A Prototype of an Intelligent Ground Vehicle for constrained environment: Design and Development

Apoorve Singhal* Vibhakar Mohta* Yash Khandelwal* Adarsh Patnaik* Manthan Patel* Jaydeep Godbole* Shruti Priya* Sombit Dey* Shrey Shrivastava* Anand Jhunjhunwala* Shreyas Kowshik* Deepank Agrawal* Siddhant Agarwal* Arvind Jha* Rishabh Singh* Kousshik Raj* Shubham Sahoo* Ashutosh Singh* Ritwik Mallik* Vaibhav Lodhi Debashish Chakravarty

***Abstract*—Autonomous vehicles are bound to take over the urban road scenario in the near future. While fully autonomous driving is yet to be deployed on urban roads unconditionally, constrained environments provide an opportunity for preliminary testing and validation as the technology emerges. Autonomous robots can be tuned to be robust in constrained environments. The technologies developed can then be extended and transferred to unconstrained environments with required safety precautions. This paper describes the design, development and testing of EKLAVYA 7.0, an autonomous differential drive robot that can follow lanes while avoiding stationary obstacles as well as navigate through a series of land markings specified by GPS coordinates. It was developed to participate in the 27^{th} Intelligent Ground Vehicle Competition (IGVC 2019). The paper describes the overall mechanical, embedded and software architecture developed for this constrained environment along with system integration, testing and results.**

***Index Terms*—Autonomous vehicle, Lane detection, Localization, Obstacle Detection, Motion Planning**

I. Introduction

The field of autonomous vehicles has seen great development in recent years owing to various conferences and competitions being organised. The advent of high computation power and high fidelity algorithms have been a crucial step towards accomplishing complex tasks. Starting from the first government initiative and encouragement with the DARPA Grand Challenge, we now have a confluence of government, industry and academia taking active interest in autonomous driving. Intelligent Ground Vehicle Competition (IGVC), founded by The Association for Unmanned Vehicle Systems International (AUVSI), is one of the leading competitions in this field where teams have to design and build an autonomous ground vehicle with the ability to navigate through a constrained environment and a set of GPS waypoint targets while detecting and following lanes and avoiding obstacles.

This paper discusses the design and prototyping of EKLAVYA 7.0, an intelligent ground vehicle built to participate in IGVC-2019. The entire system is divided into three modules that discuss the independent systems of the vehicle. The first section covers the aspect of design and analysis of the mechanical structure covering stability analysis and drive systems of the vehicle. The structural stability is compared to that of the design of Eklavya 6.0 [1]. A major part in development of such vehicles is the embedded interface for the system that includes modules like power distribution and battery management. A state of charge estimation technique as discussed in [2] is implemented for battery management and monitoring of battery life in the vehicle. The second section covers the embedded setup of the vehicle including the circuit design, sensor setup and interface modules for the deployment of the software stack on a physical system. The main focus of this competition is to see the development and implementation of new and efficient algorithms with focus on mobile robots. The third section presents the software stack developed for the vehicle that includes sub modules of localization, planning, perception and simulation. An Extended Kalman Filter based approach for accurate localization as discussed in [3] has been used in Eklavya 7.0 that can handle multiple sensor inputs including wheel encoders, GPS, Inertial Measurement Unit. Methods of optimal trajectory generation using elastic band approach as discussed in [4] have been used for Eklavya 7.0's planning module. It is assisted with path optimization in distinctive topology to ensure feasible path generation as discussed in [5]. The lane detection approach of Eklavya 7.0 has been optimized with respect to trade-off between perfomance and computation power. It combines obstacle removal with thresholding on various sub-spaces to extract lane features. RANSAC [6], an iterative algorithm robust to outliers is used for lane model prediction. The last section discusses the method of system integration and data handling for the vehicle.

II. Hardware

A. Mechanical design

The mechanical structure of the vehicle is designed to ensure structural stability, supplemented with a light weight body to allow easy maneuverability on unstructured terrains. The CAD model of the vehicle is made on Solidworks in order to simulate and visualize the vehicle in virtual environments designed for the competition.

