Monte Carlo Localization of Mobile Robots

**by**  
**Srikar Chavali**

**Supervision Essa Anas**

Project Submitted in Partial Fulfilment of the Requirements for

the Degree of Master of Science in Mechatronics and Intelligent Machines

School of Engineering

Faculty of Sciences and Technology

© [Srikar Chavali]  
UNIVERSITY OF CENTRAL LANCASHIRE  
[Term 2021]

[](https://www.google.co.uk/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=2ahUKEwijsuqcnoPZAhXBUhQKHc7RDP0QjRx6BAgAEAY&url=http://www.lancashiresciencefestival.co.uk/&psig=AOvVaw07Zwwl3bpAOrCNh-5njztj&ust=1517523633463080)

Copyright in this work rests with the author. Please ensure that any reproduction   
or re-use is done in accordance with the relevant national copyright legislation.

Acknowledgements

First and foremost, I would like to thank my parents for making this possible. Next, I would like to take this opportunity to express my gratitude to Dr Essa Anas, my project supervisor, who patiently guided me throughout the project. I would not have been able to finish this project without his constructive feedback and enlightening discussions. Many thanks to Dr Muqi Wulan for her helpful suggestions on how to prepare a professional report. Finally, I would like to thank my friends Jyothi Swaroop Tenugolla, Vijay Sudheer Nakka and Shivakarthik Varikuppala for their good advice and all the endless, but fun, discussions.

Abstract

In the field of robotics, one of the most challenging problems is localization especially in a complex environment. In certain cases, a robot must know its exact position in order to complete a task successfully. Since sensor readings are subject to noise, attempting to localize a robot using measurements from onboard sensors may not be efficient. This study investigates the Monte Carlo Localization algorithm and demonstrates how it can be used to identify a mobile robot's position in an unknown environment. In Monte Carlo Localization, the robot's belief regarding its location is represented by particles. This approach is simple to implement and computationally less demanding than other localization algorithms such as Markov localization because it only uses a large cluster of particles where it is required. Robot Operating System (ROS) is used to simulate Monte Carlo Localization algorithm using a TurtleBot in the gazebo platform. In this project, the impact of the number of particles that are used to localize a robot is analysed. This study also summarizes the effect of other secondary factors like the map, environment, and navigation on the localization process.

**Keywords**: Localization; Mobile robot; probability distribution; particles; particle filter; convergence.

Table of Contents

[Acknowledgements ii](#_Toc72791539)

[Abstract iii](#_Toc72791540)

[Table of Contents iv](#_Toc72791541)

[List of Figures vi](#_Toc72791542)

[List of Acronyms vii](#_Toc72791543)

[Chapter 1. Introduction 1](#_Toc72791544)

[1.1. Background and Context 2](#_Toc72791545)

[1.1.1. Uncertainty in the field of Robotics 2](#_Toc72791546)

[1.1.2. Probabilistic Robotics 3](#_Toc72791547)

[1.1.3. Probabilistic Generative Laws 4](#_Toc72791548)

[1.1.4. Belief 5](#_Toc72791549)

[1.1.5. Unimodality and Multimodality 5](#_Toc72791550)

[1.1.6. Localization and Filtering 6](#_Toc72791551)

[1.1.7. Localization and mapping 2](#_Toc72791552)

[1.1.8. Bayes Filter 2](#_Toc72791553)

[1.2. Scope and Objectives 4](#_Toc72791554)

[1.3. Achievements 4](#_Toc72791555)

[1.4. Overview of Dissertation 5](#_Toc72791556)

[Chapter 2. Literature review 6](#_Toc72791557)

[2.1. Categories of Localization Problems 6](#_Toc72791558)

[2.2. Localization Algorithms 7](#_Toc72791559)

[2.2.1. Kalman filter-based techniques. 7](#_Toc72791560)

[2.2.2. Topological Markov localization 8](#_Toc72791561)

[2.2.3. Grid-based Markov localization. 8](#_Toc72791562)

[2.2.4. Multi-hypothesis Tracking (MHT) 9](#_Toc72791563)

[2.2.5. Particle filters 9](#_Toc72791564)

[2.3. Monte Carlo Localization (MCL) 11](#_Toc72791565)

[2.3.1. MCL with sensor noise injection 12](#_Toc72791566)

[2.3.2. Adaptive Monte Carlo Localization 12](#_Toc72791567)

[2.3.3. MCL with Mixture proposal 14](#_Toc72791568)

[2.3.4. MCL with Gradient Ascent 14](#_Toc72791569)

[Chapter 3. Methodology 15](#_Toc72791570)

[3.1. Experimental setup 17](#_Toc72791571)

[3.1.1. Robot Operating System (ROS) 17](#_Toc72791572)

[3.1.2. Connecting MATLAB with TurtleBot 17](#_Toc72791573)

[3.1.3. Load the Map of the Simulation World 18](#_Toc72791574)

[3.1.4. TurtleBot Motion model and the Laser Sensor model setup in MATLAB 19](#_Toc72791575)

