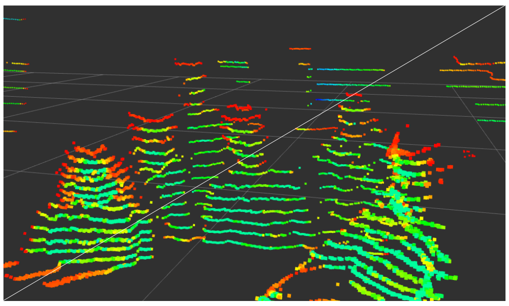
BILKENT UNIVERSITY

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING



Committee Meeting IV Report

Project Name

Sensör Füzyonu ile Navigasyon, Sahne Eşleştirme ile Hedef Tespiti Yapabilen Otonom Sistem

Group Members

Su Direkci, Yiğit Efe Erginbaş, Yusuf Umut Çiftci
Cafer Mertcan Akçay, İrem Kaftan, Özgür Bora Gevrek
(Group B5)

Academic Mentor

Prof. Dr. Levent Onural

Course Coordinator

Mehmet Alper Kutay

Teaching Assistant

Aslı Alpman

Company Mentors

Adil Can Dai
Erkan Milli
Sertaç Erdemir

Date

May 20, 2021

Project Summary

With the increasing advantages of autonomous vehicles in warfare, stakeholders of the defense industry such as Roketsan are exploring the future territory with various research and development projects. Accordingly, our project aims to construct an autonomous system that can navigate an unknown location and locate a predetermined target. For this purpose, we use a stereo camera, a LIDAR, and an INS. By analyzing the data stream obtained from these sensors, the robot decides its next maneuver and maps its surroundings. To achieve this, we have defined and achieved numerous work tasks, such as temporal and spatial calibration between the sensors, SLAM (Simultaneous Localization and Mapping), segmentation, scene matching, target detection, building the robot, and motion planning. We assigned these tasks to different group members and have achieved all of the milestones that we had determined. We have implemented the algorithms in ROS using C++, optimized the motion planning algorithms, achieved localization and mapping, constructed the robot and handled the communication between different computing units. In the end, we are able satisfy our criterion for success which is to build a robot that explores its environment autonomously and locates a predetermined target. We have tested our robot in a large-scale maze that we have constructed. The results show that the robot is able to explore and map the maze while localizing itself over this map. Furthermore, when it comes across the predetermined target in the maze, it is able to detect the target and mark it using a laser pointer.

Table of Contents

List of Figures

1 Company Information

Roketsan was founded on 14 June 1988 by the Defense Industries Executive Committee. Today, Roketsan develops and manufactures various types of rockets, missiles, guidance-control systems, fuzes, warheads, mechanical parts, algorithms, and software, and gives logistical support for these systems vehicles. Its head office is located in Kemalpaşa district, Elmadağ, Ankara, Turkey.

2 Motivation and Novelty

The importance of autonomous vehicles in the military setting is increasing all around the world. Such vehicles provide numerous advantages in warfare, such as detecting threats in a hostile area, performing target acquisition with greater accuracy compared to humans, and mapping areas that are potentially dangerous for troops [1]. For this reason, defense companies are competing to develop autonomous robots equipped with cutting edge technology. In this context, Roketsan is exploring the future territory on autonomous vehicles, and this project can shed some light in this direction. Our project is a proof of concept for a vehicle capable of navigating its environment, locating and shooting its target autonomously. The company does not plan to use the results of this project immediately. This project can become an asset for the company in the autonomous vehicle territory, provide experience, and demonstrate capability in the field. In the future, Roketsan can utilize some of the knowledge gathered from this project for their decisions in their future plans for autonomous agents.

The project is not as specialized as to suggest a patent application. There are many patents related to autonomous cars from car and car parts manufacturers. An example is US6151539A [2], a patent owned by Volkswagen named "An autonomous vehicle arrangement and method for controlling an autonomous vehicle", which describes an autonomous vehicle arrangement composed of several subsystems, sensors, and arrangements. A patent more similar to our project is US7840352B2 from Honeywell International Inc. demonstrating an autonomous vehicle system with Global Positioning System (GPS) sensors, inertial sensors, and image sensors [3].

Our project is a prototype for a vehicle capable of navigating, locating, and shooting its targets autonomously. Several similar products are demonstrating similar capabilities for different tasks. Spot robot by Boston Dynamics is an advanced showcase of the concepts that we are using [4]. It is a four-legged robot, capable of 3D mapping its environment and navigating autonomously by avoiding obstacles. It achieves these capabilities via stereo cameras and has a purchase price of $75,000 [5]. Compared to our project, Spot's advantages are its increased manoeuvre capabilities and its ability to move on rugged terrains thanks to its four legs. However, our project's advantage is the usage of a LIDAR, which can increase the mapping accuracy in long ranges. Another product similar to our project is SLAM Robot by Pantech Solutions [6]. It is a lower-end product compared to the Spot robot, used for development and educational purposes. It performs SLAM using a LIDAR, operating on a Raspberry Pi. It also uses the Robot Operating System (ROS) as we do. Compared to this robot, our project's distinctive features are the abilities of target localization and scene matching by using a stereo camera.

The target end-users of our product are military personnel and defense companies. Therefore, the expected technical backgrounds are diverse. The employees at defense companies can develop our final product to fit their needs, whereas military personnel can immediately start using the product. As the budget for our project is $5000, we assume that the users are likely to spend at least this amount on our product. Familiarity with the Linux operating system is needed to use our system, as the users will launch the application from the terminal. Therefore, users without such experience will need basic training to use our product. However, if needed, the system is suitable to be integrated with a more user-friendly graphical interface. Nevertheless, such an interface is out of the scope of this project.

3 Requirements

3.1 Functional Requirements

The final product should be an autonomous system whose aim is to locate a predetermined target. For this purpose, the autonomous robot is equipped with the following sensors: a stereo camera, a LIDAR, and an INS. These sensors generate data streams continuously throughout the operation cycle of the robot, and the robot fulfills its task by fusing these data streams to obtain a more precise estimation of its surroundings, and by approaching the target with this information. When searching for the target, the robot generates a 2D map of its current and past surroundings to understand its location, and to choose an available (traversable) path to approach the target. In order to choose the optimal path or to detect the target in the shortest time, the robot uses powerful tools such as scene matching and object detection. The results of exploration are saved in a file that can be opened via RViz, a 3D visualization tool for ROS. The system can be started from the terminal by connecting to the Jetson TX2 board remotely and we are not planning to design a GUI. Therefore, it is not listed as a requirement. Further analysis of the requirements can be found below.

a) Data Synchronization: To synchronously and optimally fuse the data streams obtained from the stereo camera and the LIDAR, the system uses location, speed, and acceleration data, obtained from the INS. With this strategy, possible errors that can arise from incongruities in the data acquisition frequencies of the sensors should be prevented. In order to ensure robust and continuous data streaming, data synchronization should be performed at a high rate. The most computationally costly part of this stage is the fusion of the point clouds and it should be performed at a rate of at least 10 fusions per second.

b) Spatial Mapping: The system contains a stereo camera and a LIDAR in order to gather data about its surroundings within 3 meters. The data gathered from these sensors in the form of point clouds should be processed simultaneously and merged with past data using SLAM algorithms. As the robot obtains point cloud data from two different sensors, it should fuse them in order to minimize errors due to any internal noise. The robot should also contain various thresholds and precautionary algorithms to preclude any possibility of error: for example, if the difference between the point cloud information obtained from the stereo camera and the LIDAR is not negligible, the robot should choose one of them temporarily. The speed of generating and updating the map should be able to follow the data acquisition rate. Therefore, we need to have a rate of at least 10 map updates per second.

c) Autonomy: The system should be completely autonomous in navigation, data acquisition, data processing, and decision making based on the environment. Except for a precautionary "terminate process" button, the final product should be able to operate on its own without any interruptions. After processing the data coming from the stereo camera and the LIDAR, the robot should decide its next maneuver based on the surrounding obstacles and passages. For this purpose, the robot should perform segmentation on the image data obtained from the stereo camera as well. Since the segmentation is not one of the crucial tasks in the overall flow, we do not require high rates. Furthermore, the segmentation task is computationally expensive and therefore it may not be feasible to perform it at high rates. For the purposes of our system, a rate around 2-3 Hz should be enough.

By combining the results of segmentation and information about depth coming from the point cloud data, the robot should decide on the best maneuver. Furthermore, the robot should adjust its speed based on the surrounding structures. Lastly, to prevent any loops in the track (due to revisiting a previously processed location), the robot should have internal decision-making mechanisms that can preclude this possibility by using the location information obtained from the INS, and the current map in the memory. We need to generate the paths at a rate of at least 10 updates per second to match the speed of map updates. In order to stably control the robot without any oscillations, delays, or other malfunctions; we require much higher rates in generating and sending the motor command: at least 50 motor commands per second.

d) Mobility: Mobility should be obtained with the help of motors powered with batteries and the maneuvering ability should be provided with wheels. The motor speed and the direction of movement should be determined by the microcontroller, which in turn can send the necessary signals to the motors. In order to ensure motor commands are issued at the required rate, the microcontroller should be able to generate PWM signals of at least 400 Hz and change the pulse-width of this signal with at a rate of at least 50 changes per second.

e) Target Localization: This capability is one of the most essential functions of the robot. The target should be described to the robot with the help of a picture that contains information about its location, which can be deduced from objects in the background etc. Throughout its steering, the robot should analyze the images captured by the stereo camera continuously. In these images, with the help of scene matching algorithms, it should try to find networks of objects similar to the one given in the target description by analyzing prominent features of the images. If the robot is confident enough about a match of the target in the image data, it should point to the target with a laser beam to let the user know that the target is found. Then, it should stop moving as long as the target is within the scene.

3.2 Non-Functional Requirements/Constraints

a) Cost: The maximum cost allowed for our project is 5,000$.

b) Size: There are not any constraints defined explicitly for our robot's size and weight. However, there are some limitations to consider while building the product. First, it is desired to have good maneuverability since it needs to move around easily to collect data. If its size becomes too large, the robot is not able to maneuver easily in confined spaces. On the other hand, it should be large enough to carry all the sensors, the board, batteries, etc. Considering these limitations, we have built our robot smaller than 50x50x50cm. Also, weight is an important limitation for our product because it increases the need for power and higher-end motors are required for carrying more weight. Thus, while building the chassis, we have used light materials such as aluminum plates with holes.

c) Power: The constraint for power is mostly be determined by the motors as they draw more power when they start moving. At the peak point, we are expecting a total of 4 amperes current from the two motors. We can also estimate the current from the Jetson TX2 board since we tested the board at maximum load in the laboratory and it draws 2.5 amperes at peak. Li-Po batteries can supply more than the estimated current we need. However, their capacity will be a constraint since it is around 1500 mAh for moderately priced batteries. Therefore, the power consumption should be as optimized as possible.

d) Environment: It is specified that the environment in which our robot needs to work is indoors. We built the setup where we tested our robot.

e) Safety and Health: There are several possible safety hazards for our product. First, since the system is autonomous, it can move out of control if an error occurs. Also, Li-Po batteries may explode if they work under extreme conditions or if they get damaged. Moreover, chemicals in the Li-Po batteries may be hazardous to human health if exposed. Health issues in our project can mainly arise from safety issues such as a robot physically harming a person due to its autonomous movement. Hence, our main health requirement is to have a shutdown button for should such cases occur.

f) Global, cultural and social factors: For the design of our robot, we do not consider cultural and social factors in the direct sense. However, the design is a direct result of global factors. We are choosing our parts for the robot, our hardware, and software packages from what is available currently in the world. And the state of the technologies that we use is the result of the combined work of many people before us.

g) Standards to follow: The standard that we found most suitable to follow for our project is OSHA 29 CFR 1910.333 [7]. It constitutes a guide for industrial robots and robot system safety. We adhered to that.

4 Big Picture

The big picture of the project can be found in Figure 1. For a larger version, see Appendix, Figure 1. As can be seen from the figure, we use the NVIDIA Jetson TX2 board as the main computing board, where all the applications run. These applications mainly include segmentation, SLAM, scene matching, point cloud fusion, and motion planning. The three main sensors are the LIDAR, the ZED2 stereo camera, and the Xsens MTi-7 GNSS/INS module. The stereo camera and the INS module are connected to the Jetson TX2 board with USB, whereas the LIDAR communicates with the board through its interface box. As the board has only one port for USB 3.0, we use a USB hub as well. The motor driver and the interface box provide analog communication with the motors and the LIDAR, respectively. The interface box is connected to the board with an Ethernet cable, whereas the motor driver has an analog connection. Lastly, the motors of the pan-tilt laser pointer are connected to the Arduino MEGA board for control signals and connected to the Jetson TX2 board for power.

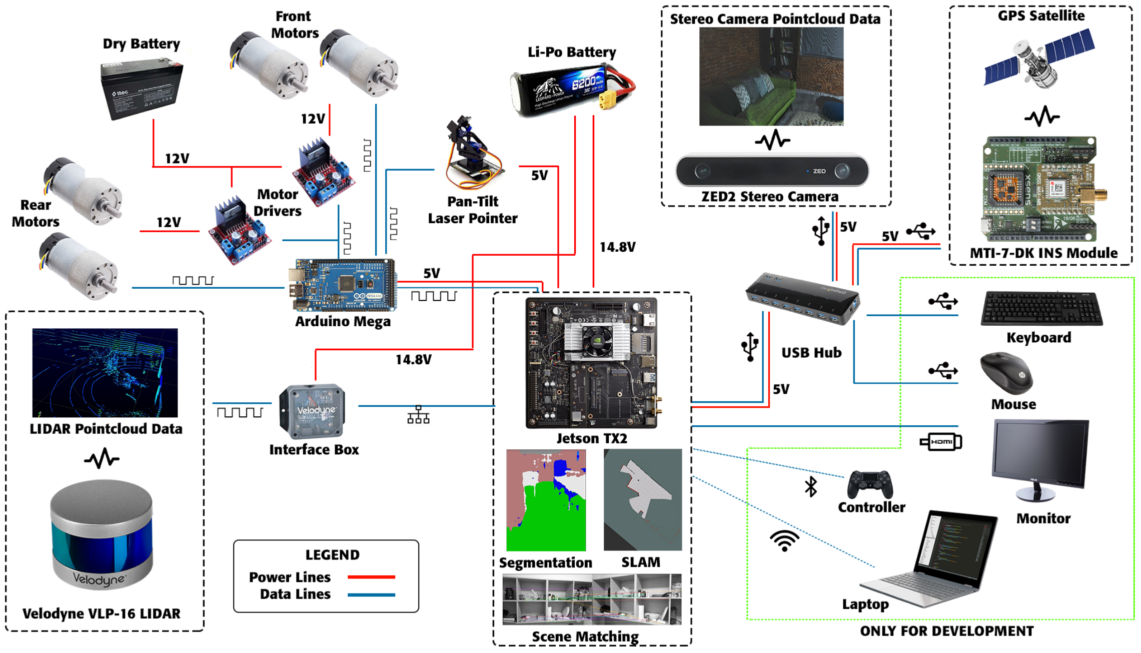


Figure 1: The block diagram of the project.

The sensors and the Jetson TX2 board are fixed on the robot. During development, Jetson TX2 is connected to output devices such as a keyboard, a monitor, and a mouse with USB and HDMI to utilize the user interface provided by the board. However, during the testing of new algorithms, the connection is established via WiFi as the robot is not stationary. The system is currently powered by a 12V dry battery and a 14.8V Li-Po battery. The power is distributed to the system via a power distribution board.

For the final product, we expect users to establish connection via SSH, which is the method that we use. Then, the system is launched from the terminal. We use RViz for mission monitory, which is a 3D visualization tool for ROS. We didn't implement any additional user interface or tool for mission monitory as it poses an additional computational cost for the system. However, we are logging the crucial data for later investigation. The system can be turned off by closing the terminal.

5 Methods and Implementation Details

This section includes our project plan and the technical details of the tasks required to complete the project.

