UNIVERSITY OF CALIFORNIA,
IRVINE

CAPSTONE PROJECT REPORT

MASTER OF SCIENCE

EMBEDDED AND CYBER PHYSICAL SYSTEMS

AUTONOMOUS GROUND NAVIGATION INITIATIVE(AGNI)

By

BHUMIL DEPANI

ARJUN SIVAKUMAR

RICKY BEVAN BABU

Guided By

MARCO LEVORATO

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We extend our sincere gratitude to all those who contributed to the success of the AGNI - Autonomous Ground Navigation Initiative project. The accomplishment of this endeavor required dedicated effort and the invaluable assistance of numerous individuals, each of whom played a significant role in shaping its outcome.

First and foremost, we express our deepest appreciation to Professor Marco Levorato for his unwavering support, expert guidance and mentorship throughout the entire duration of the project. His commitment to excellence and wealth of knowledge profoundly influenced our journey.

Our thanks are also extended to the Center for Embedded and Cyber-physical Systems (CECS) at the University of California, Irvine for fostering an enriching environment and providing essential resources that were integral to the project's execution. Their steadfast support has been pivotal in our pursuit of innovation and the attainment of excellence.

Special recognition goes to Tim Johnsen for his valuable suggestions, consistently offered whenever needed. Additionally, we acknowledge with appreciation the insights provided by Sharon L.G Contreras and Ian Harshbarger, which significantly contributed to refining the project.

The collective support, encouragement and expertise of these individuals and institutions have been instrumental in shaping the project. It is with great pleasure that we present our work on AGNI, a project that holds the potential to revolutionize autonomous navigation.

ABSTRACT

The motivation behind the project is to kick start the open source project which encompasses robotics areas such as localization, mapping, prediction and motion planning, and implements end-to-end pipeline for the autonomous navigation of the system consisting of the rover and the drone. By leveraging advanced robotics and autonomous systems, this project seeks to revolutionize the autonomous navigation of the system in any scenario, either city roads or any rough terrain.

AGNI - Autonomous Ground Navigation System is driven by the overarching goal to redefine the landscape of logistics and address the burgeoning demand for efficient, cost-effective and sustainable solutions across various applications. While our initial focus is on revolutionizing last-mile delivery, we recognize the potential of this system in diverse sectors such as agriculture, surveillance, security, disaster response, infrastructure inspection, etc.

The integration of a diverse sensor suite including advanced intelligent depth cameras, LiDAR, GPS, wheel encoder and IMU, combined with the computing power of the Jetson Nano and the flexibility of ROS, will position the project as a game-changer. The use of advanced sensors provides the rover with a heightened level of environmental awareness, allowing it to navigate through dynamic urban landscapes, avoid obstacles, and interact safely with pedestrians.

As our system consists of both rover and drone, we can take advantage of both worlds, rover as well as drone. Rover has limitations such as it can’t go on rough terrain, it doesn’t have overall visibility, not reliable GPS connection in the very congested urban area and it can’t climb the stairs for the last-mile delivery. On the other hand, the drone has several shortcomings such as very small flight range and battery capacity, and can't go above some limit in the restricted area near airports. So, our system is a combination of both smart systems. Whenever the rover lags in functionality, the drone will be at the rescue, and where the drone can’t go, the rover can help.

In summary, the Autonomous Ground Navigation System transcends the boundaries of autonomous systems, offering a versatile and adaptable solution for an array of applications. The amalgamation of advanced sensors, real-time decision-making and the collaborative synergy between rover and drone technologies positions this system at the forefront of autonomous navigation across diverse industries.

Fig 1: Rover autonomous outdoors

Fig 2: Waypoints generated from the goto script [1]

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Despite significant advancements in autonomous systems, automating machinery and control systems in the factories and automating many repetitive tasks using automation software, current solutions face limitations in navigating unstructured environments. In the factories and workplace, we can define the environment and using traditional software development, we can automate repetitive tasks very easily. But, for autonomous vehicles, diverse terrains, congested urban environments, and very very unstructured and constantly changing environments have tremendous limitations for their operation.

Moreover, existing standalone solutions, be it rovers or drones, exhibit limitations. Rovers face obstacles on rough terrain, lack overall visibility, and encounter challenges in congested urban settings or navigating stairs. Conversely, drones have restricted flight ranges, limitations in restricted airspace, and are unable to handle certain urban delivery complexities. Moreover, these systems lack the adaptability to seamlessly switch between modes to address each other's shortcomings.

The need for a comprehensive solution that bridges these gaps between rover and drone functionalities becomes apparent. An integrated system capable of leveraging the strengths of both technologies is crucial to address the shortcomings and complexities of last-mile delivery effectively. This system should offer adaptability, real-time decision-making capabilities, and the versatility to navigate varied terrains, ensuring efficient, safe, and timely deliveries in urban environments.

This project aims to revolutionize autonomous navigation by integrating rover and drone technologies, leveraging advanced sensor suites, computing power, and the flexibility of ROS (Robot Operating System). The challenge lies in creating an end-to-end pipeline that enables the collaborative functionality of both platforms, overcoming individual limitations, and ensuring a smooth transition between rover and drone functionalities as per the environmental demands.

The goal is to develop a versatile, adaptable, and robust Autonomous Ground Navigation System capable of navigating diverse scenarios—be it urban landscapes, rough terrains, or restricted areas—while addressing the demands of various industries, including last-mile delivery, agriculture, surveillance, disaster response, and infrastructure inspection.

1.2 Innovation

The AGNI - Autonomous Ground Navigation Initiative represents a pioneering leap in last-mile delivery solutions, integrating cutting-edge technologies to revolutionize conventional delivery practices. This initiative introduces a paradigm shift by amalgamating advanced robotics, intelligent sensor suites, and adaptable decision-making systems to address the persistent challenges in the logistics landscape.

Integrated Sensor Suite and Computing Power

One of the cornerstones of innovation lies in the integration of a diverse sensor suite, comprising intelligent depth cameras, LiDAR, GPS, wheel encoder and IMU. This fusion, coupled with the formidable computing power of the Jetson Nano and the flexibility of ROS, empowers the autonomous rover with unparalleled environmental awareness and computational capabilities. This amalgamation enables real-time data processing, facilitating precise navigation through dynamic urban landscapes and ensuring safe interaction with surroundings, distinguishing it from traditional delivery vehicles.

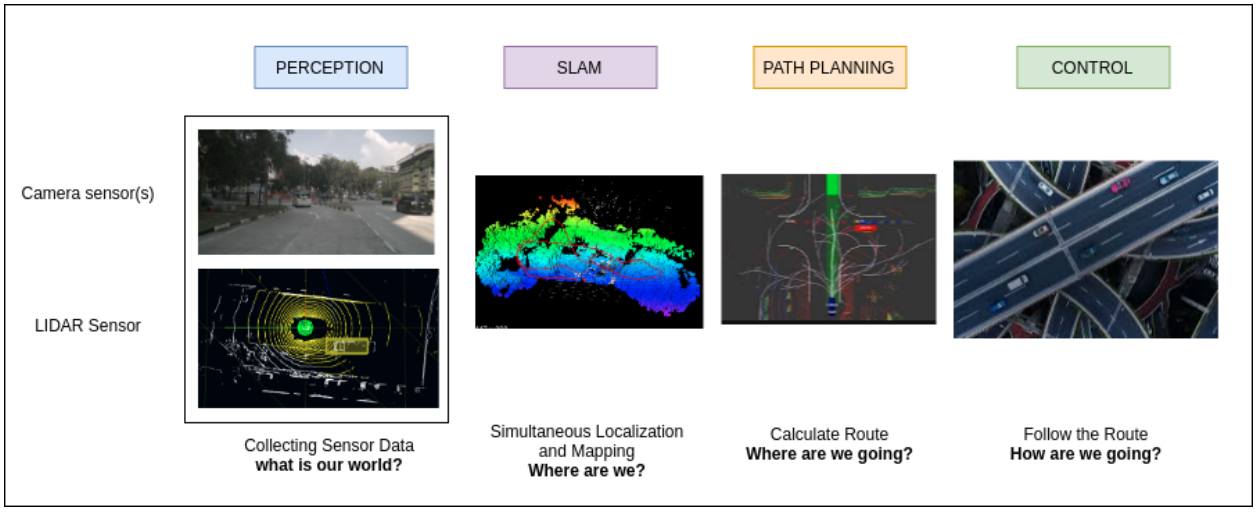


Fig 3: Techniques of Controlling the rover [2]

Adaptive Decision-Making Capabilities

The innovation extends further into the realm of adaptive decision-making. The system's ability to analyze data in real time enables dynamic route optimization, ensuring timely deliveries amidst evolving urban conditions. With the usage of advanced Machine Learning and Computer Vision algorithms, the rover is capable of traversing the unstructured and complex environment. This level of agility and adaptability surpasses traditional approaches, elevating the efficiency and accuracy of last-mile deliveries.

Synergy Between Rover and Drone Technologies

An unprecedented innovation lies in the seamless integration of rover and drone technologies. Recognizing the limitations of each individual platform, our solution leverages the strengths of both worlds. Where the rover encounters constraints in navigating challenging terrains or congested urban areas, the drone steps in, providing unparalleled accessibility. Conversely, in scenarios where the drone's capabilities are limited, the rover takes charge, ensuring continuity and reliability in deliveries. This synergistic integration showcases a comprehensive solution that maximizes the strengths of both platforms while compensating for their individual shortcomings.

In the realm of last-mile delivery, the integration of drones alongside rovers introduces a strategic approach to address specific challenges and enhance the overall efficiency of the delivery process. The unique strengths of drones complement the capabilities of rovers, providing a versatile solution to navigate the complexities of urban landscapes. Here are several key uses and advantages of incorporating drones into last-mile delivery systems:

- Precise Final Delivery: Drones are particularly well-suited for delivering items to specific locations that may be challenging for rovers to reach directly, such as upper floors and buildings or rooftops. By deploying drones for the final leg of the delivery, the system ensures precise and targeted drop-offs.
- Overcoming Terrain Limitations: Rovers may encounter difficulties navigating certain terrains or obstacles, especially in densely populated urban areas. Drones, with their ability to fly over barriers and bypass ground-level challenges, contribute to overcoming terrain limitations and reaching destinations that are not easily accessible by rovers alone.
- Battery and weight considerations: Drones face limitations in terms of battery life and payload capacity. By optimizing the delivery strategy to have the rover cover the majority of the journey and the drone handle the final stretch, the overall efficiency is maximized. The rover's extended operational time compensates for the drone's limitations.
- Time-sensitive deliveries: Drones excel in scenarios where time is of the essence such as delivering urgent medical supplies or time-sensitive packages. The swift vertical take-off and landing capability of drones significantly reduces the time required for the final delivery, ensuring timely fulfillment of critical deliveries.
- Maximizing operational hours: Rovers, designed for prolonged ground based operations, can cover longer distances and operate for extended hours compared to drones. Integrating both platforms allows for optimal use of each vehicle's strengths - rovers handle the majority of the journey, while drones are reserved for specific, challenging segments.

