System Integration of a Tour Guide Robot

Suhasa Prabhu Kandikere, Celal Savur, Ferat Sahin,
Rochester Institute of Technology,
Rochester, NY, USA
{sk4592, cs1323, feseee}@rit.edu

2022 17th Annual System of Systems Engineering Conference (SOSE) | 978-1-6654-9623-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/SOSE55472.2022.9812680

***Abstract*—In today's world, people visit many attractive places. On such an occasion, It is of utmost importance to be accompanied by a tour guide, who is known to explain about the cultural and historical importance of places. Therefore, a human guide is necessary to provide tours for a group of visitors. However, Human tour guides might face tiredness, distraction, and the effects of repetitive tasks while providing tour service to visitors. Robots eliminate these problems and can provide tour consistently until it drains its battery. This experiment introduces a tour-guide robot that can navigate autonomously in a known map of a given place and at the same time interact with people. The environment is equipped with artificial landmarks. Each landmark provides information about that specific region. An Animated avatar is simulated on the screen. IBM Watson provides voice recognition and text-to-speech services for human-robot interaction. The experimental results show that the robot takes average time of 10000 seconds to provide a tour. TEB and DWA local planner are compared by allowing the robot to autonomously maneuver the environment for 9 trials which is tabulated in section V.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Tourists visit many different places all over the world. They may visit one of the wonders of the world or they might visit a monument that holds historical significance. On such an occasion they are accompanied by a tour guide. The tour guide leads tourists to many places and explains the importance of a place. Due to the advancements in technology recently, a smartphone can be used as a guide for a person. The phone uses GPS for navigation and can lead a person to a destination. Assistants such as Apple's Siri, Google's Assistant, and Amazon's Alexa that are available on the smartphone can explain the importance of a place to a person. However, a person while looking into the phone can crash into other moving people or might trip and fall if he/she is not careful. As a result, a human tour guide can provide tours for a group of visitors leading them carefully to places of interest and explain about the importance of the place. However, a tour guide may feel tired, distracted, and get bored due to repetitive tasks. During these times, a robot can provide a tour to a group of visitors. Such a robot is called a tour-guide robot. A robot is a programmable machine, which can help in reducing the workload of humans in different tasks. there are various kinds of robots such as Mobile robots, Arm robots, legged robots, and so on. This experiment focuses solely on a mobile robot. A mobile robot uses its sensors to perceive the environment and move around the environment using its legs, wheels, tracks. This experiment focuses on a mobile robot with wheels. This robot is used to implement the functionality of a tour-guide robot. Contributions of this research are listed below:

- A tour guide robot with built-in self-driving capability: In a known map, the robot can maneuver without colliding with obstacles, acheived using navigation stack in ROS.
- An example of system of systems application: Modules such as ROS navigation stack, Animated avatar in Unity and lower level hardware required for controlling the robot, work together to function as a tour guide robot.
- Comparison of multiple planners' performance in real-world: DWA and TEB local planner comparison on the 3rd floor of KGCOE, RIT.

This paper is organized as follows: section II provides the background literature. Section III covers the research methodology: the hardware used, algorithm description, and navigation system. Section IV covers the experiment. Section V is the results, figures, and analysis of the system.

II. Related Works

Places like Museums are often filled with exhibits that are expensive and are crowded with people. The two fundamental blocks of a tour guide robot are, the robot must be able to autonomously navigate the environment safely and reliably, it must be able to interact with people around it. One such robot was presented by Sebastian Thrun et al. [10] called RHINO. This robot had a web interface that allowed people around the world to virtually move around the museum by giving goal points to the robot. When the tour is finished, the robot moves to the entrance of the museum awaiting its next set of visitors. Minerva is a second-generation tour-guide robot that was a revision of Rhino, which was a tour guide robot introduced in mid-1997. A year later Minerva was introduced to people with a lot of features that Rhino lacked. Skycall is another tour guide quad rotor designed and developed by MIT [6]. It helps college freshman find their way to their classes. A tour guide Humanoid robot by the name ASIMO was introduced by Honda in the year 2000 [7]. This robot has the capability to interact with people but is limited to 100 questions. The experiment introduced by Chung et al. [13] is about navigation, localization, path planning, and autonomous control of a tour guide robot in detail. The kinematics model of the tour guide robot introduced calculates the average angular velocity of the wheels of the robot, which can then be used to find the linear velocity. This robot has a facial expression system. The robot uses RFID tags to localize itself, the four tags are placed in the environment, RSSIs data are received from these tags, which is used to calculate distances from these tags to the robot, and by using least square methods the initial robot location is calculated.