5.1 Work Breakdown Structure and Project Plan

The main work packages of the first semester that are grouped under 1.1 and 1.2 in Figure 2 are:

a) Data Acquisition from INS: This task includes obtaining the GPS information as well as the orientation and position data which is calculated by the Inertial Measurement Unit (IMU) which is used to calibrate the stereo camera.

b) Data Acquisition from Stereo Camera and Its Calibration with INS: This task includes calibrating the stereo camera and INS to eliminate any spatial and temporal differences.

c) Data Acquisition from LIDAR and Its Calibration with Stereo Camera: This task includes obtaining point cloud data from the stereo camera and LIDAR and calibrating them to eliminate any spatial and temporal differences. The robot's location at the moment of data acquisition is also needed to adjust the speed of the robot if necessary and to apply a transformation to the point cloud data. This transformation is necessary as the point cloud data is given with respect to the camera's perspective at all times. These conversions are needed for adjusting the reference according to our current location, merging the new point cloud data with the existing information, and mapping the surroundings.

d) Point Cloud Fusion (PCF): Given the point cloud data of the stereo camera and the LIDAR, we need to fuse these points to obtain a much more informative map of our surroundings. By using the fused point cloud, our goal is to reduce the uncertainty and errors in the spatial map that is used by the robot.

e) Segmentation: This task includes semantic segmentation of the objects around the robot by using the image data coming from the stereo camera, which is essential for distinguishing convenient paths.

f) Scene Matching: This task includes detecting pivotal objects in a photo to decide whether the current scene matches with the target scene (i.e. the target and its background). When we get a match, we need to find the target in the scene.

The main work packages of the second semester that are grouped under 1.3 and 1.4 are in Figure 2 are:

a) Simultaneous Localization and Mapping (SLAM)

- **Odometry:** This task includes fusing odometry data by using a Kalman filter where the states represent the acceleration, velocity, and position in all 6-axes (x, y, z, pitch, yaw, and roll) to obtain velocity information.

- **Gmapping and PCF Integration and Localization:** This task includes generating a dynamic 2D map of the environment by using the fused point cloud and the odometry data, as well as showing the location of the robot on the created map.

b) Motion Planning

- **Navigation and Path Planning:** This task includes planning and making correct decisions about which path to follow by implementing motion planning algorithms in order to reach a particular coordinate without running into any obstacles and getting closer to the target scene.
- **Exploration:** This task includes determining frontiers and exploring the unseen locations based on these frontiers.

c) Building the Robot: This task includes building a robot that can carry the three sensors, Jetson TX2 module, and batteries.

d) Controlling the Robot: This task includes controlling the robot via a joystick in order to see whether the wheels operate correctly and the batteries give sufficient power to the components.

e) Connecting Jetson TX2 to Monitor via Wi-Fi: After ensuring that the robot can carry the components without any safety issues, the algorithms are tested on the robot to see whether it can function autonomously. For this purpose, the Jetson TX2 module shares its screen via Wi-Fi.

f) Target Detection and Laser Pointing: This task includes finding the target object in a given scene.

g) Laser Pointing: This task includes implementing a platform for a laser pointer on the robot that points its laser to the target in order to show that the target is found successfully.

The work breakdown structure is given in Figure 2. The work packages that were successfully completed in the first semester are grouped together and marked as 'completed' as seen in the chart. Specifically, Efe completed the calibration of the stereo camera with the LIDAR, İrem completed data acquisition from the INS module and its calibration with the stereo camera, Su completed point cloud fusion, Özgür completed segmentation, and Cafer completed scene matching in the first semester.

Our work breakdown structure has been changed since the second committee meeting because new work packages were determined and distributed among the group members for the second semester but there have not been any changes since the third committee meeting. We successfully completed all the work packages that we had determined for the second semester. As seen in the chart, Efe completed fusing odometry data via Kalman filtering and connecting Jetson TX2 module to the monitor via Wi-Fi and Su completed implementing simultaneous localization and mapping (SLAM) and target detection algorithms. Specifically, she worked on implementing the Gmapping algorithm and integrating point cloud fusion and localization. İrem completed implementing navigation and path planning algorithms in collaboration with Özgür who completed implementing exploration algorithms. The successful implementation and integration of these algorithms have been crucial for the autonomous movement of the robot. Cafer and Umut completed building the robot from scratch which requires custom design and integration of several components. Cafer and Umut also worked on controlling the robot and laser pointing, respectively.

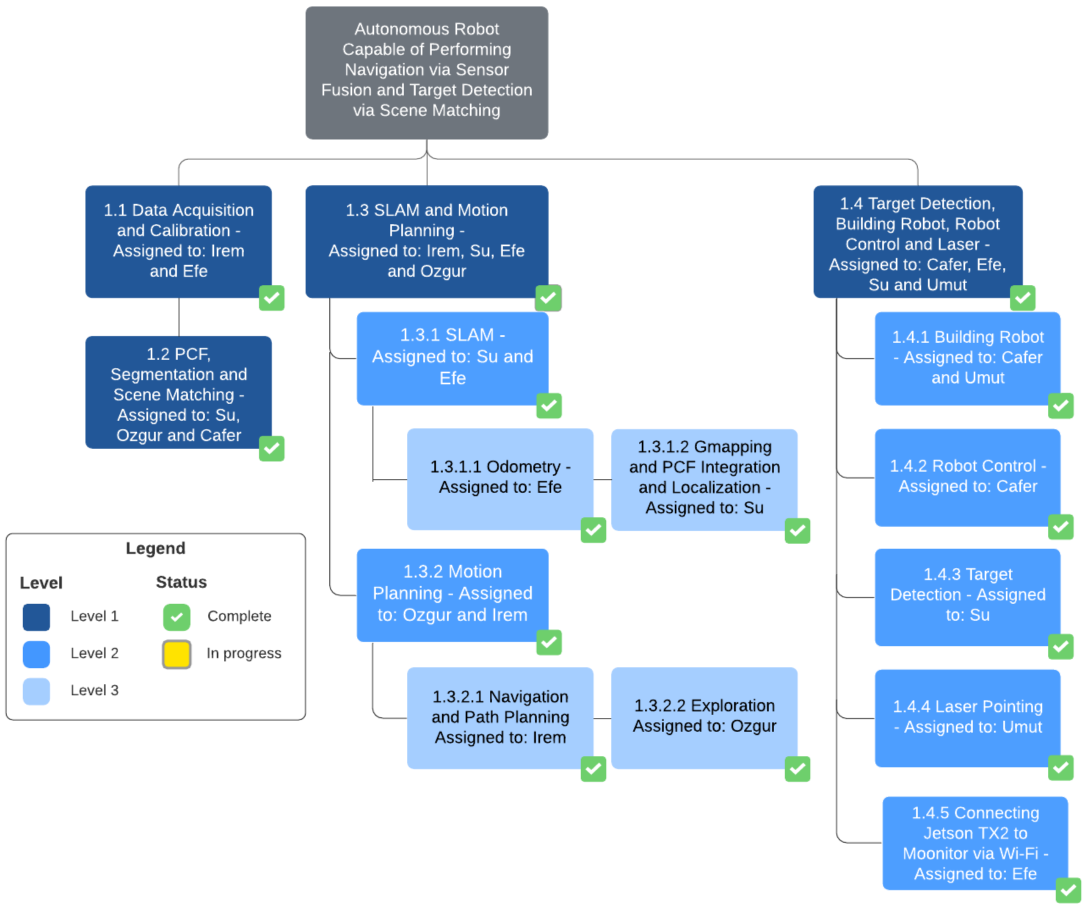
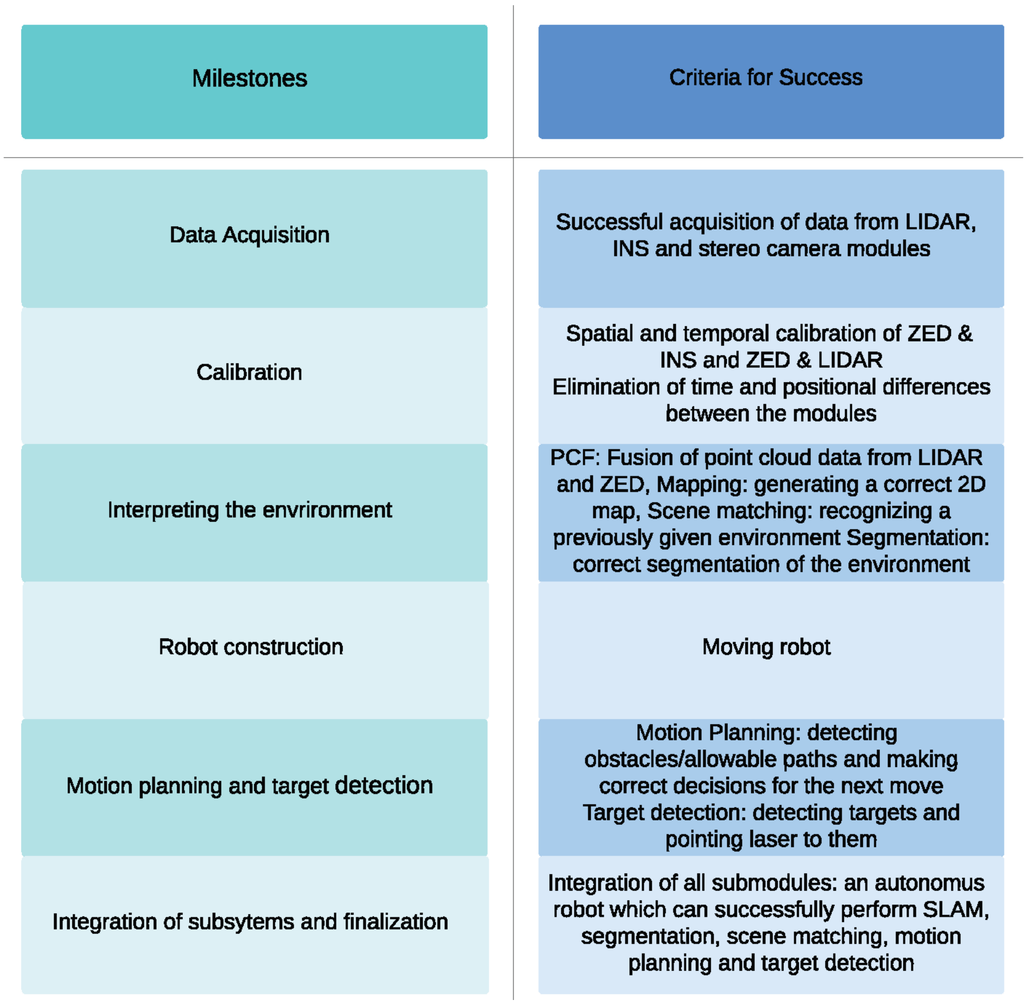

Autonomous Robot Capable of Performing Navigation via Sensor Fusion and Target Detection via Scene Matching
1.1 Data Acquisition and Calibration - Assigned to: Irem and Efe
1.2 PCF, Segmentation and Scene Matching - Assigned to: Su, Ozgur and Cafer
1.3 SLAM and Motion Planning - Assigned to: Irem, Su, Efe and Ozgur
1.3.1 SLAM - Assigned to: Su and Efe
1.3.1.1 Odometry - Assigned to: Efe
1.3.1.2 Gmapping and PCF Integration and Localization - Assigned to: Su
1.3.2 Motion Planning - Assigned to: Ozgur and Irem
1.3.2.1 Navigation and Path Planning Assigned to: Irem
1.3.2.2 Exploration Assigned to: Ozgur
1.4 Target Detection, Building Robot, Robot Control and Laser - Assigned to: Cafer, Efe, Su and Umut
1.4.1 Building Robot - Assigned to: Cafer and Umut
1.4.2 Robot Control - Assigned to: Cafer
1.4.3 Target Detection - Assigned to: Su
1.4.4 Laser Pointing - Assigned to: Umut
1.4.5 Connecting Jetson TX2 to Moonitor via Wi-Fi - Assigned to: Efe
Legend
Level
Level 1
Level 2
Level 3
Status
Complete
In progress

Figure 2: The work breakdown structure of the project.

We have determined six milestones which are shown and described in Figure 3. The first milestone is data acquisition. Our criterion for success is successfully gathering data from the three sensors (LIDAR, stereo camera, and INS) since this step is crucial for the continuation of our project. The second milestone is calibration. We need to successfully calibrate the stereo camera with the INS module, and the stereo camera with the LIDAR to eliminate the differences between the spatial and temporal properties of their data. The third milestone is interpreting the environment which includes point cloud fusion, spatial mapping, segmentation, and scene matching. The two point cloud data obtained from the stereo camera and the LIDAR need to be fused successfully in order to be used in further stages, such as spatial mapping and segmentation. The criteria for success for other algorithms are generating a 2D map of the environment based on sensor data for spatial mapping, correctly segmenting the environment for segmentation, and recognizing a previously given environment for scene matching. The fourth milestone is building the robot as our final product is a moving autonomous robot. The criteria for success are obtaining a moving robot which can carry the components without any safety issues, controlling the robot via a joystick for the initial stage, and connecting the Jetson TX2 module to the monitor via Wi-Fi. The fifth milestone is motion planning and target detection. The criteria for success are detecting allowable paths and making correct decisions for the next maneuver (navigation, path planning, and exploration), localizing the target correctly in a given scene for target detection, and pointing it with a laser in order to signal that the target has been found successfully. The final milestone is the integration of the subsystems and their optimization. The criterion for success is obtaining an autonomous robot which can successfully perform SLAM, segmentation, scene matching, motion planning, target detection, and laser pointing.



Milestones	Criteria for Success
Data Acquisition	Successful acquisition of data from LIDAR, INS and stereo camera modules
Calibration	Spatial and temporal calibration of ZED & INS and ZED & LIDAR Elimination of time and positional differences between the modules
Interpreting the envrironment	PCF: Fusion of point cloud data from LIDAR and ZED, Mapping: generating a correct 2D map, Scene matching: recognizing a previously given environment Segmentation: correct segmentation of the environment
Robot construction	Moving robot
Motion planning and target detection	Motion Planning: detecting obstacles/allowable paths and making correct decisions for the next move Target detection: detecting targets and pointing laser to them
Integration of subsytems and finalization	Integration of all submodules: an autonomus robot which can successfully perform SLAM, segmentation, scene matching, motion planning and target detection

Figure 3: The milestones of the project.

The Gantt chart for our WBS (project timeline) with responsible team members is shown in Figure 4. A larger version of the figure can be found in Appendix, Figure 2 . We worked on our tasks separately in the first semester and made sure that they worked successfully separately. We have successfully followed our timeline and finished the tasks for the first semester. These include data acquisition from the three sensors, calibration of INS and ZED2 and LIDAR and ZED2, point cloud fusion, spatial mapping, scene matching, and segmentation. Furthermore, we have completed some parts of the tasks for motion planning (navigation, path planning, and exploration) and target detection.

In the second semester, we have built the robot and achieved controlling it. Next, we have put emphasis on implementing motion planning algorithms because autonomous navigation is the core of our project. Then, we have worked on odometry fusion, target detection, and laser pointing. Lastly, we have integrated all the tasks and optimized the algorithms that we implemented within our full-blown system. At the end, we tested the robot in a scenario based environment in order to see whether the robot can function autonomously.

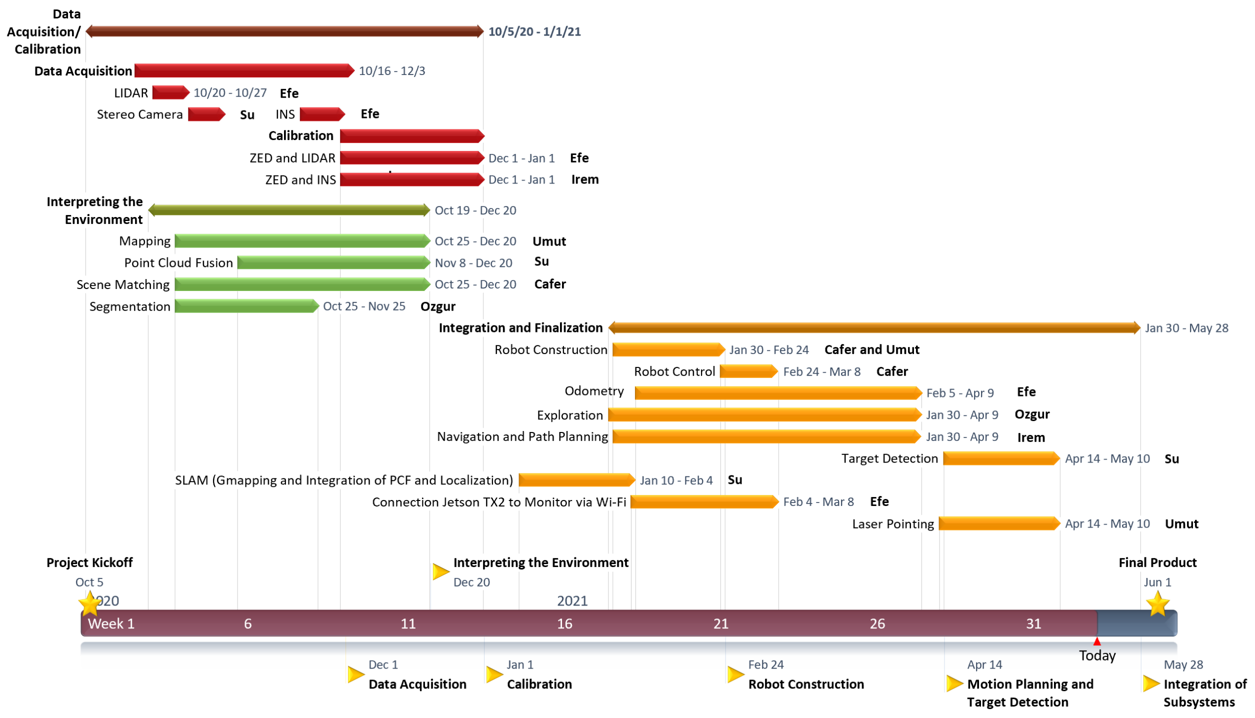


Figure 4: The timeline of the project.