* Denotes equal contribution

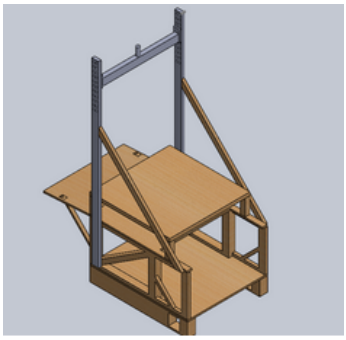

Fig. 1: Chassis Frame on SolidWorks

The chassis frame as shown in Fig 1 is carefully designed to balance all possible stresses faced by the vehicle during operation. The frame is made of teak wood to impart structural rigidity to the vehicle body and the top and base of the vehicle is made of ply for reducing the weight of the vehicle chassis. The frame is divided into several triangular members with longitudinal zero force members that provide structural stability. This design on static structural analysis on ANSYS (Fig 2) reduced the maximum stress by a factor of 10 in comparison to our previous design [1]. The camera mount is attached to the sides of the body of the vehicle to provide greater surface area of contact and less vibrations.

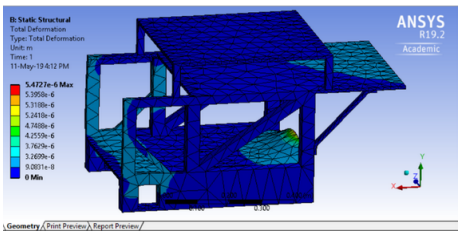

Fig. 2: Static Structural Analysis of the chassis

B. Drive Mechanism

Eklavya 7.0 has two front pneumatic wheels and a rear swivel caster with a simple differential drive mechanism with front wheels driven for maneuvering the vehicle. The pneumatic wheels provide high traction on grassy terrain to prevent slipping and provide accurate wheel odometry information. It also acts as a form of suspension for the vehicle. The differential drive mechanism allows simple implementation and easy maneuverability of the vehicle with the ability to take zero radius turns [7] about the axis of the driven wheels.

III. Embedded System

The embedded architecture links the Software System to the robot's hardware by delivering the sensor's data to the system and the actuators enact the subsequent changes. The module also handles the power distribution efficiently, amalgamated with necessary safety features.

A. Sensors and Actuators

The robot, having differential drive, is mobilized by dual high torque geared motors with planetary encoders. The motors are controlled by the Roboteq MDC2230 Motor Controller. A Global Positioning System (GPS) antenna, Inertial Measurement Unit (IMU) sensor and a front view camera are mounted on the chassis to obtain raw data which is sent to the processing unit for further processing. The Table I lists the description of the sensors and actuators used along with their power requirement.

Name	Specifications	Rated Power
VectorNav VN-200	3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer, barometric pressure sensor, GPS-aided Inertial Navigation System (INS), Low power consumption, Accurate signal output owing to internal Kalman Filtering	1W
HOKUYO UTM-30LX LiDAR	Range of 30 m in 270 degree Plane of device, Millimeter resolution in a 270 arc, Accuracy 50 mm within a range of 0.1-30m	8.4W
BFLY- 23S6 Camera	On-camera image processing: color interpolation, gamma and LUT, 16 MByte frame buffer, LED status indicator	2.5W
Planetary Encoder	2 Channel Quadrature Encoder 2000 CPR	-
ACTUATOR SPECIFICATION		
Geared Motor	Operating Voltage:- 24V Current:- Max 30 A, No Load 1.12A Rated Torque:- 142 Kg-Cm Gearbox Ratio: 1:6	2X100W
Roboteq MDC2230 MotorDriver	Built-in high-power drivers for two DC motors upto 60A output per channel. Dual Quadrature Encoder inputs with 32-bit counters. Up to 6 Digital Inputs for use as Deadman Switch, LimitSwitch, Emergency stop or user inputs.	10W
Total	-	221.9W

TABLE I: Sensor Specifications

B. Power Distribution and Safety

The circuit board, designed in-house, provides all necessary operating voltages for each of Eklavya 7.0's components. Unregulated 12V flows from one of the batteries to the power board, which is then converted to regulated 3.3V, 5V and 12V voltages and sent to the sensors. Two 12V batteries in series provide unregulated 24V, which is then fed into the motor controller for powering the motors. The power board can run the overall system for about 2.5 hours on three 17Ah Pb-acid batteries. Any surge in the current is protected against by using fuses of calculated value. The circuit also has reverse polarity protection to protect the sensors from reverse polarity being applied accidentally. A wireless receiver Xbee is used for wireless emergency stop.

C. Battery Management System

An important parameter of the battery is the State-Of-Charge vs Open-Circuit-Voltage curve of the battery. It is

estimated offline in our case with certain assumptions like temperature and discharge current invariant. For the estimation of the SOC-OCV curve, the battery cells were discharged at constant current. The data points of the current (measured by a hall effect based current sensor) and voltage were saved at constant time intervals and a n-degree polynomial is fitted minimizing the Root Mean Square error (RMSE), where n is found out empirically. The SOC is then estimated using the curve and interpolation method [2]

$$V_{oc} = a_0 + a_1 S^1 + a_2 S^2 + :::: + a_n S^n$$

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n} (V_{oc}^{(i)} \quad V_{observed}^{(i)})^2}{n}}$$

IV. Software

The Software architecture of the EKLAVYA 7.0 broadly consists of the Localization, Perception and Planning modules. Pipelines have been kept parallel so that the failure of an individual module does not affect the rest of the system.

