Intelligent Ground Vehicle Competition 2019

Team Abhiyaan

Virat

Indian Institute Of Technology Madras

Design Report

May 01, 2019

I hereby certify that the development of vehicle, Virat, as described in this report is equivalent to the work involved in a senior design course. This report has been prepared by the students of Team Abhiyaan under my guidance.

Dr. Sathyan Subbiah
Faculty Advisor, Team Abhiyaan
Associate Professor, Department of Mechanical Engineering
IIT Madras

Team Leaders
Sourabh Urade (uvw8275@gmail.com)
Bakthakolahalan Shyamsundar (kolahalvsl@gmail.com)

Team Members
Abhijith KV, Devansh Jain, Karthik Bonda, Kumud Mittal, Mahesh Balasubramani, Meenakshi Ravishankar, Sai Kumar A, Sajay V, Shashank M Patil, Skanda Swaroop, Sooryakiran, Sreerag EP, Sriram Raghunathan, Veerendra Harshal Budhi

Contents

1 Introduction

Team Abhiyaan is a group of 15 interdisciplinary students enrolled in undergraduate as well as postgraduate engineering programs of Indian Institute of Technology (IIT) Madras. Fueled by common passion for autonomy, we work in Center For Innovation (CFI), a 24x7 Student Innovation Lab in IIT Madras. We have considered all the failures and shortcomings that Kernel had faced in IGVC 2018 and reassessed our thought process and brainstormed to come up with novel and innovative ideas. Virat is our third prototype manufactured for the purpose of participating in IGVC and this design report serves to document all the details. Virat is superior than its predecessor Kernel 2.0 in terms of both performance and reliability. Virat is powered by state of the art Artificial Intelligence techniques surpassing its previous version by a large margin.

2 Team Organization

The team is organized into 4 modules: Mechanical, Electronics, Software and Management; based on the expertise and domain knowledge that is required in building the prototype and striking sponsorship deals for components and logistics. Modules have clear cut roles and responsibilities and many team members contribute to multiple modules . The team is overseen by a faculty advisor and two team heads. The team heads coordinate between each modules, organize meetings, evaluate progress and report to the faculty advisor. Each module has a head and is responsible for progress within the module. Module heads report progress to the team heads and each other. The Management team looks after all non-technical tasks including talent acquisition, public relations, strategic industrial relations etc. The Mechanical module has members working in Design, Simulation and Manufacturing of the prototype. It overlaps with the Software and Electronics team in the area of Control systems and Mechatronics. The Electronics team deals with the electronic circuitry, sensor data acquisition & signal processing and actuation. They overlap with the Software team in the area of embedded systems and micro-controller programming. The software team works with the development of software solutions for Autonomous Navigation and Inter-operability.

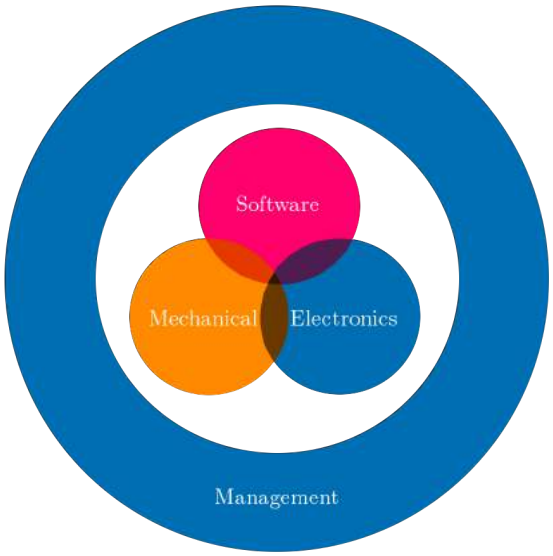

Figure 1: Team Structure

3 Innovations

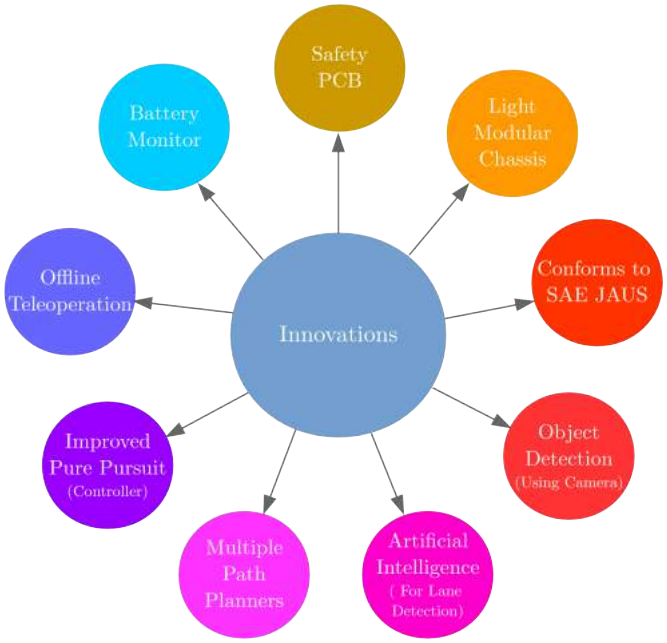

Figure 2: Innovations in Virat

- **Conformance with SAE JAUS**
  Virat is conforms to the Joint Architecture for Unmanned Systems(JAUS) standards set by the Society of Automotive Engineers (SAE).

