Speedy Navigation of Indoor Environments with Limited Sensory Inputs

Ashwin Ravindra Bharadwaj

Abstract—**Currently, there has been a significant shift towards designing buildings with enhanced accessibility features, particularly for individuals with disabilities. These advancements have resulted in the majority of rooms and entrances being accessible not only by wheelchairs but also by wheeled robots. This presents an opportunity to leverage this accessibility for developing simple yet robust algorithms for navigation. This proposal advocates for an algorithm that utilizes limited sensory inputs yet achieves precise navigation within enclosed environments. Importantly, it aims to mitigate susceptibility to minor environmental changes and small obstacle and emphasize speed of the bot.**

I. INTRODUCTION

There have been many robots and navigation pipelines that have been developed to navigate a robot in closed environments. There are robots that can navigate a campus and deliver packages over short distances. But, most of these solutions are limited to open spaces and rely on a huge array of sensors to achieve this task. Additionally, most of these robots in the market are slow, usually slower than walking speed mostly attributed to the complex algorithm used for navigation. Higher speed tends to decrease the accuracy of the navigation algorithm. This project aims to overcome both of these shortcomings: it proposes a simple yet robust algorithm that can handle navigation in closed environments with a single RGB-D camera. Keeping the pipeline simple to increase the speed of the algorithm. It incorporates a POMDP to handle some of the issues that are associated with higher speeds that cause some sensory inputs to drift from reality. In my approach the robot is given a "tour" of the environment once and all subsequent navigations will be performed autonomously.

II. PROPOSED ALGORITHM

A. Estimating the current Node

To accurately determine the robot's current location(node) within this environment, we propose employing a Partially Observable Markov Decision Process (POMDP). This method integrates sensory inputs such as RGB-D data from cameras, estimated distance traveled using inertial measurement units (IMU) and compass data, odometer from the encoders and the robot's previous node. A pretrained convolutional neural network, typically based on architectures like ResNet, processes the RGB data to a vector that can be passed to the POMDP. This localization step is crucial for subsequent navigation tasks. The main reason for the use of the POMDP algorithm is to account for the excessive drift seen in sensors when the speed of the robot is increased. In slower robots where the IMU, encoders of the wheels are usually sufficeint to track the location of the robot to a good extent. But higher speeds these encoders and IMUs drift faster from reality. This is the main reason we propose to use a POMDP. This will provide some immunity from the drift mentioed above.

B. Details on the POMDPs

To represent the above problem as a Partially Observable Markov Decision Process (POMDP), we define it using the tuple (S, A, O, T, Z, R), where:

- **State Space (S):** The state space consists of all possible nodes in the graph that represent the robot's location within the environment. Each state $s \in S$ corresponds to a specific node.
- **Action Space (A):** The action space includes actions that the robot can take, which are:

  - a_{stay}: Stay in the current node.
  - $a_{move}(n)$: Move to a neighboring node n. The set of actions available at each node depends on the connectivity of the node within the graph (i.e., which nodes are adjacent).
- **Observation Space** (O)**:** The observation space consists of sensory inputs that provide information about the robot's current state. Observations $o \in O$ are derived from:
  - RGB and depth data processed through a convolutional neural network (CNN), providing a feature vector that represents visual and depth information.
  - Inertial Measurement Unit (IMU) data, which provides information on orientation and acceleration.
  - Encoder data from the robot's wheels, giving estimates of distance traveled.
- **Transition Model** (T)**:** The transition model $T(s, a, s') = P(s'|s, a)$ defines the probability of transitioning from state s to state s' given action a. This model accounts for the dynamics of the robot and any uncertainties in movement, especially at higher speeds where slippage or drift may occur. In our case all the probabilities will be one as we are assuming the graph represents a virtual simulation the real world. Any drift with respect to the real world will be corrected using the POMDP.
- **Observation Model** (Z)**:** The observation model $Z(o, s, a) = P(o|s)$ describes the likelihood of receiving observation o when in state s. This model incorporates sensor noise and the inherent uncertainty in perception, particularly with faster movements.
- **Reward Function** (R)**:** The reward function $R(s, a)$ provides a numerical value for performing action a in state s. This function is designed to encourage the robot to reach its destination efficiently while penalizing collisions or excessive movements. The MDP will rewarded for a correct transition and heavily penalized for a mistake.
- **Solving the POMDP:** Value iteration will not be suitable for our approach as the state space is very vast. Currently, as the problem has a large observation sapce, a limited action space and a moderately sized state space I will explore the following methods to solve the POMDP.
  - **Point-Based Value Iteration (PBVI):** As the action only allows the states (nodes) to move to adjacent states (nodes) on the graph we can ignore a large chunk of the search space. This will reduce the computational burden.
  - **Monte Carlo Tree Search:** This method is very computationaly expensive to train but might be benifitial. The main benifit of this algorithm is that it has been used on many real time applications.
- **Dataset collection:** Before the robot is expected to navigate a room autonomously, the robot is given a ”guided tour” of the closed environment. The robot is moved to the important areas represneted by the nodes in Fig 2. When in these areas the robot captures the details of this area using its sensors (RGB/Depth) in all directions. Once done the robot is driven to the next node where the above process is repeated.

The POMDP framework allows for the integration of different types of sensory data and helps in managing the uncertainties associated with sensor noise and environmental changes. This is crucial for ensuring robust and accurate localization and navigation, especially in dynamic and partially observable environments. This structured approach not only aids in improving navigation efficiency and accuracy but also in achieving a balance between exploration (gathering new information about the environment) and exploitation (using known information to achieve goals).

Once the robots location is well established with in the graph (an area in the building) we can continue to monitor its position as it moves by employing the above mentioned algorithm.