5.2 Methods and Progress

5.2.1 Current State

The current state of our project is given in Figure 5. A larger version of the figure can be found in Appendix, Figure 3. Cafer dealt with technical and mechanical issues regarding the robot such as installation of new motors and laser mount. Umut implemented the closed feedback control loop for motors and the algorithms for pointing the laser to the target. Efe worked on fusing odometry data and enhanced the odometry performance. Efe, İrem and Su worked on navigation, path planning, and exploration. Efe and Özgür worked on enhancing scene matching. Su achieved target detection, computation of homography transforms and finding the required yaw/pitch angles for laser pointing. At the end all work packages are completed. The details of our progress will be explained under each work package separately.

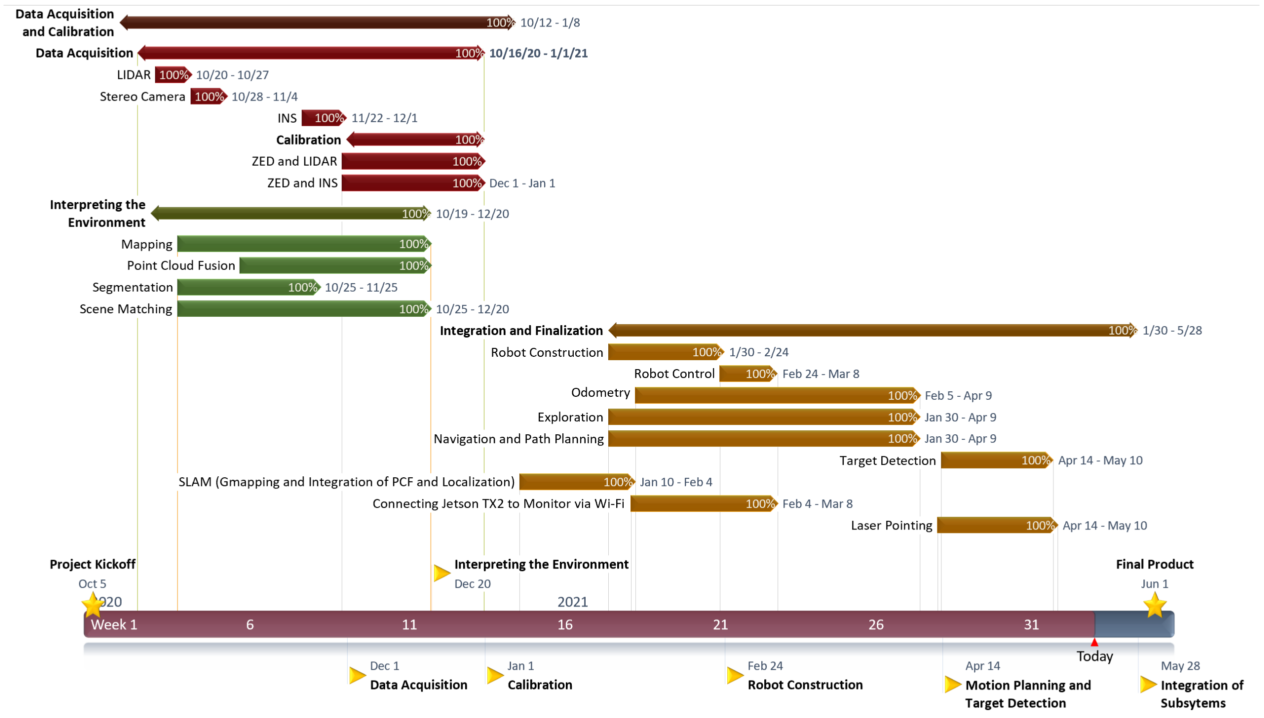


Figure 5: The current state of the project.

5.2.2 Data Acquisition from INS

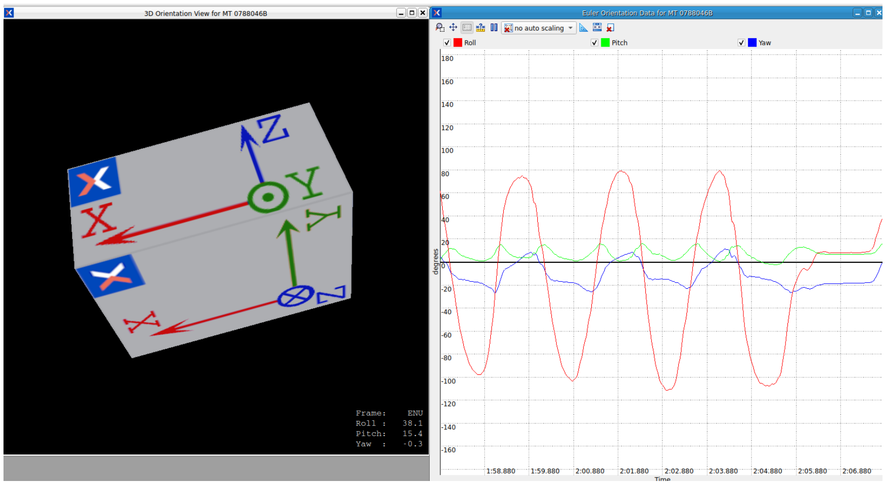
The INS module (XSens MTI-7-DK) provides the GPS information as well as the orientation and position data as calculated by an internal Inertial Measurement Unit (IMU). As the robot relies on the fusion of multiple data streams, the difference between reference frames of different modules is an integral part of our problem. The INS module measures all motion starting from an initial pose to be able to operate on a single reference frame with known position and orientation. The integrated microprocessor on our INS module can calculate this information using both IMU measurements and GPS signal and provides the translation and rotation matrices with respect to an initial global frame of reference. This data is then transferred to the central processor in order to be used within sensor calibration algorithms.

We are using an INS module with GPS because GPS signals include information about the global UTC time, which enables us to synchronize the whole system with respect to well-known time data. Even in case of GPS signal loss, which is quite probable during indoor operation, INS modules can continue to supply quite reliable time information thanks to their very low-drift internal clocks. Therefore, the INS module is also able to incessantly dictate the common time-reference that is used by all devices.

In summary, the INS module behaves as an anchor point for both spatial and temporal calibration of the rest of the system. As explained in the upcoming sections, all sensor data is converted into the temporal/spatial reference frame of IMU. The temporal conversions are simply time delays whereas the spatial conversions are rigid transformations.

a) Progress up to CM1: We decided to use which INS module to use. Irem did research about how to obtain data from the INS and perform spatial and temporal calibration of INS and ZED2.

b) Progress between CM1 and CM2: İrem obtained data from the INS module. The orientation data as the INS module is rotated mainly around its roll is shown in Figure 6.



```
Frame:    ENU
Roll :   38.1
Pitch:   15.4
Yaw  :   -0.3
```

Figure 6: INS data obtained from the sensor

5.2.3 Data Acquisition from Stereo Camera and Its Calibration with INS

The stereo camera module that we use, ZED2 from Stereo Labs, is designed to be compatible with the ROS environment. Therefore, there exists a wrapper package in the default ROS SDK, which lets us use the ZED2 stereo camera with ROS directly. This package internally handles the communication with the camera module via a USB interface and it outputs the left and right camera images, as well as a depth map, point cloud and pose information.

The native output of a stereo camera is in image format and all other outputs are obtained by processing the image data on the software level. Each of the two high-resolution cameras captures images (left and right) at the same time and transmit them to an external computer device for processing. The processing unit we use for this purpose is the Nvidia Jetson TX2 development kit and the manufacturer asserts that this device is compatible and tested with their product.

Since the relative position and relative orientation of the camera pairs in stereo camera systems are well-known and constant, simultaneously captured pairs of images can be used to represent the perceived environment in three dimensions. Likewise, ZED2 stereo cameras have a baseline length, a separation between two “eyes”, around 15 cm and the lines of vision of the cameras are parallel with each other. This configuration allows them to capture a high-resolution representation of the scene and estimate depth by comparing the displacement of pixels between the left and right images.

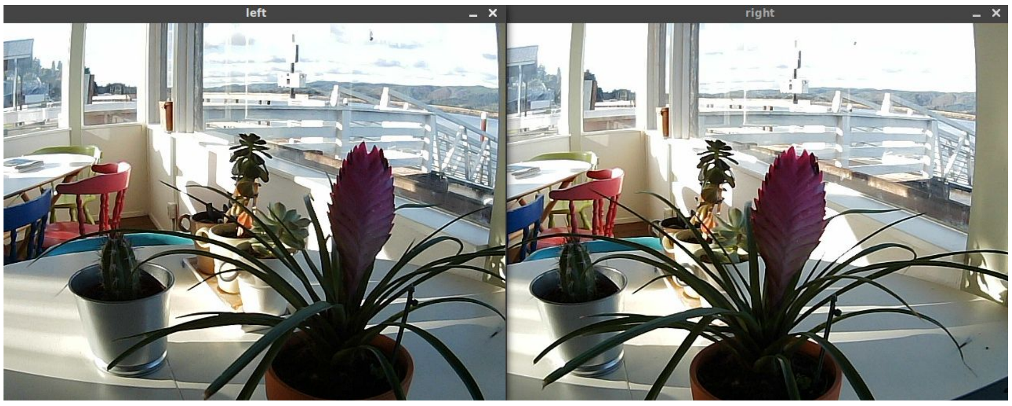


Figure 7: Images obtained from the left and right eyes of the stereo camera. [8]

There are several camera settings available for tuning a ZED2 camera module. Depending on the maximum available or reliable data transfer rate of the USB interface, the computational capacity of our main processing unit (Jetson TX2 board), and the environmental conditions, these settings can be manipulated. The most prominent ones are the resolution and frame rate settings because they combinedly determine the rate of the data produced. In order to ensure maximum transfer rate performance between the peripheral and the Jetson board, we are using a high-speed USB 3.0 connection. Apart from the image acquisition settings, the qualitative performance can also be enhanced by adjusting the settings of the onboard ISP (Image Signal Processor) that performs various image processing algorithms on the raw image captured by the dual image sensors.

After the data is preprocessed by the onboard ISP unit, it is transferred to the Jetson computer in various available image formats (rectified, unrectified and grayscale). The ZED2 Software Development Kit (SDK) collects this data and performs depth-sensing algorithms to extract the spatial information in three dimensions. The output can be represented either as a depth-map or a 3D point cloud. The depth map contains a distance value for each pixel in the image output, where the distance is calculated from the back of the left eye of the camera to the scene object. On the other hand, the point cloud is a collection of points in 3D space which represent the external surface of the scene and can contain color information. For our application, we decided to use point clouds since this type of data can be fused with the LIDAR data more easily.

As the whole robot is in motion during operation, the sequence of the images captured by the module have different reference frames. To account for temporal changes in the position of the camera, we need an estimate for the relative position and orientation of the system with respect to a global reference point. This is achieved with the help of an IMU (Inertial Measurement Unit) module which calculates the position and orientation of the camera at the time of image capturing.

One of the main challenges in translating and rotating the 3D point cloud, according to its relative position and orientation, is the temporal synchronization between the two streams of data obtained from the stereo camera and IMU. Since the amount of time delay between the sensing at the relevant module and the reception of data at the relevant script running inside the main computer might be different for these two channels (IMU and stereo camera), the transformation dictated by each IMU measurement cannot be directly applied to the most recent point cloud. In contrast, we need to transform the point clouds according to the IMU measurements made at the same time instant at which the ZED2 captures the associated image. Therefore, we need a precise way to measure and keep track of the image capture time with respect to a synchronized clock.

Another one of the challenges in calibrating the stereo camera data with IMU is related to the spatial distance between two modules. Since the two sensors, stereo camera and IMU, are located at different positions, their reference frames are not in exact match with each other. Thus, the rotation information measured by the IMU module does not match with the rotation observed according to the reference frame of the camera. In order to solve this problem, we need a way to estimate the difference between the two reference frames. Since the relative position/orientation of the modules may not be necessarily the same for each run of the robot, we have decided not to implement it based on hard-coded numerical parameters.

We use a toolbox called Kalibr [9] in order to solve both temporal and spatial synchronization problems. The calibration procedure is only performed when the robot starts to operate and done using an "Aprilgrid" calibration target which can be seen in Figure 8. The target is fixed at a position with moderate distance (1-2 meters) in front of the stereo camera and the camera-IMU system is moved in front of the target to excite all IMU axes while keeping the target in sight at all times. Firstly, the recording of the camera and the INS data is enabled to obtain a rosbag containing the grayscale image data from the left and right camera and IMU data from the INS. Secondly, the camera-IMU system is slowly rotated around its pitch, yaw, and roll 3 times and then moved up and down, side to side, and back and forth 3 times to excite all IMU cells. Thirdly, the camera calibration command of Kalibr is ran with four arguments: the recorded rosbag, the names of the recorded camera topics (raw grayscale images from left and right camera of ZED2), the model of the camera (pinhole), and the YAML file of the Aprilgrid calibration target which contains specifications, such as tag size and tag spacing. This command generates a new YAML file containing the camera intrinsic and extrinsic calibration parameters. Finally, the camera-IMU calibration command of Kalibr is ran with two arguments: the camera YAML file (generated in the previous step) and the IMU YAML file. The camera-IMU system becomes calibrated after completing these steps successfully.

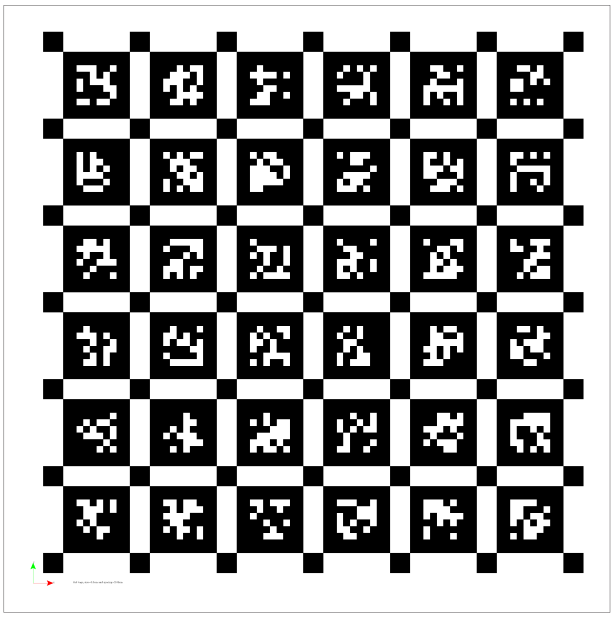


Figure 8: The Aprilgrid calibration target. [10]

a) Progress up to CM1: Su obtained the point cloud data from the stereo camera, which can be seen in Figure 9.

