International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 10 | Oct 2019 www.irjet.net p-ISSN: 2395-0072

Intelligent Autonomous Wheelchair for Indoor Navigation

Indushekhar Prasad Singh[1], Ashish Patel[2]

[1,2]*Institute for Systems Research, University of Maryland, College Park, USA*

Abstract - *Manually controlling an electric powered wheelchair is a challenging task, especially for dependent individuals with movement disabilities. A solution to improve user's mobility is to implement an autonomous navigation algorithm on powered wheelchairs. This paper presents a unique approach to achieve inexpensive and robust semi-autonomous assisted navigation for existing powered wheelchairs. Our prototype wheelchair platform is capable of localization, as well as robust obstacle avoidance, using only a commodity RGB-D camera and wheel odometry. It uses the localization data to autonomously plan and traverse to the desired goal locations in a known indoor environment.*

Key Words: Autonomous wheelchair, Differential drive, Indoor navigation, Path planning, Smart wheelchair

1. INTRODUCTION

People with cognitive or motor or sensory impairment due to a disability or a disease rely on powered wheelchairs (PWs) for their mobility needs. Since people with both upper-extremity and lower-extremity disabilities cannot use a traditional joystick to navigate the wheelchair, they use alternative control systems like head joysticks, chin joysticks, sip-n-puff, and thought control. In case of children (toddlers and pre- schoolers), a joystick control is not a viable solution. Even with a joystick control, many users face difficulties with daily maneuvering tasks. Such issues can be resolved by implementing an automated navigation system for the wheelchair. A robotic navigation system for PWs would allow the chairs to self-navigate in home and workplace environments which would drastically improve user's mobility. Currently, there are over 5 million PWs in use in Canada and the United States and an estimated 20 million PWs have been deployed in G-20 countries. It has been predicted that in the year 2018 an estimate of $3.9 billion will be spent on PWs in the US itself. Thus, the design of a cost-effective navigation system which can be retrofitted to the existing PWs is of great importance both socially and economically. Such a technology will substantially increase the quality of life and societal engagement for this segment of society. This paper focuses on robotic path planning in dynamic indoor environment. Path planning in mobile robots is broadly divided as global and local. A global path planner takes the map of the known environment (static obstacles) as input and generates a path even before the robot starts moving. The planner operates on a pre-defined cost map to find the minimum cost plan from the start

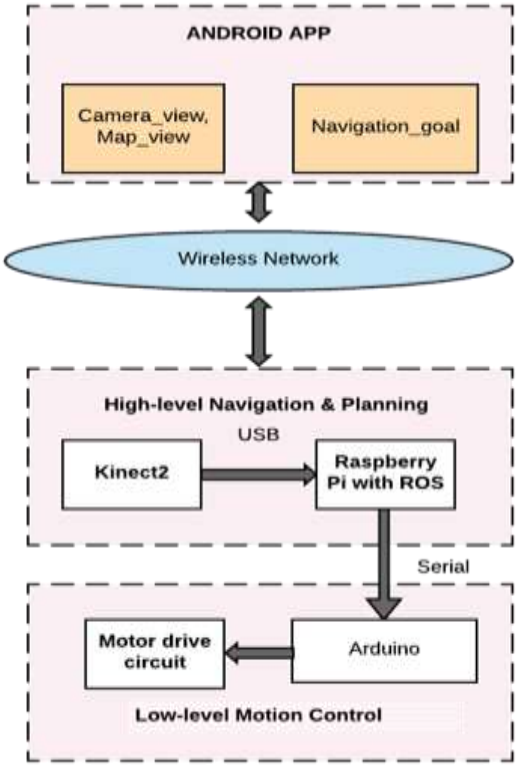

Fig -1: Proposed System Structure

position to the goal position. However, a local path planner uses the sensors (RGB-D camera in this case) to detect obstacles and plan a path to reach the nearest waypoint (points achieved from global path planner). It computes appropriate velocity commands for the robot joints, to traverse the current segment of the global path avoiding obstacles, by combining sensory and odometry data with both global and local cost maps. In other words, a local path planner complements a global path planner by taking into consideration the uncertainties encountered during path traversal by a robot. The uncertainties include state-transition uncertainties caused due to slippage of wheels, and obstacle uncertainties generally caused due to the presence of dynamic obstacles like humans or doors in an indoor environment. [3-5]

The solution presented in this paper has been verified using V-REP integrated with MATLAB. The final implementation will be on Robot Operating System (ROS) installed in a Raspberry Pi. ROS is an open source framework which provides a means for point-to-point communication and a network that connects to all sub-processes. The system structure diagram proposed in this paper is shown in Fig 1. The system uses a Kinect 2 RGB-D camera to collect information about the local environment.

The high-level computation involves the global path planner which calculates the path, and a local path planner which generates the robot joint velocities required to traverse the local paths. These velocity values are sent to an easily accessible Arduino micro-controller via serial communication. The micro-controller converts these values to Pulse Width Modulation (PWM) signals, and then transmits these signals to the motor driver circuit which amplifies the signal in order to rotate the wheelchair motors at the desired speed. Now, the system is composed of varying electrical components like Raspberry Pi, Arduino micro-controller, motor driver, and wheelchair motors. All these components have different power voltages and current requirements. Thus, a separate power supply circuit has been designed to provide the appropriate power to the different components.