C. The Graph based mapping of the floor

Every closed environment, be it a floor, a room, etc will be represented by a graph. Each node

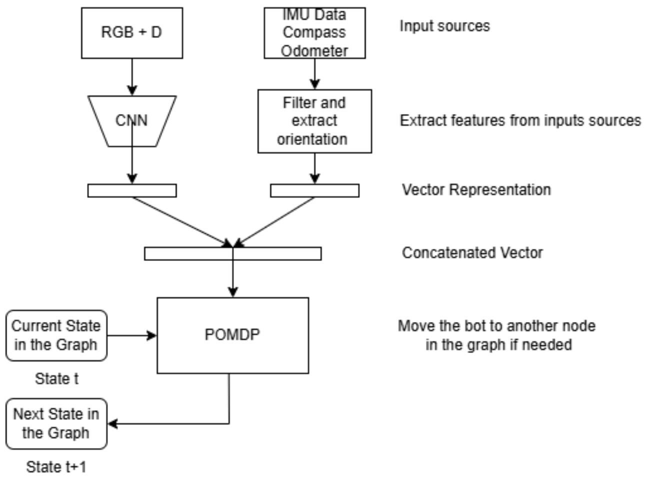
RGB + D
IMU Data
Compass
Odometer
Input sources
CNN
Filter and extract orientation
Extract features from inputs sources
Vector Representation
Concatenated Vector
Current State in the Graph
State t
POMDP
Move the bot to another node in the graph if needed
Next State in the Graph
State t+1

Fig. 1. POMDP: Estimate the current node in which the robot is in.

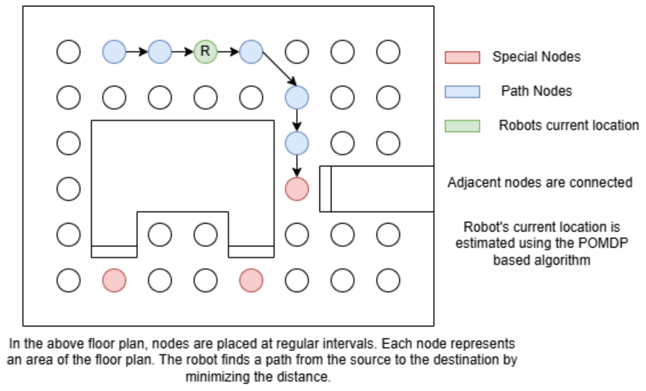
R
Special Nodes
Path Nodes
Robots current location
Adjacent nodes are connected
Robot's current location is estimated using the POMDP based algorithm
In the above floor plan, nodes are placed at regular intervals. Each node represents an area of the floor plan. The robot finds a path from the source to the destination by minimizing the distance.

Fig. 2. The graph based data structure used to map the floor.

will represent a small volume of the environment and will be fully connected with all the adjacent nodes. There will be special nodes that indicate an entry/exit to this location. These nodes will represent the doors/elevators in real life. If a robot needs to navigate within this closed environment it will merely have to follow the edges between the desired nodes. The robot's position always being updated with the previously mentioned algorithm. If the robot needs to access another environment it needs to interact with the world at the mentioned special nodes (such as opening doors/using elevators). This interaction is not part of this project. The following diagram shows a simple floor diagram with the nodes overplayed on it.

D. Actuator control

Once the robots current location and node is well established and the next node to visit is also know we can command the robot to move. This movement will be supervised and will be involve constantly checking the sensors to estimate the current location of the robot. This action will be taken in an "psudo-know" environment (the details of the nodes are know). Nodes will only have a few DoFs in which they can be arranged so a simple lookup table whose keys are the relative positions of the current and next nodes and the values are the actions that needs to be taken(move forward,left,right,back) will be sufficient to ensure the robot moves successfully to the next node.

III. Hardware

Item	**Purpose**
Intel Realsense-435	Depth and RGB camera. It also incorporates an IMU. This will be the main sensory input for the robot.
ODrive	The motor driver used to control the wheels of the robot. Connects the main computer with the motors and converts the wheel encoders into a format that can be used by the computer.
Motors and Encoders	Brushless motors used to drive the robot's wheels.
Jetson Nano	Main computer that will run the above pipleine and keep track of the robot's current location and controls its next movements. Handles all the comunication with the bot.

IV. Conclusion

The proposed methodology harnesses graph-based algorithms and POMDP-based localization techniques to enable precise and robust navigation of wheeled robots within accessible building environments at higher speeds than the current avilable methods. The main objective of this algorithm will aim to be as efficient and light weight as possible and increase the speed of the robot while utilizing minimal sensory inputs.

V. Timeline

Week	**Task**
1-4	Research related to SLAM and collecting RGB-D images of a small section of EXP 7th floor and develop a map with the necessary nodes. Assembly of the robot.
4-8	Implementing the above POMDP algorithm and fine-tuning the CNN and quantifying the results. Sampleing and validating the results by manually taking the sensors on a handheld walk of the environment.
8-12	Transferring the above algorithm onto a single-board computer which is connected to a wheeled robot. Further tuningthe robot to acomplish the above task.
12-14	Final testing of the simulated and physical bot and writing the report.

VI. Grading

Grade	**Task**
30%	Implementing the graph based system to map the entire floor. Converting given floor plans into the aboveformat. Collecting the required data for our training.
30%	Implementing the above POMDP algorithm and fine-tuning the CNN and quantifying and reporting the results.
25%	Transferring the above algorithm onto a single-board computer which is connected to a wheeled robot (off the shelf). Let the robot navigate between two given nodes.
15%	Final demo and report.