- **Object Detection Using Camera**
  We use inputs from the camera to detect regions where there is high probability for an object to be. This information helps removal of false detections of lanes. This also helps in removal of striped barrels and other objects from the camera frame.

- **Deep Learning Based Lane Detection**
  We developed lane detection algorithms using state of the art Deep Learning and Artificial Intelligence techniques. This prooved to be more robust and stable than all previous versions.

- **Multiple Pathplanners**
  The vehicle can toggle between the following path-planning algorithms before every run.

  - **RRT* Informed**
  - **A***
  - **Dijkstra**

  Rapidly exploring Random Tree (RRT) is an exploration based path-planning algorithm. It offers efficient re-planning with dynamic obstacles. A* and Dijkstra were already implemented in the previous version.

- **Improved Pure Pursuit**
  We use a dynamically varying goal directed algorithm (Pure Pursuit) in the controller module. It computes

short-term goals, plans a smooth path for the vehicle to reach it and generates the differential drive velocities based on the trajectory and vehicle kinematics. The algorithm was improved to take care of edge cases which the previous version (Kernel 2.0 IGVC 2018) was unable to handle.

- **Offline Teleoperation**
  The vehicle can be remotely controlled using a compatible joystick even if the on-board computer is off or in sleep mode.

- **Battery Monitor**
  Virat has an inbuilt battery health monitor which gives a notification on the onboard computer if the battery level drops below a threshold value.

- **Safety PCB**
  A separate PCB that distributes power to various sensors from the mother board is designed to maximize the ease of handling of multiple sensors.

- **Size Optimization**
  Unlike last year's bot which was quite big, we have optimized the size to be just above the minimum required dimensions. This will make movement easier, allow for smaller turning radius and will lead to reduction in weight. At the same time, we have not compromised on space for placing components and devices.

4 Mechanical Design

4.1 Introduction

The primary objective of mechanical module of Team Abhiyaan was to build an efficient bot considering both mechanical and electrical aspects. While designing the bot we concentrated on bot's structural rigidity, weight optimization, low center of gravity for increased stability, serviceability, aesthetics and compliance with rules. We also tried to overcome various problems faced by the previous year's bot.

Figure 3: 3D model of the bot

4.2 Chassis Design

Mechanical designing consists of chassis design and analysis, material selection , motor and gear combination, 3D modeling of parts and assembly. This year's bot has three wheels, two wheels connected to motor in the rear and one dual castor wheel setup on the front side.

- **Modularity**: The design was made from scratch to include the new changes. Keeping transportation constraints in mind, the chassis is made modular by dividing it in two parts, namely, front and rear portions. Mechanical fasteners and welding were used to join the individual members in the portions respectively. Allowance for sensor mounting and device placement was considered according their requirements while designing. We got constant feedback from the software and electrical modules for the same during the design phase.

- **Modeling**: Autodesk Fusion 360 was used for modeling of the bot and to create manufacturing drawings. The chassis, all the sensors, motor assembly, wheels and bot covering were modeled and assembled on the software. The rear portion was made by a combination of aluminum sheet and beams and the front one with a combination of aluminum strut profiles and beams. We also made sure that the bot looked aesthetically pleasing and incorporated some stylish designs while modeling. We designed the bot in Fusion 360 according to the rules and then manufactured it.

- **Material Selection**: It was decided to use aluminum for the bot frame due to its relatively low weight for its strength and cheaper cost compared to other materials. The chassis is the structure which bears overall load of the bot and it needs to be strong and rigid enough to carry this load. Initially it was planned to use beams to manufacture the rear chassis but experience from previous years required us to reduce weight and dependence on welding and hence we went with Al sheet metal of thickness 4mm. For the front portion, aluminum struts from Modular Assembly Technology were used for their versatility and weight to strength ratio.

- **Wheel and Motor Assembly** This time, we used better driving wheels which have a greater width as a result of which we get much better traction.As opposed to last year, we have used only one dual castor wheel and that too made of rubber and with inbuilt bearings to reduce vibrations and increase smoothness while driving. The driving wheels at the rear of the bot are attached to the aluminum frame by means of bearing and fasteners which provides a reliable connection to the main body. Aluminum was selected for manufacturing of motor mounts. The motor mounts carry a large percentage of stress from the wheels and house expensive motors making their structural integrity of paramount importance. Thus it was decided to use bearing plates since it provides extra strength against larger bending torques even though it commands a higher weight. The tires are attached to the body via a well distributed assembly which solved last year's problems with connection and bending of wheels. Castor wheels are mounted with the help of an adjustable mount whose height can be varied slightly if needed.

