UGV-B: V-SLAM

*Akshay Vijay Khanna
Dept. of ECE
NC State University
Raleigh, US
akhanna3@ncsu.edu

*Muskaan Bhargava
Dept. of ECE
NC State University
Raleigh, US
mbharga5@ncsu.edu

*Sumukha Manjunath
Dept. of ECE
NC State University
Raleigh, US
smanjun3@ncsu.edu

*Bingyi Su
Dept. of ISE
NC State University
Raleigh, US
bsu4@ncsu.edu

*Dr. Kevin Han
Dept. of CE
NC State University
Raleigh, US
khan6@ncsu.edu

I. Abstract

The construction industry is beginning to recognize the potential benefits of autonomous robotics in enhancing efficiency and safety on construction sites. This project aims to explore the development of a robotic system capable of autonomous navigation and leveraging its operations for various applications in construction. The proposed system is built on Clearpath's Jackal UGV with components like RGB-D sensor, and Jaco Arm for specific operations. Our focus is on localization, mapping, and path planning to ensure the successful implementation of the autonomous robotic system. The findings suggest that utilizing autonomous technology in the construction industry can lead to smarter, safer, and more efficient operations, ultimately accelerating construction projects in a cost-effective manner. The project highlights the role of autonomous robots in the construction industry and contributes to the scope of integration of robotics in traditional industries.

II. Introduction

The field of autonomous robotics has seen widespread use across various fields such as science, technology, engineering, and beyond. From the medical industry to manufacturing warehouses, from space exploration to military operations, autonomous robots have proven their versatility and usefulness. Recently, the construction industry [2] has also taken notice of the potential benefits of autonomous technology, particularly in the form of autonomous driving and robotic systems. These systems can perform heavy lifting and accelerate construction projects in a smarter, more efficient way than humans, all while ensuring safety. Construction equipment companies such as Caterpillar, Komatsu, and John Deere have recognized the value of autonomous machines and are driving the growth of robotics on construction sites in the coming years.

This project intends to develop a robotic system capable of autonomous navigation and leverage its operations for various applications in construction. The foundation of the robotic system is the Clearpath's Jackal UGV (Unmanned Ground Vehicle) along with various components (RGB-D Sensor and Jaco Arm) integrated as per requirements. Team UGV-B is particularly associated with Localization, Mapping, and Path Planning.

III. Methodology

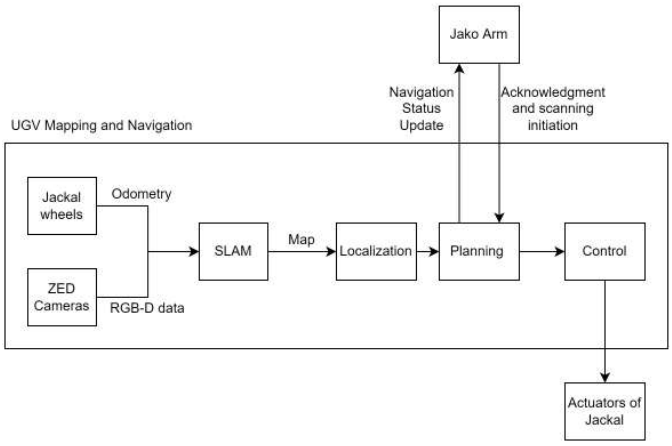

Fig. 1. High Level Architecture

A. Installations on the Jetson Nano

Installation of the Ubuntu OS The two Jetson Nanos [1] were flashed with Ubuntu 18.04, kernel version 4.9 tegra using the Balena Etcher application. The iso file for the Ubuntu OS was downloaded from https://developer.nvidia.com/jetson-nano-sd-card-image

We utilized ROS for communication between different components because of its good community support [4]. Installing ROS Melodic Based on the Ubuntu OS version, we installed ROS melodic on the Jetson Nanos. The purpose of ROS was to enable communication between the different packages or components such as SLAM, Camera and other sensors, navigation stack, and the RGB-D team arm. ROS Melodic was installed using the following command: sudo apt install ros-melodic-desktop-full

Installation of Zed Camera SDKs and ROS wrappers, To access the data captured from the Zed camera, SDKs are installed. To provide access of the data to other ROS packages such as RTAB-MAP for SLAM, ros wrappers need to be installed. Zed cameras have a CUDA requirement. The required CUDA version can be downloaded from CUDA Toolkit 12.1 Downloads — NVIDIA Developer Three ros packages related to the Zed camera need to be installed:

zed-ros-wrapper: the main package that provides the ZED ROS Wrapper node

zed-ros-interfaces: the package declaring custom topics, services and actions

zed-ros-examples: a support package that contains examples and tutorials about how to use the ZED ROS Wrapper

Installation of the Realsense RGB-D camera [3] is an alternative to the ZED camera that we explored. Similar to the Zed camera, to access the data and support communication of the data with other nodes/packages, Realsense SDKs and ros wrappers were installed.