- Adaptability to Urban Environments: Urban environments present diverse challenges, including traffic congestion and spatial constraints. Rovers excel in ground level navigation, while drones navigate seamlessly above obstacles. This adaptability ensures efficient deliveries in complex urban landscapes.

In summary, the synergy between rovers and drones optimizes the last-mile delivery process. While rovers cover the ground efficiently and operate for extended hours, drones provide the flexibility to reach specific destinations that may be logistically challenging for ground based vehicles. This collaborative approach leverages the strengths of both platforms, offering a comprehensive and adaptable solution to the intricacies of last-mile delivery in dynamic urban environments.

The AGNI project stands as a testament to innovation in last-mile delivery solutions. By harnessing advanced robotics, sensor technology, and the synergy between rover and drone systems, this initiative redefines the possibilities within the logistics industry. Its adaptability, precision, and capacity to navigate complex urban landscapes underscore its potential as a game-changer in the domain of last-mile deliveries.

1.3 Project Overview

Now, we are clear with our problem statement. Let’s explore all the available options to implement the system and take an overview of the system.

Main work of the project is to implement and use the algorithm to process data streams coming out of the different sensors. After analyzing the data streams, the rover should get a command for the further movement.

To control the rover’s movement, the rover's motor rotation should be controlled. We are using Pixhawk Cube as a low-level flight controller. This flight controller will be running Ardupilot autopilot software. This flight controller’s output will control the motor driver. The main input to this flight controller is from a companion computer(Jetson Nano). This input is high level using Dronekit-Python API and MAVLink-MAVProxy communication protocol. Apart from main input Pixhawk also takes GPS coordinates from GPS receiver and wheel rotation information from wheel encoder. Fusing all this data, Pixhawk gives low-level output to the motor driver in the form of PWM (Pulse Width Modulation).

The project’s pretty much work has been done on the Jetson Nano. We successfully installed the Robot Operating System (ROS) on the Nano. This ROS acted as a backbone of our software stack. We used ROS nodes, topics, packages, launch files and algorithms to implement Navigation Stack. Main sensors we are integrating with Nano are Intel Realsense D455 depth

camera and Slamtec RPLiDAR A3. These both sensors stream data on the ROS topics using their respective launch files and nodes.

We have designed multiple ROS nodes to consume these data streams and data such as GPS coordinates, IMU and orientation from Pixhawk and give out high level maneuver instructions to the Pixhawk. We have implemented and used advanced algorithms such as Semantic Segmentation, Split and Merge Segmentation, Canny Edge Detection, Object Detection and Classification (for traffic sign detection), Gaussian Smoothing Filter for noise cancelation, Extended Kalman filter to process and fuse data coming out of the different sensors. To get way points to follow, we are using OpenRouteService API.

The actor will act as an end-user and will give GPS coordinates as an input to the Jetson Nano. In the below flowchart, one can take a graphical overview of the project.

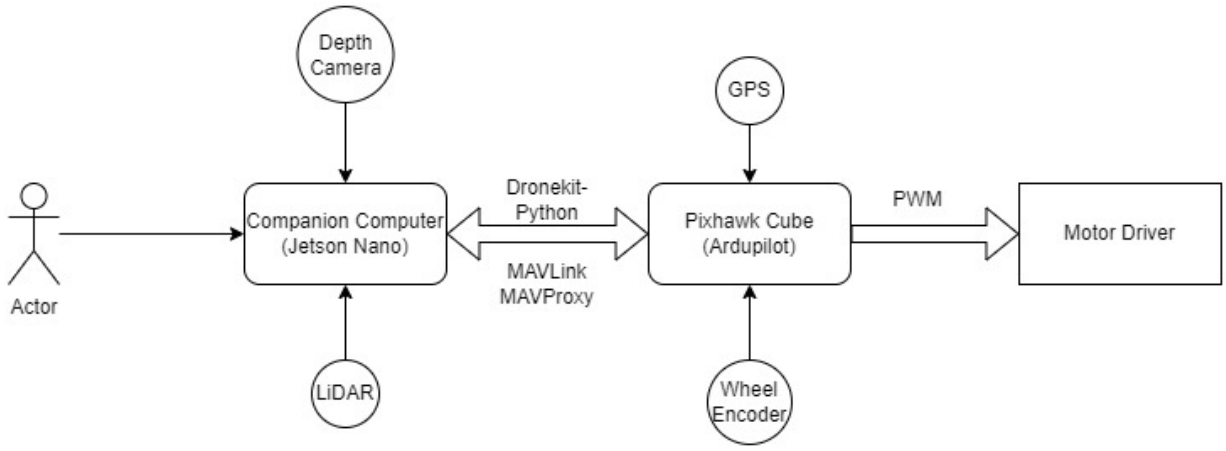


Fig 4: Basic flowchart of actions in the controllers and motors of the rover

1.4 Objective

The primary objective of AGNI - Autonomous Ground Navigation Initiative is to develop and implement a groundbreaking last-mile delivery system that overcomes the challenges inherent in traditional logistics approaches. The project aims to achieve the following key objectives:

- Integrate Cutting-edge Technologies: Implement state-of-the-art algorithms to process data streams from diverse sensors, including Depth cameras, LiDAR, GPS, and IMU, providing the rover with a comprehensive understanding of its environment.
- Real-Time Decision Making: Enable the rover to perform real-time analysis of incoming sensor data and autonomously generate precise navigation commands. This involves the development of sophisticated algorithms for obstacle detection, path planning and trajectory execution.
- Seamless control integration: Utilize pixhawk cube flight controller running Ardupilot software to seamlessly integrate high-level commands from the companion computer(Jetson Nano) with low-level motor control. Ensure precise and responsive control over the rover's movement.
- Enhanced adaptability: Enhance the rover's adaptability to diverse terrains and urban environments, addressing challenges such as traffic congestion and pedestrian interactions. Implement intelligent navigation algorithms to navigate through complex scenarios effectively.
- Efficient Last-Mile Delivery: Revolutionize last-mile delivery practices by creating a system that optimizes delivery routes, reduces delays and minimizes environmental impact. The objective is to achieve efficient, sustainable and timely deliveries in urban landscapes.
- Integration of Rover and Drone functionalities: Bridge the gaps between rover and drone functionalities by creating a hybrid system that combines the strengths of both technologies. Ensure that the integrated system can handle diverse delivery scenarios, including congested urban settings and varying terrains.

By achieving these objectives, the AGNI aims to set new standards in last-mile delivery solutions, providing a versatile adaptable and efficient system that addresses the complexities of urban logistics and revolutionizes the autonomous navigation landscape.

CHAPTER 2

AGNI HARDWARE

2.1 Rover chassis

The rover chassis serves as the foundational structure for our autonomous ground navigation system, offering robustness and adaptability across diverse terrains. Constructed from lightweight yet durable materials, the chassis features a modular design that facilitates easy customization and integration of various components.

We are using the R1 Autonomous Robot chassis from Aion Robotics. The R1 rover is an autonomous rover designed for educational and research tasks. The R1 comes in two flavors, one based around the PixHawk autopilot system and another version based around the NVidia Jetson platform running the popular Robot Operating System (ROS). We are using the first one. This rover chassis offers skid steering. The rover can be operated remotely via RC system or from the ground control software available for the platform. The ground control software can be used for planning and executing autonomous missions and is available for desktops and laptops as well as iOS and Android.

Fig 5: Aion Robotics Rover[3]

The rover chassis forms the robust backbone of our autonomous ground navigation system, providing a versatile and adaptable platform that facilitates the integration of advanced technologies, enabling effective navigation and functionality across diverse environments.

2.2 Pixhawk Cube (Flight Controller)

Pixhawk Cube flight controller is a powerful and versatile autopilot hardware system designed for unmanned aerial vehicles and other robotic applications. This controller is known for its reliability, flexibility and extensive range of features. As the core component of an autonomous system, the flight controller is responsible for managing low level sensors, processing data and controlling the vehicle's actuators to ensure stable and controlled flight.

The Cube flight controller is equipped with a range of sensors including accelerometers, horoscopes, magnetometers, barometers, IMU, enabling it to gather comprehensive data about the vehicle's position, orientation and environmental conditions.[4]

Fig 6: Pixhawk Cube

In our rover system, the pixhawk cube flight controller serves as a central hub for autonomous control, seamlessly integrating with our hardware and software stack to facilitate smooth and precise navigation. The pixhawk cube is a pivotal component, receiving essential inputs from the Jetson Nano Companion Computer, which houses an array of sensors responsible for environmental perception and data collection. We are using Ardupilot autopilot software on the Pixhawk.

The Jetson Nano, housing the range of sensors and perception algorithms, serves as the cognitive brain of the rover. It processes data from depth cameras and LiDAR to gain a comprehensive understanding of the environment. The high-level commands generated by the Jetson Nano encapsulate sophisticated decision-making processes including obstacle avoidance, path planning and environmental mapping. These high level commands represent the rover's intended actions and responses to the perceived surroundings. The pixhawk cube acts as the

executor, translating these high-level commands into low-level control signals that directly influence the motor drivers.

The seamless integration allows for a division of labor where the Jetson Nano focuses on perception, mapping and decision-making while the pixhawk cube specializes in the precise execution of those decisions through control of the rover's motors. With a robust set of features and real-time processing capabilities, the flight controller ensures that the rover responds swiftly and accurately to the high-level commands provided by the Jetson Nano.

This design philosophy promotes ease of use, enabling users to concentrate on defining mission objectives and high-level behavior rather than getting bogged down in low level controls of motors. The Nano's perception capabilities and decision-making processes are complemented by the Cube's precision in translating those decisions into actions. This integrated approach results in a robust and user-friendly autonomous rover system capable of navigating complex environments with agility and intelligence.

Overall, the integration of the Cube in the rover system streamlines the autonomous navigation process, providing a reliable and adaptable solution for users. Its compatibility with the Nano, abstraction of complex control tasks and generation of low-level motor control signals contribute to the overall efficiency and user-friendly nature of our autonomous rover platform.

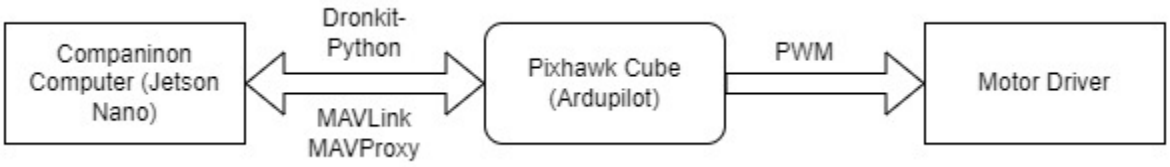


Fig 7: Simple flowchart showing communication between the controllers

2.3 Motor/Motor Driver

The motors within the rover system are pivotal components that drive its movement and locomotion across diverse terrains. These motors are specifically selected for their efficiency, torque output, and compatibility with the rover's design and intended functionalities. Motor requires very current to drive the rover and the payload.[5]

The motors within the rover system are pivotal components that drive its movement and locomotion across diverse terrains. These motors are specifically selected for their efficiency, torque output, and compatibility with the rover's design and intended functionalities. The motor driver takes signal and current(power) as an input and drives the motors. The R1 Autonomous Robot chassis from Aion Robotics comes with a RoboClaw motor controller.