- **Motor and Gear combination**: The possible motor types and gear ratio are selected by calculating the torque required for driving the vehicle at maximum required speed. Total effective effort can be given as:

$$T = \frac{TTE * D}{2 * n} \tag{1}$$

where, T = torque per wheel, TTE = Total tractive effort, D = Diameter of powered wheel.

$$TTE = RR + GR \tag{2}$$

where, TTE = Total tractive effort, RR = Force necessary to overcome rolling resistance, GR = Force required to climb an inclination.

$$RR = M * g * C_{rr} \tag{3}$$

where, M = Total weight of the vehicle, g = gravity, C_{rr} = Surface friction.

$$GR = M * g * sin(\theta) \tag{4}$$

where, θ = Maximum inclination

$$\omega = \frac{2V}{D} * \frac{60}{2\pi} \tag{5}$$

where, ω = Angular velocity of wheels.
For M = 50kg, g = 9.81 m/s^2, $C_{rr} = 0.055$ (grass), θ =15 and D = 254 mm (10 inch):
$RR = 27.06N$, $GR = 127.34N$, $TTE = 154.4N$, $T = 9.8044Nm$
Therefore,

$$V_{max} = 2.23m/s(5mph) \Rightarrow \omega_{max} = 167.67RPM$$

$$V_{min} = 1.34m/s(3mph) \Rightarrow \omega_{min} = 100RPM$$

According to above results, following combination of motor and gear chosen:

S No	Specification	unit	Value
1	Nominal Voltage	V	24
2	Nominal Current	A	10.8
3	No load speed	RPM	5950
4	Nominal Torque	mNm	405

S No	Specification	unit	Value
1	Reduction	-	66:1
2	Max. continuous torque	Nm	30
3	Mass inertia	gcm^2	16.7

4.3 Manufacturing of the bot

Figure 4: Front portion manufacturing and Total Chassis.

- **Process Planning** - The bot is divided into two parts,front and rear berth. The rear berth was manufactured first followed by motor to wheel coupler and then the front portion was manufactured.
- **Manufacturing** - The aluminum chassis was manufactured at the IIT-M Central Workshop and a few other workshops around Chennai. For high precision, CNC machines were used. For the sheet metal components, laser-cutting and press brakes were used. Wherever aluminum beams were to be welded, tig welding was carried out using highly skilled labor. We also used 3-D printing for manufacturing some sensor mounts. A strict timeline for manufacturing was followed to devout to troubleshooting and reiterating the design.
- **Waterproofing** - Virat can be operated in the event of light rains as all the electrical and software components have been covered with suitable waterproofing material. GPS and the LIDAR body are waterproof by themselves and hence, are left open.

5 Electrical Design

5.1 Introduction

The electrical circuitry of Virat is an improved revision of the previous one. It includes a custom designed central Printed Circuit Board that coordinates and caters to all features of the vehicle. The design is more safer and efficient, hence increasing the runtime of the vehicle by a considerable amount. We implemented a distributed power system this year that uses a single battery. We also developed a compact and customized Printed Circuit Board to cater to all features of the bot. TIVA TM4C123GXL is the micro controller used. Motors and motor drivers have been calibrated in a better way to reduce odometry errors to a large extent.

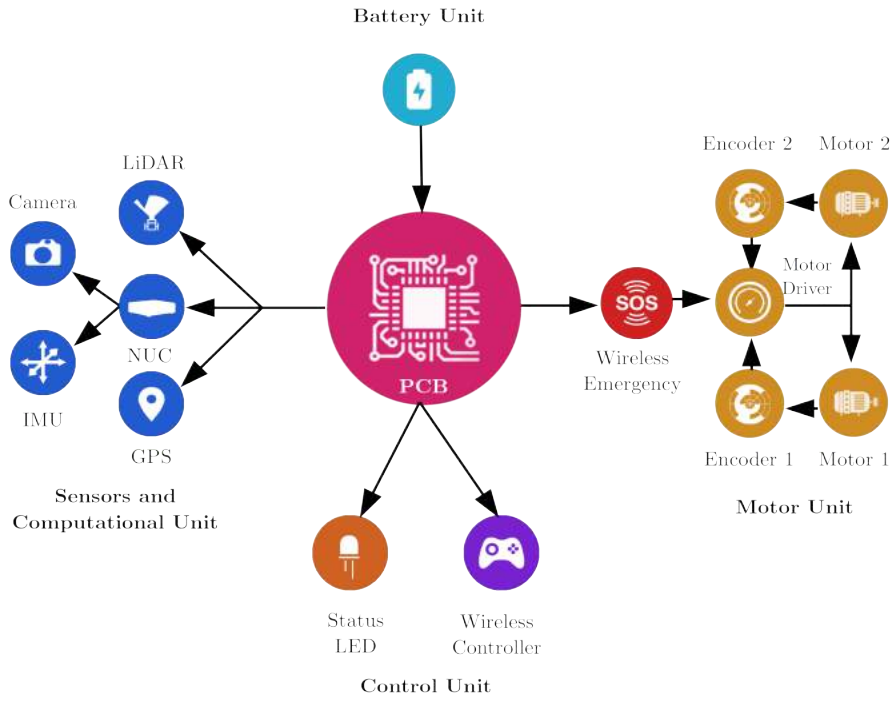

Figure 5: Electrical Flow diagram.

5.1.1 Safety PCB

The main components and sensors which include GPS, LIDAR, Router, Roboteq motor controller, NUC and the USB Hub are connected to their respective power supply via a fuse and switch, in a PCB separate from the main one. These are then connected to the main PCB. This is done to provide additional safety to these components, and also for easier switching of required components, while testing the bot.