2. RELATED RESEARCH WORK

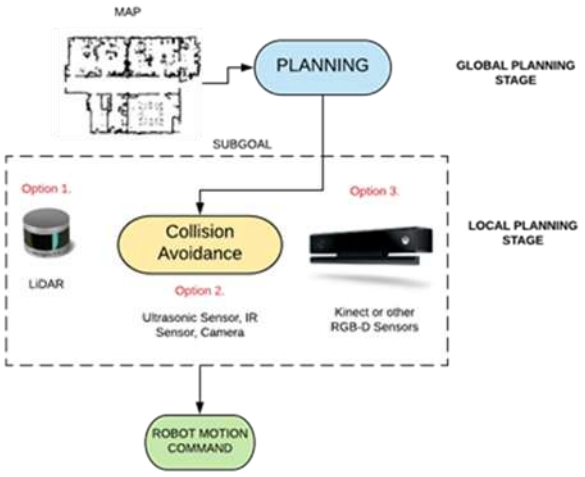

Fig -2: Obstacle avoidance

A wide variety of assisted wheelchair navigation systems have been developed over the past 30 years. Low-cost solutions are generally either limited and semi-autonomous [6] [11], or require an external localization system [8], or lack global localization planning capabilities. Maya Burhanpurkar et al., 2017 [5] presented a cost-effective and robust autonomous navigation system for existing PWs.[7] According to the paper, based on an inexpensive sensor suite (an RGB-D sensor and wheel odometry), the various modules of the system (Simultaneous Localization and Mapping (SLAM), navigation, and door traversal) functioned synergistically to enable reliable operation under real-world conditions. Zhengang Li et al., 2017 [9] presented a wheelchair which adopts two-wheel differential control structure and used RGB-D camera to perceive the environment. [10] The position and orientation of wheelchair was estimated by Adaptive Monte Carlo Localization (AMCL) algorithm. A* algorithm was applied for global path planning and Dynamic Window Approach (DWA) algorithm was used for local path planning.[12] Their experimental results were consistent with the experimental expectation.

Our approach in this paper is inspired from the latter approach. As such for obstacle avoidance there were many approaches adopted using the different combination of sensors. Most of these approaches were based on the system given in Fig 2. [13-15] Our system is a commercial product, so we wanted it to be cheap, robust and reliable. Thus, we selected Kinect sensor for detecting obstacles, over LiDAR sensor which would have made the system expensive. Ultrasonic and IR sensors, though cheaper, were rejected as they are not robust and reliable.

3. METHOD

3.1 System Component

Fig -3: Autonomous Wheelchair Prototype

Setup: The setup consists of an autonomous wheelchair prototype with Microsoft Kinect v2 sensor mounted on an elevated frame to ensure an adequate field of view. Raspberry Pi is being used for high level planning, along with Arduino as a controller. The base of the chair is rectangular with an approximate weight of 10 kg. The base is adaptive to fit different types of wheelchairs (Fig 3). The map (based on Children's National Medical Center, Washington DC) is pre-defined with a possibility of presence of dynamic obstacles. The user has the freedom to choose up to N locations on the map, in order of traversal, using an interactive mobile application which would interact with the Raspberry Pi on-board the smart system. [16-19]

Software: The global planning algorithm (A*) has been currently developed in MATLAB. A* is an informed search algorithm, or a best- first search, meaning that it solves problems by searching among all possible paths to the goal for the one that incurs the smallest cost and among these paths it first considers the ones that appear to lead most

quickly to the solution. The algorithm considers the differential drive constraints while planning the path. As our system is a rectangular robot with front actuated differential drive, its line of action does not coincide with the center of the plat- form. So, the first step in developing the differential drive constraint is to study the kinematic constraints of a standard fixed wheel. [20-23]. A fixed standard wheel has no vertical axis of rotation for steering. Its angle to the chassis is thus fixed, and it is limited to motion back and forth along the wheel plane and rotation around its contact point with the ground plane. Fig 4 depicts a fixed standard wheel and indicates its pose relative to the robots local reference frame. The position of the wheel is expressed in polar coordinates by distance l and angle α. The angle of the wheel plane relative to the chassis is denoted by β, which is fixed since the fixed standard wheel is not steerable. The wheel, which has radius r, can spin over time, and so its rotational position around its horizontal axle φ(t) is a function of time.

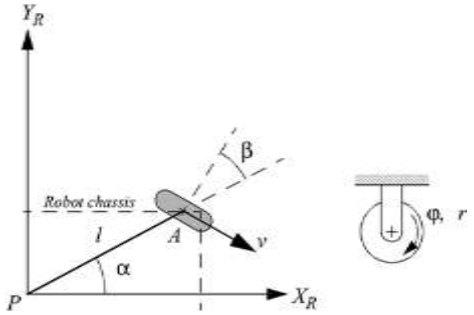

Fig -4: Fixed Standard Wheel

The rolling constraint for this wheel enforces that all motion along the direction of the wheel plane must be accompanied by the appropriate amount of wheel spin so that there is pure rolling at the contact point:

$$A(\alpha,\beta,l)R(\theta)\dot{\zeta}_I - r\dot{\phi} = 0 \qquad (1)$$

$$A = \begin{bmatrix} sin(\alpha+\beta) & -cos(\alpha+\beta) & -lcos(\beta) \end{bmatrix}$$

The first term of the sum denotes the total motion along the wheel plane. The three elements of the vector on the left represents mappings from each of $\dot{x}$, $\dot{y}$, $\dot{\theta}$ to their contributions for motion along the wheel plane. Note that the $R(\theta)\dot{\zeta}I$ term is used to transform the motion parameters that are in the global reference frame into motion parameters in the local reference frame. This is necessary because all other parameters in the equation, *α*, *β*, *l*, are in terms of the robot's local reference frame. This motion along the wheel plane must be equal, according to this constraint, to the motion accomplished by spinning the wheel, $r\dot{\varphi}$.[1-4]