B. Configuration of Jackal and Nanos

The UGV used for navigation tasks in this project is Clearpath Jackal mobile robot, which is an entry-level robotics research platform. It is a four-wheeled platform that has been optimized for use in a variety of environments, from indoor labs to outdoor fields. In addition, the Jackal mobile robot is equipped with an onboard computer, GPS, and IMU. Its rugged design and customizable platform make it an ideal choice for researchers and developers looking to experiment with robotics in real-world environments.

One of the key advantages of the Clearpath Jackal is its support for the ROS (Robot Operating System) framework, which is a widely used open-source platform for developing and controlling robotic systems. This allows researchers and developers to quickly and easily program and control the robot using a range of programming languages, making it an ideal platform for rapid prototyping and experimentation. The Jackal can be seen in the following picture.

Fig. 2. Clearpath Jackal Mobile Robot

Connecting Jackal to Nano through ethernet: Establish a wired connection between Jackal and the Nano through an ethernet cable. Manually add a new connection (type: ethernet) on nano. Make sure that it is set on the 131 subnets as the jackal version we are using here is kinetic. Ping 192.168.131.1 to check the connection. Get control of the Jackal using the ssh command.

Setting up wifi on Jackal: Set up the wifi interface using the wicd-curses command. Delete the wired interface if any. Select DHCPCD under external programs. Set up the wifi by selecting the DHCP hostname and suitable encryption. Shift + C to connect to the required wifi connection. Make sure that the Nano and Jackal both are on the same network to ensure communication between the two.

Setting up ROS on Jackal: Install all necessary ROS packages from the Clearpath website. Export the ROS Master URI to locate the ROS Master. Use the echo command to check the communication between Nano and Jackal through ROS Topics.

C. Simulation of Jackal on Gazebo

The navigation function is first simulated in a user-specified Gazebo world. To achieve this goal, a front laser was attached to the top of the Jackal robot for both creating the map and localization. ROS packages, such as Gmapping, AMCL, and move-base, are applied for navigating Jackal in the simulated Gazebo world, which are explained as follows.

Map generation using the Gmapping package: The gmapping package provides laser-based Simultaneours Localization and Mapping (SLAM), which uses laser and odometry data collected by the mobile robot to create a two-dimensional occupancy grid map. Jackal is manually moved around the whole world in Gazebo to scan the environment. Once the scan process is finished, the map can be saved accordingly as yaml and pgm files. The simulated world and created map can be seen in Figure 2.

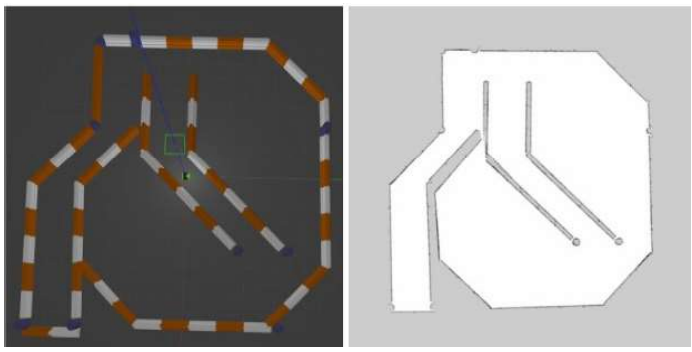

Fig. 3. The simulated world in Gazebo and the map from Gmapping

Localization with the AMCL package: The AMCL package implements the adaptive Monte Carlo localization algorithm, which is a probabilistic localization system for robots moving in a two-dimensional environment. It tracks the robot's pose in a known map by using particle filters with the information from odometry and laser scan. With the map created by the Gmapping package, the mobile robot can be globally localized in the Gazebo world. Before navigation, the initial localization should be set by using 2D Pose Estimate in rviz.