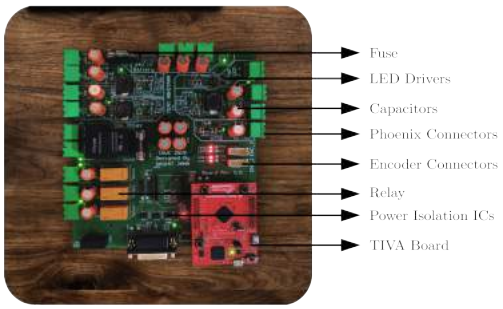

Figure 6: Components of PCB.

5.1.2 Power Distribution

The primary power for all systems is provided by the designed Printed Circuit Board. All the components in the vehicle are powered by a single source of 25.9V, 46.8 Ah Lithium ion battery. The PCB supplies 3.3V, 5V, 12V, 19V and 24V to the required components. Different buck converter ICs are used from efficient stepping down

of voltages. These ICs turn on and off using PWM. Thus by using the desired duty cycle, we get appropriate power supply with an efficiency of more than 95% in conversion of voltages. This is better than the Linear Voltage Regulators used last year which were not energy efficient. ICs used for power distribution are

1. TPS54202 - For converting 24 volt input to 5 volt output. This 5 volt is used for powering other IC's and for powering SC189
2. TPS54202 - For converting 24 volt input to 12 volt output. This 12 volt is used for powering relay and router
3. TPS54302 - For converting 24 volt input to 19 volt output. This 19 volt is used for powering NUC.
4. SC189 - For converting 5 volt input to 3.3 volt output. 5 volt is output of TPS54202 and output 3.3 voltage is used to power XBEE and SI8621.

5.1.3 Battery Level Indicator

We process the data required to calculate the battery voltage level as analog readings using the TIVA microcontroller, and apply a moving average filter on it to smoothen the spikes and noise. The battery voltage levels will then be sent as messages to the ROS system from PCB. So, in case the battery levels go below 22V, all controls from software would be stopped to prevent damage to the components.

5.1.4 Signal and Power Isolation for Roboteq

No path must be created between the ground terminal of the Input Output DB15 Connector of Roboteq and ground terminal of the main power supply. In case the controller's ground terminal is disconnected but power terminal is still connected, high current may flow through controller which may damage the controller. Thus the signal and power should be isolated. We use RE0505S to power isolation and to create Isolated continuous power supply for input/output connector of Roboteq. Thus they create an electrically isolated supply from the main power supply from the battery and works like a secondary battery. SI8621 is used to isolate 3.3 V signals data transmission signals. SI8620 isolates 5V encoder channel signals.

5.1.5 LED Driver

LEDs are current controlled devices. Hence the LED driver AL8860 maintains constant current. The IC can take upto 1.5A and the inductor used regulates it to 300mA. The input Supply is up to 40V. By changing the frequency, the LED can be made blinking or solid depending on manual or autonomous mode of Virat.

5.1.6 Evaluation of PCB

Several test points have been placed around the PCB for troubleshooting any problem. LED indicators green in color has been used to indicate power transmission whereas red LEDs are used to indicate data transmission.

5.1.7 Minimizing EMI Radiations

1. As the LED Driver switches on and off at a higher frequency, an alternating magnetic field is created which leads to production of electromagnetic radiation. These waves affects the IMU data, NUC or any sensitive electronics. Thus the magnetic field is shielded by using ferromagnetic material.

2. Loops in the PCB act as inductor coils and enhance electromagnetic interference. Thus the area of loops is minimized to reduce the EMI radiation.

3. Thin ground wires have been used to prevent back flow of EMF and flow to other components since it has higher resistance.

4. Noise generation during communication has been reduced by keeping the communication lines away from noise generation circuits and keeping them near ground.

5. Schottky diode is used for fast switching of relays and it also has very less voltage drop. Ferrite bead is used to reduce the electromagnetic interference generated due to fast switching of relays

5.1.8 Safety

1. Resettable Fuses are installed in the PCB which provides resettable overcurrent protection. Once the circuit is broken for higher currents, it makes the circuit again after a small time interval.

2. Buck Converters have short-circuit protection. Thus the maximum current won't go above a specified limit in any case in PCB.

3. A diode is placed to prevent plugging input terminals in the reverse polarity.

4. If the input voltage is less than 24 volt, the power circuit is cut off which provides protection against low voltage.

5.1.9 Miscellaneous

1. Relays are operated by providing signals from TIVA through ULN2003 Darlington Transistors.

2. Output voltages of three terminals can be chosen between 5V, 12V, 24V by populating the corresponding resistor.

3. The TIVA board can be powered either through PCB or through USB while priority is given to USB by using a diode.

5.2 Power Consumption

Virat uses a Lithium-ion battery of rating 25.9V-46.8Ah with total capacity of 1200Wh. The Battery has a built-in battery monitoring system (BMS) for cell balancing, over-voltage, under-voltage and short-circuit. As the figure shows, the Lithium-ion battery provides 24V to the LiDAR, Motor Controller, GPS and Buck converter that converts 24V to 19V to power up the NUC, 12V to power up the Router and LED, 5V to power up the TIVA-board and 3.3V to power up the XBEE. It also provides the required voltage for powering up the DC to DC buck converter ICs, Relays, Led Driver IC etc. Having a single source for everything makes Virat more compact and simple. The maximum estimated power consumption including consumptions by each component, DC-DC conversion losses and technical losses (including Heat losses) is

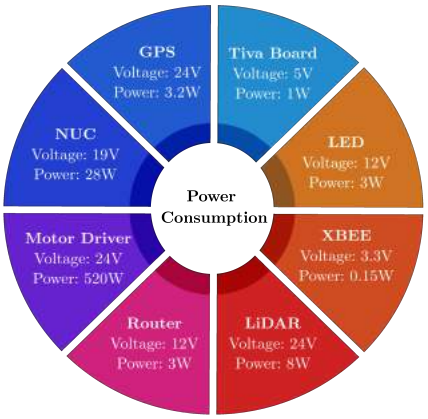

Figure 7: Power Consumption Chart.