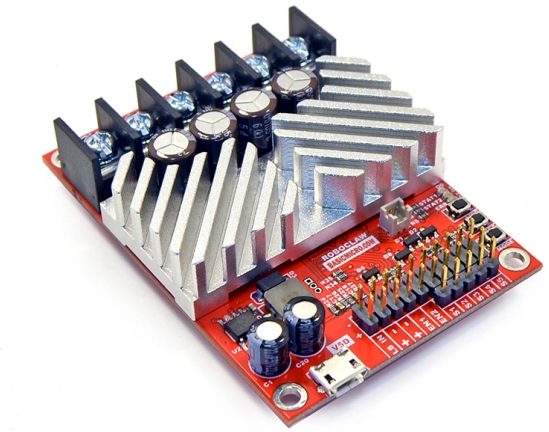


Fig 8: Motor Driver

2.4 Wheel Encoder

Wheel encoders play a pivotal role in autonomous navigation, providing a robust solution for maintaining accurate localization and executing precise movements, especially in scenarios where GPS signals are unreliable or unavailable.

In situations where GPS signals are compromised, such as during cloudy weather or signal interference, relying solely on GPS for localization becomes impractical. Wheel encoders offer a crucial alternative, providing redundancy to ensure continuous and accurate localization of the rover.

Wheel encoders enable the implementation of dead reckoning techniques, allowing the rover to estimate its position based on the rotation of the wheels. This ensures that the system can maintain accurate localization relative to a known starting point, mitigating the impact of temporary GPS unavailability.

Trajectory planning: when receiving trajectory instructions from an API, the integration of wheel encoder data becomes paramount. The rover leverages this information to precisely follow instructions such as “go straight for 200 meters” or “ take a right and go for 1 mile” ensuring accurate execution of navigation commands.

The wheel encoder data seamlessly integrates with our navigation algorithms, offering a complementary source of information to GPS data. This integration enhances the rover’s adaptability to dynamic environments and contributes to a more accurate and reliable

navigation system. It provides a reliable solution for navigating through challenging weather conditions. In these scenarios, where the GPS signals may be attenuated, the rover maintains its trajectory and accurately follows navigation instructions.

2.5 Joystick

The joystick interface serves as the user input device enabling intuitive control and navigation of the rover system, facilitating direct human interaction with the autonomous ground navigation system.

The joystick interface serves as the bridge between the user and the autonomous ground navigation system, offering intuitive, responsive, and user-friendly control over the rover's movements. Its ergonomic design and compatibility with the rover's control systems ensure efficient and effective user interaction within the autonomous navigation framework.

Fig 9: Joystick and rover

2.6 GPS Receiver

A GPS receiver for an autonomous navigation robot is a crucial component that enables the robot to determine its precise position on Earth. Here's an overview of its functionalities in the autonomous navigation:

- Localization: The GPS receiver provides accurate positioning data, allowing the autonomous system to know its exact location on Earth.
- Navigation: With precise location information from the GPS receiver, the autonomous system can plan routes, navigate to specific waypoints, and determine the best paths to reach destinations efficiently.
- Global Coverage: GPS provides global coverage, allowing autonomous systems to operate virtually anywhere on Earth, making it highly versatile for various applications, from urban settings to remote areas.
- Real-Time Updates: The receiver continuously receives signals from satellites, enabling real-time updates of the robot's position.

The R1 Autonomous Robot chassis from Aion Robotics comes with *Here2* GPS receiver. The Here2 GPS receiver is a high-precision positioning module designed for use in autonomous vehicles, drones, and robotics. It's known for its accuracy and reliability in providing global positioning data. The Here2 utilizes signals from multiple satellite constellations (like GPS, GLONASS, BeiDou, and Galileo) to deliver precise location information. With its capability for Real-Time Kinematic (RTK) positioning and integration with other sensors, the Here2 offers centimeter-level accuracy, making it ideal for applications that demand extremely precise navigation and positioning capabilities.[6]

Fig 10: Here2 GPS Receiver

2.7 Jetson Nano (Companion Computer)

We already have a flight controller, Pixhawk, why do we want another processor on-board? In our system Jetson Nano plays the role of Companion Computer. So, what is this companion controller and why do we require it if we already have Pixwawk? A companion computer is a device that travels on-board the vehicle and controls/communicates with the Flight controller over a low-latency link such as MAVLink and MAVProxy. Applications running on a companion computer can perform computationally intensive or time-sensitive tasks, and add much greater intelligence than is provided by the Flight Controller alone.[7]

Why is Jetson Nano only as a companion controller? We can summarize our decision by comparing contemporary available processor/controller boards.

	Jetson Nano	Raspberry Pi 4B	BeagleBone Black
Performance and Processing Power	Equipped with a quad-core ARM Cortex-A57 CPU and an NVIDIA Maxwell GPU with 128 CUDA cores, optimized for AI and machine learning applications	Features a quad-core ARM Cortex-A72 CPU, suitable for general-purpose computing and IoT applications	Includes a 1GHz ARM Cortex-A8 processor, which is less powerful compared to the CPUs in the Jetson Nano and Raspberry Pi 4, making it more suitable for simpler tasks
Graphics and AI Capabilities	Offers dedicated AI processing power with an NVIDIA Maxwell GPU and CUDA cores, ideal for machine learning and computer vision applications	Provides basic GPU capabilities but lacks specialized hardware for AI, making it less suitable for AI-focused projects compared to the Jetson Nano	Does not have dedicated AI or GPU capabilities like the Jetson Nano or Raspberry Pi 4
Memory and Storage	Comes with 4GB LPDDR4 RAM	Available in models with 2GB, 4GB, or 8GB LPDDR4 RAM options	Offers 512MB of DDR3 RAM
Connectivity and	Provides USB 3.0	Offers USB 3.0 ports,	Includes USB ports,

Ports	ports, HDMI, GPIO headers, and a CSI camera interface	dual HDMI ports, GPIO pins, a CSI camera port, and options for additional accessories via various expansion boards	HDMI, Ethernet, and expansion headers for GPIO and other peripherals

Table 1 : Characteristic of different processors

The Jetson Nano is a powerful, small-sized single-board computer designed by NVIDIA primarily for AI and robotics projects. It's equipped with a quad-core ARM Cortex-A57 CPU and an NVIDIA Maxwell GPU with 128 CUDA cores, providing significant computational power for AI applications.

With a compact form factor and low power consumption, the Jetson Nano supports various neural network models and frameworks like TensorFlow, PyTorch, and Caffe, making it ideal for deep learning tasks, computer vision, and autonomous machines. It features 4GB of LPDDR4 RAM, multiple USB ports, HDMI, GPIO headers, and a CSI camera interface, offering versatility for diverse projects. Its compatibility with popular peripherals and software tools makes it a favored choice among developers and hobbyists for prototyping and deploying AI-based applications in edge computing scenarios.

Fig 11: Jetson Nano

2.8 Slamtec RPLiDAR A3

LiDAR (Light Detection and Ranging) sensors are pivotal components in various industries, primarily known for their precise distance and mapping capabilities. Operating on a principle of emitting laser pulses and measuring their reflections to create detailed 1D, 2D or 3D maps in the sensor's field of view, LiDARs have numerous applications in robotics, autonomous vehicles, surveying, and more.

In the context of robotics and autonomous systems, LiDAR sensors provide crucial environmental data essential for navigation, obstacle detection, and mapping. Obstacle avoidance is a fundamental aspect of autonomous navigation systems, ensuring the safety and efficient movement of vehicles in dynamic environments. These sensors enable robots to perceive their surroundings with accuracy, enabling them to make informed decisions and navigate safely in complex environments.

Choosing the most suitable LiDAR sensor involves evaluating the specific requirements of a project or application against parameters like range, resolution, scanning frequency, and cost-effectiveness.

The Slamtec RPLiDAR A3 is a popular LiDAR sensor known for its reliability, affordability, and applicability in various robotics, mapping, and navigation applications. Comparison with Other LiDAR Sensors:

- Velodyne LiDAR: Compared to higher-end Velodyne LiDAR models, the RPLiDAR A3 may have a shorter range and lower resolution but offers a more cost-effective solution for applications with moderate range and accuracy requirements. As our rover’s speed is slow, to choose cost efficient options RPLiDAR A3 is better compared to Velodyne LiDAR.
- Hokuyo and SICK LiDAR: While lacking the range and scanning frequency of some Hokuyo and SICK LiDAR sensors, the RPLIDAR A3 provides a broader field of view and competitive performance at a more affordable price point.
- Ouster and Quanergy LiDAR: These LiDAR sensors often offer higher ranges and resolutions than the RPLIDAR A3 but at a higher cost, catering to applications with more demanding precision needs.

Here's a summary of Slamtec RPLiDAR A3 specification:

- Range and Accuracy: The RPLiDAR A3 typically offers a scanning range of up to 25 meters with high accuracy in distance measurement.
- Resolution: It provides a resolution of up to 0.2 degrees, generating detailed point clouds for mapping and localization.

- Scanning Frequency: Capable of scanning at a frequency of up to 20 Hz, capturing data points rapidly.
- Field of View (FoV): The sensor usually provides a wide field of view, covering 360 degrees for comprehensive environmental perception.
- Interface and Compatibility: Offers interfaces like USB and UART, compatible with various platforms and operating systems. We are using USB as it is the simplest to use with Jetson Nano.
    - Price: Positioned competitively in terms of cost compared to other LiDAR sensors, making it an attractive choice for budget-conscious projects.[8]

Fig 11: Slamtec RPLiDAR A3

2.9 Intel RealSense D455 Depth Camera

The camera in our system serves as a versatile sensory tool, contributing to various aspects of perception and decision-making. Integrated seamlessly into the Robot Operating System (ROS), the camera's raw feed becomes a foundational element for multiple algorithms each designed to fulfill specific functionalities critical to the rover's autonomy.

Fig 12 : Intel RealSense D455 Depth Camera

There are different kind of cameras available for the robotics application, such as

- Color Camera: A color camera, also known as an RGB camera, captures images in full color using red, green, and blue sensors to reproduce a wide spectrum of colors.
- Tracking Camera: When we talk about our tracking camera on the other hand, we are talking about a camera that can track its own position and movement in space. Tracking cameras like the T265 are used to track anything they are attached to with a high degree of precision. They don't provide the depth information.
- Depth Camera: A depth camera, often utilizing technologies like time-of-flight (ToF), structured light, or stereo vision, measures distances to objects in a scene, providing depth information in addition to color. In addition to RGB data, the depth camera provides depth information for each pixel. It captures depth information, creating a 3D representation of the environment alongside color data, enabling accurate distance measurements and object segmentation. As we want our rover to traverse on the walkway, to detect the curb using depth information, as the walkway would be a little bit higher than the road.