A. Localization

EKLAVYA 7.0 uses IMU, GPS and planetary encoders for localization. Their data is passed through an Extended Kalman Filter (EKF) [3], which is a non-linear version of the Kalman Filter, in two levels. In the EKF, the state transition and observation steps don't need to be linear functions and may instead be differentiable functions. This algorithm performs sensor fusion by nonlinear optimal filtering of the uncertain data, leading to an improvement of over ten fold in the location accuracy of the vehicle over the data given by a regular GPS sensor.

B. Trajectory Planning and Controls

Trajectory planning is the task of generating an optimum path from the current robot position to the waypoint, taking into consideration vehicle dynamics and obstacles. The planning module generates this optimal path using a combination of a Global planner and a Local planner. The Global planner takes into consideration the global costmap and plans a path in the global frame. Using this global plan, the local planner generates an optimal trajectory for the mobile base taking into consideration dynamic obstacles and vehicle dynamics.

The Controls module kicks in after the optimal trajectory has been planned. It generates the left and right wheel velocities and sends these commands to the low level control unit which controls the motor actuation via a closed loop feedback. This approach has been elaborated in the following sections.

1) Planner: The planning module consists of a global and a local planner.

Global Planner: Given a set of waypoints, the Global Planner interploates between consecutive waypoints to generate the global path. The sources for these waypoints are the lane detection module whenever lanes are present and the GPS navigation module for cases where lanes are absent. The global obstacle map is taken into account when generating the global path. The A* heuristic search algorithm is used for this purpose which guarantees an optimal path in a grid-based search space.

Local Planner: A local planner is necessary to replan at short intervals and generate locally traversable paths, especially in the case of dense obstacles. EKLAVYA 7.0 optimizes the local trajectory through the use of an elastic band based approach [4]. An elastic band represents a trajectory that can be freely deformed in a region free of collisions, while respecting the kinematic and holonomic constraints of the robot. We have used an implementation of the Timed-Elastic Band as the local planner which samples various trajectories to find the most optimal one [5]. The complex homotopy invariant is given by

$$H(T) = \int_T F(z)dz$$

The H-signature of a discrete path (composed of line segments) is calculated by

$$H(T) = \sum_{k=1}^{N \quad 1} H_s(Z_k; Z_{k+1})$$

The H- signature determines the homotopy class of a trajectory, this is used to eliminate all paths except one from each homotopy class to keep the sampling efficient.The presence of paths in each homotopy class prevents the planner from being stuck at a point by developing alternate feasible trajectories.

2) Controls: The high level planner discussed above is assisted by a high level controller that generates linear and angular velocity for the tracking of the optimal trajectory solutions. Over that we use a low level controller to generate the required RPM for the individual motors. We use a simple PID controller to ensure closed loop stability for the vehicle. Use of PID is justified because of the ease of implementation and the robustness achieved by the controller after proper tuning of the parameters. The feedback from the motors is obtained from wheel encoders attached at the base of the motors. As discussed in [1] the state configuration of the vehicle can be represented as:

$$X = \begin{bmatrix} x & y & ! _l & ! _r \end{bmatrix} \tag{1}$$

with x and y being world frame coordinates and theta being the orientation with respect to the world frame and $! _l$ and $! _r$ being the rotational speed of the left and right motors respectively. As discussed in [1] the relation between the configuration variables are:

$$\begin{bmatrix} V \\ - \end{bmatrix} = \begin{bmatrix} R=2 & R=2 \\ R=2b & R=2b \end{bmatrix} \begin{bmatrix} ! _l \\ ! _r \end{bmatrix} \tag{2}$$

where R is the radius of the wheel and b is the wheelbase. Using this the robot configuration in the world frame is derived as:

$$\begin{bmatrix} \underline{x} \\ \underline{y} \\ - \end{bmatrix} = \begin{bmatrix} \cos & 0 \\ \sin & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ - \end{bmatrix} \tag{3}$$

C. Perception Module

This module identifies the drivable region for EKLAVYA 7.0. It includes identifying obstacles and lanes and generating

a feasible waypoint as the goal point for the local planner. This section has been divided into four sub-sections, that presents our approach towards obstacle detection, lane detection, pot hole and ramp detection and waypoint generation respectively.