680W.

Estimated Runtime considering Maximum power consumption = Total Capacity divided by estimated power consumption Max = 1.76 hr

The runtime of 1.76 hours is when the bot is under maximum power consumption. In normal running cases, we had a runtime of over 4 - 5 hours during testing period. The batteries takes about 2 hours to be completely charged. Virat uses Lithium ion batteries having more volumetric as well as specific energy densities compared to last year's Li-Po batteries. This reduces payload as well as storage space. All battery connections are made by using XT-60 connectors with appropriate power ratings.

5.3 Motor Interface

Virat uses the Roboteq SDC2160 2x20A high performance dual channel brushed DC motor controller with HEDS-5540 quadrature encoders. The controller is operated by Robot Operating System (ROS) through serial communication. It includes an elite 32-bit microcomputer and quadrature encoder input to perform motion control algorithms in closed loop speed mode. Closed loop speed modes ensure that the motor(s) will run at a precise desired speed. It collects the motor‘s feedback from the encoder through digital input pins of the controller which utilizes PID loop control to improve the precision of speed. This controller has internal memory which is able to hold configuration settings that controls operation of motors.

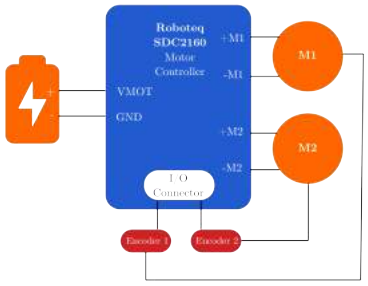

Figure 8: Motor Interface.

5.4 Wireless Joystick

Figure 9: Flow of Data.

A wireless joystick has been designed for controlling the bot manually when it is not in ‘autonomous’ mode. The joystick is an analog 2-axis thumb device. Data is transmitted through XBEE S2C which has an operating range of 2000 feet. The wireless Joystick can also switch between manual and autonomous mode. It could also stop the bot in case of an emergency.

5.5 Emergency Stops

The following are the implementation of emergency stops in Virat:

- Mechanical E-Stop - Hardware Controlled
- Wireless E-Stop - Hardware Controlled

- JAUS Emergency message - Software Controlled

The motor power is connected through the RF module on the bot. This RF Module helps to control the circuit wireless by switching it ON/OFF using a remote control and has a range up to 100 meters. This serves as the Wireless E-stop, added to the Mechanical E-stop (Push-button) on the vehicle. Both the Wireless E-Stop and the Mechanical E-Stop turn off the power of only motor instead of turning off the entire vehicle and it's components which may cause damage to NUC, Router and other electronic devices. Both these devices are hardware controlled. The vehicle can also be entered in emergency mode through software control using appropriate Joint Architecture for Unmanned Systems(JAUS) transmissions by a controlling agent. When Virat receives an emergency message through a JAUS message, the velocity of the vehicle is immediately set to zero and the vehicle enters an emergency state. A 'clear emergency' message has to be transmitted through JAUS for the vehicle to return to normal state.

6 Software Strategy

Virat's software stack is built on the Robot Operating System(ROS) and runs on Linux. ROS offers simple interfaces with sensors through *nodes*, *topics* and *services* to help manage the hardware abstraction and low-level control. Moreover, ROS is very popular and it has a large and active community. There are ever-growing contributions and the APIs are well documented which accelerated the development of our software stack. It is scalable, provides graphs of processes for easy debugging and allows control of the robot through multiple networked machines.
Virat uses state of the art techniques like Artificial Intelligence and Deep learning for perception.

6.1 Introduction

We obtain noisy data from the environment at every instant of time. Our goal is to process and filter this noisy data in real time and take optimal actions that will yield better results in the long run. The actions that we choose in turn affect the environment. Our software stack is designed to be modular so that any component of the stack can be replaced independent of the other components. This allows for major changes and replacements in the code-base with minimal change to the existing code. We also developed our own algorithms that are robust and can be customized to suit our needs.
Our navigation stack can be sub divided into Perception, Processing, Planning and Pursuit.

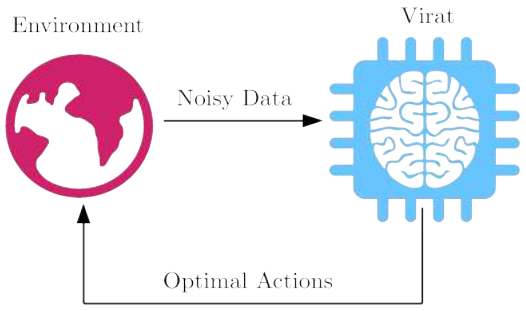

Figure 10: Role of Software Stack.

