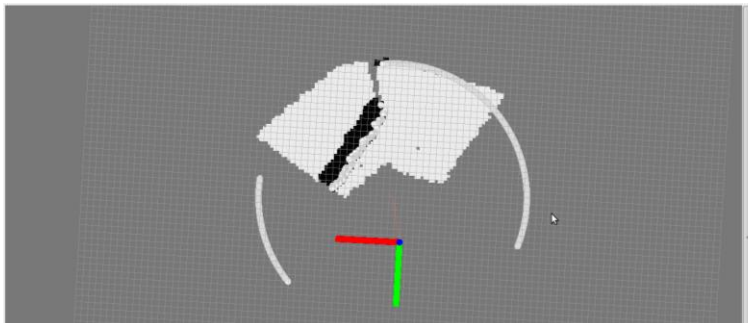

Figure 19. Line map

6.3.2 Obstacle Map

Polar map with a 240 degree field of view obtained from the LIDAR is converted into a 2D occupancy grid map of same resolution as that of line.

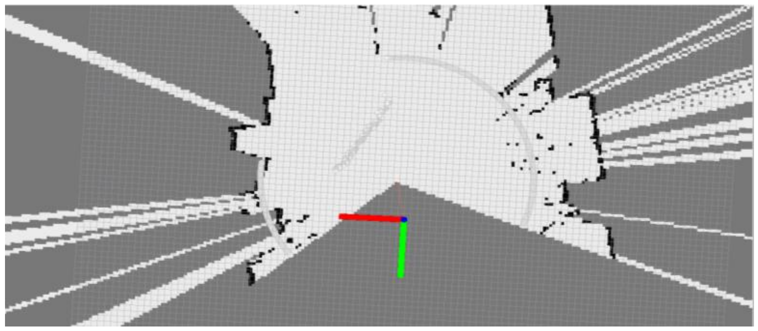

Figure 20 Obstacle Map

6.3.3 Map Fusion

Individual occupancy grid maps generated above are combined together and are stitched to together with a global occupancy grid map (generated in an incremental fashion using previous local grid maps).

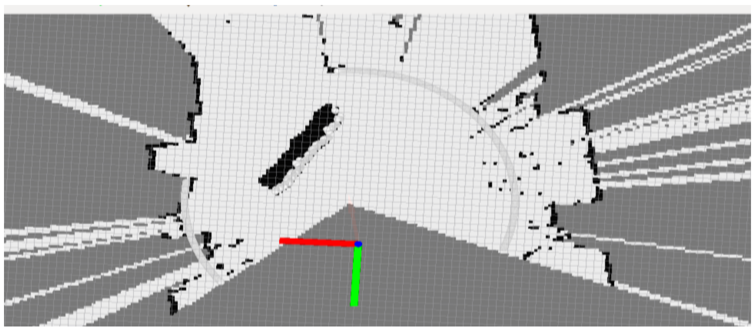

Figure 21. Combined map: line & obstacle

6.4 Localization

The environment in IGVC is sparse, thus traditional SLAM algorithms such as gmapping were found to be unreliable. The environment in IGVC is sparse, thus traditional SLAM algorithms such as gmapping were found to be unreliable. In order to increase the accuracy of our positioning, we use a Kalman Filter to integrate data from the GPS system with data from an INS (Inertial Navigation System) and encoders. Positioning even if no satellite is visible was also possible using this approach. Kalman Filter is an extremely effective and versatile procedure for combining noisy sensor outputs to estimate the state of a system with uncertain dynamics. In GPS/INS/Encoders integration case, noisy sensors include GPS receivers INS and Encoders components, and the system state include the position, velocity, acceleration, attitude, and attitude rate of our vehicle. Uncertain dynamics include unpredictable disturbances of the vehicle and unpredictable changes in the sensor parameters. The relatively low data output rate of GPS receivers does not meet the cm level accuracy requirements for robot localization in a global plan. This problem becomes more serious when the potential temporarily loss of a GPS signal occurs or phase ambiguity resulting from cycle slips considered. INS provides the dynamics of motion between GPS epochs at high temporal resolution and complements the discrete nature of GPS in the occurrence of cycle slips or signal loss.