A. SLAM and Path Planners

One fundamental rule for a tour guide robot is it has to navigate the environment autonomously. to achieve this, the robot needs a map of the environment. The robot has to map the environment and also has to be aware of its position. This problem in robotics is known as Simultaneous Localization and Mapping. A new algorithm that fuses the laser slam and visual slam, to obtain a highly accurate robot position in an environment is introduced by Chan et al. [10]. Saman et al. [14]. go on about implementing an extended Kalman filter to solve the slam problem. Kalman filters are used to predict the position of the landmarks with odometry sensors and update using the values from the sensors using mathematical equations. A system for occupancy grid mapping using a mobile robot equipped with an ultrasonic range finder is discussed by Hadji et al [3]. In this experiment, an ultrasonic ping sensor is mounted on the robot and has the ability to sweep 90 degrees on either side. The inverse sensor model is used to create a grid map. The map cells occupied by obstacles have a probability of 1 and cells with no obstacles have a probability of 0. Global planners and local planners are the two types of planners required for navigation. Global planners plan a route from source to destination and local planners provide control commands to robot such as velocity, acceleration to follow the path constructed by global planner and avoid obstacles at the same time. Naotunna et al. [4] compares different local planners available in ROS for differential drive heavy robots based on their goal reaching accuracy, the path chosen to reach the destination, and time consumption. Fox et al. [5] describes a planner for avoiding obstacles and maneuvering to reach the goal. Dynamic Window Approach planner is introduced for this purpose. This planner samples from the velocity space of the robot (v, w). For each sampled value from this space, a simulation is performed to understand the effect of applying these values to the robot. Quinlan et al. introduce an Eband planner that relaxes the global planner [8]. As the name suggests, EBand stands for Elastic Bands, which considers the path planning and control capability of the robot. The Eband planner deforms this path in real-time when it comes near an obstacle. The main concept here is the bubble around some point on the robot.

B. Fiducial Markers and Speech system

Fiducial markers help the robot in identifying those places in the environment. A new visual fiducial system is described that improves upon the previous systems by incorporating the fast line detection system and avoids occlusion, lens distortion to its maximum extent. A graph-based image segmentation algorithm is proposed which allows precise estimation of lines in the image. A digital coding system is introduced that is not susceptible to false positives. The detection method introduced by this system has a better localization accuracy as proposed by Olson et al. [2]. An experiment on ArUco markers, to localize the robot globally and locally is introduced by Babinec et al. [11]. The experiment is conducted with a webcam and HDR camera. The webcam provides a high resolution and HDR provides a high dynamic range. ArUco markers are 7x7 square markers, in which the outer rows and columns are black, each marker is represented by 5 words, each represented by 5 bits. It uses 3 bits for encryption and the rest 2 bits carry information about the markers. Since there are 4 combinations for 2-bits, and there are 25 bits in total for each marker, there is a total of 1024 markers available. Since it has a limited number of markers, it can be used for localizing a robot in a small area. The overview of a speech recognition system is introduced by Rawat et al. [15]. There are two types of speech recognition systems, speaker-dependent, and speaker-independent speech recognition systems. The Speaker-dependent version must be trained with a large library of voice datasets. However, the speaker-independent version does not dependent on the voice of a person, hence these have limited vocabulary. After the conversion to digital signals, features must be extracted from these speech signals, which removes redundant information and other noise-related components which makes the signals contain only essential information that is required to recognize speech. The most popular features for speech signals are the Mel frequency cepstral coefficient (MFCC), which is commonly used to recognize numbers spoken through a telephone. There is also Perceptual linear prediction defining the human auditory system in an effective manner and Linear predictive code which assumes that the sound is produced at the end of the tube. Decoding of these speech signals requires the phonemes and the words in the dictionary to be compared to the processed analog signals using the Hidden Markov model.