Navigation with the move-base package: The move-base package achieves the implementation of the sequence of actions to reach a desired position, which provides an interface for interacting with the navigation stack. It combines both a global planner and a local planner to accomplish the navigation task. In addition, two cost maps are maintained by the move-base for the planners. Given a goal pose, the move-base

package will use the information from the odometer and the laser scan to calculate a safe speed command for the mobile robot to reach the goal position while avoiding static collisions. When the robot senses that it is stuck, move-base will perform the recovery behavior by rotating itself in place.

D. RTAB-MAP SLAM

RTAB-Map [5] is a widely-used SLAM approach that utilizes incremental appearance-based loop closure to track camera/sensor motion and orientation. It is compatible with various sensors and provides tutorials for different robotics arrangements. However, it may be computationally demanding and may require a powerful computing platform to run in real-time, as noted in the study. The authors switched to a laptop with a more powerful processor, graphics card, and RAM to ensure that the sensor reading, calculation, and control could keep up with the robot's kinematics. The generated 2D occupancy map can be used to plan a collision-free path for the robot. In summary, RTAB-Map is a robust SLAM approach with various applications, but it may require a powerful computing platform to run efficiently.

RTAB-Map is a popular and well-documented SLAM approach that is widely used in robotics applications. One of the key benefits of RTAB-Map is that it provides a series of tutorials for a wide range of sensors and mounting scenarios, allowing users to build fully functional SLAM systems that are tailored to their specific needs. These tutorials are easy-to-follow and provide a comprehensive guide for implementing RTAB-Map in various robotics applications. One of the tutorials provided by RTAB-Map is called "handheld mapping," which involves using a handheld RGB-D camera to create a 3D map of an indoor environment. This tutorial has been replicated in various studies and has been found to be effective in creating accurate and detailed maps of indoor environments. However, despite the benefits of RTAB-Map, it has been found to be computationally heavy, which can limit its use in certain applications.

E. Adding the ZED Camera on the robot model using the jackal_description package

The ZED camera [6] is a stereo camera system designed for depth sensing and 3D mapping. The camera uses a combination of passive stereo and active depth sensing technologies to capture high-quality images and depth maps. It has a field of view of 120 degrees, a maximum resolution of 2K (2560x1440) for color images, and a depth range of up to 40 m. The ZED camera also has an adjustable baseline distance that allows users to optimize depth sensing performance for their specific applications.

One of the key features of the ZED camera is its real-time 3D mapping capability. Using the depth information captured by the camera, the ZED SDK can create a 3D point cloud of the environment in real time. In addition, the ZED camera also has an integrated IMU (Inertial Measurement Unit), which provides additional motion tracking data to improve the accuracy of the depth maps and 3D point clouds. The ZED camera can be seen in the following Figure.

Fig. 4. ZED camera

In this work, the map is created using a ZED camera with the Rtabmap SLAM algorithm. In the process of creating the map, all environment information is sensed based on the location of the ZED camera (zed_base_link). While navigating the Jackal mobile robot in the given map using move_base, the localization is based on the position of Jackal (base_link). Since the ZED camera and Jackal mobile robot are from different nodes, there is no information to define the link relationship between zed_base_link and base_link. Therefore, before creating the map using the ZED camera, it is necessary to add zed_base_link to the URDF description file of the Jackal. The added zed_base_link_joint connects the parent link base_link of Jackal and the child link zed_base_link of the ZED camera. Also, the relative position between these two links if specified as $x : 0.21$, $y : 0$, and $z : 0.35$. The detailed information added into the $jackal.urdf.xacro$ file is as follows.

$$
\begin{aligned}
&< link\ name = "zed\_base\_link" >\\
&\quad < inertial >\\
&\qquad < mass\ value = "0.2" / >\\
&\qquad < origin\ xyz = "0.21\ 0\ 0.35"\ rpy = "0\ 0\ 0" / >\\
&\qquad < inertial\ ixx = "dummy\_inertial"\ ixy = "0.0"\ ixz = "0.0"\ iyy = "dummy\_inertial"\ iyz = "0.0"\ izz = "dummy\_inertial" / >\\
&\quad < /inertial >\\
&< /link >
\end{aligned}
$$

$$
\begin{aligned}
&< joint\ name = "zed\_base\_link\_joint"\ type = "fixed" >\\
&\quad < origin\ xyz = "0.21\ 0\ 0.35"\ rpy = "0\ 0\ 0" / >\\
&\quad < parent\ link = "base\_link" / >\\
&\quad < child\ link = "zed\_base\_link" / >\\
&< /joint >
\end{aligned}
$$