Now, given the wheelchair has 4 fixed standard wheels, we can now compute the kinematic constraints of the system. The key idea is that each wheel imposes constraints on robot motion, and so the process is simply one of appropriately combining all of the kinematic constraints arising from all of the wheels based on the placement of those wheels on the robot chassis. So, the rolling constraints of all wheels have been collected in a single expression:

$$J_1 R(\theta)\dot{\zeta}_I - J_2\dot{\phi} = 0 \qquad (2)$$

This expression bears a strong resemblance to the rolling constraint of a single wheel, but substitutes matrices in lieu of single values, thus taking into account all wheels. J_2 is a constant diagonal matrix whose entries are radii r of all standard wheels. J_1 denotes a matrix with projections for all wheels to their motions along their individual wheel planes:

$$J_1 = \begin{bmatrix} sin(\alpha_1+\beta_1) & -cos(\alpha_1+\beta_1) & -l_1cos(\beta_1) \\ sin(\alpha_2+\beta_2) & -cos(\alpha_2+\beta_2) & -l_2cos(\beta_2) \\ sin(\alpha_3+\beta_3) & -cos(\alpha_3+\beta_3) & -l_3cos(\beta_3) \\ sin(\alpha_4+\beta_4) & -cos(\alpha_4+\beta_4) & -l_4cos(\beta_4) \end{bmatrix}$$

In summary, equation (2) represents the constraint that all standard wheels must spin around their horizontal axis an appropriate amount based on their motions along the wheel plane so that rolling occurs at the ground contact point. In case of our system, the parameters for front left wheel are α1 = 36.57 degrees, β1 = 53.43 degrees, l1 = 0.52 meters, and r1 = 0.0762 meters. The parameters for front right wheel are α2 = −36.57 degrees, β2 = −53.43 degrees, l2 = 0.52 meters, and r2 = 0.0762 meters. The parameters for back right wheel are α3 = −155.37 degrees, β3 = 65.37 degrees, l3 = 0.46 meters, and r3 = 0.0762 meters. The parameters for back left wheel are α4 = 155.37 degrees, β4 = −65.37 degrees, l4 = 0.46 meters, and r4 = 0.0762 meters. Using these parameters, we get

$$J_1 = \begin{bmatrix} 1 & 0 & -0.3086 \\ -1 & 0 & -0.3086 \\ -1 & 0 & -0.1905 \\ 1 & 0 & -0.1905 \end{bmatrix}$$

$$J_2 = \begin{bmatrix} 0.0762 & 0 & 0 & 0 \\ 0 & 0.0762 & 0 & 0 \\ 0 & 0 & 0.0762 & 0 \\ 0 & 0 & 0 & 0.0762 \end{bmatrix}$$

It should be noted that only the front two wheels of the robot are actuated, that is, $\dot{\varphi}_3 = \dot{\varphi}_4 = 0$. The A* algorithm implemented in MATLAB first computes the velocity of the robot in the world (inertial) frame using equation (2) and then computes the new position of the robot by taking its integral with respect to time. The final equation employed in the MATLAB code is:

$$\zeta_{I_{new}} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \zeta_{I_{old}} + \int_0^{\Delta t} R^{-1}(\theta) J_1^{-1} J_2 \dot{\phi} \quad (3)$$

From equation (3), it can be inferred that the action space that has been used by the algorithm are the wheel velocities.