III. Methodology

ROS framework provides numerous tools to work on a simulated robot if a user doesn't have access to a real robot. This experiment focuses on the Gazebo simulator which is default to ROS. This engine is built on OpenGL which is a specification and allows the program developed to be cross-platform. The simulation engine has all the physics implemented and can be altered as well. To be able to simulate the Permobil c300, It was first designed in CAD software and later converted into a Unified Robot Description Format (URDF in short). Gazebo is used as a simulation environment. ROS provides a framework to write controls to any robot. There are many prebuilt controls that can be used as a plugin. One such plugin called the Differential Drive plugin is used.

The base of the Tour guide robot is salvaged from Permobil C300 WheelChair. It consists of two 12v DC batteries, sufficient to power the entire hardware on the robot. The robot consists of a lidar, Intel Realsense camera D435I with an IMU and wheel odometry. The LIDAR is mounted in front of the robot so that it can detect obstacles with lower heights. This limits the lidar to sweep the area in front of the robot instead of the 360 degrees around it. The initial hardware setup of the robot consisted of a Raspberry Pi 3, Teensy 4, Sabertooth motor driver, YDLidar, Intel Real sense D435i depth camera, Router, and a CUI encoder. Since the robot does not move straight when both motors are commanded the same value, a PID controller is used to make the robot move straight. The process here is the rotation of wheels and encoders are responsible for reporting the speed of the rotating motors. This controller has three constants known as gains. These gains have to be tuned such that the robot moves straight. The PID controller is implemented in the ROS framework. The control board running ROS sends a command to teensy through the UART port and in turn teensy is responsible for commanding the motors at a given speed.

$$u(t) = K_p\, e(t) + K_i \int e(t)\, dt + K_d \,\frac{de(t)}{dt} \tag{1}$$

In the above equations, u(t) is the output of the PID equation. *"K_p"* refers to the proportional gain, *"K_i"* refers to the Integral gain, *"K_d"* refers to the differential gain. The proportional gain describes "How fast the system should reach the setpoint" If the value is too less the system may never reach the setpoint. If it is too large it may overshoot beyond the setpoint and starts oscillating. The differential gain tries to smoothen out the oscillations caused by proportional gain. The integral gain is responsible for minimizing the steady-state error that exists after tuning proportional and integral gain. Usually, the integral gain is assigned a small value, since it is scaling the accumulated error.

A. *Processing and Fusion*

The position and orientation are calculated solely based on the ticks provided by the encoder motors. The encoder is accurate when the robot moves forward in a straight line. However, when the robot rotates, there is a lot of drift and the position estimate provided by the odometry is not reliable. To address this issue another reliable sensor that provides the required orientation estimate of the robot is needed, which in this case is an IMU. The Real sense depth camera comes equipped with an IMU that provides linear acceleration and angular velocity. However, raw values provided by IMU cannot be used directly. In general, IMU has a gyroscope and an accelerometer. The gyroscope provides the angular velocity and the accelerometer provides linear acceleration. The gyroscope gives an orientation estimate when the robot is moving and the accelerometer is good at estimating orientation when the robot is stationary. Therefore, Sebastian Madgwick et al. [17], a filter is used known as a Madgwick filter to obtain a good estimation of the orientation of the sensor mounted on the robot. In order to obtain close to accurate position and orientation of the robot, the values from these two sensors are fused using the Extended Kalman filter. The resultant position and orientation of the robot are obtained, which can then be further used by navigation algorithms to move the robot safely to its destination position on the map.