F. Creating a map with RTABMAP using ZED stereo camera

The RTABMAP node subscribes to the odometry, RGB data stream, depth stream, and camera and depth camera-related information of the ZED stereo camera through ROS topics. We launch the RTABMAP node using the following command:

roslaunch zed_rtabmap_example zed_rtabmap.launch

We navigate the jackal around the environment by publishing geometric messages to the jackal cmd_vel topic. We

observe the loop closure detection window in the rviz window of RTABMAP and save the map accordingly using the map_server package. Since we are using the map generated using the sensor data from ZED Camera, we can save the map using the following command:

rosrun map_server map_server -f map /map:=/zed/map

This command saves the file in Portable Gray Map format and also saves an associated yaml file containing details regarding the Portable Gray Map file in the working directory.

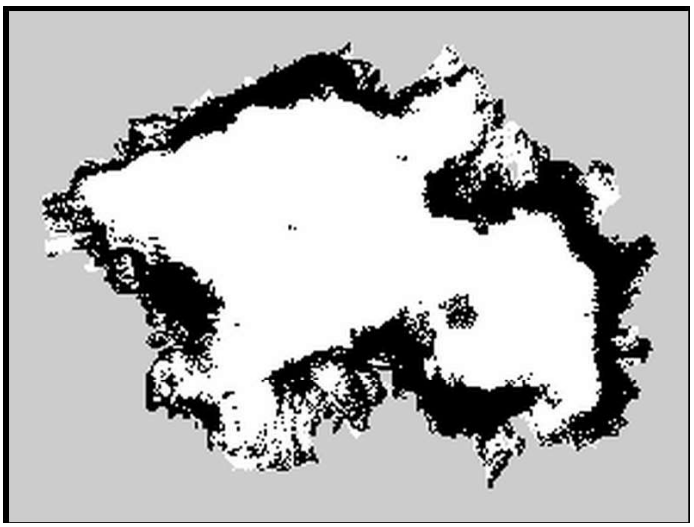

Fig. 5. Saved map of the lab using RTABMAP with Zed Camera

G. Localization using Adaptive Monti Carlo Localization (AMCL)

Adaptive Monte Carlo Localization (AMCL) is a localization algorithm used in robotics and autonomous systems to estimate the position and orientation of a robot within an environment. It is a probabilistic technique that utilizes a particle filter-based approach, also known as a Monte Carlo localization (MCL) algorithm, with the additional capability of adaptively adjusting the number of particles based on the current situation and sensor measurements.

The basic idea behind AMCL is to represent the belief about the robot's pose (position and orientation) as a probability distribution using a set of particles. Each particle represents a hypothetical pose of the robot in the environment. The particles are sampled from the distribution based on the robot's motion model and updated using sensor measurements, such as odometry, laser scans, or other sensors, to refine the estimate of the robot's pose.

What makes AMCL adaptive is its ability to dynamically adjust the number of particles based on the uncertainty of the robot's pose estimate. When the robot has high uncertainty, such as when it is in a new or ambiguous environment, AMCL increases the number of particles to explore more hypotheses and obtain a more accurate estimate. Conversely, when the robot has low uncertainty, such as when it is in a familiar and well-mapped environment, AMCL reduces the number of particles to save computational resources.

AMCL also incorporates resampling techniques to prevent particle degeneracy, where a few particles have much higher weights than others, which can result in poor convergence. Resampling allows for maintaining a diverse set of particles and prevents the filter from converging to a single hypothesis prematurely.