Technical specification of Intel RealSense D435 camera:

- The Intel RealSense D435 camera is based on the Intel RealSense D430 module with an attached, but separate, RGB camera. The depth algorithm is based on precise placements of sensors, so not having them fixed on the same stiffener may make calibration between depth and RGB problematic. However, users of previous-generation cameras said they really wanted the flexibility to choose their own RGB sensors, and so the D435 was born, and while it is more challenging to precisely calibrate color and depth data together using this camera, it is still possible.
- The Intel RealSense D435 is a compact and versatile depth camera renowned for its advanced depth perception capabilities. Featuring active infrared stereo depth technology, it combines depth sensing with a high-resolution RGB camera, delivering impressive visual and spatial data. With a depth stream output of 1280x720 at up to 90 frames per second and an RGB resolution of 1920x1080 at 30 FPS, it offers detailed depth perception and vivid color imaging. [9] The D435's wide field of view, precise depth range from 0.2m to 10m, and compact form factor make it ideal for robotics, 3D scanning, AR/VR, and industrial automation applications. Its compatibility with various operating systems and support for the Intel RealSense SDK 2.0 make it a versatile tool for diverse depth-based projects.

CHAPTER 3

AGNI SOFTWARE STACK

3.1 Jetpack/Ubuntu Operating System

JetPack is a comprehensive software development package provided by NVIDIA, specifically tailored for their Jetson family of AI-focused single-board computers, running on an Ubuntu-based Linux distribution. This software bundle encompasses a range of tools, libraries, and frameworks curated to optimize AI and deep learning applications on Jetson devices within an Ubuntu environment.

As an integral part of the Jetson ecosystem, JetPack includes CUDA for parallel computing, cuDNN for deep neural networks, TensorRT for optimizing inference, VisionWorks for computer vision applications, and various APIs. These components, combined with Ubuntu's robust and developer-friendly environment, streamline the development process for AI applications on Jetson devices.

Moreover, JetPack offers wrapper tools and utilities that aid in simplifying tasks such as software installation, system setup, and managing device configurations. These wrappers facilitate smoother integration and utilization of the hardware capabilities of Jetson devices, empowering developers to efficiently deploy AI models, computer vision algorithms, and robotics applications on the Ubuntu-based Jetson platform.

Command to know list of connected devices and their port: python -m serial.tools.miniterm

3.2 Ardupilot

Ardupilot is an open-source autopilot software suite specifically designed for unmanned vehicles, including drones, rovers, helicopters, and more. It operates on a variety of hardware platforms such as Pixhawk, Cube Black/Orange, Sparky. Ardupilot provides a comprehensive set of tools and firmware that enable autonomous and semi-autonomous control of vehicles. Ardupilot leverages an Ubuntu-based environment on compatible hardware, offering a robust and flexible platform for developers and enthusiasts to create, customize, and deploy autonomous vehicle systems.

Ardupilot's software suite includes

- Flight control firmware
- Ground control station software (such as Mission Planner and QGroundControl)
- Libraries for navigation, control, and mission planning

Key components and functionalities of flight control firmware include:

- Sensor Integration: Interprets data from various onboard sensors such as gyroscopes, accelerometers, magnetometers, and GPS receivers to determine the vehicle's orientation, velocity, altitude, and environmental conditions.
- Control Algorithms: Implements control algorithms to stabilize the vehicle, maintain its orientation, and regulate its movement in response to pilot commands or autonomously generated commands from the flight path or mission.
- Autonomous Functions: Executes autonomous functions, including waypoint navigation, return-to-home procedures, obstacle avoidance (if equipped), and various predefined missions or tasks.
- Communication Protocols: Establishes communication protocols with ground control stations or remote control units to receive commands, transmit telemetry data, and ensure real-time monitoring and control.
- Safety Features: Incorporates fail-safe mechanisms, such as emergency landing protocols or return-to-home procedures triggered by loss of communication or critical system malfunctions, to ensure safe operation.
- Customization and Configuration: Allows users to configure and tune various parameters, such as PID controllers for stabilization or mission-specific settings, to optimize performance for different applications or environments.[10]

Ardupilot simplifies the complexities of vehicle control and navigation, enabling users to focus on higher-level applications and customization within an Ubuntu environment, making it a favored choice for autonomous vehicle enthusiasts and developers.

3.3 Mission Planner

Mission Planner is a robust ground control station software primarily used with ArduPilot-powered vehicles like drones, rovers, and other autonomous systems. It operates within the Windows environment and offers a comprehensive set of tools for configuring, planning, and monitoring autonomous missions.

Developed as an open-source application, Mission Planner provides an intuitive interface for users to manage various aspects of their vehicles. It allows for easy configuration of vehicle parameters, firmware updates, and calibration of sensors, ensuring optimal performance.

Mission Planner facilitates mission planning through features like waypoint creation, geofencing, and automated mission execution. Users can visualize their planned missions on maps and define complex routes or tasks for their vehicles to follow.

Additionally, the software offers real-time telemetry and monitoring capabilities, displaying critical data such as vehicle status, sensor readings, GPS information, and battery levels. This information assists operators in making informed decisions during missions.

Overall, Mission Planner serves as a central hub for users to interact with and control their ArduPilot-based vehicles, providing a user-friendly interface within the Windows environment for configuration, mission planning, and real-time monitoring.

Fig 13: Mission Planner [11]

3.4 OpenRouteService API

- OpenRouteService is a multifaceted and versatile online routing service that harnesses the extensive and continually updated data from OpenStreetMap (OSM). Developed by the Heidelberg Institute for Geoinformation Technology (HeiGIT), it's an open-source platform designed to provide a range of geospatial solutions, primarily focused on routing and accessibility.
- The key strength of OpenServiceRoute lies in its ability to offer various types of routing services that cater to different modes of transportation, including driving, cycling,

walking. It goes beyond simple point-to-point navigation, integrating advanced features like route optimization, isochrones for accessibility analysis, and matrix routing for complex route planning.

There are many reasons why we used OpenServiceRoute over other APIs.

1. We tried a lot of APIs calls such as OSM(OpenStreetmap), Mapbox navigation, OSM Tile, QGIS etc. All these Api calls were either expensive or did not give us the right accuracy that we needed. Using mapbox navigation api call in our navigation script it was able to give the drive route of navigation waypoints correct but the accuracy was bad when the pedestrian/cycle path was along the path.

2. Hence we found OpenServiceRoute to be the best API call to trigger our script to trigger the waypoints through GPS. As it was free to use and gave us unlimited API calls during our testing purposes and also produced the right waypoints for both drive and cycle/pedestrian paths , making this the best API call for our project.

OpenServiceRoute Api has a url for each of the driving modes such as drive,cycle etc. It also includes geocoding (turning addresses into coordinates) and reverse geocoding services, making it a comprehensive tool for both location-based services and spatial analysis.We generate waypoints for each 5 meters from the start to end point and also generate list of waypoints as a array for the same. The folium map is a feature which generates the map which reflects the path traced by the start and end points.[12]

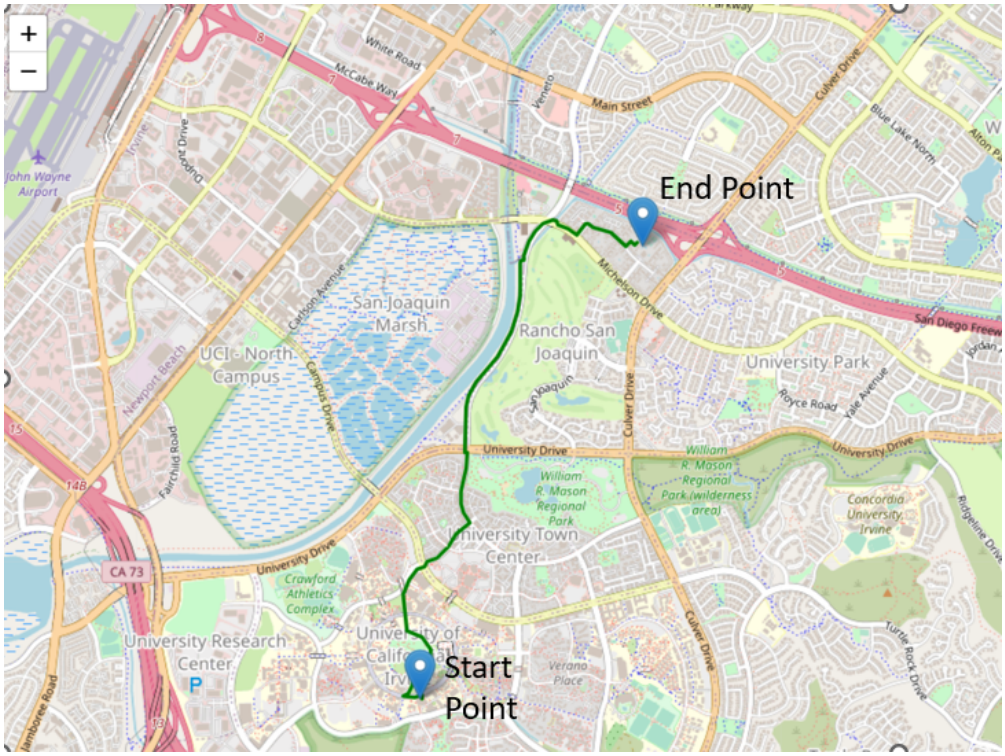


Fig 14 : Folium Map of the waypoints generated by the API

The below image is a map generated by folium through openstreetmap data indicating the path traced from the start to end point.