B. *Navigation*

Once the robot has been set up with the above requirements. It has to be interfaced with the Navigation Stack of ROS. This stack is responsible for making any mobile robot autonomous. However, before implementing the navigation stack. There is a need to build a map of an environment. The navigation stack works effectively when it has a prebuilt map. When it comes to building a map of an environment. There are many algorithms that are developed over the past years such as RTAB map, occupancy grid map, octomap to name a few. In this experiment, the map to be built is called the occupancy grid map. Occupancy grid map uses inverse sensor model [18] to build a map, where the robot uses its current position and a laser scanner to estimate the placement of obstacles. This map is a grid map, as the name suggests, the entire environment is perceived as a 2D grid world to the robot, with each cell in the grid containing information whether it is being occupied by an obstacle or not. The probability of a cell being occupied is considered to be one [p(the cell has obstacle) = 1], the probability of a cell not containing an obstacle is zero [p(the cell has no obstacle) = 0], and the unexplored grid cells have values -1. This way a map is being constructed. ROS makes this easy to implement by providing the necessary tools to generate such a map and visualize the process of map building. After building a map, the Navigation stack can use this map to maneuver the robot in an environment.

There are many localization techniques available in ROS such as AMCL, EKF localization, and so on. AMCL stands for Adaptive Monte Carlo localization. It is a probabilistic localization system for robot maneuvering in a 2D environment [34]. Here The robot is interpreted as a particle. These particles are randomly spread in the given map. each particle has a specific weight. Depending on those weights the particles are sampled from the distribution. AMCL matches the LaserScan of the particles to the borders of the given map. If the scan matches the map, then those particles are sampled from distribution as they have a higher weight than all other particles. AMCL depends on the robot's odometry sources and LaserScan measurements.In this experiment, Extended Kalman filter is used for correction of drift in encoder source. This fused odometry is an input for AMCL localization.

Move Base as the name suggests is responsible for moving robots from one point to another in a known map. Move base has two important helper algorithms, which are Global planner and Local planner. In this experiment, A* algorithm is used as a Global planner. The local planner used here is the Dynamic-Window Approach planner, which is an effective algorithm for making the robot autonomous. Global planner uses the global cost map to trace out the path between the start and endpoint. The local planner is responsible for maneuvering the robot in the path provided by the global planner by avoiding obstacles that are in the way of the robot. It does so by using the information provided by Local CostMap. The global cost map consists of the map borders, local cost map consists of the obstacles that are not present on the map while mapping. move base takes its input from odometry source, map server and frame transforms to drive the robot autonomously in a known map.

This experiment also consists of visual fiducial markers as artificial landmarks in the environment. April Tags are a kind of visual fiducial marker that consists of a black background with white foreground with a specific pattern [2]. Due to the black border of the April Tags, it is easy to detect using computer vision techniques under various conditions like poor lighting, different orientations, etc. April tags look like a QR code. However, a QR code holds around 3Kb of information whereas April Tag can hold only 7 to 12 bits of data. This is the important feature of the April tag since it has less payload and is detected at far distances without any difficulties. In this experiment, Tag 36h11 April tags are used. This is a standard April tags family. ROS provides support for detecting April Tags and visualizing them in a tool called RQT image view. This ROS package requires the size of the April tag and the tag family used for this experiment to detect the tags and finally camera stream is provided as an input.