6.2 Perception

Perception includes getting data about the surrounding environment. Virat uses a 2D Light Detection And Ranging (LiDAR) as a range sensor. The LiDAR gives data about the location of the immediate obstacle in every direction from the bot. This helps in determining the location of barrels and other obstacles. Virat has 2 forward facing cameras slightly inclined to the left and right. Optical data is used for lane keeping. The wheel

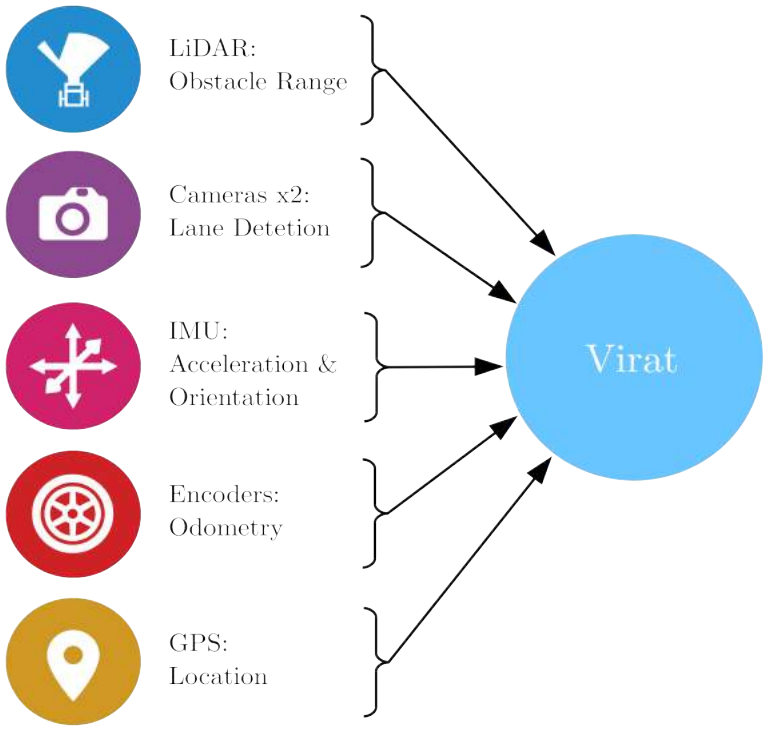

Figure 11: Sensors and Perception in Virat

encoders give information on how much each wheel has moved from the start. This information can be used to track the position of the vehicle. We also have an Inertial Measurement Unit (IMU) to get the orientation of the vehicle. The IMU also acts as an accelerometer.

6.3 Processing

Mapping and Obstacle Avoidance

For indoor regions, the GPS data is insufficient for accurate localization. So we use a 2D LiDAR to perform Simultaneous Localization And Mapping (SLAM). It scans the environment horizontally and laser scans are fused with odometry data to map the surroundings. In outdoor regions, localization is performed using Extended Kalman Filter and the obstacles are detected using laser scans. An occupancy grid is constructed using this information. Information about lanes and potholes are also added to the map.Localization is essential for getting accurate pose estimates of the robot. The robot must be able to locate itself in a particular frame of reference and must be able to keep track of its state as it moves. Since we have to localize in a sparsely populated outdoor environment, SLAM is not a good choice as the features are sparse. Hence we resort to using GPS coordinates (latitude and longitude) as the major source of location data. We also get pose estimates from the wheel encoders which track the odometry of the robot. However we observe much more drift and increase in covariance of the wheel odometry data and hence we use the GPS as the primary source of position data. The IMU helps in getting accurate orientation and angular acceleration data which can be fused together with the GPS position data to get an accurate pose estimate of the robot. The IMU also offers us data about linear acceleration however we observe this to be noisy and hence discount this data. We fuse all the sensor data using Extended Kalman Filter which uses a non linear model to propagate the system state. Although vanilla KF which uses a linear model has been theoretically proven to converge unlike EKF, the latter performed much

Figure 12: Lane Detections (Raw outputs from the network) illustrated in the blue channel. *Test Image From: SeDriCa, Indian Institute of Technology Bombay, IGVC 2017 Winning Run*

better in our case.

Lane Detection

The vehicle has to detect white lanes on green grass. We use a Fully Convolutional Neural Network to segment out lanes in the images. The architecture of the network is custom designed to obtain best results and to reduce the time for inference. The network is trained with images of lanes from IGVC from the past. The architecture was implemented in Tensorflow and integrated with ROS. The image was pre-processed using histogram equalization. This is done to nullify any effects arising due to the different white balance in cameras. The output is then resized to 512x512 and fed to the neural network. The network uses a stride of 2 and does not use max-pooling as opposed to the convention. This is because having a stride of 2 approximates the result of a convolution operation with stride one and max-pooling with kernel size 2x2, with significantly lower amount of computation (nearly 1/4th of the original computational expense). Padding with zeros were done to make the output have the same size as that of the input. Transpose convolutions (aka Deconvolutions) were used for image generations. Rectified Linear Units (ReLUs) were used as activation functions in all layer except the last where sigmoid was used to squish the output values between 0 and 1. The output is a 512x512 matrix with each pixel representing the probability of that pixel being in a lane.