One of the requirements of the AMCL package in ROS is that it expects laser data as one of the inputs. Since we do not have a lidar, we need a way to transform the depth information from the ZED Camera to the laser format. We utilize zed_laserscan_nodelet.launch file which starts ZEDWrapperNodelet and the DepthImageToLaserScanNodelet in the same nodelet manager. We obtain this conversion using the following command:

roslaunch zed_nodelet_example zed_laserscan_nodelet.launch camera_model:=zed

The depth-converted laser data is published to the topic /zed/scan. Apart from the laser data, the AMCL node also expects odometry data and a saved map (in .pgm format) as prerequisites. For the odometry, we use the $/zed/zed\_nodelet/odom$ topic publishing the odometry data of the jackal. With the prerequisites satisfied, we launch the AMCL node using the following command:

roslaunch jackal_navigation modified_amcl.launch

We launch rviz for the purpose of visualization. Initially, we have to guide the jackal to learn the initial pose and location. We use the "2D Pose Estimate" option in the rviz.

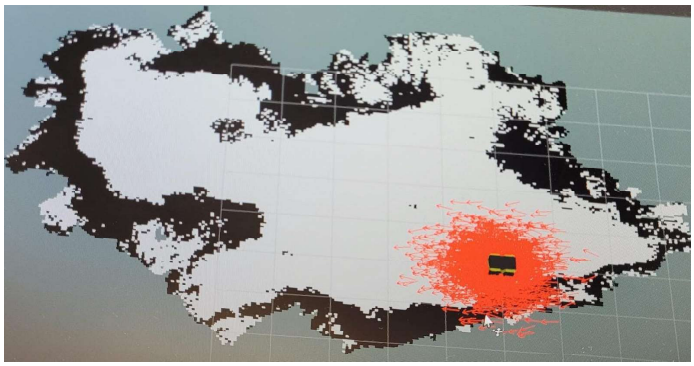

Fig. 6. Inaccurate localization of the jackal in the saved map using AMCL

One issue during localization using AMCL is that the Jackal robot keeps jumping from the initial and estimated positions. Several measures were taken to solve this problem, which are described as follows,

1) Remove redundant joint_state_publisher node: One possible reason for the issue is that two different nodes are publishing joint state information to the $/joint\_states$ topic, while the information from these two sources conflicts with each other. Therefore, the rqt_graph of the navigation system was examined. It

is found that both the $/joint\_state\_publisher$ node and $/jackal\_node$ node are publishing $/joint\_states$ information to the $/robot\_state\_publisher$ node, which can be seen in Fig. 5. The measure we took was removing the $/joint\_state\_publihser$, and the result rqt_graph can be seen in Fig 6. The problem was alleviated to some extent, but the issue still remained.

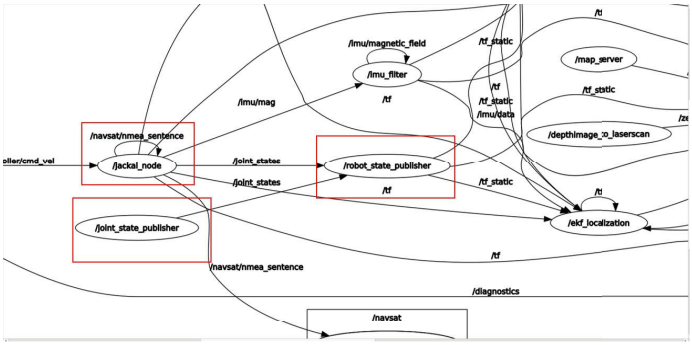

Fig. 7. Redundant nodes for $/joint\_states$ topic

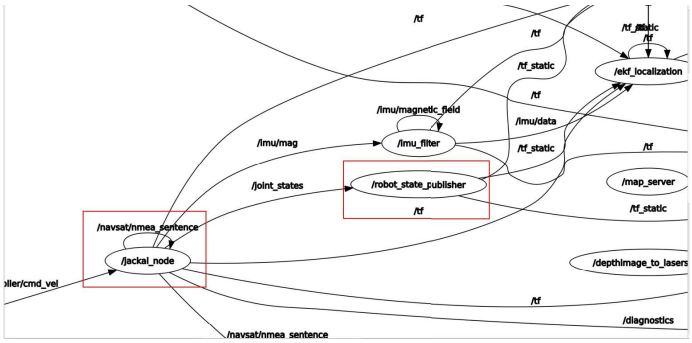

Fig. 8. rqt_graph after removing redundant node

2) Re-calibrate the ZED camera and create a new map : Since localization highly relies on the sensor data and the generated map, the next step taken was to re-calibrate the ZED camera and to create a new map of the lab. The calibration was done by using the ZED Calibration tool following this tutorial $https://support.stereolabs.com/hc/en-us/articles/360011828773-How-do-I-recalibrate-my-ZED-stereo-camera-$. After the re-calibration, another map creation using RTAB Map was performed. However, the problem was still not completely solved after this step.