C. System Integration

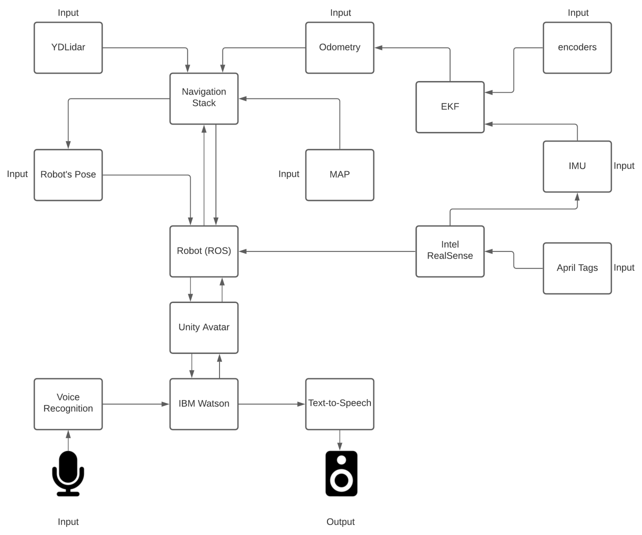
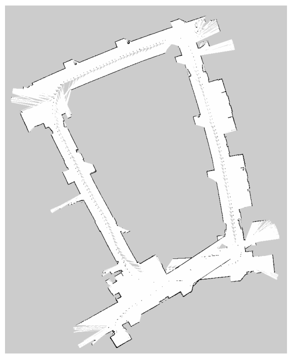
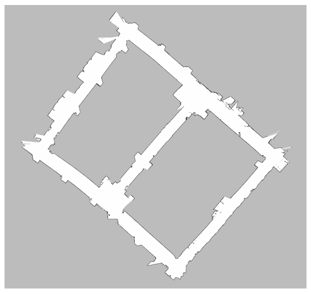


Fig. 1: Complete System Block Diagram

In fig 1, each block represents a different part of the system. The experiment here is a tour-guide robot. The robot should be able to move from its starting point to its destination avoiding obstacles in its way and It should also be able to interact with humans. To achieve this, each room is stuck with April Tags. Each tag has a unique code that can be deciphered from the vision system of the robot. These tags are 7x7 large and around 1024 unique markers can be generated. These tags are stuck to the laboratories in the environment. The robot can be made to maneuver near the tags. The robot is equipped with a vision system capable of detecting the markers. Once the tags are detected, the information about the marker id is sent to IBM Watson which is a dialogue system. if the specific tag id exists in the dialogue system, the corresponding dialogue is submitted to the text to speech conversion system. This system can access the speaker where the speech is being played. Once the speech is finished, the unity pings the ROS framework



(a) map generated with encoder



(b) Map generated with UKF fusion

about its current status, the robot starts moving to the next goal point. This keeps repeating until all the tags have been visited in the environment. The odometry source of the robot comes from two sources as shown in fig 1. IMU data from Intel Realsense and odometry data is fused using extended Kalman filters. This fused odometry is then fed to the SLAM block for mapping and navigation. The navigation stack is responsible for the autonomous navigation of the robot. The data from the perception sensors are fed to the navigation stack or move base. The localization node AMCL, Global planner A* algorithm, local planner TEB planner work together to maneuver the robot in the given environment.