We notice that the network has even learned to differentiate between sky and lanes. We also see that no striped barrels are detected as lanes. White potholes can also be detected by the Network.

6.4 Planning

Path planning deals with the problem of finding the optimal path from start state to goal. There are two types of planners viz.Global Planners,Local Planners. Local Planners deal with finding optimal path over short distances in order to avoid dynamic obstacles whilst the Global Planner finds the overall optimal path from

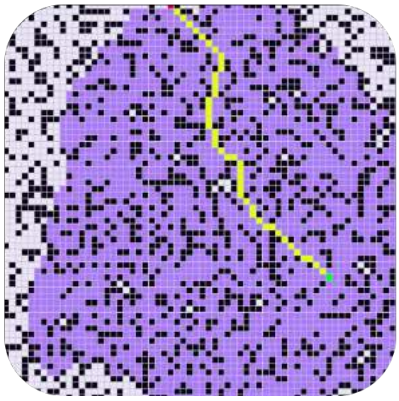

Figure 14: Dijkstra Path Planning.
Purple cells indicate the cells
visited before finding an optimal path.

Figure 15: A* Path Planning.
Purple cells indicate the cells
visited before finding an optimal path.

start to goal. The Local Planner that we are using is integrated with the Pure Pursuit Controller(see 6.5).

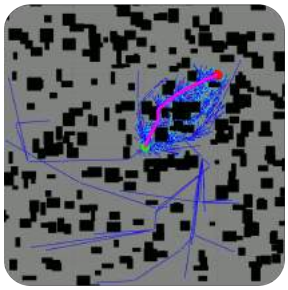

Figure 13: RRT* Informed Path generation

RRT(Rapidly exploring Random Trees) can efficiently deal with unseen obstacles without the need to re-run the algorithm multiple times. RRT takes in cost of each grid given by the costmap and gives the optimal path by growing a tree towards the goal. Only a part of the tree is removed whenever an obstacle is encountered and the forest of trees thus formed are recombined into one tree by random growth of vertices. RRT finds the solution to single query problems efficiently but RRT* extends RRT to find the optimal path but asymptotically and in doing so finds the optimal path from a given state to every other possible state within the planning domain. This is inefficient for a single-query problem. Informed RRT* intelligently chooses only a subset of these possible states and finds the optimal path. This subset is chosen such that inclusion of any state outside this set does not significantly improve the solution. The states in this subset are described by a prolate hyperspheroid(heuristic). Also for the given subset the problem converges at a faster rate towards the optimal solution.

Dijkstra's Algorithm finds the shortest path by expanding vertices in the graph starting with it's initial pose/vertex. It expands the closest unexpanded vertices in an iterative manner until it reaches the goal. Dijkstra's will always find the shortest path as long as none of the edges have negative cost. This planner is the easiest to implement and will always find the shortest path.

A* is very similar to Dijkstra's algorithm, the only difference being that it takes into account both the exact cost of expanding a vertex as well a heuristic cost. In each iteration is maintains a balance between the two. It chooses the vertex with the least of the sum of exact cost and heuristic cost. A* like Dijkstra is simple to implement and is very reliable.

6.5 Pursuit

Pure Pursuit Controller

The controller gives appropriate control inputs to the robot to ensure that the planned path is followed. We use a pure-pursuit based controller to move the robot smoothly while taking its non-holonomic constraints into account. The controller essentially tries to mimic the human driving nature by taking a point at a fixed distance in front of the vehicle as the immediate goal. This goal changes continuously as the vehicle moves on the planned path, traversing through a coarse set of waypoints. Once this look-ahead goal point is found, the controller algorithm plans a smooth curvilinear path from the current position of the vehicle to this point which is used to generate velocities to be given to the wheels for following the intended course.

6.6 Simulation

Gazebo, an open-source platform has been used for the simulations as it is highly integrated with the ROS communication framework and the codebase also uses ROS's latest communication framework. A simplified 3D model of the bot in the form of URDF file was then created that adds metadata to the meshes. The main aspects of metadata include collisions, joint limits, inertia. The sensor data collected from the plugins is made available through topics to other sub-modules that we have deployed. This allowed us to have an easy understanding of what the vehicle is seeing when it encounters a situation, which facilitates easier debugging and optimization.

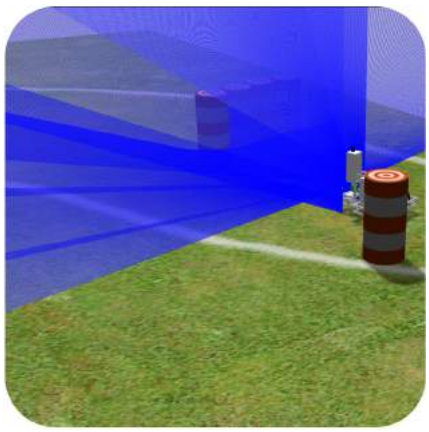

Figure 16: Simulation in Gazebo

6.7 Interoperability

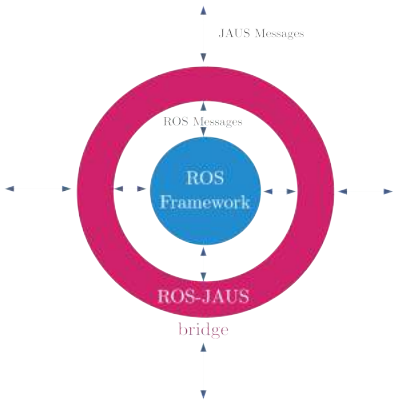

Figure 17: JAUS-ROS Bridge

Virat conforms to the Joint Architecture for Unmanned Systems (JAUS) standards set by the Society of Automotive Engineers. We implemented the bridge between ROS and JAUS using OpenJAUS. This bridge acts as an interface between Virat's JAUS components and ROS implementation. Virat's JAUS subsystem consists of 2 components:

- **Platform Management Component**
  The platform management component implements management services such as Discovery, Liveness, Access Control, Events and Transport (all services implemented have version 1.0). Events like report heart beat pulse and report access control are also implemented in the Platform Management Component.

- **Navigation and Reporting**
  The Navigation and Reporting component implements services like Management, Waypoint Driver, Waypoint List Driver, Velocity State Sensor, Local Pose Sensor and Primitive Driver other than those inherited from the Platform Management Component.

7 Initial Performance Assessments

- Lanes and potholes are being detected upto 10 feet with nearly 140° field of view at more than 10fps.
- Lane Detection algorithm is able to differentiate between objects similar to lanes like striped barrels.
- Barrels and obstacles are getting detected up to 10m distance.
- The University of Michigan Benchmark(UMBmark) test, a bi-directional Square Path Experiment is used as a dead reckoning test for localization of the bot.The bot is programmed to move in a square of side 1m in both directions clockwise and counter clockwise.However due to uncertainity in odometry there will be some error in final pose when compared to initial pose.Moving in both directions removes the error due to unequal wheel diameter as we are interested in uncertainity due to GPS,Wheel Odometry and IMU.

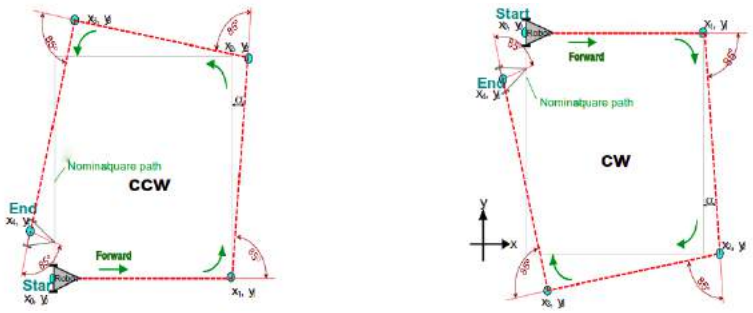

Figure 18: UMBmark experiment

The final position is not always exactly same and in order to calculate the offset we find the centre of gravity of the cluster of final positions.An accuraccy of 10cm is obtained using GPS,Wheel Odometry and IMU.

- The bot is able to move and turn as per the instructions from controller.
- Ramps of an inclination of 14 degrees are covered with average speed of $1.5m/s$ with a payload of 5 kg.
- Runtime of Virat with nominal load was found to be 4 hours and with full load is estimated to be 1.76 hours.
- Emergency and safety features are tested for different scenarios and are found to be working.

8 Failure Modes

S.No.	Failure Mode	Resolution
1	Low frame rate for lane detection and frame out of sync with real time.	Distribution of computational load to GPUs
2	Blind spot in front of the bot for cameras	Position another camera top down or use a 3D camera
3	Errors in transformation of detected lanes to real world frame.	Use automatic ground plane detection
4	Immediate goal directly behind the vehicle & out of sensor fields and dead end in front.	Implement higher level planner using Finite State Machines concept to handle recovery behaviours
5	Slippage during tight turns	Implement adaptive linear velocity control to reduce speed when in tight turns and areas of higher obstacle density
6	Vibrations of the camera mount leading to distorted images	Implement digital image stabilization or use mechanical stabilization techniques
7	Holes off-centering due to temperature change	Slotted L- clamps
8	Vibrations caused by loosening of the bolts	Use washers and lock nuts
9	Wobbling or bending of wheels	Tighten the nuts joining the wheels to the coupler as soon as problem is observed
10	One or more component failures	Implement a health monitoring system to immediately stop the vehicle automatically to prevent damage or injuries.

9 Cost Estimation

S.No.	Component	Retail Cost[1]	Team Cost[1]
1	Sick LMS1xx LiDAR	5000	0
2	RPLiDAR	600	600
3	Hemisphere A101 GPS	3000	0
4	Sparton AHRS-8P IMU	1500	0
5	Intel NUC	1000	0
6	Logitech Camera	150	150
7	Printed Circuit Board	70	70
8	Roboteq SDC2160	125	125
9	TIVA TM4C123GXL	15	15
10	XBEE S2C	15	15
11	Maxon Motors	937	629
12	Maxon Gears	738	508
13	Encoders	218	146
-	**Total**	**13,368**	**2,258**

10 Acknowledgements

We would like to thank our institute IIT Madras for providing us a platform to work on our own autonomous robot and giving us the chance to participate in IGVC 2019. We would like to thank the Center For Innovation (CFI) for supporting us through this journey to IGVC. We would also like to thank our faculty advisor Prof. Sathyan Subbiah for guiding us all along.

[1] All costs are in US Dollars