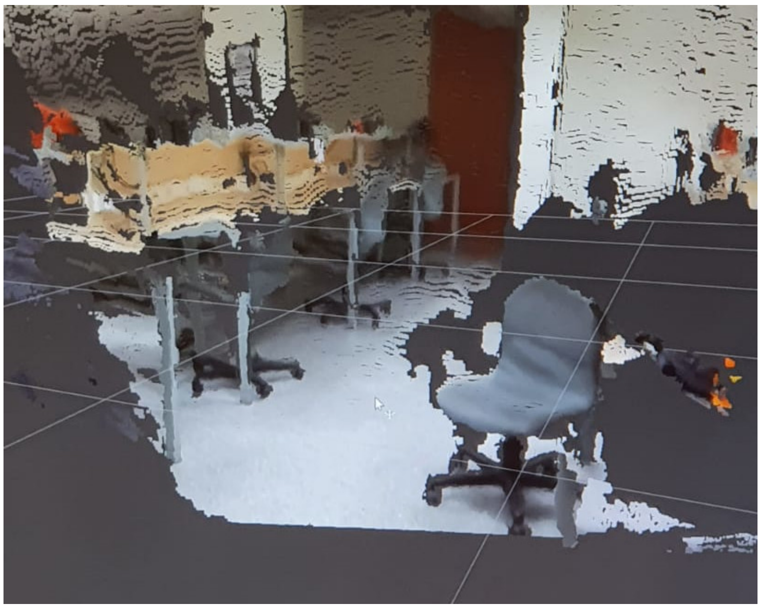


Figure 9: Point cloud data obtained from the stereo camera.

b) Progress between CM1 and CM2: İrem started working on the spatial calibration of INS and stereo camera by using the Kalibr [9] package. The locations of the right and left eye of the stereo camera are shown in Figure 10.

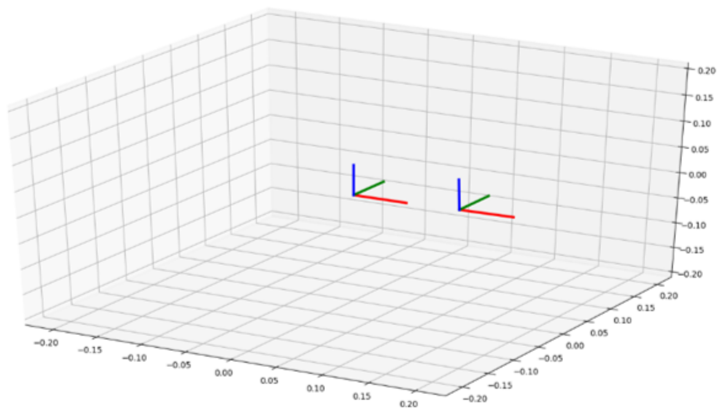


Figure 10: The locations of the right and left eye of the stereo camera.

c) Progress between CM2 and CM3: İrem completed the spatial calibration of INS and stereo camera. The calibration results (errors) are shown in Figure 11.

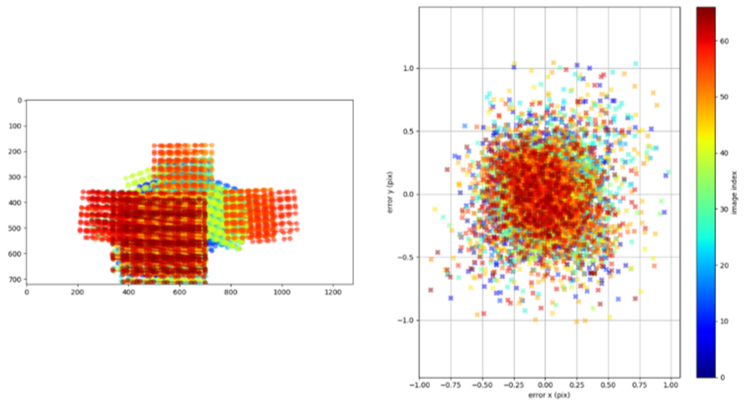


Figure 11: Calibration results.

5.2.4 Data Acquisition from LIDAR and Its Calibration with Stereo Camera

The LIDAR device that we use, Velodyne VLP-16, is also designed to be compatible with the ROS environment. Therefore, there exists a wrapper package in the default ROS SDK, which lets us use VLP-16 LIDAR with ROS directly. Similar to the stereo camera, this package internally handles the communication with the LIDAR via an ethernet interface and outputs the point cloud representation of the received data. The native output of LIDAR is in “laser scan” (sometimes called LSA) data format which consists of the raw measurements made by each laser/sensor pair and corresponding measurement timestamp. Then, this raw data is processed by the ROS Velodyne driver on the software level to generate the point cloud data. Similar to the stereo camera, the LIDAR might also suffer from spatial and temporal synchronization problems. Therefore, we need to synchronize its data with respect to the common reference frame achieved within the camera-IMU system.

Our LIDAR module can provide its data together with timestamps with respect to an internal clock, and this clock should be synchronized to an external clock with the help of various interfaces. The internal clock consists of two parts called sub-second, ToH (Top of Hour) counters. As the name suggests, the sub-second counter rollovers every second and counts in microsecond precision. It is zeroed at each rising edge of an externally provided Pulse Per Second (PPS) signal. Since the internal clock has sufficiently low time drift, it is adequate to supply the synchronization signal once per second. On the other hand, ToH counter measures the "Minutes and Seconds" part of the internal clock and its synchronization is done through an NMEA message. As LIDARs are usually designed to be used in conjunction with global geolocalization, their synchronization interfaces generally support the acquisition of NMEA messages. This format includes the information about GPS fix time (in at least second-precision), latitude/longitude, lateral speed, and true heading as observed or calculated by an external GPS receiver module. Using the GPS fix time information, LIDAR updates its ToH counter.

Therefore, we are required to supply both a PPS signal and an NMEA message. As our INS module is designated to be the central reference frame for all other peripherals, we should use its output data to provide these signals. However, some INS modules do not support NMEA message output, and their output needs to be converted into an NMEA message. If it turns out to be the case, the Jetson board acts as an intermediary between INS and LIDAR in order to convert the output of INS to NMEA message format.

The last part of the synchronization process is to integrate the LIDAR module into the common spatial reference frame achieved within the camera-IMU system. This can be done by calibrating the LIDAR either with the INS or with the stereo camera. However, the literature mostly includes previous practices of LIDAR-camera calibration rather than LIDAR-INS calibrations. Therefore, we follow the common practice and optically calibrate the LIDAR-camera pair. We use the ROS module named "LIDAR-Camera Calibration using 3D-3D Point correspondences" [11] that follows a point cloud matching approach in finding the translation matrix. The module requires the setting in Figure 13. We have two cardboards with Aruco markers on their corner. LIDAR can detect the cardboards, whereas ZED2 can detect the Aruco markers, as it has the image information. The algorithm requires premeasurements about the alignment and the size of the Aruco markers, as well as the size of the Aruco markers. By using the point cloud data from both of the sensors and the given measurements, it finds the transformation matrix between the reference frames of ZED2 and LIDAR. This matrix is used to link the data obtained from the sensors. Calibration is performed once at the start of each operation cycle.

a) Progress up to CM1: Efe obtained the point cloud data from the LIDAR, which can be seen in Figure 12. Furthermore, Efe did research about how to perform spatial and temporal calibration of LIDAR and ZED2 and started working on it.

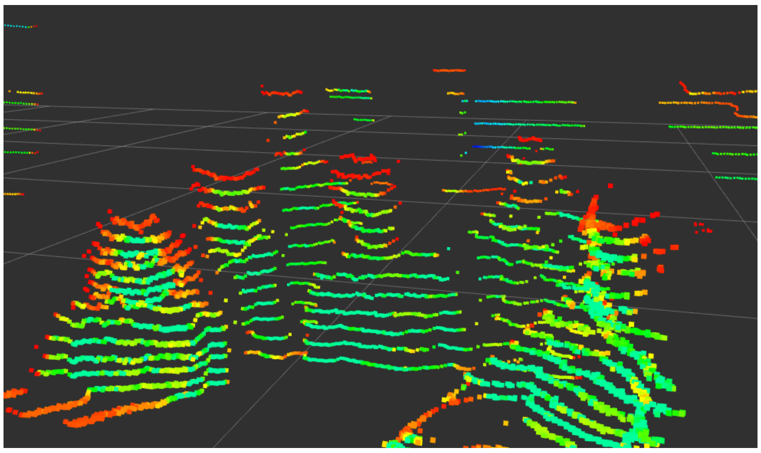


Figure 12: Point cloud data from the LIDAR.

b) Progress between CM1 and CM2: Efe worked on the spatial calibration of LIDAR and stereo camera. The setup used for the calibration is shown in Figure 13. The points detected on the edges of the cardboard are shown in Figure 14. In order to detect the edges of each cardboard by LIDAR, they are placed diagonally such that the horizontal laser scans of the LIDAR maximally overlap with the edges. Since spatial calibration can be performed better if we use multiple sources of data with small to no noise correlation, our set-up uses two card-boards separated by some distance.

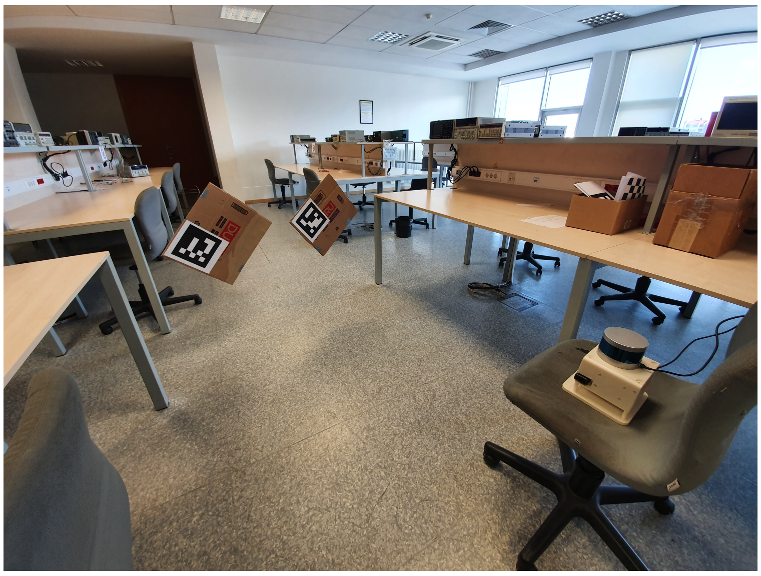


Figure 13: Setting for spatial calibration of stereo camera and LIDAR.

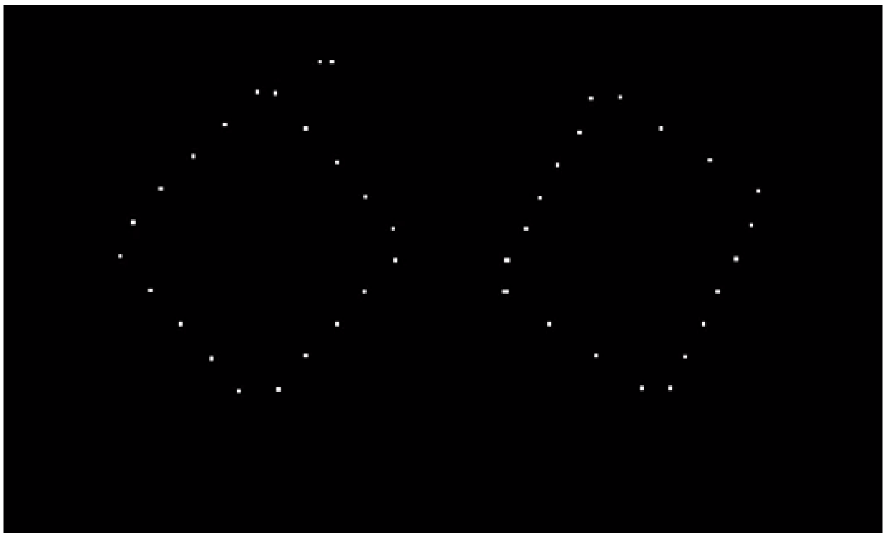


Figure 14: Points detected on the edges of the cardboard.

5.2.5 Point Cloud Fusion (PCF)

For this step, we aim to fuse the point cloud data generated continuously by ZED2 and LIDAR to obtain a new point cloud stream. However, the point clouds that were fused have different reference frames. After we obtain a common spatial frame by calibrating ZED2 and LIDAR as described in the previous section, the point cloud data obtained from both sensors overlap. An example for two point clouds before and after being calibrated to the same reference frame can be found in Figure 15, which summarizes the process explained in the previous section. To easily process the point cloud data, we use the Point Cloud Library (PCL), which is an open-source library that is used for image and point cloud processing. It contains state-of-the-art algorithms for various applications, and it can be used through ROS [12]. PCL provides functions such as downsampling, filtering, and object recognition. To use PCL methods on our point clouds, we convert the data published by ROS in the form of "sensor_msgs/PointCloud2" to "pcl::PCLPointCloud2" first. Then, we convert "pcl::PCLPointCloud2" data to "pcl::PointCloud⟨pcl::PointXYZ⟩" for manipulations.

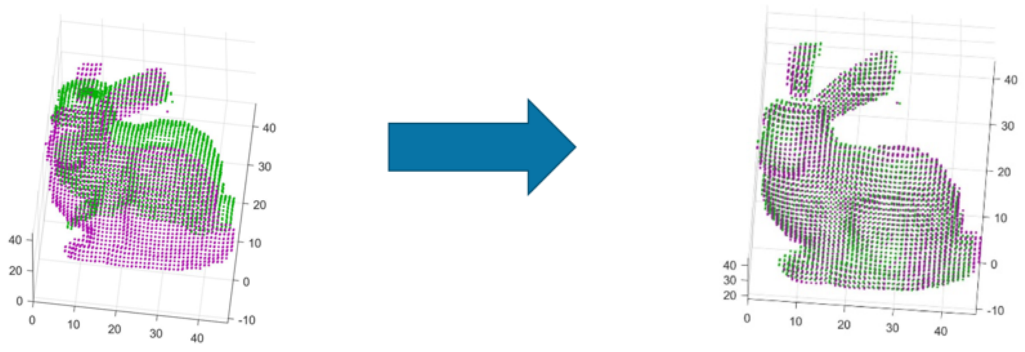


Figure 15: Two different point cloud data, calibrated [13]

With fusion, we aim to obtain a result that contains more information compared to its constituents. Therefore, our goal is to combine the strengths of the data obtained from the two sensors to get a much more precise and informative point cloud. As ZED2 is better at depth-sensing in short-range and provides a much more dense point cloud compared to LIDAR, it is sensible to utilize mostly its point cloud data for short distances. For distances larger than 3.0 m, LIDAR provides a more stable and accurate point cloud compared to ZED2, as it is designed for long ranges.

For this fusion algorithm, we add up the point clouds. As they are already spatially calibrated, they fit. However, to decrease the memory burden, we also need to downsample the resulting point cloud. For this purpose, we use the ApproximateVoxelGrid of PCL. This library creates 3D voxel data from the point cloud. The points present in each voxel are approximated with their centroid to downsample the data. This approach provides a small-sized cloud that is able to represent the surroundings accurately. In the end, the resulting point cloud is transformed to "sensor_msgs/PointCloud2" and is published from the point cloud fusion node with a frequency of around 3 Hz.

We should note that we do not use all the point cloud data obtained from the sensors because the publishing frequency of the sensors is higher than the speed of the algorithm. ZED2 publishes point clouds with a frequency of 5 Hz, whereas LIDAR publishes them with 15 Hz. However, as the speed of our robot is not high, it is enough to have a fused point cloud publishing frequency higher than 1 Hz, which is already achieved. Lastly, after we obtain the fused point cloud, we merge it with the existing data by using the SLAM algorithms to obtain a 2D map of the environment while steering.

a) Progress up to CM1: Su did research about how to perform point cloud fusion and started working on it.

b) Progress between CM1 and CM2: Su performed point cloud fusion between the data that comes from the stereo camera and LIDAR. The calibrated point clouds of stereo camera and LIDAR are shown in Figure 16.

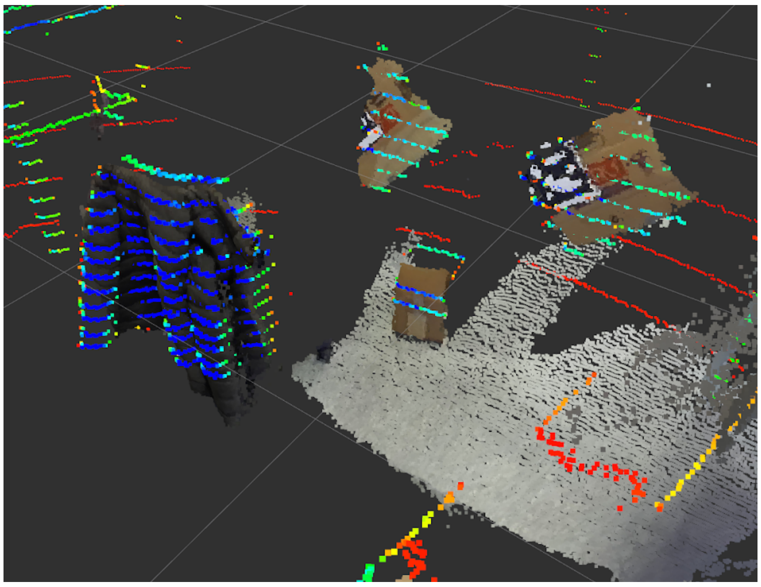


Figure 16: Calibrated point clouds of stereo camera and LIDAR.