[3.1.5. Initialization and configuration of AMCL object in MATLAB 20](#_Toc72791576)

[3.1.6. TurtleBot navigation 21](#_Toc72791577)

[Chapter 4. Experimental results 23](#_Toc72791578)

[Chapter 5. Conclusion and future work 31](#_Toc72791579)

[5.1. Conclusion 31](#_Toc72791580)

[5.2. Summary and evaluation 31](#_Toc72791581)

[5.3. Project impact 32](#_Toc72791582)

[5.4. Future Work 32](#_Toc72791583)

[References 34](#_Toc72791584)

List of Figures

Figure 1 HMM illustrating the progression of states, measurements, and control (Thrun et al., 2000). 5

Figure 2 Gaussian distribution 6

Figure 3 Population of Gaussians 6

Figure 4 Scan matching 6

Figure 5 Pose filtering. 2

Figure 6 Particle filter algorithm (Thrun et al., 2000) 10

Figure 7 Properties of different localization algorithms (Fox, 2003) 10

Figure 8 Comparison of different localization algorithms 11

Figure 9 MCL algorithm (Thrun et al., 2000) 12

Figure 10 AMCL algorithm (Zeineh, 2001) 13

Figure 11 System design 16

Figure 12 Simulated office environment. 18

Figure 13 Map of the office environment 19

Figure 14 Particle distribution when the localization algorithm was initialized with 5000 particles. 23

Figure 15 Accurate localization with 5500 particles 24

Figure 16 Accurate localization with 5000 particles 24

Figure 17 Error in localization when HP is initialized with a low value (4000 particles). 25

Figure 18 Increased error in localization on further reducing HP value (2500 particles) 25

Figure 19 Robot failed to localize when HP value was initialized to 1000. 26

Figure 20 Particle clusters created at probable robot locations. 27

Figure 21 Comparison of localization error for different particle sizes 27

Figure 22 Robot localized by 15th update in the area of map with more features. 28

Figure 23 Robot not localized by 15th update in the area of map with less features. 28

Figure 24 Comparison of localization error for different zones. 29

Figure 25 Robot driven in circles localized in the 27th update. 30

Figure 26 Robot driven in a straight line not localized even after 36th update. 30

Figure 27 Comparison of localization error for different navigation schemes 30

List of Acronyms

|  |  |
| --- | --- |
| MCL  ROS  Bel  PDF  MHT  PC  AMCL | Monte Carlo Localization  Robot Operating System  Belief  Probability Distribution Function  Multi-hypothesis Tracking  Personal Computer  Adaptive Monte Carlo Localization |
|  |  |

# Introduction

Mobile robots are devices that can navigate in diverse environments like remote areas, dangerous places (Deep sea, Other planets) or common surroundings like offices, museums etc. to perform a task. Typical applications of mobile robots include material recovery from dangerous locations, surveying and mapping of remote places, delivery. Due to the growth in the number of applications of mobile robots in the last decade, robot localization has been identified as the most important problem in mobile robotics (Borenstein, Everett, & Feng, 1996). Robot localization is the process of determining where a robot is relative to its environment. There are two types of localization problems *position tracking* and *global positioning*. In the first case a robot knows its starting position and just needs to account for minor odometry errors as it moves. The global localization problem involves a robot that does not know its initial position and must therefore solve a much more complicated localization problem: estimating its position from scratch. However, in a collection of applications it was observed that a robot’s ability to localize globally as well as locally played a significant role in mobile robots (Andrew, 1999) (Endres, Feiten, & Lawitzky, 1998) (Burgard, Wolfram *et al.*, 1998).

Most of the early research in mobile robots was carried to solve tracking problem, later researchers developed a very effective family methods like Markov localization, Kalman filters and grid-based methods which can solve either of the localization problems. The core concept behind Markov localization is to reflect the robot's belief as a probability distribution over all possible locations, and to update the belief using Bayes rule and convolution whenever the robot senses or travels (NOURBAKHSH, POWERS, & BIRCHFIELD, 1995a). To reflect the robot's belief with a Kalman filter, (which involves estimation of probabilistic states) multivariate Gaussians are used. Kalman-filters are typically used only for position tracking due to the restrictions associated with Gaussians (Gelb, 1974). Markov localization solves the problem of global localization by using distinct yet multi-modal representations to describe the robot's beliefs. Since kinematic states are real-valued and multi-dimensional, Markovian approaches can only approximate the robot's perception, and accurate estimation is typically needed. Such approaches cannot be used for high resolution applications that use localization to avoid collisions with obstacles that cannot be detected by sensors (Fox *et al.*, 1999).

MCL represents the robot's confidence using quick sampling techniques. Importance re-sampling is used to approximate the posterior distribution as the robot travels or senses (Rubin, 1988). To balance computation and precision, an adaptive sampling scheme is used, which calculates the number of particles on-the-go (Koller & Fratkina, 1998). As a result, MCL uses a large number of particles for global localization, when they are most required, but a small number of particles during monitoring, when the robot's location is only approximated.