3.5 ROS

- Why ROS? -
  - Without ROS - Monolithic and sequential;
  - Challenges in robotics software stack development - interconnected algorithms, Concurrency
  - With ROS - Modular and parallel distributed computing
- What is ROS?
  - It’s not an OS. But is is a software framework to help developing the software stack for the robotics applications
  - It is set of libraries and tools

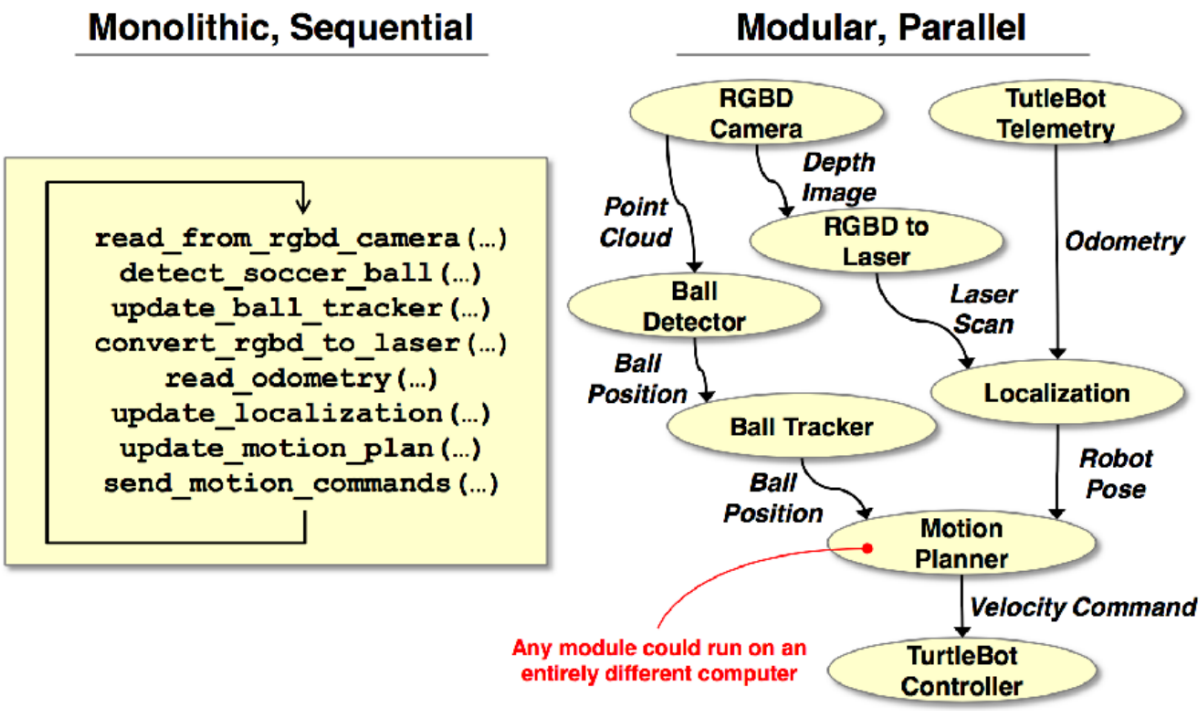


Fig 15: The process inside the ROS nodes

- ROS in detail
  - ROS Masters - roscore
  - ROS Nodes - Create graphical view of nodes, topics, messages and packages
  - ROS Topics - Subscriber Node, Publisher Node
  - ROS Messages
  - ROS Packages
  - ROS Launch files
  - Gazebo
  - Rviz
- Steps to create work space and packages:
- Step 1: create the folder
- >> mkdir -p ~/catkin_ws/src
- To make a new directory for our workspace called caktin_ws. The dash -p means create all parent directories if needed (that is if catkin_ws is not there, create it as well).
- You can create a workspace folder anywhere except inside another workspace folder
- Step 2: Use ROS scripts to initialize this folder to be a ROS workspace
- >> cd ~/catkin_ws/src

- >> catkin_init_workspace
- Now when you use “ls” you will see a new file there called CMakeLists.txt. This is the “top-level” cmake file for the whole workspace. Every time you create a new package, this file is going to change automatically. Usually, no need to edit this file.
- Now let’s create the rest of the folders. First, we go to the root of the workspace and use catkin_make
- >> cd ..
- >> catkin_make
- Now you will see that two more folders are going to appear namely devel and build. Let’s go
- inside the devel folder
- >> cd ./devel
- You will see several files. One important one is the setup.bash and setup.sh which have several shell scripts that makes sure that ROS understand that this folder contains ROS packages. This makes sure that commands like roscd and rosls are aware of this folder and treat it exactly as if like all other pre-installed ROS packages.
- You can just execute this bash file here
- >> source ~/catkin_ws/devel/setup.bash
- Or you can add it to the bashrc which is a set of shell scripts that are executed everytime you open a new shell:
- >> echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
- >> source ~/.bashrc
- You can see the effect by opening the bashrc file (check the last line there)
- >>gedit ~/.bashrcCreate Package Folder:
- Now it is time to create your first package folder. First, go to the src folder and use the catkin create package command. Remember that this command takes the package name followed by all the dependencies that are needed for this package:
- >> cd ~/catkin_ws/src
- >> catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
- Now since we made a change to the workspace, let’s re-compile the whole workspace again. So we go up to the root of the workspace and then use the catkin make command:
- >> cd ~/catkin_ws
- >> catkin_make
- Create a new folder named “scripts” in the

- Transfer all the python scripts from the cloned repository to the newly created scripts folder.
- >> cd ~/catkin_ws
- >> catkin_make

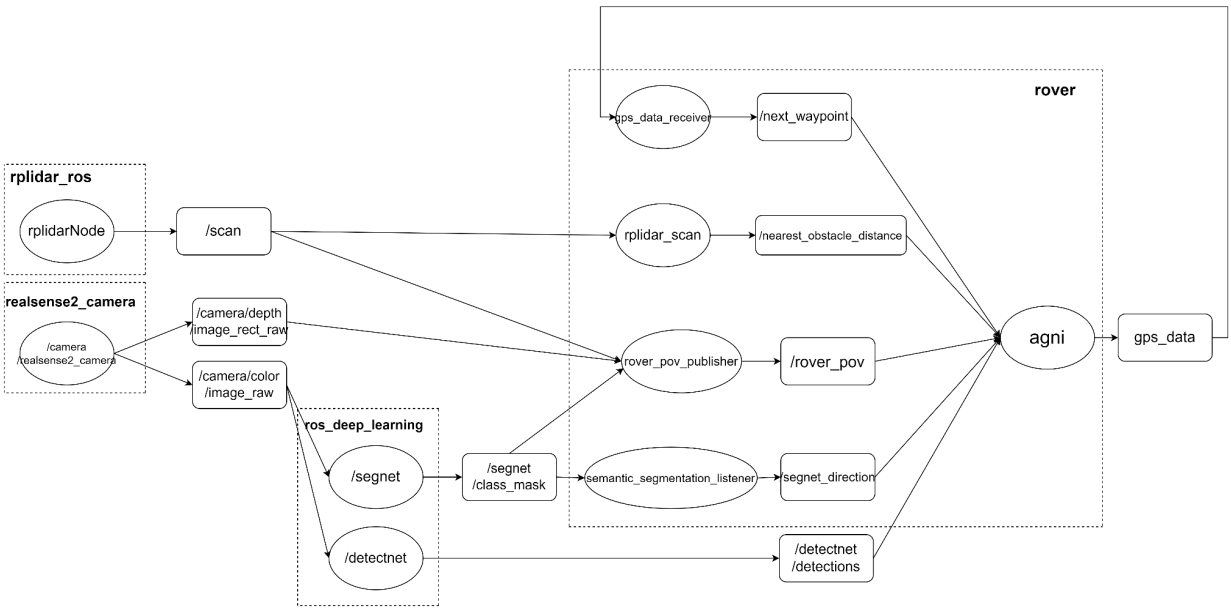


Fig 16 : This flowchart shows the ROS Nodes interaction in the rover

3.6 DroneKit - Python

- DroneKit-Python: It is an API or tool using which developers can develop applications, which runs on Companion Computer, applications can communicate with Ardupilot(Pixhawk). DroneKit uses the Python script to send command and collect data from Ardupilot.
  - It is an API or tool using which developers can develop applications, which runs on Companion Computer, applications can communicate with Ardupilot(Pixhawk).
  - DroneKit uses the Python script to send commands and collect data from Ardupilot.
  - Using DroneKit,a developer can build applications in Python.

- DroneKit-Python is an API, which helps in developing apps in Python.
- This API allows developers to create applications. These applications will run on Companion Computer (Raspberry Pi or Jetson Nano) and communicate with Flight Controller (Pixhawk) using low latency MAVLink protocol.
- This API provides programmatic access to
    - Vehicle's telemetry
    - Vehicle's state
    - Vehicle's parameter information
- And control:
    - Mission Management
    - Direct control of the vehicle
- API features:
    - Connect to a vehicle
    - Get and Set telemetry and states
    - Receive Asynchronous notifications
    - Guided Mode
    - Auto mode

Jetson Nano can read GPS coordinate data from the GPS receiver sensor through Dronekit - Python.

3.7 Mavros

- MAVROS is a popular open-source communication bridge used in the robotics community, specifically designed to link the MAVLink protocol (used by ArduPilot and PX4) with the Robot Operating System (ROS) environment. Operating within the ROS framework, MAVROS facilitates seamless communication between ROS-based systems and MAVLink-enabled autopilots.
- As a ROS package, MAVROS acts as a middleware, providing a set of ROS nodes that enable interaction and control of MAVLink-enabled vehicles. It offers functionalities such as interfacing with autopilots, handling telemetry data, sending control commands, and accessing various vehicle parameters and sensor readings.
- MAVROS simplifies the integration of MAVLink-based vehicles into ROS ecosystems, allowing ROS users to leverage the capabilities of MAVLink-compatible autopilots within their robotics projects. It provides

ROS-based tools for mission planning, visualization, and control, enhancing the autonomy and capabilities of robotic systems.[13]

- Moreover, MAVROS facilitates the development of complex robotic applications by enabling seamless communication between ROS nodes and MAVLink-enabled vehicles, making it an essential bridge for utilizing ROS functionalities in conjunction with MAVLink-based autopilots for advanced robotics applications.

3.8 Lidar

- ROS RPLiDAR: http://wiki.ros.org/rplidar
- Object avoidance
- For forward motion: divided the forward scene in left, middle and right segments.
- For backward motion: using back cloud points
- We are pushing LiDAR data on "XYZ" topic. And Consuming that data using RPLiDAR_Scan.py script
- Lidar_advanced_goto.py
- 2d map creation for local navigation
  Triangulation for finding an angle

3.9 Camera

- Why is vision powerful?
  - extremely rich source of information
  - Allows us to extract information beyond geometry (e.g., semantic information)
  - Computer vision seems easy (since we as humans are so good at it), but is in fact extremely challenging
  - As an exercise, think about what makes a cat a cat.
- What is an image?
  - digital representation of visual information
  - Image is an array of pixels
- Semantic segmentation
  Semantic segmentation plays a pivotal role in the autonomous outdoor ground navigation system, specifically in differentiating between various surfaces such as roads, vegetation, and other non-traversable paths. Efficient semantic segmentation relies on high-quality, annotated datasets.