5.2.6 Segmentation

Segmentation is the process of delineating the given scene when multiple objects are to be recognized. Since our robot is designed to operate in an indoor space, segmentation plays a crucial role in the decision-making process of the rover. We should distinguish obstacles from suitable paths for correct motion planning. The results of the segmentation can be used to determine the phase and direction of the rover to avoid any confrontation with an obstacle. The existing objects in the path are classified in terms of their features. For this sake, we utilize a deep-learning-based inference model. The image acquired from the ZED2 stereo camera is further processed in this segmentation model. During the inference time, the segmentation model receives the left camera image as the input and produces a segmentation map according to the classified objects.

Our model utilizes a residually connected encoder-decoder architecture to extract the relevant features from a two-dimensional image to perform segmentation. We carried out experiments with multiple neural network models and decided to utilize the ResNet-18 model since it provides high accuracy and fast response. The residual connections in the neural network enable the construction of deeper architectures and avoid gradient vanishing problem since the problem of predicting the output from scratch is mitigated with the help of prior information provided at the output of the residual layer. Moreover, the convolutional layers provide a high capability for feature extraction with fewer parameters. Therefore, the encoder and decoder of the segmentation model contain cascaded convolutional neural networks with 3x3 filters preceded by batch normalization. The activation function is determined as Leaky Rectified Linear Units (Leaky ReLU) to achieve nonlinear classification. The pre-processing and post-processing steps adjust the size according to the size of the input obtained from the stereo camera and produce an output with the same size as the input. Furthermore, the transfer learning approach is employed to determine the weights of the parameters on the PyTorch environment. The neural network model trained in a supervised manner is capable of producing accurate segmentation maps, however, lacks fast response due to high computational complexity. The TensorRT module provided by Nvidia optimizes the network architecture and helps to deploy the segmentation model on real-time applications. After the implementation of optimizations, TensorRT chooses kernels that are unique for the program in order to maximize performance on the Jetson TX2 board. Thanks to this backend algorithm, the implementation of the inference model is accelerated.

a) Progress up to CM1: Özgür performed segmentation by using ResNet 101 residually connected neural network model. The segmentation algorithm was tested in the EEE 493 Lab and the performance of the results was benchmarked. It took around 5.5 seconds to perform segmentation with this model. The results are shown in Figure 17 which demonstrate high accuracy. The depth of the network can be reduced and different network architectures can be tested based on the time constraints.

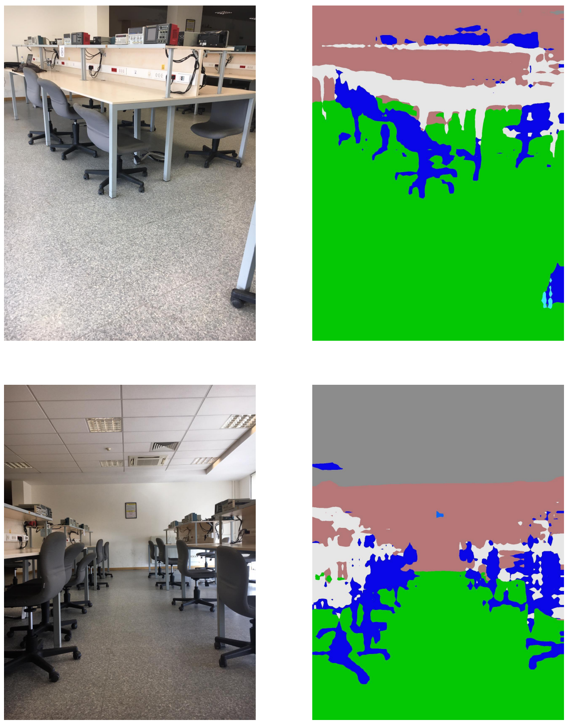


Figure 17: Initial trials for automatic delineation of the EEE 493 lab indoor scene.

b) Progress between CM1 and CM2: Özgür downgraded the segmentation model to fulfill the time requirements. He utilized a ResNet-18 model, which is trained with the SUN RGB-D dataset commonly available on the web. He performed the experiments with a wide range of manually annotated labels available with the dataset in order to avoid underfitting. In particular, the training procedure included 20 types of labels as ground truth references including wall, floor, ceiling, table, door, and person. Efe and Özgür integrated the segmentation model with the main system constructed on Jetson TX2. They also achieved high throughput for the segmentation system by performing the experiments on the Jetson TX2 GPU. The inference rate for the segmentation system is 10 FPS in the standalone mode and 5 FPS while Jetson GPU is heavily used. The results of the segmentation model is shown in Figure 18.

Figure 18: A representative image from our segmentation model.

5.2.7 Odometry

The core of our project is to achieve autonomous navigation. To this end, we are required to obtain the odometry data of the robot, which is the estimate of the differential position with respect to a reference point. In the most robotic applications, this reference point is selected as the point where robot starts to operate at each run. Since our robot is not also required to position itself with respect to a global reference frame, the origin for the odometry data is the starting point.

In order to achieve this goal, we utilize multiple sources of odometry data and fuse them to obtain a less noisy and more robust estimate for the odometry. The fusion is done with the help of a Kalman filter where the states represent the acceleration, velocity, and position in all 6-axes (x,y,z, pitch, yaw, and roll). In order to be able to keep track of all 6 states stably, we are required to use data sources that can provide both position and acceleration estimates. Even though acceleration data can be integrated twice to obtain a position estimate, or position data can be differentiated twice to obtain an acceleration estimate; this approach will result in erroneous results due to accumulation of errors.

Keeping this possible problem in mind, we determine two different sources of odometry that is able to stably provide the required data to the Kalman filter. The first one is the INS which can provide low-noise information about the acceleration of our system. The second one is the stereo camera (ZED2) which generates odometry data with its internal IMU. This data is strengthened with visual odometry, where the camera perceives the changes in motion by stacking its images and keeping track of the differences.

However, we observed that the performance was not sufficient despite using two sources of information because of the accumulation of errors. In particular, the odometry data was wobbling in the y and z axis, which correspond to left-right and up-down translational motion respectively. Since our robot cannot move in both of these directions, we decided not to consider the outputs of the sensors and to input them as zero to the Kalman filter. Similarly, our robot cannot pitch or roll, therefore we input them as zero to the Kalman filter as well. Furthermore, we have observed that the yaw data obtained from the INS was prone to errors, therefore we used the output of the stereo camera only. After these changes, the wobbling stopped and the odometry data became sufficient for exploration.

a) Progress between CM1 and CM2: Efe did research about odometry fusion by using Kalman filtering. He also found a suitable package to handle odometry fusion using Kalman filtering and implemented the initial version for our robot.

b) Progress between CM2 and CM3: Efe made changes to the configuration of the Kalman filter and changed the frame transforms between the input and output odometry data in accordance with the project structure.

c) Progress between CM3 and CM4: Su and Efe improved the odometry by considering only the axes of interest, as described above.

5.2.8 Simultaneous Localization and Mapping (SLAM)

Given the odometry data and the laser scan obtained from the sensors, the next goal is to construct a map of the environment and to position the robot on that map. These data are fed into the gmapping package of ROS, which is a wrapper for the open-source GMapping library of OpenSLAM [14]. The GMapping package uses the Rao-Blackwellized particle filter technique to perform simultaneous localization and mapping. In this technique, each particle holds a map of the environment and the goal is to minimize the number of particles. This is done by adaptive techniques considering the movement of the robot and the most recent observation of the environment. This approach helps to decrease the uncertainty in the pose of the robot [15]. Subsequently, the gmapping package publishes a 2D occupancy grid, which is a floor-plan-like map of the environment. The laser scan input required by GMapping is obtained with the package “pointcloud_to_ laserscan” [16]. This package converts the 3D fused point cloud data to 2D laser scan data, which is then used to create the 2D map.

a) Progress up to CM1: Umut did research about existing packages for spatial mapping and ran a tutorial on the robot simulator platform Gazebo with a prepared environment and robot. A predefined robot is controlled in a simulated environment while performing mapping and localization.

b) Progress between CM1 and CM2: Umut found a suitable bag file and used the gmapping package on Jetson TX2 board. The resulting map is shown in Figure 19.

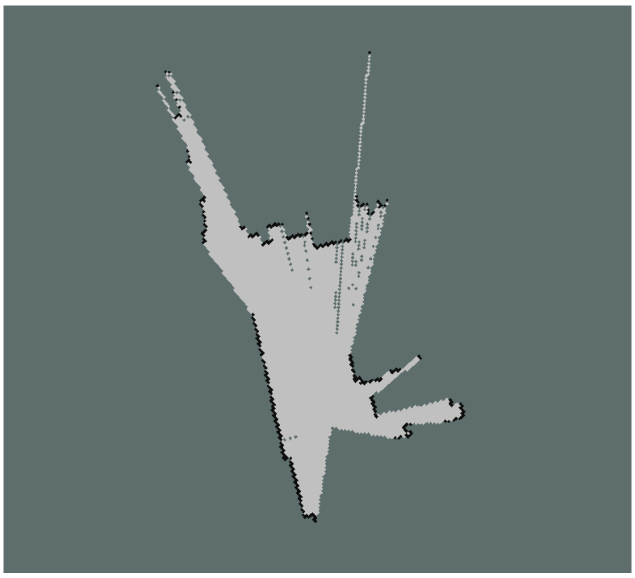


Figure 19: Spatial mapping results.

c) Progress between CM2 and CM3: Su implemented the system to convert point cloud data to laser scan data and therefore launched GMapping. An example laser scan on top of a map constructed in the EEE-102 laboratory and displayed with RViz can be found in Figure 20.

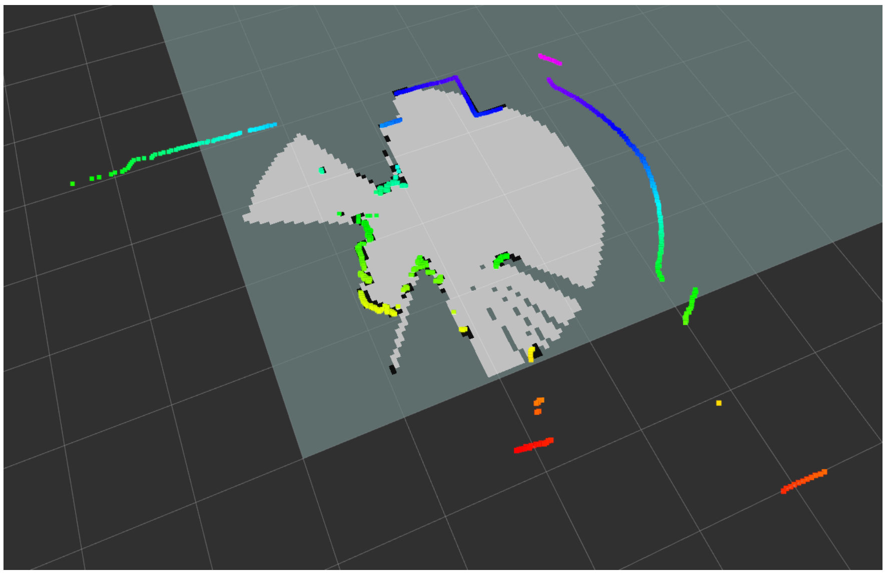


Figure 20: An example laser scan on top of a map.

5.2.9 Scene Matching

Scene matching is the process of matching two images that have different illuminations or scales, or that are taken from different angles. This task is necessary for our project because the robot will have a picture of the scene with a target, and it needs to recognize that scene while moving around. However, the given picture and what the robot sees differ by perspective, brightness, etc. Therefore, we need scene matching algorithms for the robot to recognize the scene. After this step, the robot is able to look for the target in that specific area.

To match the image of the scene with the image seen by the robot, features in the two images are detected first. Then, these features are compared and matched if they have similarities higher than the threshold. Features can be thought of as the key-points in an image. A corner of an object, boundaries, interestingly shaped patterns are examples of features and can be detected by filters. On the other hand, flat surfaces, for instance, do not contain any features.

In our project, feature matching is performed by using OpenCV library [17]. OpenCV has various feature detection algorithms which have their pros and cons. Currently, we tested commonly used scene matching algorithms provided by the OpenCV library and we obtained the best trade off between the accuracy and the speed with ORB (Oriented FAST and Rotated BRIEF) algorithm. The main advantage of ORB is its speed, which makes it suitable for real-time applications like our project. Additionally, ORB displays superior trade off between the computational complexity and the accuracy of the detection algorithm. The ORB detection algorithm both relies on variance and correlation of the distribution of the spatial signals. Furthermore, it utilizes orientation components unlike its ancestor FAST algorithm which makes ORB more robust than FAST. The ORB algorithm also makes use of rotation aware version of BRIEF algorithm. In this way, the ORB algorithm receives smoother input patches denoised by the BRIEF algorithm and constructs a steered version of them to detect the rotation changes. During the validation procedure, we observed that the ORB algorithm outperforms FAST and BRIEF algorithms both in computational complexity and the robustness against rotational variances. More specifically, run time of ORB algorithm is less than 50 ms which is the least run time score when compared with other existing solutions. This algorithm also captures the same scene when it takes the rotated version of the desired scene as an input.

a) Progress up to CM1: Cafer started testing scene matching algorithms on existing datasets whose results are shown in Figure 21. The algorithm is able to match the objects in two photos of a scene which are taken from different angles.

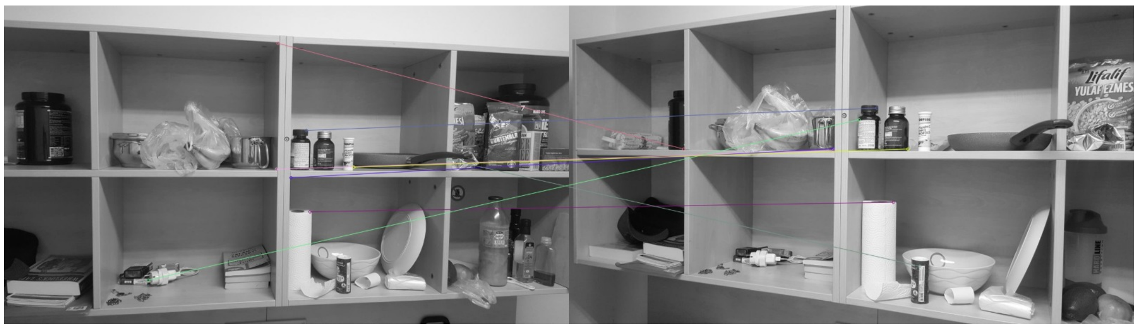


Figure 21: Scene matching results.

b) Progress between CM1 and CM2: Cafer tested scene matching algorithms in different conditions whose results are shown in Figure 22.

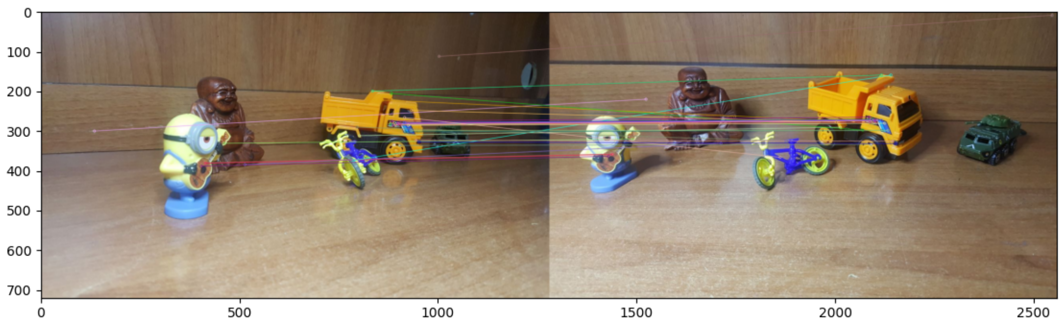


Figure 22: Scene matching results.

c) Progress between CM3 and CM4: Ozgur and Efe investigated the OpenCV library detectors and compared the accuracy and reliability of these detectors. For this purpose, they designed different testing environments and placed realistic targets to these environments. They found out that the ORB algorithm outperforms other algorithms in terms of computational complexity and accuracy. This algorithm produces better feature matching results which are robust to angle change and object type change. Therefore, they decided to integrate this algorithm into the general framework for the decision making process of the robot.