IV. Experiment

The experiment was first conducted in a space where the robot had more freedom to choose its path. The mapping of such an environment does not require significant effort. So, a simple odometry source like Encoder was enough to fulfill mapping capabilities. The gmapping relies heavily on accurate odometry sources and laser scans. Therefore, it is not recommended to use gmapping if the odometry source is not accurate. The robot in this experiment has a good CUI Encoder which provides around 118,784 ticks per revolution of the wheel. However, The encoder is useful when driving in a straight line and has issues during the turning of the robot, where there is a lot of drift. To reduce drift in the odometry source and create a good map, an IMU sensor is used as another odometry source. The Intel real sense D435i comes with an IMU built-in. IMU's linear acceleration in the x-direction, its angular velocity in z-axis, and its orientation values are fused with encoder's forward velocity, angular velocity, and orientation using Extended Kalman filter package available in ROS to create an Odometry source. In the map shown in the fig **??**-b, even though the map looks similar to a rectangle and portrays the 3rd floor of KGCOE, RIT, this map has a key error that makes it unfit for navigation. Since the floor consists of corridors, the corridor reading is the same for at least 2-3m, which makes the algorithm believe that it has not moved and is in the same place as it first registered the scan of the corridor. This makes some pathways in this map smaller, which confuses the localization node and the robot gets stuck at some point. To overcome this problem. A map was created from the floor plan of the environment as described in this algorithm as shown in [9]. This algorithm uses the floor plan and it also uses two points on the x-axis and 2 points on the y-axis. It requires the distance between those points in the real world. Using the information provided in the floor plan and the distance data from the real world, the algorithm scales the floor plan and an occupancy grid map is created that can be visualized in ROS. The initial setup consisted of Dijkstra as the global planner. However, the heuristics of Dijkstra is the distance from the robot's current position to the start point. This would make the robot move very close to the walls and took a longer route instead of the shorter one. To overcome this problem, the A* algorithm was used. The heuristics of the A* algorithm is the sum of the distance from the current position of the robot to the starting point and the distance from the robot. Many local planners are available in ROS. The default local planner in ROS is the DWA planner. This planner works well in a given map with static obstacles. If the local cost map has local obstacles, this planner stops as soon as it encounters an obstacle and this planner takes time to calculate trajectory score from its sampled velocities (v, w) and then moves past avoiding the obstacle in its vicinity. However, it sometimes fails to produce a path. TEB local planner is suitable for car-like robots. It does not take advantage of the differential drive turning mechanism. This planner requires more space in the environment to avoid obstacles. EBand local planner is also suitable for dynamic obstacles. However, in this experiment, this planner made the robot waver a lot and the motion of the robot wasn't smooth. So it was discarded.

V. Results

The first step towards making the robot autonomous with a prebuilt map is to observe the planners capability in maneuvering the robot autonomously in the presence of local obstacles in the costmap. The table I represents the accuracy of a planner in avoiding obstacles in 9 trials. In all the trials that were conducted, the robot was autonomously navigating in the given map, while doing so, an obstacle is introduced in its path which would appear as a local obstacle on a local costmap, which means that the local planner is responsible for avoiding the obstacle. The TEB planner was successful in avoiding obstacles in more trials than the DWA planner. Hence TEB is chosen as the local planner in this experiment.

TABLE I: Dynamic obstacle test

Local Planners	Number of Trials	Successful trials
Time Elastic Band (TEB)	9	8
Dynamic Window Approach (DWA)	9	6

Once the planner has been selected, it is important to test out the planner in the environment. In this experiment, the autonomous capability of the robot is tested on the 3rd floor Kate Gleason College of Engineering (KGCOE), RIT. In figures 2 and 3, the robot starts at the same start point and is able to traverse three different goal points. This shows that the robot can maneuver all part of the map autonomously.

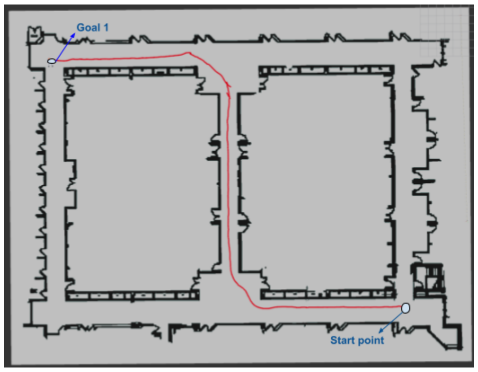


Fig. 2: The trajectory followed by the robot to reach Goal 1

In the fig. 2, In this figure, the robot travels the longest distance on the map from one corner to the diagonally opposite corner. There are no sub-goals while traversing the map. As seen from the figure, the red line depicts the trajectory of the robot while traveling along the three corridors of the environment and reaching its goal location.