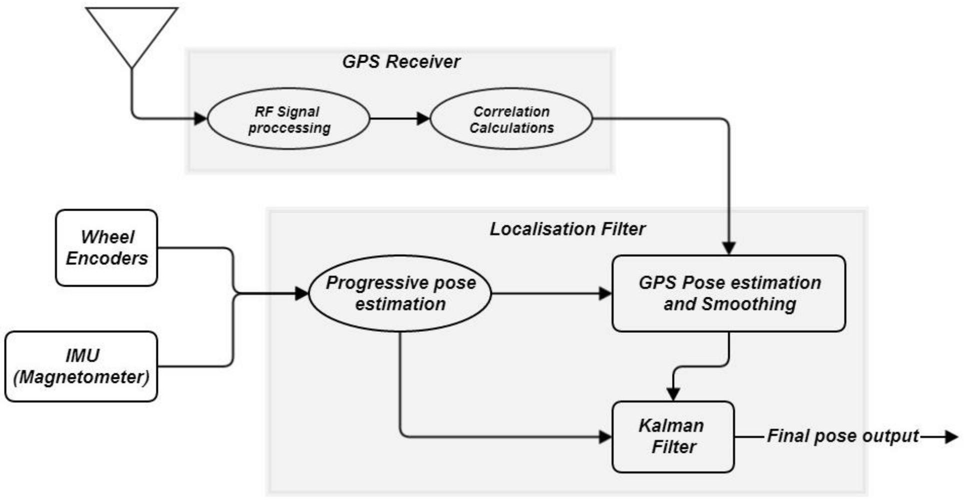

Figure 22. Localization Block Diagram

In addition, positioning with INS requires the integration with respect to time of accelerations and angular rates, the measurement noise accumulates and results in long wavelength errors. GPS errors do not accumulate, but in short term, they are relatively larger and the measurements have poorer resolution. Thus using these characteristics a novel EKF based algorithm was used to effectively fuse the data coming from these sensors. Positional accuracy of 25cm was achieved using this methodology.

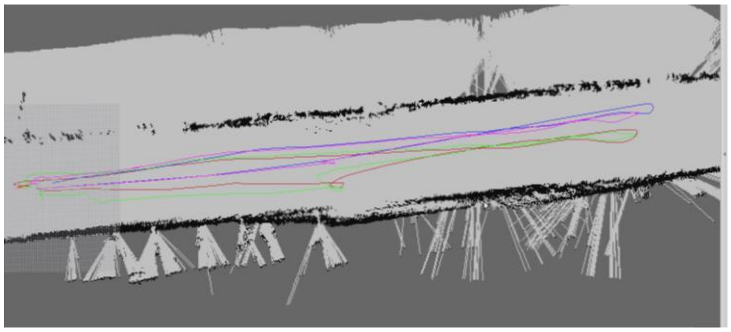

Figure 23. Comparison of trajectory using Kalman Filter

6.5 Navigation

The above discussed global and local maps are fed in to navigation planner. Navigation planner works in three steps, it calculates a path from current position to next GPS waypoint considering global map, it finds out the local direction for moving forward and obstacle density of the environment using local map and then it converts it into motor commands.

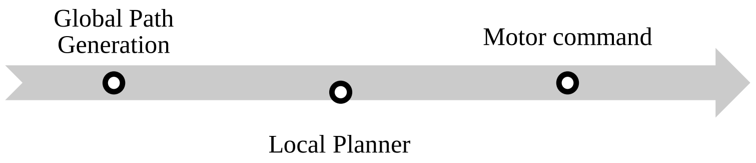

Figure 24. Hybrid global and local path planning

6.5.1 Global path generation

Dijkstra's algorithm supported by heuristics based on distance transform is used for path finding. The path planner works on distance transforms of obstacle and line map to find a possible and optimised route.

A basic filter is implemented on the line map to remove the remaining noise before taking its distance transform. It thresholds the area of the cluster of continuous line and filters out relatively small chunks.