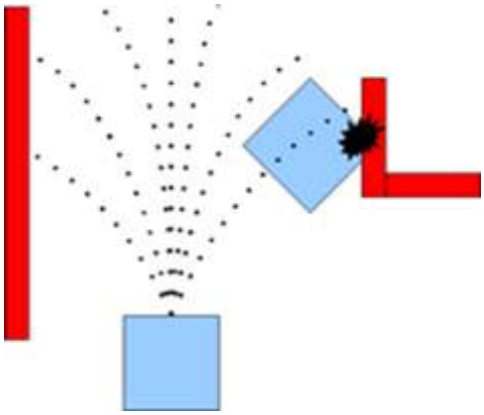

Fig -5: Dynamic Window Approach

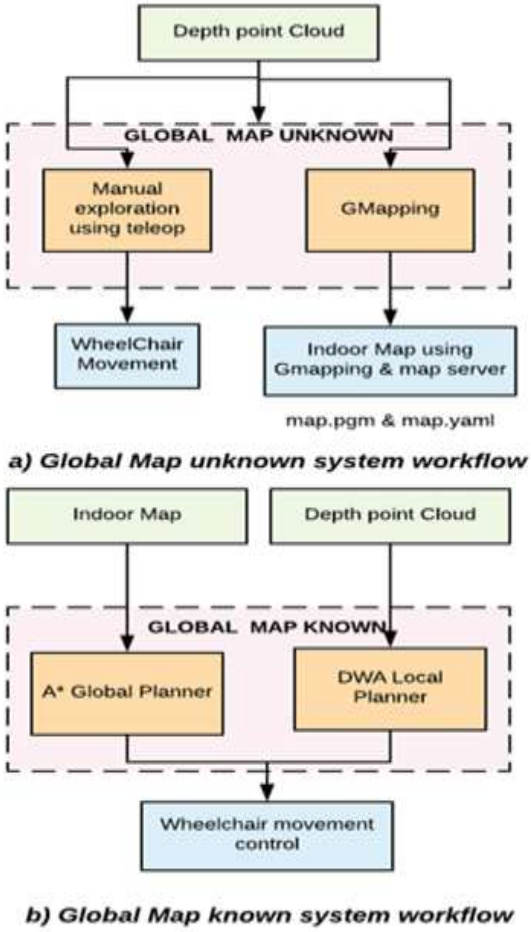

Fig -6: Workflow Chart

For local path planning, Dynamic Window Approach as shown in fig 5 has been used. It is a collision avoidance strategy for mobile robots developed by Dieter Fox, Wolfram Burgard, and Sebastian Thrun [7]. It consists of two main components, first generating a valid search space, and second selecting an optimal solution in the search space. The planner in ROS is implemented by using the package dwa local planner. This package provides a controller that drives a mobile base in the plane. This controller serves to connect the path planner to the robot. Using a map, the planner creates a kinematic trajectory for the robot to get from a start to a goal location. Along the way, the planner creates a local value function, represented as a grid map. This value function encodes the costs of traversing through the grid cells. The controllers' job is to use this value function to determine $dx^{\cdot}$, $dy^{\cdot}$, $d\theta^{\cdot}$[24-27].

3.1 System Workflow

This paper assumes that the map of the indoor environment is known. A* algorithm takes the map as input to carry out global path planning. DWA algorithm will be deployed for obstacle avoidance in real time based on depth point cloud images from Kinect camera. Combining the results of the two path planning algorithms, ROS will establish a shortest path to the target point on the premise that there is no collision. Workflow chart of the algorithm is shown in Fig 6.

4. SIMULATION

4.1 Gazebo

Creating Robotic Model: To simulate the whole system on Gazebo, the design of a robotic model representing our system is needed, as well as the indoor environment on which the robot is supposed to be simulated is required. To design this on Gazebo there are two options, using Unified Robot Description Format (URDF) which is an XML format for representing all the elements of robot model, or using Simulation Description Format (SDF) which is also an XML format that describes objects and environments for robot simulators, visualization, and control.

While learning to build a model on Gazebo, we came across many ways to design model with URDF and SDF. But we first had to make a choice as to which format Was to be used. So, we investigated the properties of the two formats. For instance, URDF can be used for specifying only the kinematic and dynamic properties of a single robot in isolation. It cannot specify the pose of the robot itself within a world. It is also not a" universal" description format since it cannot specify inertia, friction and other properties. Moreover, URDF cannot be used to model the environment. SDF solves all these problems. It is a complete description of everything from the world level to the robot level. It is highly scalable and extremely easy to add and modify elements. However, SDF is best suited only for use of model in gazebo environment. In case one wants to integrate the robot with ROS (to control the robot), it is better to create the model in URDF format as there are gazebo plug-in supports which are only available in URDF format. Thus, we decided to go with URDF, as our final goal is to integrate the simulation with ROS. The robot

model and the environment developed in Gazebo are shown in Fig 7 and Fig 8 respectively.

ROS Control: ROS Control has been developed by the ROS community to allow simple access to different actuators. ROS control is an API to control all parts of a robotic model. Using this standard API, the controller code is separated from the actuator code. For example, one can write a new controller or use the built- in one to implement a control strategy and test it on hardware without changing a single line of code.

It also provides functionality like real time capabilities, which allows the user to run control loops at any desired frequency; a simple manager, which gives access to the actuators and handles resource conflicts (for instance, one can assign different controllers to different actuators); a safety interface, that knows the hardware limitation of the joints and ensures that the commands sent to the actuators are between their limits; and off-the-shelf controller that is readily available to use as per the requirement.

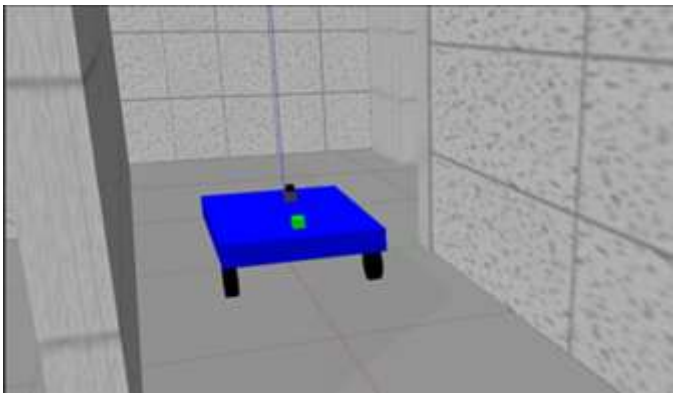

Fig -7: Model of the Robot (Gazebo)

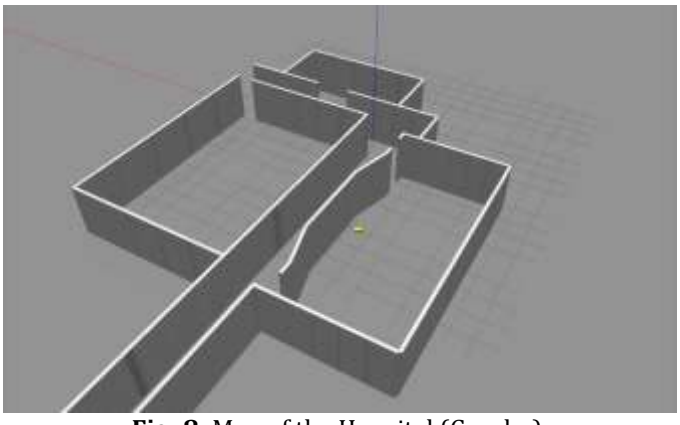

Fig -8: Map of the Hospital (Gazebo)

ROS and Gazebo Integration: We integrated ROS with Gazebo through the gazebo ROS package. This package provides a Gazebo plug-in module that allows bidirectional communication between Gazebo and ROS. Also, simulated sensor and physics data can be streamed from Gazebo to ROS, and actuator commands can be streamed from ROS back to Gazebo through this package. We have provided properties like friction, inertia and collision properties to our URDF model after integration with ROS.