1) Obstacle Detection: The costmap represents the top view of perceivable region with each point having a cost based on its probability of being an obstacle or not. A 2D LiDAR is used to populate the costmap. Since the costmap is from a bird's perspective, masking the obstacles in the camera image requires a perspective transformation to the front view (camera feed).
This process is, however, not error free; caused due to false obstacles detection by LiDAR as well as parameter inaccuracies. Since the problem statement comprises of only a few kinds of obstacles, like orange obstacles with white stripes and blue obstacles. These were removed using standard thresholding techniques on certain combinations of colour channels and morphological transformations.

2) Lane Detection: Lane Detection has been an open research problem for a long time. The advantage that we had was that our problem statement specified that the lanes would be made on grass. So we exploited this constraint to solve this problem using simple image processing strategies and mathematical models. This module can be further divided into three sections which highlight lane extraction, lane model estimation and lane classification respectively.

Lane Extraction: As the presence of obstacles and shadows in the image induces errors in the lane extraction process, obstacle and shadow removal is done aprior [8] [9] as shown in 3(a). It was empirically found that lanes were considerably highlighted in the $2B \quad G$, B and $2B \quad R$ channels. A suitable combination of the channels was taken to create a greyscale intersection image 3(b). Due to problems like varying lighting conditions and glare, regions of high intensity emerged which were removed using a combination of adaptive thresholding [10] and median blur. To ensure uniformity in the number of inlier and outlier points for lane model estimation, central pixel suppression was carried out on 3x3 kernels 3(c).

Lane Model Estimation:
Representing the lanes mathematically serves several important purposes. It allows to embed the costmap with continuous lanes, constrain the waypoint within the lanes and also provide an orientation for the destination waypoint which is required by the planner for smoother path planning. First the lanes are checked to be horizontal straight lines since our choice of parabolas are unable to fit such lines properly. We use the Probabilistic Hough Transform algorithm with the conditions that the slope of the obtained line lies between -20 to 20 , its number of inliers is greater than a certain threshold and length of the line is greater than a pre-defined length.

When the above mentioned method fails, lanes are assumed to be parabolas with the equation, $y^2 = (x \quad c)$. Random Sampling Consensus Algorithm (RANSAC) is used to estimate and c. RANSAC algorithm [6] is based on the property that the true curve will have the maximum number of inliers (points that lie on the curve).

Lane Classification: Classifying the lanes as left or right is trivial when both the lanes are visible.

Difficulty arises in the case of ambiguous single lanes: for which we use the classification of the previous frame. The current frame classification is done on the basis of distance to the previous lanes (centroid to centroid distance).

3) Pot Hole and Ramp Detection: Pot holes and ramps cannot be detected using a 2D LiDAR, whose detection is crucial for planning and velocity profile generation. For pot hole detection, we use the fact that pot holes are circular, thus applying the following mathematical constraint:

$$\frac{P}{A} = \frac{2 r}{r^2} = 2^{-}(Constant)$$

The ramp looks like a trapezoid in the front view. For a more robust detection, the problem can be further simplified to detect 3 geometrically constrained lines, 1 horizontal and 2 near-vertical, instead of 4. This approach worked well in our experiments.

4) Way-point Generation: After lane extraction and obstacle detection, we need to generate an optimal goal point for the local planner. The goal point should be in the center of the drivable region at optimum distance from obstacles for smooth traversal and continuity in vehicle motion. An empirical cost function is defined as $_P = C_P + \frac{1}{O_P}$ where C_P is the distance from the lane center and O_P is the distance from the nearest obstacle. This cost is minimized using iterative sampling from the set of discrete feasible points. For goal point orientation, the generated optimal point is projected on the lanes and the lane model derivative is calculated at the projected point. Lane classification and average lane width are used to tackle central line estimation in the case of single lane visibility.

D. Simulation

Testing the planning and perception modules directly on a mechanical chassis is generally not advisable and can lead to physical damage. Therefore, an open-source simulation platform Gazebo was used to simulate and test these modules. A testing environment was created similar to the IGVC arena, and was interfaced with the ROS environment. The sensor noise was modelled as a Gaussian and fused with the sensor output to make the simulations more realistic. The result was the creation of a robust simulation environment where a sanity check and integration of the whole code base was tested.

V. System Integration

Considering the heavy amount of data transfer and concurrent data processing involved in robotic applications, Robot Operating System (ROS) [11] is used for peer-to-peer communication simplifying the method of data sharing between different modules. The highly efficient protocols used by ROS for data transfer through nodes and topics handles large amount of data by parallelly processing high frequency operations.