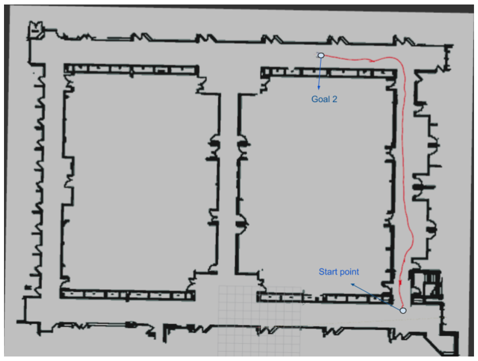


Fig. 3: The trajectory followed by the robot to reach Goal 2

TABLE II: Distance travelled and Time taken to reach goals

Goal	Distance Travelled along X-axis in meters (m)	Distance Travelled along Y-axis in meters (m)	Time taken (sec)
Goal 1	42.778	31.055	419.448
Goal 2	10.831	30.813	274.789

Table II represents the distance travelled and time taken by the robot to reach goals as shown in fig 2 and 3. As per the table, the robot takes 419 seconds to reach goal 1 since it has to travel a long distance. Goal 2 is near to the start point of the robot compared to goal 1. The final system consisted of a robot navigating autonomously in the indoor environment and an avatar for human-robot interaction. The robot needs a prebuilt map to navigate autonomously. The April tags are stuck to the doors of the laboratories and their locations are known with respect to the map frame. The navigation stack is provided with the points which are closer to April tags. The Intel real sense camera can detect the April tag effectively and relay information to Unity which would trigger a dialog in IBM Watson explaining the people about the experiments that are going inside these labs. The entire system is not as it is expected to be. There are issues that need to be addressed. The local planner's parameters had to be tuned to match the environment. Local planners like DWA, TEB, Eband were tested and finally TEB planner was chosen since this planner is smooth and executes trajectory avoiding obstacles. Care should be taken to keep a minimum distance of 3-4 ft when giving tours so that the robot wouldn't crash when it is driving backwards.

TABLE III: Tour guide trials

Tour guide robot trials	**Time Taken (secs.)**
Trial 1	10686
Trial 2	10069
Trial 3	9847

From the table III, it is observed that all the three trials took approximately the same time for the robot to provide the tour.

VI. Conclusion and Future Works

This experiment deals with the system integration of an Autonomous Tour Guide robot. The self-driving nature of the robot is implemented using the ROS navigation stack. The Avatar system is handled by Unity and IBM Watson is used for limited interaction with humans. The robot in this experiment is able to detect obstacles and drive past them by avoiding a collision. The environment contains April tags at known locations which are stuck to the laboratories and the robot drives from one April tag location to another describing the experiments that are going behind those doors to the people. IBM Watson, Unity, and ROS interact using existing libraries that use web sockets. The robot's effectiveness in operating as a tour guide robot is tested on the 3rd floor of KGCOE, RIT in the department of Electrical and Micro electrical engineering which consists of many laboratories with corridors. The robot is integrated with multiple systems. They are ROS navigation stack, Unity's Avatar system, and IBM Watson for speech recognition system. For the robot to provide a tour of the environment, It has to be supported with visual fiducial markers such as April tags. However, In RIT, each laboratory has its own labels that are stuck beside the door. The robot can use OpenCV techniques to detect those labels and provide information about the experiments going in those labs. The camera mounted on the robot could be adjusted so as to not turn the robot as much to detect tags. All subsystems on the robot are running on a laptop. ROS and Unity 3D game engine are battery-hungry applications. In the future, ROS could be run on a single-board computer like Nvidia Jetson Nano, which will handle all the applications needed to drive the robot autonomously. The laptop can run the Unity game engine and visualization tools of ROS.

Acknowledgment

The authors are grateful to the staff of Multi-Agent Bio-Robotics Laboratory (MABL), the CM Collaborative Robotics (CMCR) Lab, and the Electrical Engineering Department at RIT for their valuable inputs. This material is based upon work partially supported by the National Science Foundation under Award No. DGE-2125362. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Z. Xuexi, L. Guokun, F. Genping, X. Dongliang and L. Shiliu, "SLAM Algorithm Analysis of Mobile Robot Based on Lidar," 2019 Chinese Control Conference (CCC), 2019, pp. 4739-4745, doi: 10.23919/ChiCC.2019.8866200.