3) Changing Odom information from $/odometry/filtered$ to $/zed/zed\_nodelet/odom$: Another important information used in localization is odometry. When the Odom information of Jackal was plotted in RViz, it can be seen that it was not stable and accurate. On the other side, the Odom information from the ZED camera was more stale and very accurate. Therefore, we changed the Odom information from $/odometry/filtered$ to $/zed/zed\_nodelet/odom$. So far, the robot's position on the map is more stable, while the localization accuracy is still sensitive to the initial position. This issue is the limitation of the present work and should be improved in the future.

H. Navigating the jackal using move base package

We navigate the jackal in the saved map using the "move_base" package available in ROS. The "move_base" package is designed to work with a robot's sensor inputs, such as odometry, laser or RGB-D sensor data, and a costmap representation of the environment, to generate safe and optimal trajectories for the robot to follow. It consists of several components, including a global planner, a local planner, and a costmap, which work together to enable the robot to navigate autonomously.

The global planner is responsible for generating a high-level plan or path from the robot's current location to the goal location in the global coordinate frame. It uses algorithms such as Dijkstra's algorithm, A* algorithm, or Trajectory Rollout to find an optimal or near-optimal path while avoiding obstacles.

The local planner is responsible for generating a low-level plan or trajectory that the robot can follow in real-time to avoid dynamic obstacles and reach the goal. It uses algorithms such as the Dynamic Window Approach or the Timed Elastic Band to generate local trajectories that take into account the robot's dynamic constraints, sensor data, and environmental information.

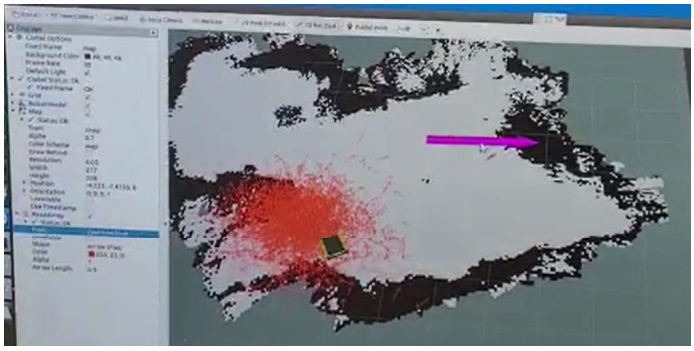

Fig. 9. Providing a target location in rviz to navigate using move base

The costmap is a representation of the environment that provides information about obstacles, free space, and other relevant information for navigation. It is used by both the global and local planners to plan paths that avoid obstacles and ensure safe navigation.

"move_base" also provides interfaces for controlling the robot's motion, such as sending velocity commands to a robot's actuators to follow the planned trajectory, and monitoring the robot's state, such as its position, velocity, and sensor readings.

"move_base" node is launched using the following command:

roslaunch jackal_navigation move_base.launch

We fine-tuned certain parameters with respect to the local planner and costmap parameters. The observation was that increasing the controller frequency improves the smoothness

of the movement of the jackal. Reducing the planar frequency accordingly improves the smoothness of the movement of the jackal. We also fine-tuned the parameters "obstacle_range" and "raytrace_range" which impact how close the jackal can move toward an obstacle before changing the local plan.

The demonstration of the navigation system can be seen in the following video, which can be accessed via the link $https://drive.google.com/file/d/1nK5Zt9yueamZvRrsrZ-imyUjoBWARp5F/view?usp=share\_link$

I. Communication system and components of the final system

Communication between the two teams was a critical job. The UGVB team first performs localization and mapping of the whole region of interest. The UGVB team then identifies the location of the object of interest with respect to the map created. Once the target location is given as input, the unmanned ground vehicle goes independently from its starting point to the location of the object on the map. The UGV is positioned in a way that scanning with the Jaco robotic arm is simple.