GMapping: The gmapping package pro- vides laser-based SLAM, as a ROS node called slam gmapping. Using this node, 2D occupancy grid map (like a building floor plan) was created from laser and pose data collected by the wheelchair model (Fig 9). As a future work, we are planning to incorporate Kinect sensor into the robot model. There is a package called point cloud to laser scan data which will be used along with gmapping, to get the 2D occupancy grid map (OGM).

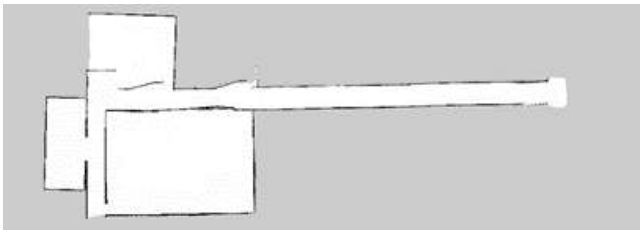

Fig -9: OGM using gmapping and teleop

4.2 V-REP

Creating Robotic Model: Similar to Gazebo, before running any simulations, it is important to develop the model of the robot and create the environment in V-REP. The model developed for ROS cannot be transferred to V-REP directly. The same is true for the environment. So, some time was spent on developing the model of the robot and creating the environment independently on V-REP. As V-REP takes care of all the dynamic properties of the objects on its own, it was easy to develop the model of the robot in V-REP as compared to Gazebo. There were a few glitches such as the integration of the motors with the base of the robot. V-REP does not allow direct connections. So, a force sensor was introduced in between to link the motor supports with the base of the robot. The final robot model is shown in Fig 10.

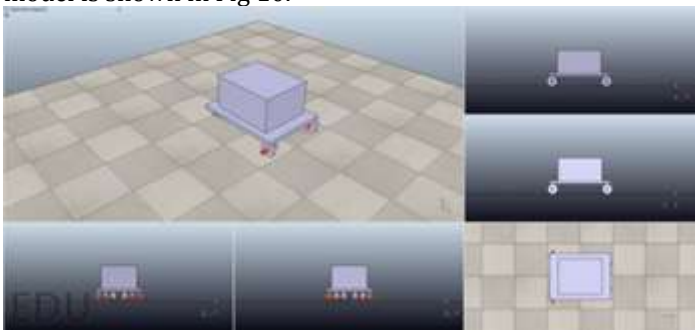

Fig -10: Model of the Robot (V-REP)

Similarly, the indoor environment was developed in V-REP (Fig 11). This took a lot of time as compared to Gazebo because V-REP has fixed dimensions for its walls and they cannot be modified. But it was finally created with good accuracy.

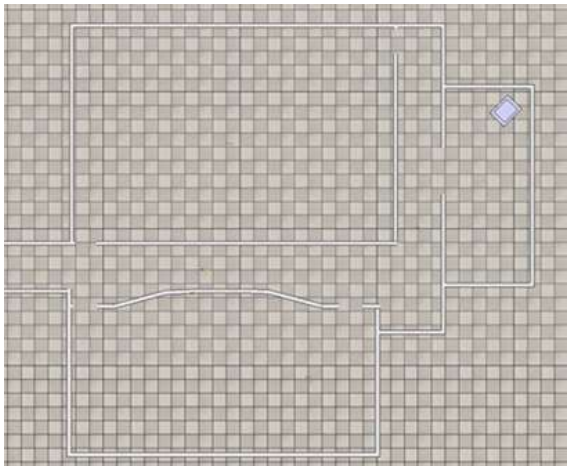

Fig -11: Map of the Hospital (V-REP)

MATLAB Control: As described before in section III, the A* algorithm was developed in MATLAB. This algorithm worked perfectly for safe paths, that is, paths not involving narrow doorways. But, when trying to pass through narrow doorways, it took infinitely long time to compute the path. So, we looked into another planning algorithm, RRT.

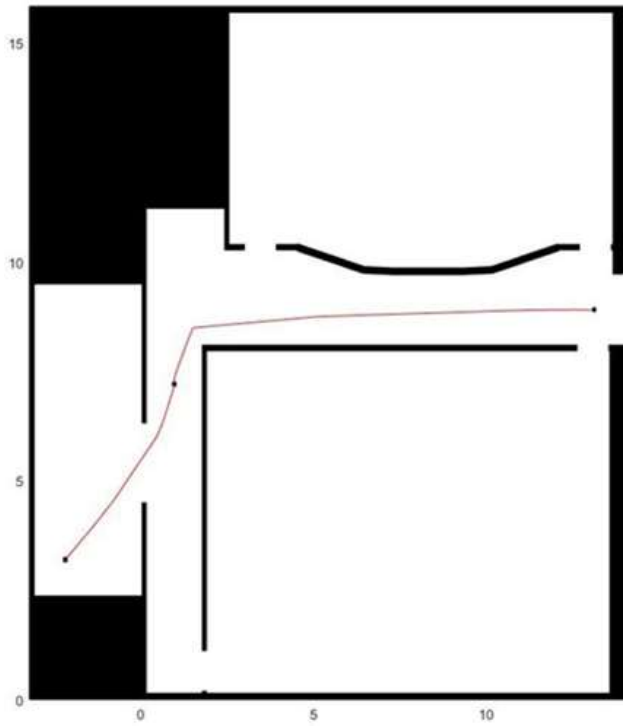

Fig -12: Path with no Narrow Passage

A rapidly exploring random tree (RRT) is an algorithm designed to efficiently search non-convex, high-dimensional spaces by randomly building a space-filling tree. The tree is constructed incrementally from samples drawn randomly from the search space and is inherently biased to grow towards large unexplored areas of the problem. This algorithm was employed as it can easily handle problems with obstacles and differential constraints. It also gave an output in a shorter time duration as compared to A*. For some time, we considered using weighted A* but as it does not give an optimal path, this approach was rejected.