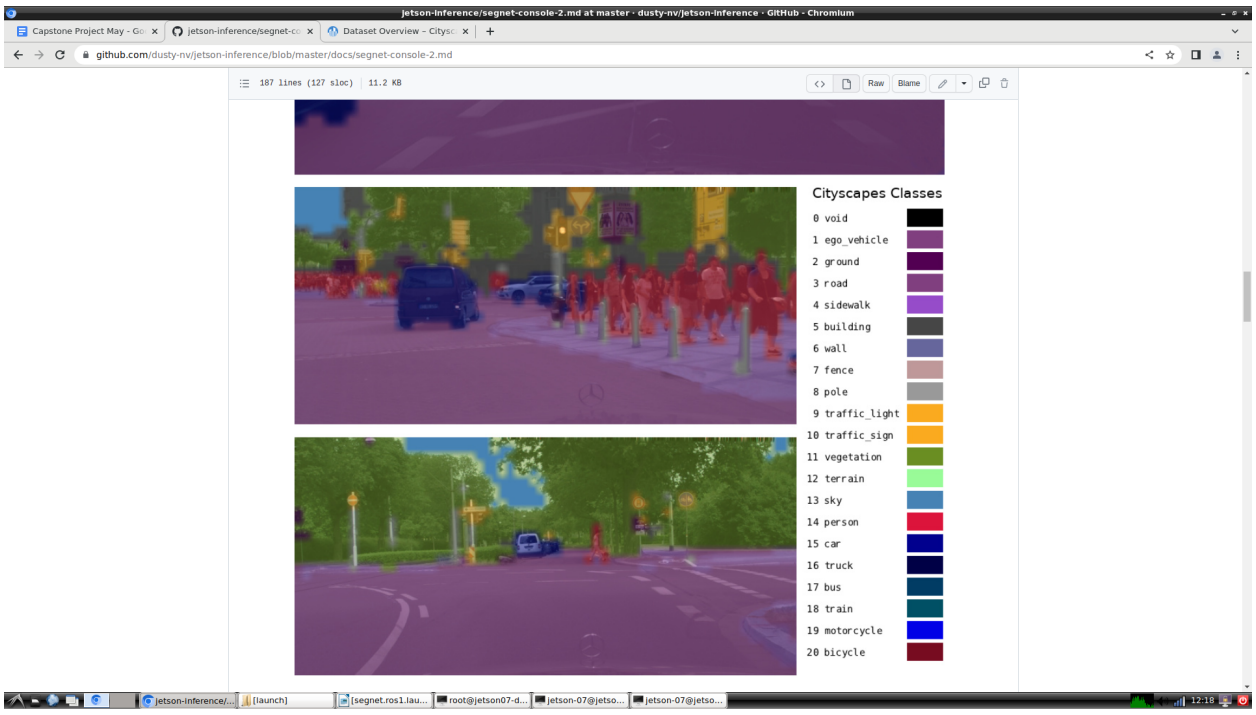
Semantic segmentation models leverage the camera feed to classify and differentiate various objects and surfaces in the rover's environment. This includes identifying roads, vegetation, enhancing the rover's understanding of its surroundings.

Depth data extracted from the camera feed plays a crucial role in curb detection. By analyzing the depth information, the rover can accurately identify and navigate around curbs, ensuring safe traversal through urban or structured environments.

The camera also contributes to sensor fusion, working in tandem with the LiDar data to construct a comprehensive 3d map of the rover's surroundings. This fused data enables a more holistic perception of the environment, improving obstacle avoidance and path planning.

For lane detection, the camera supports Canny edge detection algorithms. By identifying edges in the camera feed, the rover can determine the boundaries of lanes on roads, aiding in autonomous navigation and ensuring it stays within the designated path.[14]

- Different options for the semantic segmentations.
    - City scapes



    - Deep scene

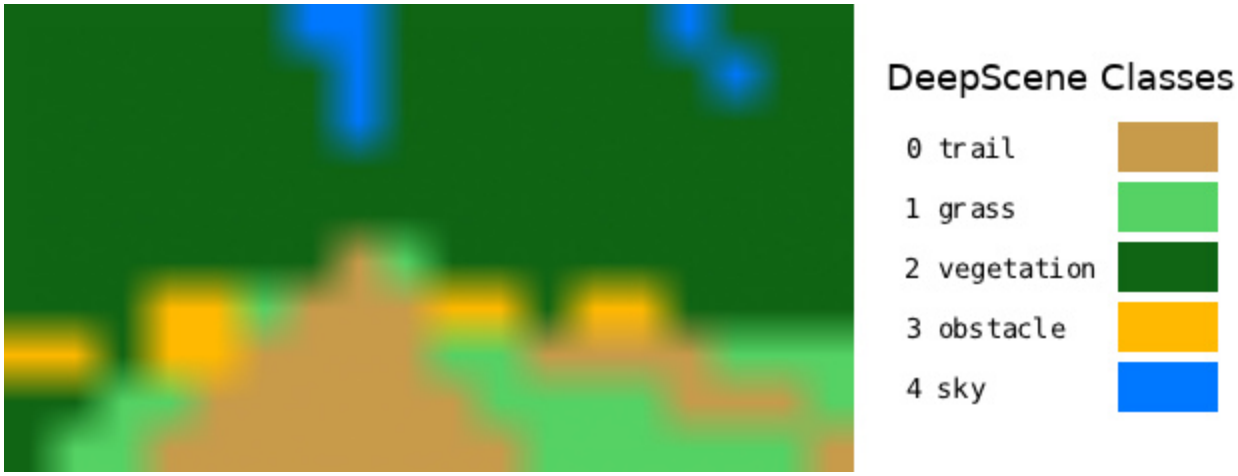


Fig 16 : List of Semantic Segmentation labels

Write about our dataset

Deeplearning models, particularly convolution Neural Networks are commonly used for such tasks. These models are trained to learn intricate patterns and features in the input images, enabling them to accurately classify each pixel into specific semantic classes.

Accurate road segmentation is critical for safe navigation. Identifying and segmenting vegetation is essential for avoiding collisions and navigating through areas with dense greenery. Recognizing non-traversable paths such as obstacles, cars, vegetation, etc is vital for ensuring the safety of both the autonomous vehicle and pedestrians. Semantic segmentation aids in creating a clear distinction between traversable and non-traversable areas.

Real-Time input: The segmented output from the model serves as real-time input for the path planning algorithm. The system dynamically adjusts its planned path based on the segmentation results, ensuring it follows the designated traversable paths while avoiding obstacles and non-traversable areas. It also enables the system to respond to changes in the environment, such as sudden appearance of obstacles or alterations in the road layout.[15]

- ROS topics we are getting from rs_camera.launch file:
    - /camera/color/camera_info
    - /camera/color/image_raw
    - /camera/color/image_raw/compressed
    - /camera/color/image_raw/compressed/parameter_descriptions
    - /camera/color/image_raw/compressed/parameter_updates

- /camera/color/image_raw/compressedDepth
- /camera/color/image_raw/compressedDepth/parameter_descriptions
- /camera/color/image_raw/compressedDepth/parameter_updates
- /camera/color/image_raw/theora
- /camera/color/image_raw/theora/parameter_descriptions
- /camera/color/image_raw/theora/parameter_updates
- /camera/color/metadata
- /camera/depth/camera_info
- /camera/depth/image_rect_raw
- /camera/depth/image_rect_raw/compressed
- /camera/depth/image_rect_raw/compressed/parameter_descriptions
- /camera/depth/image_rect_raw/compressed/parameter_updates
- /camera/depth/image_rect_raw/compressedDepth
- /camera/depth/image_rect_raw/compressedDepth/parameter_descriptions
- /camera/depth/image_rect_raw/compressedDepth/parameter_updates
- /camera/depth/image_rect_raw/theora
- /camera/depth/image_rect_raw/theora/parameter_descriptions
- /camera/depth/image_rect_raw/theora/parameter_updates
- /camera/depth/metadata
- /camera/extrinsics/depth_to_color
- /camera/motion_module/parameter_descriptions
- /camera/motion_module/parameter_updates
- /camera/realsense2_camera_manager/bond
- /camera/rgb_camera/auto_exposure_roi/parameter_descriptions
- /camera/rgb_camera/auto_exposure_roi/parameter_updates
- /camera/rgb_camera/parameter_descriptions
- /camera/rgb_camera/parameter_updates
- /camera/stereo_module/auto_exposure_roi/parameter_descriptions
- /camera/stereo_module/auto_exposure_roi/parameter_updates
- /camera/stereo_module/parameter_descriptions
- /camera/stereo_module/parameter_updates
- Depth data
- Canny edge detection
- Object detection (signals and signs)

The incorporation of the camera into the ROS framework streamlines data flow and facilitates seamless communication between perception algorithms. This not only enhances the efficiency of each individual algorithms but also allows for collaborative decision-making, where multiple

algorithms can leverage the camera feed simultaneously to generate a more comprehensive understanding of the rover's surroundings.

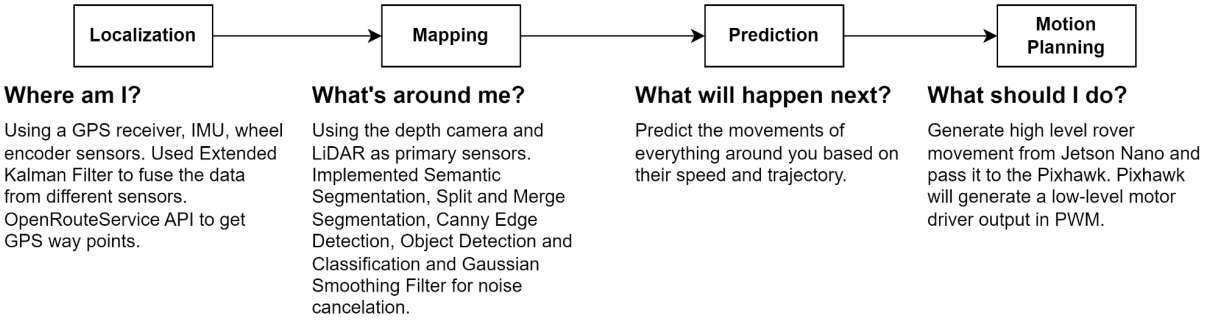
In essence, the camera is a multi-functional sensor that significantly contributes to the rover's perception capabilities, playing a central role in tasks ranging from semantic segmentation to lane detection.