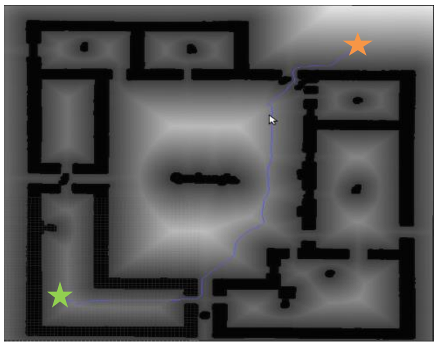

Figure 25. Obstacle heuristics

Distance transforms of obstacle and line are created individually according to the Euclidean distance to nearest obstruction. These values for each grid are incorporated in its cost using an inverse exponential function which decides the spread of the obstacle. The spread of line and obstacle are configured such that vehicle doesn't finds a way through small unmapped regions of the lines. The algorithm provides a lot of parameters for different platforms and environmental conditions.

6.5.2 Local Planner

It converts the global path into motor commands with the help of local map. The algorithm runs in the polar coordinate system in the frame of the robot

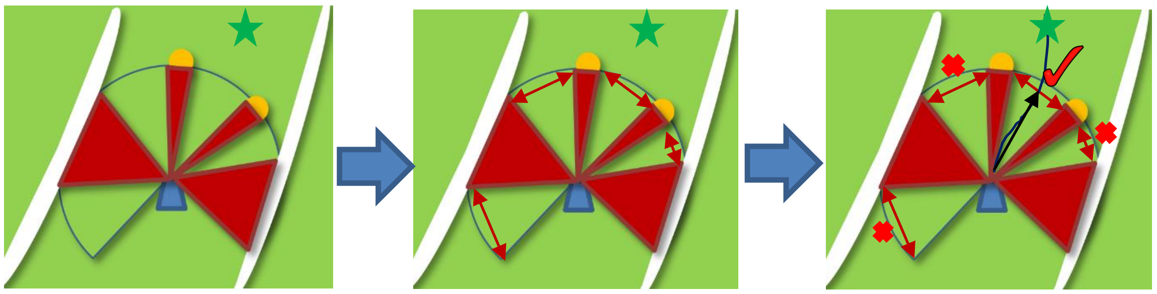

Figure 26. Local map and planning

A threshold is set and all angles with distance lesser than this threshold qualify as obstacles (assigned value 0), while others qualify as open path (assigned vale 1). Thus sectors of 1 and 0 are obtained. Each sector is analyzed and if its width is greater than vehicle's width, it qualifies as valid path. This gives an array of valid paths that the vehicle can take safely. The vehicle is asked to follow the valid path obtained from the A-star algorithm. If, however, the path isn't valid or for some reason the path cannot be calculated, the vehicle is asked to follow closest to absolute heading obtained

Even if it doesn't get a global path, it switches smoothly to move in a local direction. It is responsible for the reactionary avoidance as well.

6.5.3 Motor Command generation

The above module gives an absolute heading on which the vehicle is supposed to move. The vehicle has a feedback control based on digital compass and inertial measurement unit which helps the vehicle achieve the heading very quickly. The vehicle also has individual feedback control on both driving motors based on rotary encoders. This control loop ensures both the wheels are moving with the desired angular velocity. The motor commands are determined by using intricate relations between velocity, omega and difference between present heading and the desired heading obtained from the navigation module, minimum obstacle distance and direction of obstacle scarcity.

7. PERFORMANCE ANALYSIS

Table 4. Performance parameters of Pushpak 3

Speed	Maximum RPM of the motors up to 88 and wheel radius of 19 cm restricts the speed to 4 mph
Ramp	Pushpak 3 is tested to climb slopes up to 30 degrees
Reaction time	The current reaction time of the vehicle is about 60 milliseconds
Battery life	Main driving battery of the vehicle last about more than 3 hours
LIDAR range	The LIDAR is able to detect obstacles up to 18 m, but we are using 8 m range to avoid noise
Waypoint accuracy	Vehicle has localization accuracy of 25 cm within the map, GPS points can be achieved with a tolerance limit of 50 cm