MATLAB and V-REP Integration: The V- REP built-in path planner uses RRT* to compute an optimal path. So, instead of writing our own code all over again, we went ahead with the built-in planner. Two paths were generated in this way with varying difficulty of traversal and read into MATLAB using remote API functions. Two paths on maps focusing on the region of interest are shown in Fig 12 and Fig 13.

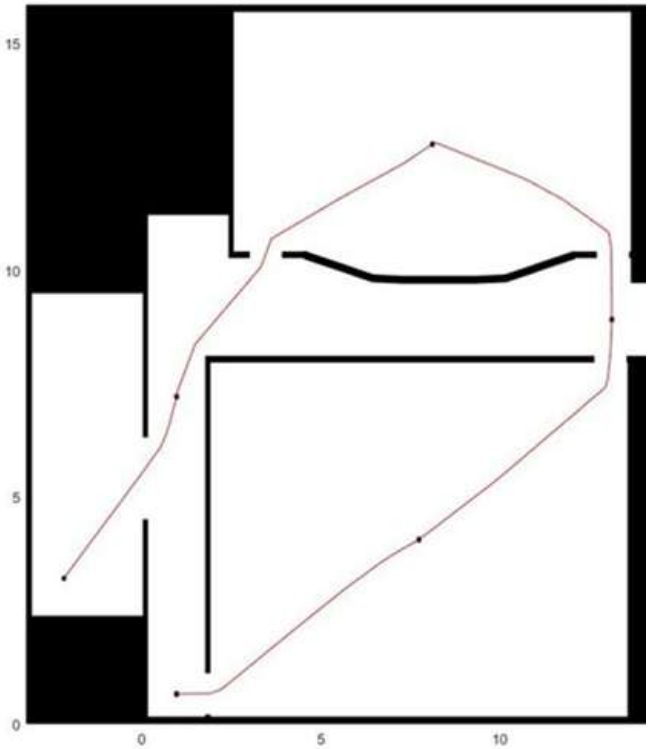

Fig -13: Path with Narrow Passages

It should be noted that these paths were not generated on a single try. As the paths are generated using RRT*, multiple paths were generated, and the most optimum path was selected manually. When the simulation was run in V- REP for these paths, the robot was easily able to traverse path 1 (Fig 12) without hitting any obstacles. But the same cannot be said for the other path. The robot was able to reach the door but could not pass through it. After running the simulation several times, it was concluded that the issue was not with the controller but the way the paths are designed. These are global paths and thus do not take into account the state-space uncertainties. We had decided not to use the local planner yet, so we had to figure out a way to pass through the doors using the global planner only. This was done by introducing some new waypoints along the path which ensured that the robot would be in a particular configuration before approaching the door. An example path is shown in Fig 14. As can be seen, the path ensures that the traversal through the doors

is in a straight line. So, when the simulation was run with this path, the robot was able to pass the narrow doorways as the state-space uncertainty was low while such traversal.

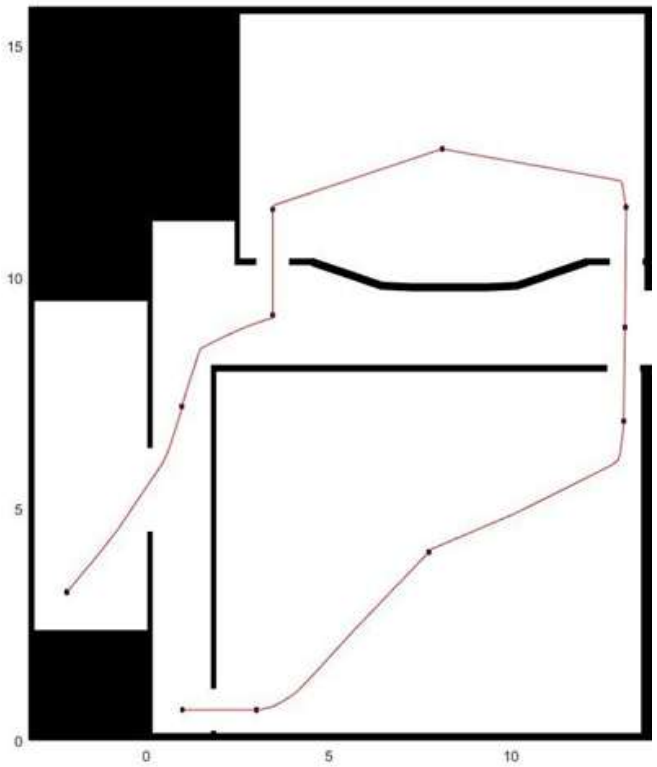

Fig -14: Path with Narrow Passages

5. RESULT

We have made tremendous progress in this project after our last visit to the hospital. Once we had the dimensions of the real robot and the indoor environment, the robotic model and the map were developed in Gazebo (Figure 7 and Figure 8) as well as in V-REP (Figure 10 and Figure 11). The A* algorithm developed in MATLAB was updated to accommodate the real robot dimensions and the hospital map. This updated algorithm has been successfully verified using the simulation on V-REP. The robotic model is able to traverse the computed path during simulation in V-REP. The same has not been verified in Gazebo due to some inertia definition issues with the developed model.