5.2.10 Building and Remote Controlling the Robot

To perform the functions for our desired goals, we needed a rover-like robot capable of moving in indoor and mild outdoor environments. As the goals of our project are mainly software-based solutions, we were encouraged, by our company mentors, to use a pre-built robot platform for our project. However, the robot platforms and pre-built robots sold in the market usually target a hobby-grade consumer base that isn't capable of supporting the hardware we are using. Also, the weight and shape requirements due to our hardware were not compatible with the pre-built platforms we investigated. Therefore, we had built the robot ourselves from scratch.

The main constraints that we considered while designing the robot were weight and shape. It is necessary for the robot to handle the weight of our hardware and battery easily, and to have enough space to carry all the components. The battery makes up a large portion of the robot's weight. Hence, the main constraint while designing our robot was power use. If stronger motors, which are able to carry more load, were to be used, they would consume more power. Hence, we would need a bigger battery which would weigh more. There is a power-weight trade-off that we wanted to balance. For this, we decided to use two motors instead of four which is widely used in skid steering as an easy and robust solution. For steering with two motors, we investigated designs with idle wheels. However, we couldn't find an idler wheel that met our quality requirements in the market. Hence, we continued to investigate different steering designs. Initially, we used omni wheels as idler wheels to make the robot move freely. However, we had to change the wheels as they turned out to be too small and made the robot touch the ground from time to time. There were not larger omni wheels available in the market, thus we used the same wheels which are attached to the motors as idler wheels in the final design. The use of normal wheels also allowed the robot to move freely and did not create a problem.

Moreover, another main consideration in our robot design was modularity. While our first objective was to make a robot that can support our existing hardware, we had to be ready for changes in the project, such as a possible need for a second Jetson board for computational power or a change of sensors' places. Hence, we needed to be adaptable as we went along. For this, we designed our robot to be composed of adjustable layers mounted to an aluminum frame, which are attached to our support beams. For instance, if we need to add a second board to the design, we can simply add another layer to the robot frame. Finally, while designing the robot, we factored in the shape of the popular robot platforms, as we needed to build a robot similar to them to be able to use existing software libraries more conveniently.

Since our aim is to have a mobile robot as our final product, it should have a power source mounted to it, which supplies power for at least 20 minutes. For our case, we should supply power to the Jetson TX2 board, LIDAR, and DC motors. Arduino and the motor controller draw their power from the Jetson board. To supply power, we considered different kinds of battery solutions such as lead-acid/SLA, lithium-ion, lithium polymer and nickel-metal hydride batteries. At first, we chose lead-acid batteries only as they are the cheapest option for the same amount of power. They can output sufficient current, are easy to charge, and easy to maintain as they don't easily breakdown. The main reason they are not widely used in robot projects is their heavier weight compared to their alternatives, but we had sufficient capacity to carry their load. However, while using lead-acid batteries only, we encountered problems during the operation of the robot. Thinking that the power output was not sufficient, we used a setup with two lead-acid batteries in parallel, but we still had problems with the Jetson board shutting down during operation. After our investigations, we concluded that the lead-acid battery could not supply the necessary current as fast as needed during the take off of the robot, resulting in a momentarily current drop which makes the Jetson board shut itself down due to safety as it demands a stable power supply. Hence, we moved on to Li-Po batteries as they have high discharge rates. At the end of our trials, we concluded that a setup where a Li-Po is supplying the Jetson board and the LIDAR, and a lead-acid battery is supplying the motors was a good solution, as the Li-Po can provide a stable current and the lead-acid battery has the capacity to power the motors for long periods of time.

In order to control the robot before implementing the motion algorithms, we connected a joystick to the Jetson board via Bluetooth. Then, we implemented a package to perform the motion commands coming from the joystick with the robot.

a) Progress up to CM1: Cafer and Umut did research about how to build the robot and which materials to use in the process.

b) Progress between CM1 and CM2: Cafer and Umut decided on the first design of the robot and drew schemes using TinkerCad which are shown in Figure 23. They also decided on which components to use and purchased them.

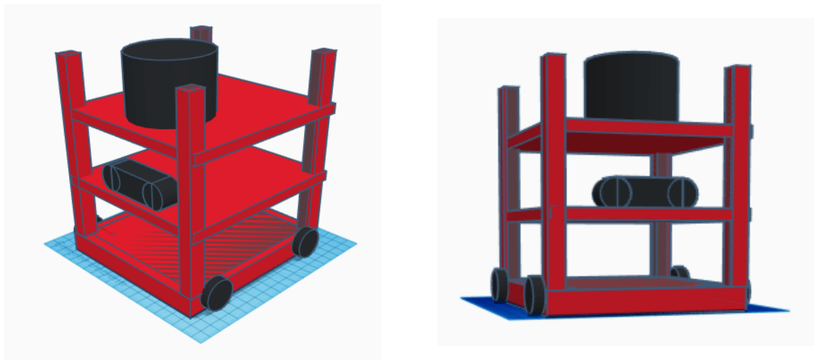


Figure 23: Sketches of robot design.

c) Progress between CM2 and CM3: Cafer and Umut built the robot from scratch based on the initial design between CM2 and CM3. First, the design in Figure 23 was updated slightly and sketched by professional computer-aided design (CAD) software, SolidWorks. Then, the design was arranged by a professional engineer to make it ready for production. The final design of the parts can be seen in Figure 24. For the chassis, aluminum sheets with 2mm thickness were used. The aluminum sheets were drilled and bent by workers. Aluminum has the advantage of being light and still durable enough to carry the required weight. This is especially important since the weight is an important constraint for the project as explained before. Moreover, the design has a hole pattern on it for two purposes. The holes make the chassis lighter and they provide places to assemble components by using bolts and nuts. The hole pattern is uniform across all parts which provides flexibility while changing the places of chassis parts and components. Also, the pattern is consistent with the standard motor frame hole pattern.

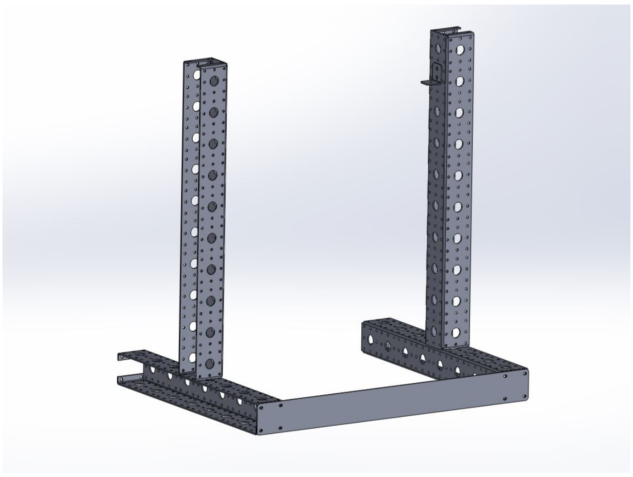


Figure 24: Final design of chassis parts.

In order to place the sensors, batteries and other materials, shelves were placed on the chassis. The material for the shelves was selected as acrylic glass. The biggest advantage of acrylic glass is being an insulator as electronic components are placed on it. Also, acrylic glass is easy to drill and cut by using a hand drill so that we were able to place the components of the robot at the desired places by drilling holes to place bolts. The final look of the robot with all components attached to the shelves is given in Figure 25. As it can be seen, there are 2 shelves on the robot, one at the bottom carrying the batteries and the motor driver, and one in the middle carrying the Jetson board, stereo camera, INS, Arduino, and USB hub. LIDAR is placed at the top, on the aluminum sheet between two beams.

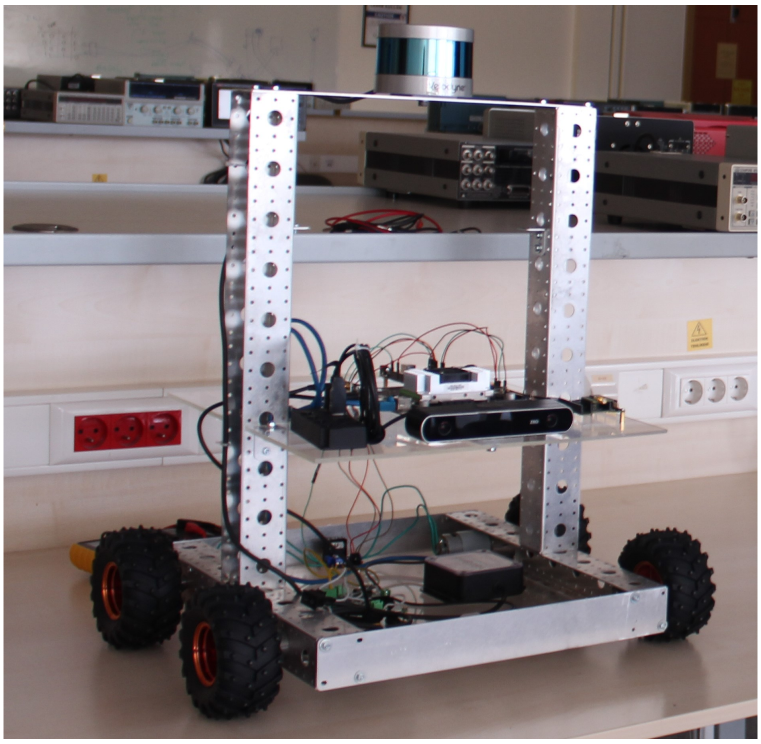


Figure 25: Final look of the robot with all components attached.

d) Progress between CM3 and CM4: Two brushed DC motors are replaced with four encoded DC motors. Four motors give the robot better traction because the total torc of the motors is increased. It also provides greater maneuverability. The robot's base is now in the middle of the robot and it can turn around this axis. The newly mounted motors are also faster. The motor encoders generate a square wave whose frequency is proportional to the angular velocity of the DC motor. Counting the rising edges of this signal allows us to find the frequency of the encoder signal. The encoder signals are connected to interrupt pins of the Arduino microcontroller which counts the rising edges of the signal. Before CM4 we were using Arduino Uno but to use four encoded motors we changed it to Arudino Mega microcontroller because Arduino Uno didn't have enough interrupt pins. We drilled new holes in the robots platform to place the Arduino Mega. Counting the rising edges of the signals, we can calculate the velocity of each motor in the unit of rounds per minute. Because we need to measure the velocity of this signal long enough to calculate velocity information, the velocity information updates every 500 milliseconds. This may be improved using better technologies to measure the velocity but our encoders introduce a bottleneck. We have built a serial communication system between the Jetson board and the Arduino microcontroller. For the movement planned by the movebase, the required velocity commands are sent to Arduino in real-time through the serial communication. The measured velocities of the motors are also provided to the Jetson board through this communication link. To control the four encoded DC motors Arduino Mega microcontroller takes the desired velocities for each motor from the Jetson board. It measures the actual velocities of each motor. As a result it outputs a pulse with modulated control signal to the motor driver. The output signal is calculated with a PID controller which takes the actual velocity as the feedback to achieve the desired velocity input. The proportional, integral and derivative control parameters are tuned for the robot with trial and error. We set them to good enough parameters for the movement of the robot.

The height of the motor is kept the same contrary to what we have planned. We were worried that because the Lidar was standing that high the vibrations in the robot could affect the performance of the Lidar. However, we observed that we didn't had any significant problems in that way. Hence, we decided to keep the height of the robot as is.

5.2.11 Path Planning and Exploration

Path planning and exploration are the most crucial yet complex tasks, as they build on the previous work described above. We ensured that all the components in the system are working in harmony in order to obtain autonomous motion. Path planning takes a goal location as its input. Then, it uses information obtained from the sensors to find a path that is free of obstacles. On the other hand, exploration uses the map of the environment and the robot's location data from the beginning of its operation cycle to suggest previously unvisited places to explore. We use "move_base" [18] and "explore_lite" [19] packages for path planning and exploration respectively, which are both ROS packages.

"move_base" has two planners and two costmaps, one local and one global for each. The cost-maps hold information about the location of the obstacles around the robot. The obstacles are detected with data coming in two formats: a laser scan and a point cloud. For our system, laser scan data is obtained from changing the format of the fused point cloud data as described above. Additionally, we use the point cloud data of ZED2. The incoming data is used for marking or clearing, which correspond to inserting or removing obstacle information into the costmap, respectively. The cells in the cost-map can be in one of the following three states: occupied, free, or unknown. We label a cell as occupied when there is an obstacle in that location, free when there is not any, and unknown if we have not seen that location yet. The cost of the cell depends on the state that it is in. Each map is updated with a frequency of 5 Hz. The localization of the robot in the map is achieved with GMapping. Lastly, global and local costmaps are used to accomplish global and local navigation goals respectively. Therefore, the local costmap is a smaller but more detailed map of the surroundings, whereas the global map is larger and roughly sketched.

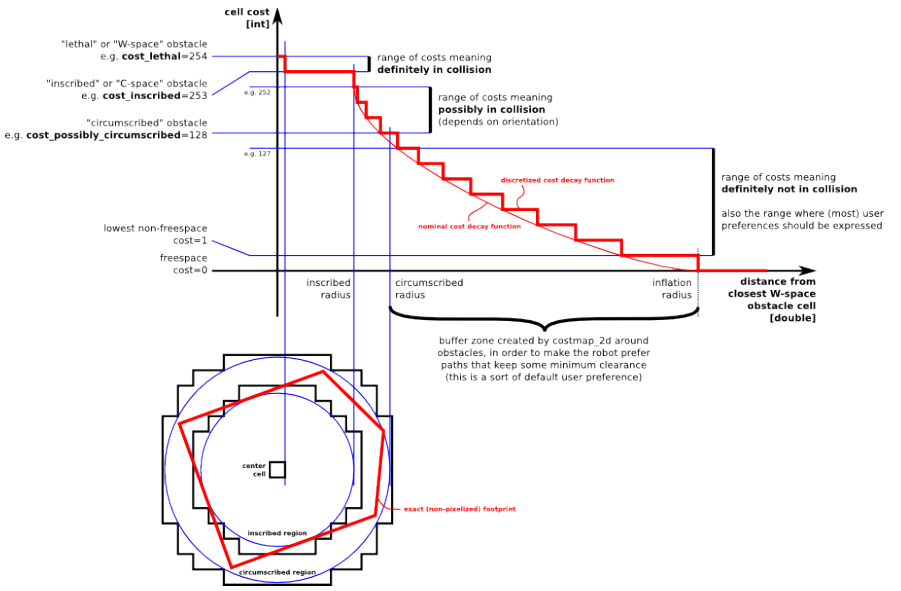

cell cost [int]
"lethal" or "W-space" obstacle e.g. cost_lethal=254
"inscribed" or "C-space" obstacle e.g. cost_inscribed=253
"circumscribed" obstacle e.g. cost_possibly_circumscribed=128
e.g. 252
e.g. 127
lowest non-freespace cost=1
freespace cost=0
range of costs meaning definitely in collision
range of costs meaning possibly in collision (depends on orientation)
range of costs meaning definitely not in collision
also the range where (most) user preferences should be expressed
discretized cost decay function
nominal cost decay function
inscribed radius
circumscribed radius
inflation radius
distance from closest W-space obstacle cell [double]
buffer zone created by costmap_2d around obstacles, in order to make the robot prefer paths that keep some minimum clearance (this is a sort of default user preference)
center cell
inscribed region
circumscribed region
exact (non-pixelized) footprint

Figure 26: The exponential decay of the cost from an occupied state to a free state, and the corresponding inflation radius.

Some important parameters associated with the configuration of the costmaps are the following: width and height of the map, footprint, inflation radius, obstacle range and raytrace range. For the global costmap, the width and height are both set to 10 m, whereas for the local costmap, they are set to 6 m. The footprint corresponds to the dimensions of the robot. The center is assumed to be (0, 0), and the user enters the values by considering the coordinates of the wheels by taking the center of the robot as the origin. Inflation radius is a crucial parameter as it determines the costs associated with certain cells. The costs of the cells have the following structure: an occupied cell has a high cost (e.g. 128), whereas the free space surrounding the obstacle has a cost of zero. The cost decreases in an exponential fashion starting from the occupied space, and converges to zero in the free space. The inflation radius determines the decay length from maximum cost to minimum cost. A scheme of this process can be seen in Figure 26. The obstacle range corresponds to the maximum range for which a sensor reading results in inserting an obstacle to the map. It is taken to be 2.5 m by default. Lastly, raytrace range corresponds to the maximum range for which we detect free space.