CHAPTER 4

AGNI - An Autonomous System

4.1 Autonomous Vehicle Software

- Where am I? - **Localization** Detailed three-dimensional maps that highlight information such as road profiles, curbs and sidewalks, lane markers, crosswalks, traffic lights, stop signs, and other road features
- What's around me? - **Mapping** Scan constantly for objects around the vehicle—pedestrians, cyclists, vehicles, road work, obstructions—and continuously read traffic controls, from traffic light color and railroad crossing gates to temporary stop signs.
- What will happen next? - **Prediction/Planning** Predict the movements of everything around you based on their speed and trajectory
- What should I do? - **Control** Determine the exact trajectory, speed, lane, and steering maneuvers needed to progress along the route safely



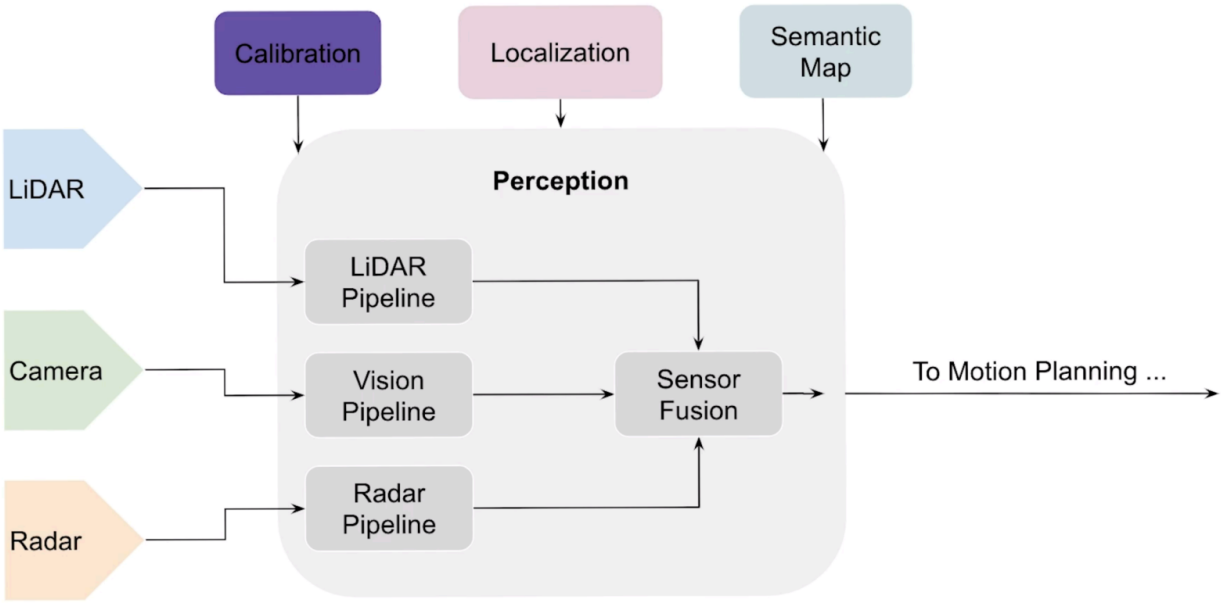
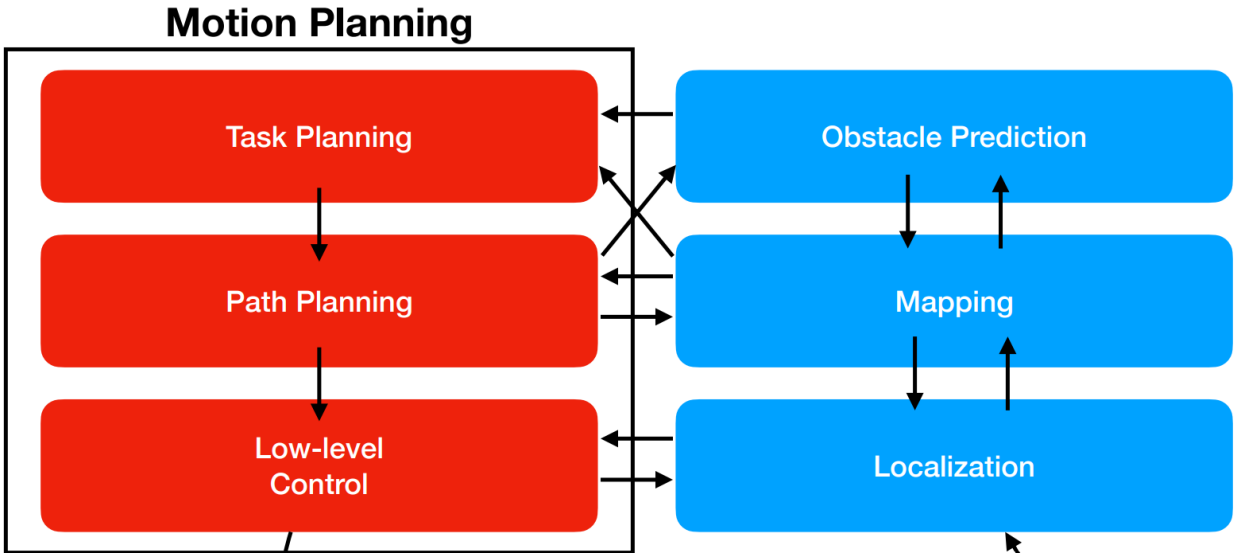


Fig 17: Actions in the Motion Planning of the rover

4.2 Autonomous Systems Stack

- Motion Planning
  - Low level control
    - PID
- **Navigation (Our main work is here)**



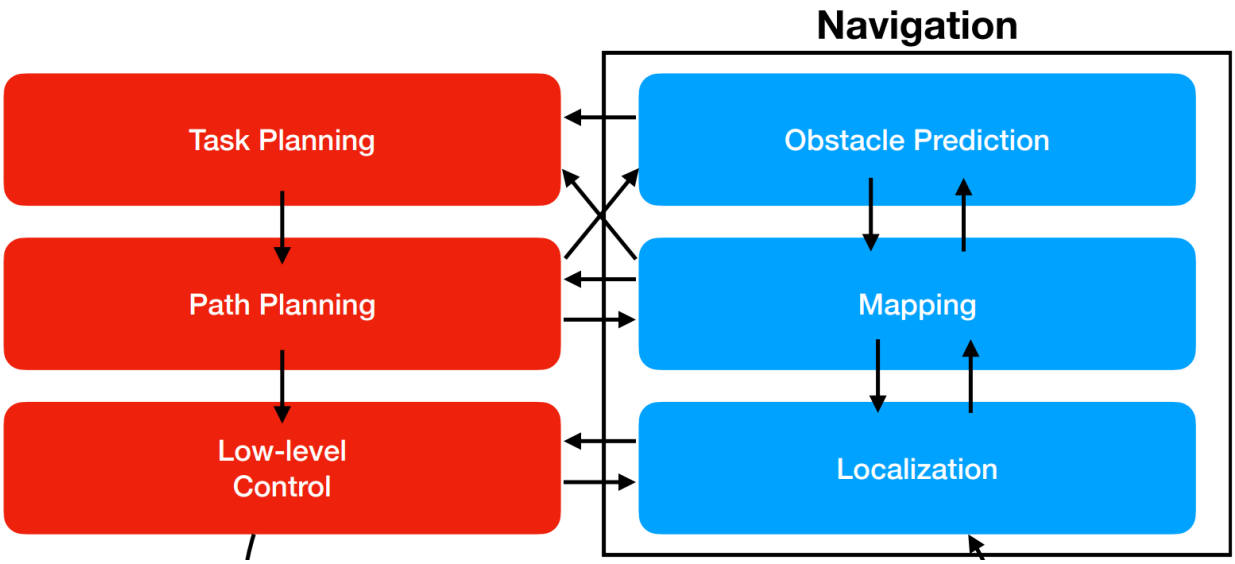
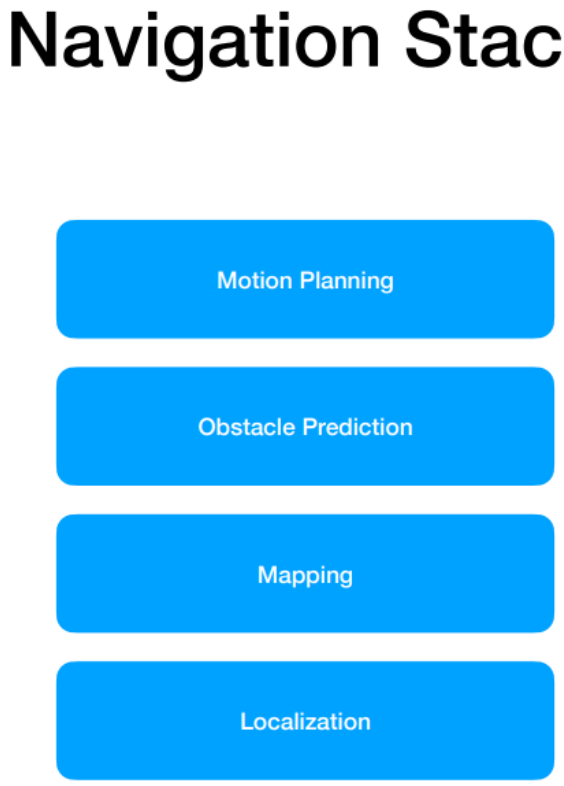


Fig 18: Block Diagram of the Path Planning in our rover

4.3 Navigation Stack



- Localization

        - Determine the state (position and orientation) of a robot with respect to the environment - faster faster.
        - IMU
        - GPS
            - 2-10 meter error
        - Localization can be within a global frame of a map, or relative to one's starting point
        - We are using Global map
        - Relative localization is more helpful in indoor mapping
        - Extended Kalman filters
        - Using SLAM for local navigation
        - Used wheel encoders
        - Using OpenRouteService API for global reference
    - Mapping
        - Used LiDAR for creating a 2D point cloud of the vicinity and getting sense of the distance of the rover from nearby obstacles.
        - Used Intel Realsense Depth camera as a vision sensor. In depth cameras we do get RGBD value for each pixel.
        - We are using semantic segmentation algorithm on the RGB output of the camera. Semantic segmentation will give the traversable path, avoiding grass, stones and uneven terrain.
        - We are using depth information to distinguish curb.
    - Obstacle Prediction/planning
    - Control
    - Split amd merge segmentation

4.4 State of the rover

- There are multiple rover states possible, such as:
- Go straight
- Turn due to boundary conditions
- Turn due to obstacle
- Stoppage due to obstacle
- Reverse due to all way blockage
- Critical point transition
- Turn right
- Turn left

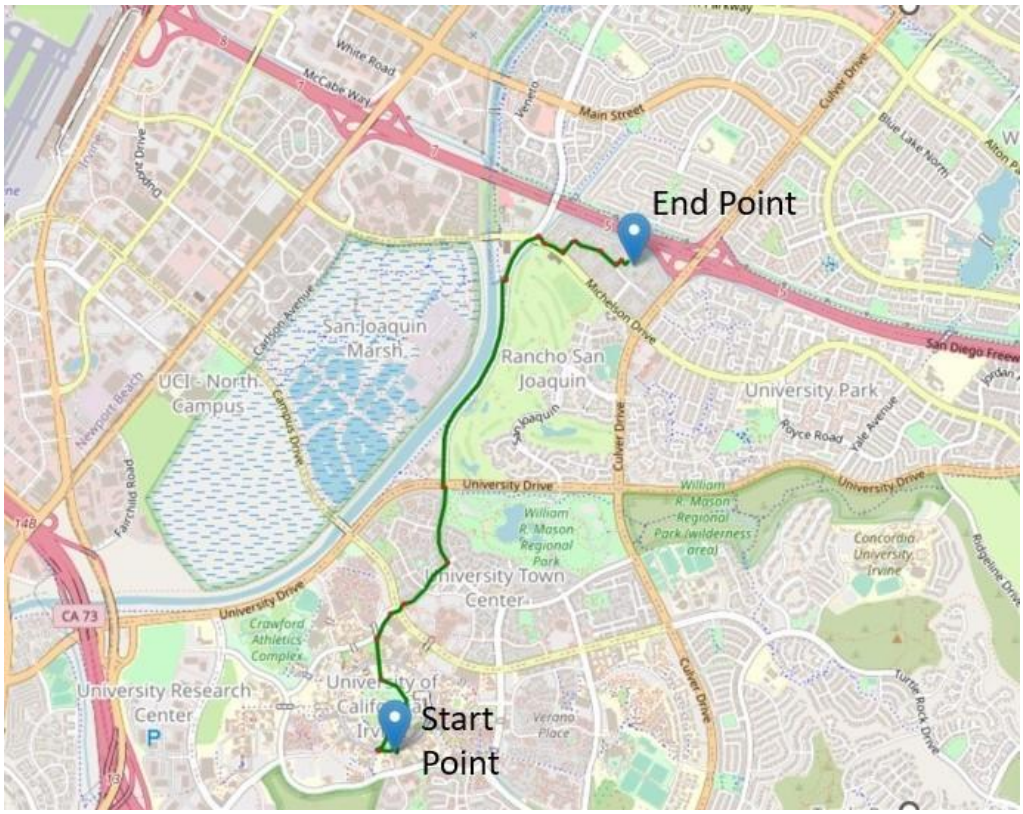


Fig 19: Folium map with waypoints generated along with the critical points

- Traversable path planning
  Autonomous outdoor ground navigation relies heavily on effective traversable path planning to ensure safe and efficient movement in complex environments. The key concepts are path planning, encompassing algorithms, sensor integration, obstacle detection and adaptability to different terrains.