The local planner has not been used yet to traverse the path as no dynamic obstacles were introduced yet. But, the algorithm for the same has been extensively explored. If the built-in planners are to be used, then the only changes required are the integration of the robotic models with Kinect v2 sensor. The main component left is to run our algorithms on the real robot. From the simulation results, we can confidently state that it is not an impossible task now. The final path traversed by the robot are shown in Fig 15 and Fig 16 respectively.

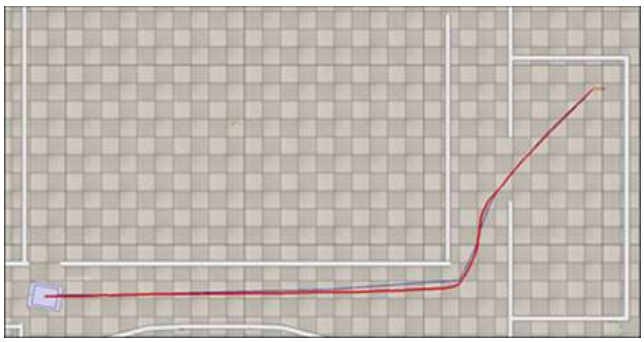

Fig -14: Simulation for Path 1

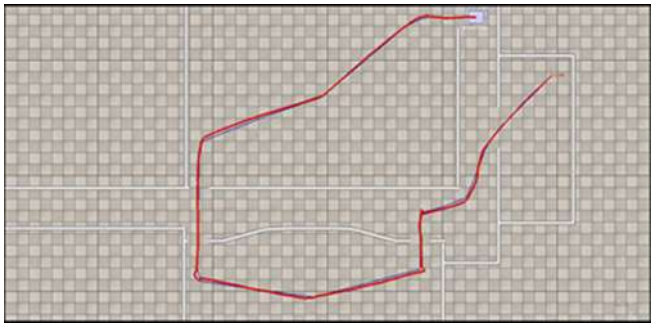

Fig -15: Simulation for Path 2

As can be observed that the robot trajectory (red) does not exactly coincide with the path generated (blue). This is an evidence of state- space uncertainty occurring while traversal. This has been taken care of by the controller developed in V-REP.

6. CONCLUSIONS

There is a long way to go before this project can be said to have been completed. But, our rate of progress is good. We plan to work on the implementation of our algorithms on the real robot during the summer break and if needed, incorporate a local path planner in MATLAB and V-REP.

In case of ROS and Gazebo, we were not able to incorporate move base package which is used as a navigation stack for a mobile robot. After incorporating it, we will focus on simulating robot to move through narrow pathway and doors without colliding, using point cloud data acquired from the Kinect sensor.

ACKNOWLEDGEMENT

The authors would like to thank Institute for Systems Research, James Clark School of Engineering to carry out the experiment.

REFERENCES

[1] Suresh A., Arora C., Laha D., Gaba D., Bhambri S. (2019) Intelligent Smart Glass for Visually Impaired Using Deep Learning Machine Vision Techniques and Robot Operating System (ROS). In: Kim JH. et al. (eds) Robot Intelligence Technology and

Applications 5. RiTA 2017. Advances in Intelligent Systems and Computing, vol 751. Springer, Cham

[2] Kumar V.S., Aswath S., Shashidhar T.S., Choudhary R.K. (2017) A Novel Design of a Full Length Prosthetic Robotic Arm for the Disabled. In: Kim JH., Karray F., Jo J., Sincak P., Myung H. (eds) Robot Intelligence Technology and Applications 4. Advances in Intelligent Systems and Computing, vol 447. Springer, Cham

[3] Suresh A., Gaba D., Bhambri S., Laha D. (2019) Intelligent Multi-fingered Dexterous Hand Using Virtual Reality (VR) and Robot Operating System (ROS). In: Kim JH. et al. (eds) Robot Intelligence Technology and Applications 5. RiTA 2017. Advances in Intelligent Systems and Computing, vol 751. Springer, Cham

[4] Bhardwaj S. et al. (2019) Isolated BioRegenerative System for Vertical Farming Through Robots in Space Explorations. In: Kim JH. et al. (eds) Robot Intelligence Technology and Applications 5. RiTA 2017. Advances in Intelligent Systems and Computing, vol 751. Springer, Cham

[5] Burhanpurkar, Maya, Mathieu Labb, Charlie Guan, Fra- nois Michaud, and Jonathan Kelly. "Cheap or Robust? The practical realization of self-driving wheelchair tech- nology." In Rehabilitation Robotics (ICORR), 2017 Inter- national Conference on, pp. 1079-1086. IEEE, 2017.

[6] Suresh A., Laha D., Gaba D., Bhambri S. (2019) Design and Development of Innovative Pet Feeding Robot. In: Kim JH. et al. (eds) Robot Intelligence Technology and Applications 5. RiTA 2017. Advances in Intelligent Systems and Computing, vol 751. Springer, Cham

[7] Fox, Dieter, Wolfram Burgard, and Sebastian Thrun. "The dynamic window approach to collision avoidance." IEEE Robotics & Automation Magazine 4, no. 1 (1997): 23-33.

[8] Kim, Bong Keun, Hideyuki Tanaka, and Yasushi Sumi. "Robotic wheelchair using a high accuracy visual marker lentibar and its application to door crossing navigation." In Robotics and Automation (ICRA), 2015 IEEE Interna- tional Conference on, pp. 4478-4483. IEEE, 2015.