Given the local and global costmaps, the corresponding local and global planners aim to find the optimal path to a given navigation goal. With its different cost functions, it can aim to minimize the distance to the obstacles, the time the robot spends on going backwards, etc. The global planner tries to find the optimal route to the given navigation goal by taking the large-scale structure of the environment into consideration, whereas the local planner aims for finding the optimal path given the immediate surroundings. After finding a route, “move_base” outputs the necessary angular and linear velocities to follow it. During this process, it also updates its outputs with respect to the resulting trajectory of the robot. The minimum and maximum possible velocities are determined by the user. For our system, the minimum and maximum linear velocities are in the range 0.25 m/s and 0.4 m/s. The reason for such a small range is due to the large weight of the robot: if the minimum velocity is too small, the robot cannot move, and it cannot go with a speed larger than 0.4 m/s.

“explore_lite” provides greedy frontier-based exploration in which frontier means a border separating a known territory from an unknown one. The robot explores its environment in a greedy fashion until there are no unvisited places left (no frontiers left to explore) when the “explore_lite” node actively runs. As this package uses “move_base” for navigation, it is important to make sure that the “move_base” node is configured and initialized correctly before running the “explore_lite” node. This package requires a map in order to look for frontiers and plan exploration so it either obtains a costmap published by “move_base” or a map constructed by simultaneous localization and mapping (SLAM) algorithm. For this purpose, it is subscribed to two messages which are OccupanyGrid and OccupanyGridUpdate, which correspond to a map from SLAM or a costmap from “move_base” and incremental updates on map or costmap, respectively.

The inflation radius parameter of “move_base” costmap may allow the robot to explore very small frontiers so it may be an advantage over using the map constructed by SLAM. These very small frontiers may also be explored by setting the min_frontier_size parameter of “explore_lite” to a reasonably small number. The information coming from SLAM is used for exploration purposes in our project, which contains the map of the environment along with the current location of the robot. The min_frontier_size is set to 0.5 m in order to explore small frontiers. After the “move_base” node is correctly initialized and the map is obtained from SLAneed to critically assess your system performance and compare your results withM, “explore_lite” package allows the robot to explore its environment by finding the frontiers that have not been visited before. As the robot explores new frontiers, it sends movement commands to “move_base” so that it can find a path that is free of obstacles.

a) Progress between CM2 and CM3: We have integrated both packages into our main flow and test them in different settings. An example global costmap that we obtained can be found in Figure 27. However, we have observed problems with autonomous movement in difficult conditions. The robot cannot properly move when it is in an environment full of obstacles which are hard to discern.

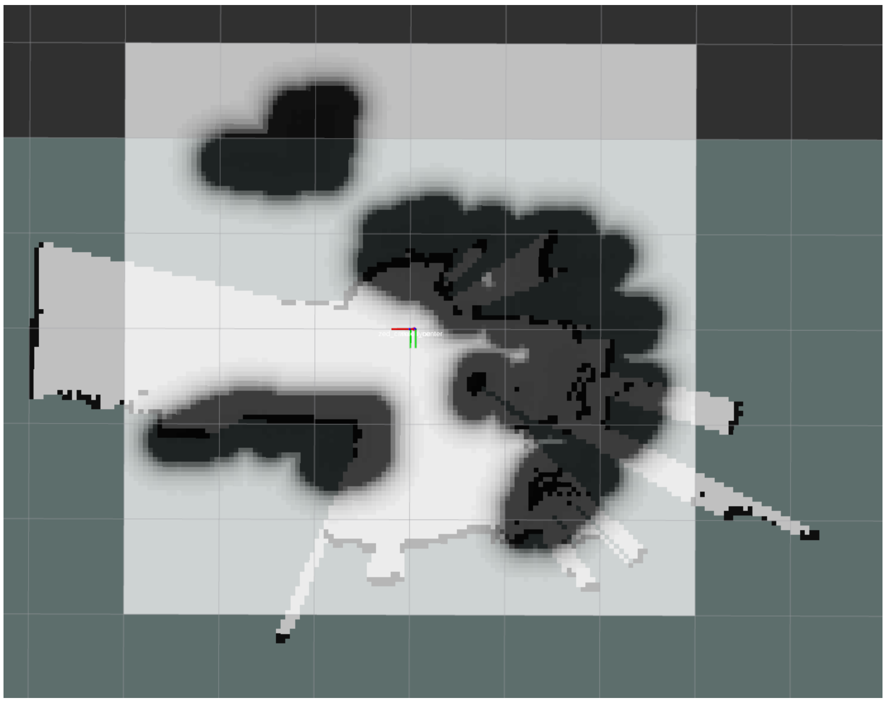


Figure 27: An example global costmap. Dark regions correspond to cells with a high cost.

b) Progress between CM3 and CM4: Efe and Su have configured the algorithm for optimal performance. They have changed the transform frame of operation for exploration algorithms since the transformations given by the mapping algorithm changes abruptly and with low frequency. Instead of using the transform frame provided by the mapping, they used the global transform frame of the odometry data. By this way, the performance of the exploration and localization is largely improved.

5.2.12 Target Detection and Laser Pointing

Should the robot detect a scene match, the next goal is to detect the target object within the scene and to mark it with a laser pointer. To this end, we used the *find_object_2d* module of ROS to perform a similar algorithm as we did in the scene matching stage [20]. Using the color video feeds from the stereo camera, this algorithm enables us to detect the object and also determine the spatial position of the detected object.

It outputs the homography transformations between the detected object and the image plane which enables us to find the position of the target center. This matrix includes information about the position of the object in the given scene. Using this, we are able to find the coordinates of the center point of the object. Furthermore, we know the resolution of the video feed from ZED2 (1280x720 pixels). Since the laser pointer is located very close to the stereo camera (and at the same altitude), the (x, y) coordinates of the object center are sufficient to calculate the angle between the laser pointing mount and the object. We need two angles that correspond to pitch and yaw, and we find them by taking the inverse tangent of the ratios $y/720$ and $x/1280$ respectively. This approximation assumes that the object is at a close distance but considering that we do not detect the objects at long distances, it is sufficient.

In order to mark the target, we used a pan tilt mount that aims a laser pointer at the detected target. Using a 2-axis motion structure, we are able to control the laser pointer and aim at the target with sufficient precision. The angle information is send to Arduino Mega via serial communication along with the information of weather the target is found. When the target is found the microcontroller lights the LED on the side of the robot and the points the laser to the target by controlling the servo motors in the pan tilt mount according the angle information provided.

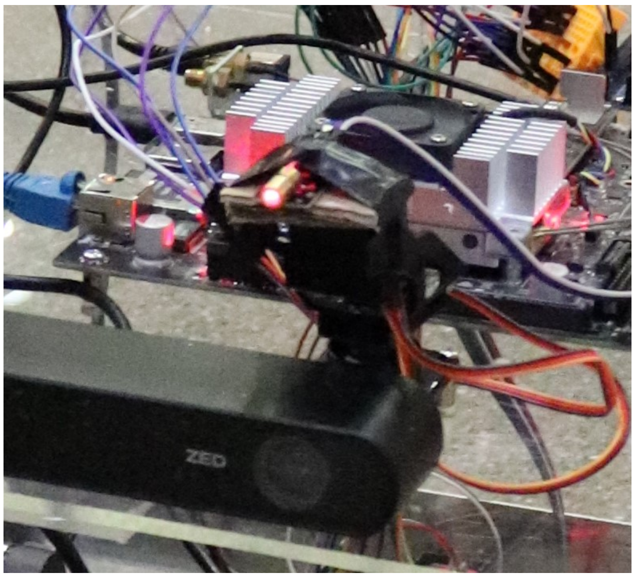


Figure 28: Two-axis pan-tilt laser mount.

a) Progress between CM2 and CM3: We installed the required libraries into our Jetson board and tried running initial tests.

b) Progress between CM3 and CM4: We mounted the pan tilt kit near the camera where the processed image is taken so that the angle information is similar for in both the image and the laser.We established communication between the board and the laser pointer. We mode the necessary connections and succesfully implemented our design.

5.2.13 Possible Risks and Solutions

a) Point Cloud Fusion for SLAM: The stereo camera and LIDAR provide distinct point clouds and these outputs are fused to perform simultaneous localization and mapping. However, the point cloud data provided by LIDAR demonstrates sufficiently accurate performance for determining the depth of the objects in its surrounding. For this reason, the benefit-to-cost ratio of using fused point cloud data is quite low.

b) Jetson Performance Issues: The performance issues of the Jetson TX2 module are considered while running the algorithms and performing experiments. However, since all of the demanding tasks have not been loaded yet and deployed simultaneously, it is not clear whether a single Jetson TX2 GPU is adequate to fulfill the time and performance constraints. In the case of inadequacies, our solutions are lowering the point cloud data rate of the sensors, downsampling the point clouds, and lowering the segmentation resolution.

c) The Overlapping Goals of Segmentation and SLAM: The most crucial algorithms are road segmentation and SLAM while performing motion planning. Road segmentation is crucial for determining the obstacles and making a decision to avoid them. However, the SLAM algorithm also considers the obstacles and plans motion accordingly. Therefore, it is not clear whether we can get the maximum benefit from these two algorithms when they are utilized together. Furthermore, the computational cost of the segmentation algorithm overweights the benefits of utilizing segmentation and SLAM simultaneously. Therefore, we have forgone the segmentation in favor of computational efficiency, even though we have previously implemented the core of the segmentation algorithms.

6 Results, Discussions, and Further Directions

6.1 Results

We have successfully built an autonomous robot equipped with a LIDAR, a stereo camera, and an INS module which can navigate and explore an unknown location and locate and point to a predetermined target. Our autonomous robot satisfies most of the determined functional requirements. Firstly, our autonomous robot does not perform data synchronization which means that it does not fuse the data streams obtained from the LIDAR and the stereo camera in order to perform SLAM. There are two reasons for not fusing the two point clouds: the fusion of two point clouds is computationally costly and the point cloud data obtained from the LIDAR demonstrates sufficient performance in determining the depth of the objects in its environment. As the cost of fusing the two point clouds is more than its benefits, we only used the point cloud data obtained from the LIDAR.

Secondly, our autonomous robot performs spatial mapping by integrating the data gathered from the LIDAR with SLAM algorithms. The speed of updating the map was given to be 10 updates per second in the functional requirements but our system can update 1 map per second due to the performance issues of the Jetson TX2 module. This was the main reason of the problems that we had encountered in the exploration stage. Thirdly, our robot is completely autonomous in data acquisition, navigation, exploration, and decision-making based on the environment. It decides its next maneuver based on the surrounding paths and obstacles through SLAM, path planning, and exploration algorithms. Although we implemented segmentation, we decided not to use it in motion planning because the SLAM algorithm also considers the obstacles and plans motion accordingly. Hence, the computational cost of the segmentation algorithm overweighs the benefits of utilizing segmentation and SLAM simultaneously. In order to assess their performance, we have constructed a maze in the EEE building and tested our robot there. Our robot can successfully generate the map of the maze as given in Figure 29.

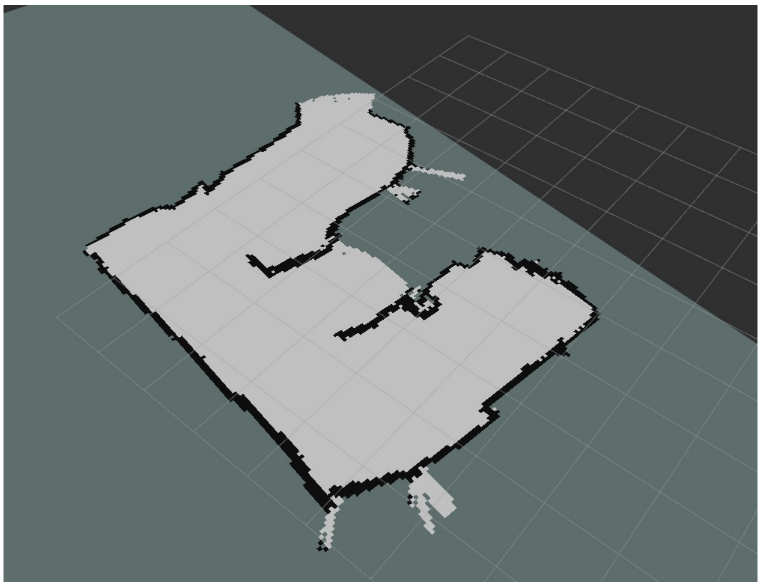


Figure 29: The map of the constructed maze.

Next, mobility is provided by the motors powered with batteries and the maneuvering ability is provided by the wheels. The microcontroller is able to generate PWM signals at various rates in order to maintain the requested wheel speed using a closed feedback loop.

Finally, our autonomous robot can locate a predetermined target by using scene matching and target detection algorithms and can point a laser to the target once it is found. In the case of a target detection, the robot stops exploration and keeps pointing at the target object as long as the target is visible. If the target becomes invisible or undetectable, the robot continues exploring to find the target again. An illustration of our robot pointing the target with a laser is shown in Figure 30.

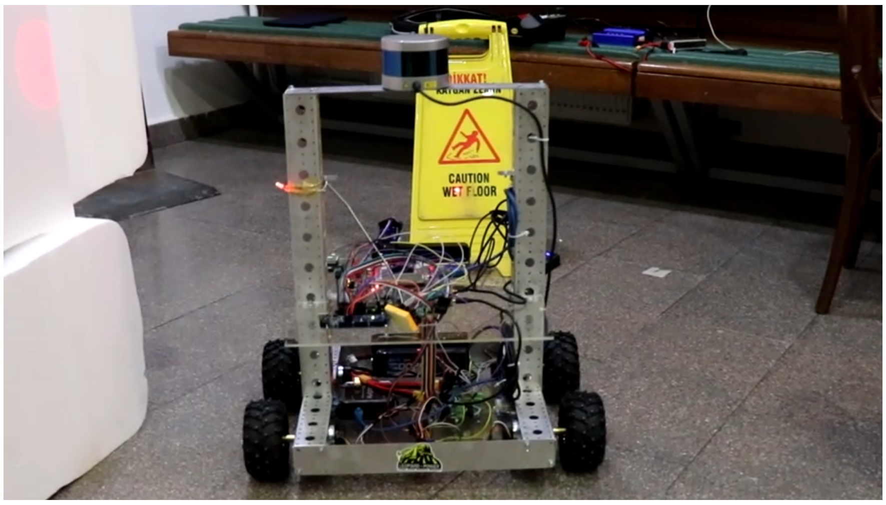


Figure 30: An illustration of the robot pointing the target with a laser.

Our autonomous robot satisfies most of the non-functional requirements. Firstly, its design and operation follows the OSHA 29 CFR 1910.333 [7] standards for industrial robots and robot system safety. Secondly, it successfully navigates and explores indoor locations as given in the non-functional requirements. Thirdly, the modular design of our robot allows us to arrange the sensors and components according to our needs. Finally, we try to optimize the power consumption of our robot by supplying the LIDAR and the Jetson TX2 board with Li-Po batteries and the motors with lead-acid batteries.

6.2 Discussions

The project goals were fully attained since we were able to construct an autonomous robot which can navigate an unknown environment and locate a predetermined target. Overall, we were succesful in integrating separate algorithms and optimizing their operation but there were also some problems. Firstly, we spent more than half of the first semester for implementing point cloud fusion and segmentation but we realized that they are actually not needed in the second semester. We found out that performing point cloud fusion leads to more problems than benefits and that the point cloud data obtained from the LIDAR is sufficient for utilizing in the SLAM algorithm. We also found out that the SLAM algorithm works similar to segmentation in the sense that both algorithms consider the obstacles and plan motion accordingly. Therefore, we decided to use only the SLAM algorithm because utilizing them simultaneously does not result in any performance improvements. We could have done more research about these algorithms and performed multiple trials before fully implementing them in order to save time and effort. Hence, we advise others who might be involved in similar projects to do more research about the possible algorithms and pick the most optimal ones before starting to implement them.