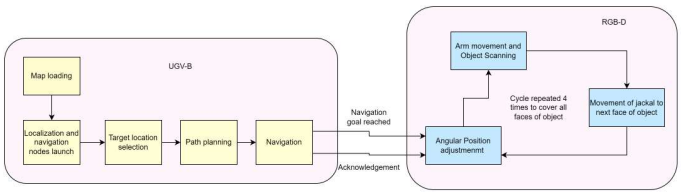

Fig. 10. Components of the demonstration

Once the destination is reached, UGVB publishes a message through ROS stating "Navigation Goal Reached". RGBD team responds with an acknowledgment message and proceeds to initiate the scanning process. The scanning of the item is dome using the camera and the RTabmap algorithm, while the Jaco arm is also started and begins path planning based on the STL file. Once one side of the item has been scanned, the Jaco Robotic arm returns to the location it started the scanning from. After that, the RGBD crew moves the UGV to the next side of the object for scanning, and so on for the other sides. After capturing the object's point cloud, the image file is loaded into meshlab and the picture is post-processed.

Figure 10 shows our completed system performing the scanning of the first side of the object during our final demo.

Since autonomous motion of our system often results in severe shaking of the system and is not always reliable, we decided to hardcode the motion of the Jackal around the object to be scanned. This "hardcoding" is still semi-dynamic because it reads the size of the object from an STL file and can adjust the length of the movements based on this information. The shape and motion pattern are fixed, and this information only serves to scale the path. This leaves the system vulnerable to errors in the position of the jackal relative to the object when the scan begins. If the Jackal does not start in the correct desired position, there is currently no way to know that and adjust automatically. As we saw in our demo, the Jackal arrived at a location that was slightly different than the desired and we were forced to adjust the location of the object during our demo to account for this. A more robust solution would utilize the information from the RGB-D camera to create a closed-loop system. This would require an object detection algorithm implementation that could identify an object as the desired scanning object. This object's location would then have to be compared to the estimated location of the Jackal using the odometry from the RTABMapping. This would have allowed us to dynamically determine the location of the Jackal and adjust for errors in the Jackal's location at the start of the scan.

IV. Conclusion

This project-based course requires substantial effort from diverse teams collaborating towards a common goal - developing a fully functional robot capable of autonomous navigation through a construction site environment, identifying dynamic obstacles, and performing collision avoidance for smooth navigation toward the objective. Our team, UGV-B, is responsible for simultaneous localization and mapping, as well as navigation using a ZED Stereo camera on the Jackal robot while working closely with the RGB-D team to track the object of interest and generate a point cloud of the object. We have successfully tested and demonstrated the integration of autonomous navigation with the RGB-D team on the Jackal by mapping the unknown environment, updating the known map for changes, and localizing the robot in the map using the SLAM and AMCL algorithm on the ZED Stereo camera. Our implementation of the Vision SLAM solution on Nvidia Jetson Nano has enabled us to generate a real-time map of the environment and navigate in it, visualizing a simulated Jackal model combined with a Zed camera model in RVIZ, which simulates the actions of the Jackal in the real world. We have been able to monitor the navigation status and communicate it to the RGB-D team in real-time. These efforts demonstrate a viable solution for an autonomous system capable of navigating through a construction environment, localizing an object, and performing 3D scans. For future work, we believe it would be worthwhile to optimize system performance by running the visual-SLAM algorithm with CUDA to fully utilize the computing power of Jetson Nano. This would involve utilizing an embedded device such as Jetson Nano with all the libraries compiled to use CUDA acceleration, which could potentially improve performance significantly.

References

[1] Jetson Nano. Available: https://developer.nvidia.com/embedded/jetsonnano

[2] Evers, "How autonomous robots are changing construction," CNBC, 30-Nov-2020.

[3] Intel© RealSense™D435. Available: intelrealsense.com/depth-camerad435/

[4] "ROS Wiki" Open Robotics, 08-Aug-2018. [Online]. Available: ROS.org.

[5] "RTAB-Map: Real-Time Appearance-Based Mapping" Available: http://introlab.github.io/rtabmap/

[6] Zed Stereo Camera. Available: www.stereolabs.com/docs/tutorials/depth-sensing/

[7] ”Dellaert, F., Fox, D., Burgard, W. and Thrun, S., 1999, May. Monte carlo localization for mobile robots. In Proceedings 1999 IEEE international conference on robotics and automation (Cat. No. 99CH36288C) (Vol. 2, pp. 1322-1328). IEEE.”
[8] Navigation with Move-base using ROS- http://wiki.ros.org/move_base