[9] Li, Zhengang, Yong Xiong, and Lei Zhou. "ROS-Based Indoor Autonomous Exploration and Navigation Wheelchair." In Computational Intelligence and Design (ISCID), 2017 10th International Symposium on, vol. 2, pp. 132-135. IEEE, 2017.

[10] Suresh A. et al. (2017) An Advanced Spider-Like Rocker-Bogie Suspension System for Mars Exploration Rovers. In: Kim JH., Karray F., Jo J., Sincak P., Myung H. (eds) Robot Intelligence Technology and Applications 4. Advances in Intelligent Systems and Computing, vol 447. Springer, Cham

[11] Shen, Jiajun, Bin Xu, Mingtao Pei, and Yunde Jia. "A low-cost tele-presence wheelchair system." In Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pp. 2452-2457. IEEE, 2016.

[12] Krishnan, A.B., Aswath, S., Udupa, G.: Real Time Vision Based Humanoid Robotic Platform for Robo Soccer Competition. In: Proceedings of International Conference on Interdisciplinary Advances in Applied Computing, India, October 10-11 (2014)

[13] Suresh A., Sridhar C.P., Gaba D., Laha D., Bhambri S. (2019) Design and Development of Intelligent Self-driving Car Using ROS and Machine Vision Algorithm. In: Kim JH. et al. (eds) Robot Intelligence Technology and Applications 5. RiTA 2017. Advances in Intelligent System

[14] Alajlan, Abrar, Khaled Elleithy, Marwah Almasri, and Tarek Sobh. "An Optimal and Energy Efficient Multi-Sensor Collision-Free Path Planning Algorithm for a Mo- bile Robot in Dynamic Environments." Robotics 6, no. 2 (2017): 7.

[15] Aswath Suresh, Sri harsha,Kolluri Surya,Debrup Laha,Dhruv Gaba,Shivam Bhardwaj,Vinay Teja,Siddhant Bhambri and Suvaansh Bhambri, "Innovative Human Mars Mission with Vertical Farming" ,Mars Society Convention,Marspapers,2017

[16] Aswath Suresh, Gautam Ranjan,Sri Harsha, Kolluri Surya, Adarsh Ranjan, Nitin Ajithkumar, Sajin Sabu,Vinay Teja, Abel Varghese David, Ganesha Udupa, "Innovative Low Cost Mars Flyby Spacecraft for Safe Interplanetary Human Mission",Mars Society Convention, Marspapers,2016

[17] Cavanini, Luca, Flavia Benetazzo, Alessandro Freddi, Sauro Longhi, and Andrea Monteriu. "SLAM-based au- tonomous wheelchair navigation system for AAL scenar- ios." In Mechatronic and Embedded Systems and Ap- plications (MESA), 2014 IEEE/ASME 10th International Conference on, pp. 1-5. IEEE, 2014.

[18] Aswath, S., Tilak, C.K., Suresh, A., Udupa, G.: Human Gesture Recognition for Real-Time Control of Humanoid Robot. In: Proceedings of International Conference on Advances in Engineering and Technology, Singapore, March 29-30 (2014)

[19] Leaman, Jesse, and Hung Manh La. "A comprehensive review of smart wheelchairs: Past, present, and future." IEEE Transactions on Human-Machine Systems 47, no. 4 (2017): 486-499.

[20] Aswath, S., Tilak, C.K., Sengar, A., Udupa, G.: Design and development of Mobile Operated Control

System for Humanoid Robot. In: Advances in Computing, 3rd edn., vol. 3, pp. 50– 56

[21] Morales, Yoichi, Nagasrikanth Kallakuri, Kazuhiro Shi- nozawa, Takahiro Miyashita, and Norihiro Hagita. "Human-comfortable navigation for an autonomous robotic wheelchair." In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pp. 2737-2743. IEEE, 2013.

[22] Nasri, Yassine, Vincent Vauchey, Redouane Khemmar, Nicolas Ragot, Konstantinos Sirlantzis, and Jean-Yves Ertaud. "ROS-based autonomous navigation wheelchair using omnidirectional sensor." International Journal of Computer Applications 133, no. 6 (2016).

[23] Aswath Suresh, Sri Harsha,Kolluri Surya,Debrup Laha,Dhruv Gaba,Siddhant Bhambri,Suvaansh Bhambri and Karthik Rangarajan, "ExplorationProbe to Jupiter Moon Europa", Mars Society Convention Marspapers,2016

[24] Tang, Robert, Xiao Qi Chen, Michael Hayes, and Ian Palmer. "Development of a navigation system for semi- autonomous operation of wheelchairs." In Mechatronics and Embedded Systems and Applications (MESA), 2012 IEEE/ASME International Conference on, pp. 257-262. IEEE, 2012.

[25] Aswath S. et al. (2015) An Intelligent Rover Design Integrated with Humanoid Robot for Alien Planet Exploration. In: Kim JH., Yang W., Jo J., Sincak P., Myung H. (eds) Robot Intelligence Technology and Applications 3. Advances in Intelligent Systems and Computing, vol 345. Springer, Cham

[26] Wu, Zhenyu, and Lin Feng. "Obstacle prediction-based dynamic path planning for a mobile robot." International Journal of Advancements in Computing Technology 4, no. 3 (2012): 118-124.

[27] Aswath Suresh, Unnikrishnan VJ, Sreekuttan T Kalathil, Abin Simon, and Ganesha Udupa, " Design And Development Of An Intelligent Rover For Mars Exploration",Marspapers,2015