[2] E. Olson, "AprilTag: A robust and flexible visual fiducial system," 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 3400-3407, doi: 10.1109/ICRA.2011.5979561.

[3] S. E. Hadji, T. H. Hing, M. S. M. Ali, M. A. Khattak and S. Kazi, "2D occupancy grid mapping with inverse range sensor model," 2015 10th Asian Control Conference (ASCC), 2015, pp. 1-6, doi: 10.1109/ASCC.2015.7244705

[4] I. Naotunna and T. Wongratanaphisan, "Comparison of ROS Local Planners with Differential Drive Heavy Robotic System," 2020 International Conference on Advanced Mechatronic Systems (ICAMechS), 2020, pp. 1-6, doi: 10.1109/ICAMechS49982.2020.9310123.

[5] D. Fox, W. Burgard and S. Thrun, "The dynamic window approach to collision avoidance", IEEE Robotics & Automation Magazine, vol. 4, no. 1, pp. 23-33, March 1997.

[6] Harihar Subramanyam et al. "SKYCALL", [online]. https://senseable.mit.edu/skycall/

[7] Yuri Kageyama, "Honda robot Asimo makes balky tour guide", [online]. https://www.usatoday.com/story/driveon/2013/07/06/honda-robot-asimo/2494143/

[8] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," [1993] Proceedings IEEE International Conference on Robotics and Automation, 1993, pp. 802-807 vol.2, doi: 10.1109/ROBOT.1993.291936.

[9] Automaticaddison, "How to Create a Map for ROS From a Floor Plan or Blueprint", [online]. https://automaticaddison.com/how-to-create-a-map-for-ros-from-a-floor-plan-or-blueprin

[10] S. Chan, P. Wu and L. Fu, "Robust 2D Indoor Localization Through Laser SLAM and Visual SLAM Fusion," 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Miyazaki, Japan, 2018, pp. 1263-1268.

[11] Babinec, Andrej & Jurišica, Ladislav & Hubinský, Peter & Duchoň, František. (2014). Visual Localization of Mobile Robot Using Artificial Markers. Procedia Engineering. 96. 10.1016/j.proeng.2014.12.091.

[12] A. Al-Wazzan, R. Al-Farhan, F. Al-Ali and M. El-Abd, "Tour-guide robot," 2016 International Conference on Industrial Informatics and Computer Systems (CIICS), Sharjah, 2016, pp. 1-5.

[13] Woojin Chung, Gunbee Kim, Munsang Kim and Chongwon Lee, "Integrated navigation system for indoor service robots in large-scale environments," IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, New Orleans, LA, USA, 2004, pp. 5099-5104.

[14] A. B. S. H. M. Saman and A. H. Lotfy, "An implementation of SLAM with extended Kalman filter," 2016 6th International Conference on Intelligent and Advanced Systems (ICIAS), Kuala Lumpur, 2016, pp. 1-4.

[15] S. Rawat, P. Gupta and P. Kumar, "Digital life assistant using automated speech recognition," 2014 Innovative Applications of Computational Intelligence on Power, Energy and Controls with their impact on Humanity (CIPECH), Ghaziabad, 2014, pp. 43-47.

[16] ROS, Robot Operating System" [online]. https://www.ros.org/about-ros/

[17] Madgwick, Sebastian & Harrison, Andrew & Vaidyanathan, Ravi. (2011). Estimation of IMU and MARG orientation using a gradient descent algorithm. IEEE ... International Conference on Rehabilitation Robotics : [proceedings]. 2011. 5975346. 10.1109/ICORR.2011.5975346.

[18] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. 2005. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press.

[19] Tony, Pigram, "Create a 3D Digital Human with IBM Watson Assistant and Unity3D" [Online]. https://developer.ibm.com/recipes/tutorials/create-a-3d-digital-human-with-ibm-watson-assistant-and-unity3d/