- Local path planning with lidar, camera. Global path planning with gps

- OpenRouteService api for waypoints
- Curb detection
- Point Cloud formation
- Intent prediction
- Anomaly detection
- 2d to 3d point cloud projection and sensor fusion with camera
- Last mile delivery
- Imitation learning

CHAPTER 5

RESULTS AND ANALYSIS

5.1 Application

The autonomous rover and drone system has diverse applications across several industries due to its flexibility, adaptability, and ability to navigate complex environments. Some key applications include:

1. Last-Mile Delivery
    - Addressing the challenges in urban logistics by efficiently delivering packages to customers' doorsteps, overcoming traffic congestion and ensuring timely deliveries.
2. Agriculture
    - Crop monitoring, analysis, and management by utilizing drones for aerial surveys and rovers for ground-based data collection, aiding in precision agriculture and yield optimization.
3. Surveillance and Security
    - Patrols in sensitive areas or large premises using drones for aerial surveillance, complemented by rover patrols for ground-level monitoring, enhancing security measures.
4. Infrastructure Inspection
    - Assessing critical infrastructure such as bridges, pipelines, or power lines, where drones provide aerial inspections while rovers navigate difficult terrains for closer inspection.
5. Environmental Monitoring
    - Conducting environmental surveys, mapping, and monitoring biodiversity in remote or challenging terrains where drones assist in aerial assessments and rovers gather ground-level data.
6. Search and Rescue Operations
    - Utilizing drones for rapid aerial search and providing real-time information to guide rovers in navigating challenging terrains during search and rescue missions.
7. Disaster Response
    - Deploying drones to assess disaster-affected areas for damage evaluation, while rovers navigate through debris to provide on-site assistance or gather vital information.
8. Exploration and Mapping

- Conducting exploration missions in remote or hazardous environments, where drones gather aerial imagery and rovers traverse the terrain, aiding in mapping and data collection.

9. Scientific Research

- Assisting researchers in data collection for various scientific studies, such as geology, biology, or archaeology, by combining aerial and ground-level observations.

10. Entertainment and Events

- Enhancing event experiences through aerial photography or videography using drones, complemented by rovers for ground-level interactions or assistance in event management.

The synergy between the rover and drone enables versatility in addressing a wide array of tasks across different industries. Their combined capabilities facilitate comprehensive data collection, efficient navigation through various terrains, and adaptability to different environments, making this integrated system highly applicable and versatile in numerous scenarios.

5.1 Future Work

The AGNI - Autonomous Ground Navigation Initiative has laid a solid foundation for revolutionizing autonomous navigation, but there are several avenues for future work to enhance the system's capabilities and address emerging challenges:

1. Advanced Sensor Integration: Explore and integrate emerging sensor technologies to further enhance the system's environmental awareness. Investigate the use of 3D imaging, multi-modal sensors, and improved LiDAR technologies to provide more detailed and accurate perception of the surroundings.

2. Machine Learning and AI Algorithms: Implement and refine machine learning algorithms for better decision-making in dynamic environments. Develop models that allow the system to learn from real-world scenarios, improving its ability to adapt to unforeseen obstacles and changes in the environment.

3. Enhanced Collaboration Between Rover and Drone: Investigate ways to optimize the collaboration between the rover and drone components. Develop algorithms that enable seamless handovers between the two systems, ensuring efficient and coordinated navigation across diverse terrains and scenarios.

4. Localization and Mapping Improvements: Focus on refining localization and mapping algorithms to enhance precision and reliability, especially in GPS-challenged environments. Investigate the use of simultaneous localization and mapping (SLAM) techniques to improve the system's ability to navigate with minimal reliance on external positioning systems.

5. Scalability and Customization: Work on making the AGNI system more scalable and customizable for different applications and industries. Explore modular designs and configurations that can be adapted to specific use cases, such as disaster response, agriculture, or surveillance.

6. Human-Robot Interaction: Research and implement improved human-robot interaction features. Enhance the system's ability to communicate with and respond to human presence in urban environments, ensuring safety and smooth integration into daily activities.

7. Energy Efficiency and Autonomy: Focus on optimizing energy consumption for both the rover and drone components. Research and implement strategies for prolonged autonomy, including efficient charging solutions and energy-aware navigation algorithms.

8. Real-world Testing and Validation: Conduct extensive real-world testing in diverse environments to validate the system's performance and robustness. Gather feedback from practical deployments to fine-tune algorithms and address any unforeseen challenges that may arise in complex, dynamic scenarios.

9. Regulatory Compliance: Stay abreast of regulatory developments in autonomous navigation and ensure that the AGNI system aligns with industry standards and safety regulations. Collaborate with regulatory bodies to facilitate the integration of autonomous systems into various sectors.

10. Community Involvement and Open Source Contribution: Foster community involvement by open-sourcing relevant components of the AGNI system. Encourage collaboration with researchers, developers, and industry experts to collectively advance the field of autonomous navigation.

By pursuing these future directions, the AGNI project aims to continually push the boundaries of autonomous navigation, making strides towards a more adaptive, intelligent, and widely applicable system across diverse industries and environments.

5.2 Conclusion

In conclusion, the AGNI - Autonomous Ground Navigation Initiative stands at the forefront of addressing the inherent challenges in autonomous navigation across diverse and dynamic environments. While significant strides have been made in automating controlled and structured settings, the limitations become pronounced when faced with the complexities of unstructured terrains, congested urban landscapes, and ever-changing environments.

The project recognizes the shortcomings of existing standalone solutions, be it rovers or drones, and emphasizes the critical need for an integrated system that seamlessly combines the strengths of both technologies. The vision is to create a comprehensive solution capable of adaptive navigation through various scenarios, from rough terrains to congested urban settings, and restricted areas. This integrated approach not only overcomes individual limitations but also introduces a new level of versatility by allowing the system to transition seamlessly between rover and drone functionalities.

The core innovation of AGNI lies in its holistic approach to autonomous navigation. By integrating cutting-edge technologies, such as advanced sensor suites, powerful computing, and the flexibility of the Robot Operating System (ROS), the initiative aims to revolutionize conventional delivery practices. The end-to-end pipeline being developed is poised to set new standards in the industry, providing real-time decision-making capabilities and adaptability that are essential for navigating the complexities of diverse environments.

As AGNI paves the way for a more versatile, adaptable, and robust Autonomous Ground Navigation System, it not only targets last-mile delivery challenges but also envisions applications across various industries, including agriculture, surveillance, disaster response, and infrastructure inspection. The project's commitment to overcoming the limitations of current autonomous systems reflects a pioneering leap in autonomous vehicle solutions, promising transformative advancements in the way we navigate and interact with dynamic environments.

References

[1] “ClearPathRobotics”, Available: https://clearpathrobotics.com/assets/manuals/outdoornav/archived/0.6.0/ui_overview.html [Accessed 15th April 2023].

[2] “TowardsDataSciences”, Available: https://towardsdatascience.com/sensor-fusion-interpolation-for-lidar-3d-point-cloud-data-labeling-38ec670b481 [Accessed 20th April 2023].

[3] “Discuss”, Available: https://discuss.ardupilot.org/t/ardurover-r1-available-now/22109 [Accessed 5th May 2023].

[4] ARTUR SHURIN, "The Autonomous Platforms Inertial Dataset”, *IEEE Access,* 2022.

[5] VISHAL SHARMA, SAKSHAM SANGWAN , KARAN SINGH BORA, “Design, Development and Control of a Planetary Rover using ROS”, Second International Conference on Electronics and Sustainable Communication Systems (ICESC), 2021

[6] MINGXING WEN, YUNXIANG DAI, TAIRAN CHEN, “A Robust Sidewalk Navigation Method For Mobile Robots Based on Sparse Semantic Point Cloud”, International Conference on Intelligent Robotics and Systems(IROS), 2022

[7] “Hackster”, Available: https://www.hackster.io/Matchstic/connecting-pixhawk-to-raspberry-pi-and-nvidia-jetson-b263a7 [Accessed 28th April 2023]

[8] “GitHub”, Available: https://github.com/Slamtec/rplidar_sdk [Accessed 10th May 2023]

[9] “GitHub”, Available: https://github.com/IntelRealSense/realsense-ros/tree/ros1-legacy [Accessed 16th May 2023]

[10] “Ardupilot”, Available: https://ardupilot.org/rover/docs/common-pixhawk-overview.html [Accessed 25th April 2023]

[11] “ Mission Planner”, Available: https://ardupilot.org/planner/docs/mission-planner-installation.html [Accessed 19th April 2023]

[12] “Digital-Geography”, Available: https://digital-geography.com/openrouteservice-api-a-leaflet-example-for-isochrones/ [Accessed 15th May 2023]

[13] “GitHub”, Available: https://masoudir.github.io/mavros_tutorial/ [Accessed 1st May 2023]

[14] “GitHub”, Available: https://github.com/lukewenMX/Robust-Navigation-Method [Accessed 1st October 2023]

[15] “GitHub”, Available: https://github.com/dusty-nv/jetson-inference/blob/master/docs/segnet-camera-2.md [Accessed 1st October 2023]

ACRONYMS

AGNI	Autonomous Ground Navigation Initiative
GPS	Global Positioning System
API	Application Programming Interface
ROS	Robot Operating System
IMU	Inertial Measurement Unit
SLAM	Simultaneous Localization and Mapping
HDMI	High Definition Multimedia Interface
GPIO	General Purpose Input Output
RC	Radio Control
MAVL	Micro Air Vehicle Link
MAVP	Micro Air Vehicle Proxy