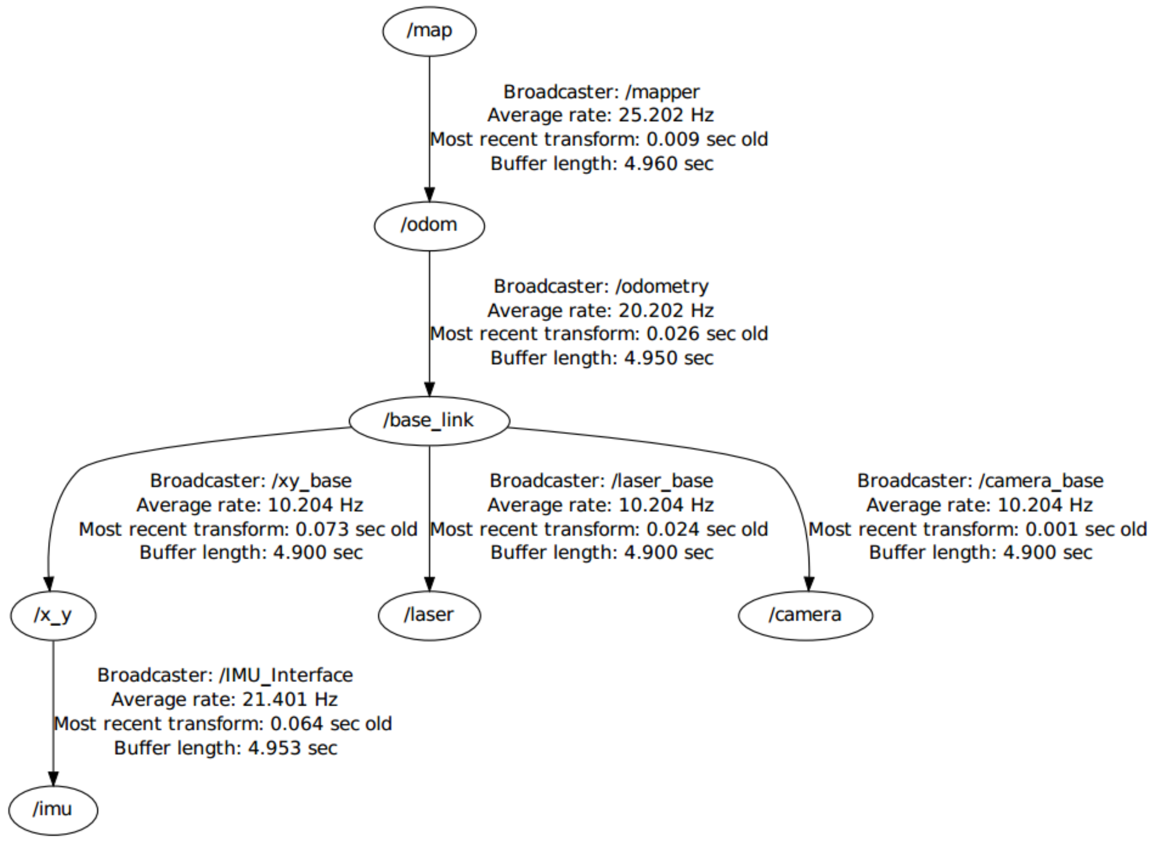

Figure 27. Kinematic transforms between different frames in Pushpak 3

8. CONCLUSION

Pushpak 3 performed magnificently during the test runs and we expect the same in the main competition also. The robustness in both mechanical and software design has improved the reliability of Pushpak 3 over its earlier versions. It has been designed considering the scalability to a full scale self-driving car. We hope that this report clearly illustrates the innovations we have made and help readers develop better understanding of our work.

9. ACKNOWLEDGEMENT

We would like to thank IIT Bombay for being our chief patron and providing us major funding for vehicle construction, travel and logistics. We are thankful to our industry sponsors for providing us financial and technical support pertaining to usage of the sensors in the most efficient way. We would like to thank our faculty advisors for their extreme support and invaluable guidance. We would also like to acknowledge never ending support from our distinguish alumni network of IIT Bombay.