Secondly, we changed the batteries and motors multiple times in order to reach the most optimal power consumption because it is hard to anticipate the possible power consumption of the robot without finishing all tasks and testing it in a realistic setting. Moreover, a single Jetson TX2 board is merely adequate in fulfilling the time and performance requirements. We did not have time to buy a second board but we advise others who might be involved in similar projects to consider using two boards in order to prevent any unforeseen performance limitations. Thirdly, it is important to determine the main work packages and divide them among the group members in the initial stages of the project in order to progress steadily. Implementing the required algorithms separately and then integrating them into a full-blown system is an effective idea with only one downside: it is not possible to determine the performance of the board and power consumption of the robot without loading all tasks. However, this problem can be avoided by having a modular design which allows the designers to add components and change the locations of sensors.

6.3 Future Directions

We have identified three possible future directions. Firstly, our robot is designed and constructed to operate only in indoor environments. Hence, a possible extension of the project can be making the robot operate in outdoor environments by utilizing the GPS data obtained from the INS module. We think that using the GPS data can improve the performance of the path planning and exploration algorithms. Secondly, our robot is trained to locate and point to a single target in a given scene but there are generally multiple targets in more realistic scenarios. Hence, a possible extension of the project can be training the robot to locate and point to multiple targets in multiple scenes. Finally, we tried to optimize the power consumption of our robot by utilizing both Li-Po and lead-acid batteries but it can further be optimized by considering different alternatives. Hence, a possible extension of the project can be constructing a robot with more efficient power usage.

7 Detailed Equipment List

The set of equipment which includes sensors, computing board, and miscellaneous parts that have been borrowed or purchased are listed together with their prices in Table 1. The sensors and the computing board are borrowed from the company. Peripherals required to control the board, such as the keyboard are borrowed from the senior year project lab. After Committee Meeting 1, motors, wheels, and motor controller circuits have been purchased. Considering the size of these components, chassis parts are also purchased during the semester break. Overall, the total cost of the project is below the project's budget. The estimated costs of the parts which were not yet purchased are also listed in the table.

The Equipment Name	**Cost**	**Acquired from**
Velodyne VLP-16 LIDAR	Borrowed (4000 $)	Roketsan
ZED2 Stereo Camera	600 $	Purchased
XSens MTI-7-DK INS	400 €	Purchased
Jetson TX2 Module Board	Borrowed (400 $)	Roketsan
Arduino MEGA	20 $	Purchased
Monitor, keyboard, mouse	Borrowed	EEE 493 Lab
USB 3.0 Hub	Borrowed (40 $)	Roketsan
Motors and wheels	70 $	Purchased
Chassis parts	60 $	Purchased
Batteries and distribution transformers	300 $	Purchased
Laser Mount and Laser	25 $	Purchased

Table 1: List of Equipment and Prices

References

[1] A. Mishra and S. Kumari, “Military robots play a pivotal role as a tactical and operational tool for armed forces,” May 2018.

[2] R. Bergholz, K. Timm, and H. Weisser, “Autonomous vehicle arrangement and method for controlling an autonomous vehicle,” *United States Patent and Trademark Office*, no. US6151539A, 1998.

[3] D. W. Strelow and A. B. Touchberry, “Method and system for autonomous vehicle navigation,” *United States Patent and Trademark Office*, no. US7840352B2, 2006.

[4] “Boston dynamics.” [Online] Available at `https://www.bostondynamics.com/`.

[5] C. Stieg, “This $75,000 boston dynamics robot ’dog’ is for sale-take a look,” Jun 2020.

[6] “Slam robot.” [Online] Available at `https://www.pantechsolutions.net/slam-robot`.

[7] “Industrial robots and robot system safety,” *Occupational Safety and Health Administration*, 1992. [Online] Available at `https://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html`.

[8] A. Armea, “Calculating a depth map from a stereo camera with opencv.” [Online] Available at `https://albertarmea.com/post/opencv-stereo-camera/`, Oct 2017.

[9] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, “Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4304–4311, 2016.

[10] E. Olson, “Apriltag: A robust and flexible visual fiducial system,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3400–3407, 2011.

[11] A. Dhall, K. Chelani, V. Radhakrishnan, and K. M. Krishna, “LiDAR-Camera Calibration using 3D-3D Point correspondences,” *ArXiv e-prints*, May 2017.

[12] “Point cloud library,” *ROS*. [Online] Available at `http://wiki.ros.org/pcl_ros`.

[13] “Parallel point cloud registration.” [Online] Available at `https://hanzhoulu.github.io/Parallel-Point-Cloud-Registration/`.

[14] G. Grisetti, C. Stachniss, and W. Burgard, “Openslam.org,” 2020.

[15] G. Grisettiyz, C. Stachniss, and W. Burgard, “Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling,” in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 2432–2437, 2005.

[16] “pointcloud_to_laserscan.” [Online] Available at `http://wiki.ros.org/pointcloud_to_laserscan`.

[17] “Feature detection and description,” *OpenCV*. [Online] Available at `https://docs.opencv.org/master/db/d27/tutorial_py_table_of_contents_feature2d.html`.

[18] "move_base." [Online] Available at http://wiki.ros.org/move_base.

[19] J. Hörner, "Map-merging for multi-robot system," 2016.

[20] "find_object_2d." [Online] Available at http://wiki.ros.org/find_object_2d.

Appendices

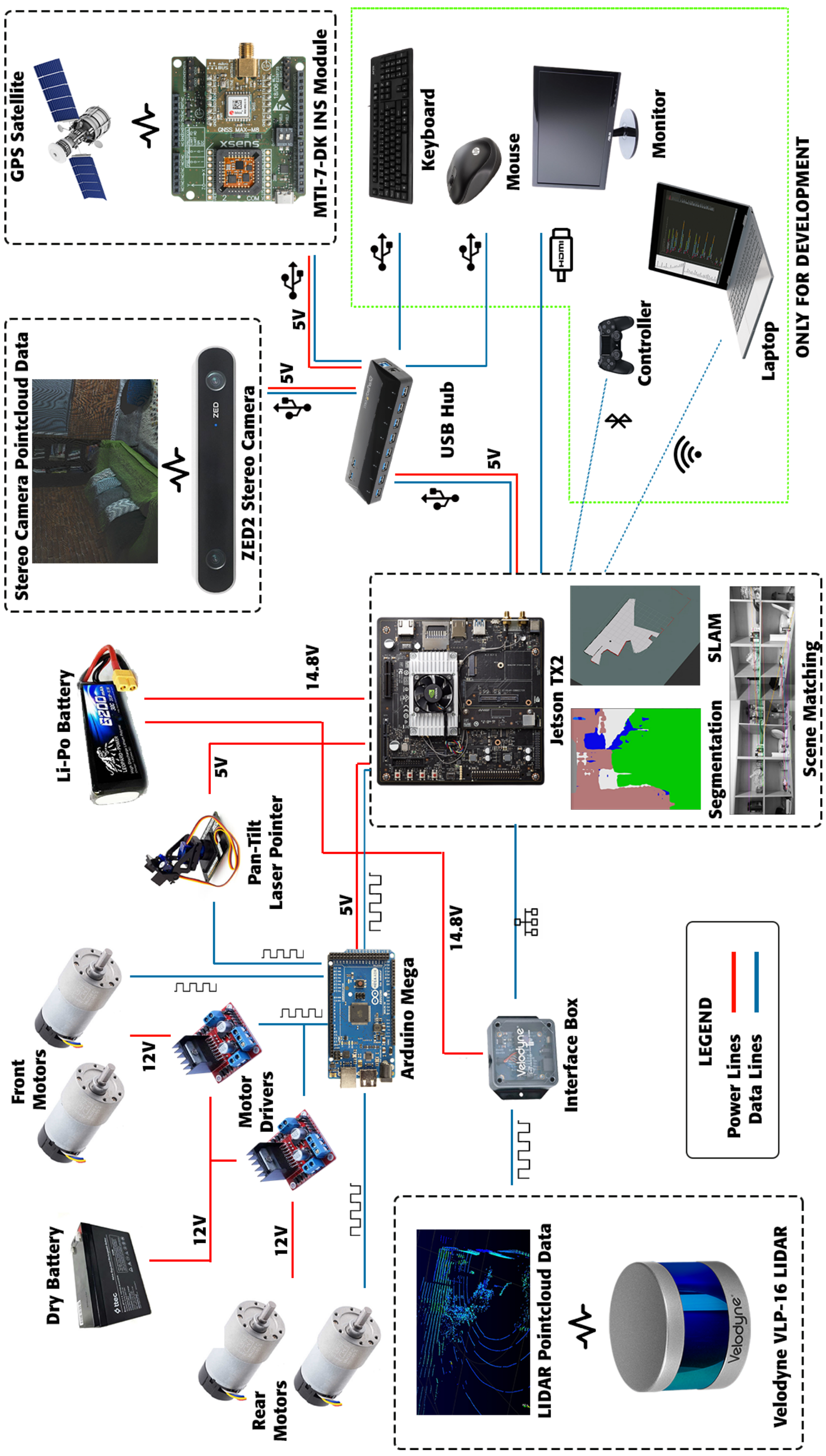


Figure 1: The block diagram of the project.

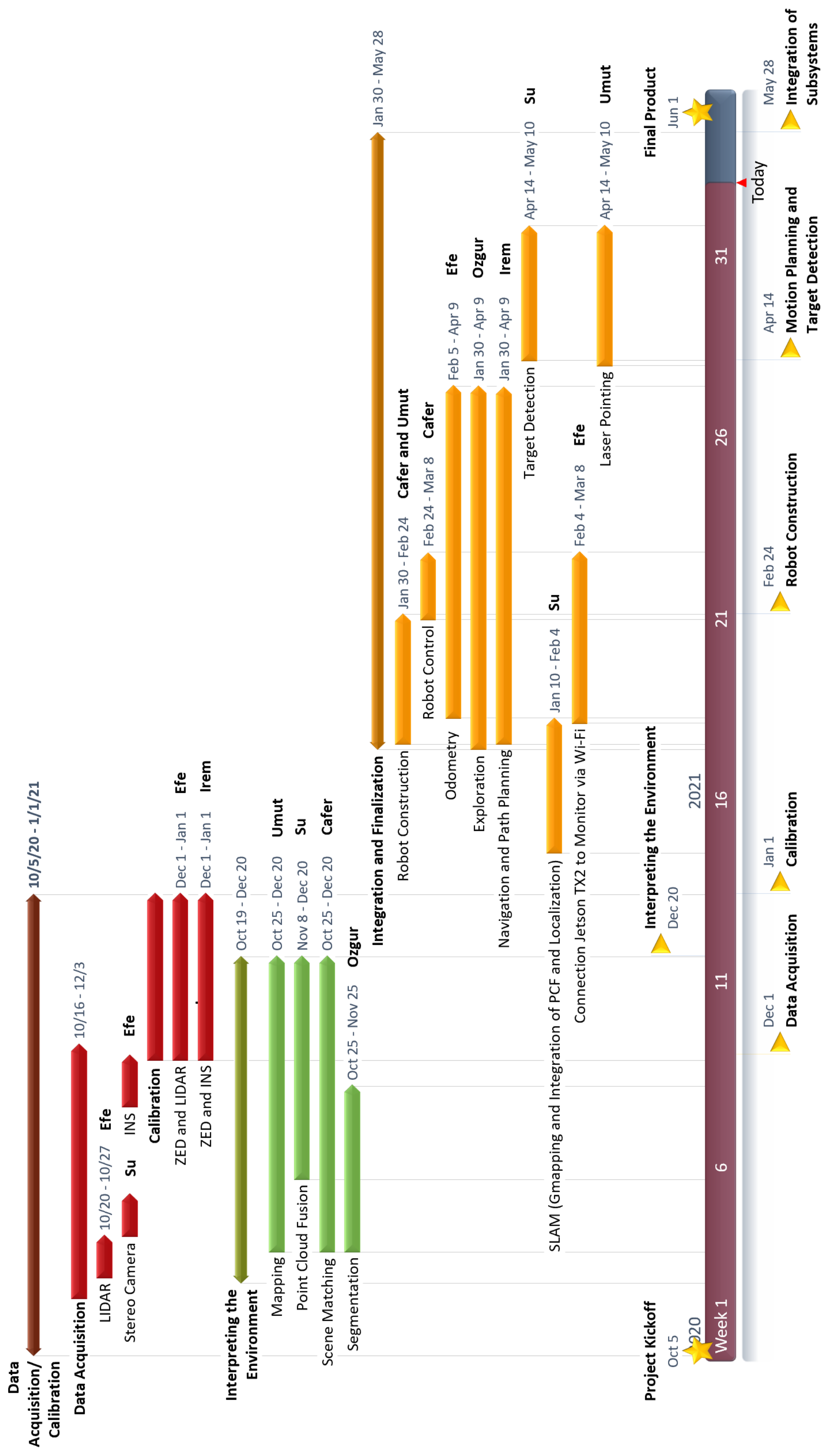


Figure 2: The timeline of the project.

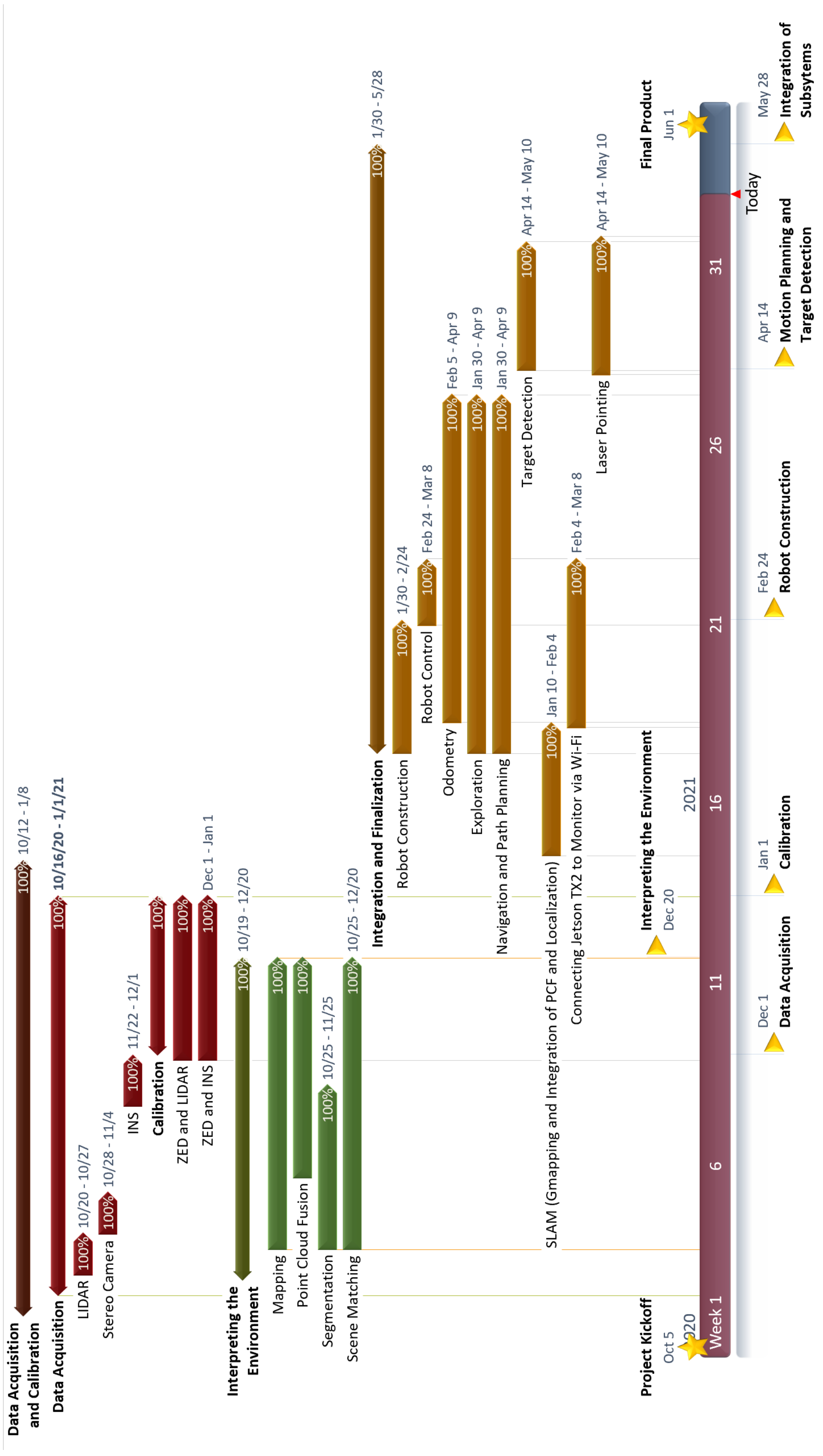


Figure 3: The current state of the project.