MCL has many main advantages over previous work in the field because it uses a particle-based representation:

* Unlike current Kalman filtering-based localization, MCL can describe multi-modal distributions and hence can localise a robot globally.
* As opposed to grid-based Markov models, it needs significantly less memory and can process measurements at a much higher rate.
* Since the state described by the samples is not discretized, it is more reliable than Markov localization with a predefined cell size.
* It is a lot easier to understand and apply.

## Background and Context

This section will provide context information required to define and solve the problem of robot localization.

### Uncertainty in the field of Robotics

Robotics is the science of using computer-controlled robots to perceive and manipulate the physical world. For example mobile rovers for interplanetary expeditions, self-driving vehicles, robotic arms used in industrial assembly lines etc (Thrun, Fox, & Burgard, 2000). A robot typically has four key components: (i) Physical body, which allows it to function in the real world. (ii) Sensors, for sensing its surrounding environment. (iii) Actuators and effectors: which allow it to perform an action. (iv) Controller: to make it autonomous.

In order to perform tasks in the real world, robots must account for a variety of uncertainties (Thrun *et al.*, 2000), which are induced by a range of factors. To begin with, the robot's surroundings are often unpredictable, particularly in ever changing environments like offices and highways. Second, the sensors have drawbacks in terms of what they can detect, such as limitations regarding their resolution and range. Third, the unpredictability due to imperfections in motor which is used as an actuator for a robot. Uncertainty may also be introduced as a result of mechanical failure and control noise. Fourth, since the simulated models of the physical world are approximate and limited, the robot's software can also cause uncertainty. Lastly, algorithmic approximations are another source of ambiguity. Accuracy must often be compromised in real-time systems in order to get a quick response.

Uncertainty has become a major problem for the design of robots as robotics systems have become common in the world around us. The first step towards developing a stable robotic system is to learn how to handle uncertainty.

### Probabilistic Robotics

Probabilistic robotics is a modern approach to solving the issue of uncertainty in robot perception. The key idea behind probabilistic robotics is to use probability theory to describe uncertainty.

#### State

In probabilistic robotics, the environment is a dynamical system that consists of internal state. Robot can perceive the information about its environment through sensors, and the robot can also affect the environment through its actuators. Since the enormous of uncertainty exists, robot needs to maintain an internal belief about the state of its environment. The state is defined as the collection of all aspects of the robot and its environment that can impact the future. It has been categorized into two groups. State variable that changes along with time is called dynamic state, such as walking people around the robot. Static states, on the other hand, are those that appear to remain static or do not change at all, like walls in houses. The state of the robot is often influenced by its own variables, such as its pose and velocity. Variables that typically characterise a robot's state include: (1) the pose of the robot (2) In robot manipulation (kinematics state) (3) velocity of the robot (dynamic state). (4) State variables include the position and characteristics of nearby objects in the environment. A desk or a box could be an entity, and features could be visual perceptions like texture or color (5) the positions and velocities of people and objects in motion could also be state variables. Since the number of possible state variables is infinite, robot environments can be represented using several state variables, depending on the level of granularity required.

If the present state is the best predictor of the future, it is said to be complete. This means that no other information from previous states, tests, or controls can be used to improve the future prediction. In reality, however, finding a complete state for any robot device is difficult. A complete state includes not only all aspects of the world that could have an effect on the future, but also the robot's own variables. There are times when it is impossible to obtain all of this knowledge.

### Probabilistic Generative Laws

The probabilistic laws govern the state evolution (Thrun *et al.*, 2000). The state xt, according to the concept of state, is dependent on all previous states, controls, and measurements, and it can be represented by a PDF of the form p(xt| x0:t-1, z1:t-1, u1:t). The measurement zt is accompanied by a control action ut in the series of environment interactions. xt-1, in particular, is an adequate description of all the past controls and measurements up to period t-1. If the state xt-1 which we have is a complete state, only the current control data ut will affect the previous expression. As a result, the equality in the expression is as follows: **p(xt| x0:t-1, z1:t-1, u1:t) = p(xt| xt-1, ut)**. Conditional independence is the property used in this expression, which implies that if we know the conditioning variables (ut, xt-1), then certain variables, (xt), are independent of other variables (z1:t-1 and u1:t-1). If xt is complete, the conditional independence also applies to the modelling of the measurements: **p(zt| x0:t, z1:t-1, u1:t) = p(zt| xt)**. All other variables, like past states, controls, and measurements are insignificant in this case as the complete state xt is adequate to predict the measurement zt.

The state transition probability is defined as **p(xt| xt-1, ut)**. It depicts how state xt changes over time as a function of the controls ut. The measurement probability is defined as **p(zt| xt)**, which defines how measurements z are derived from state x. A mixture of motion and measurement models presents the dynamical framework of the robot and its surroundings. The evolution of measurements and states is depicted in the diagram below. The control ut and the previous state xt-1 are stochastically related to the current state xt. The condition xt has a stochastic effect on the measurement zt. This model is also known as a hidden Markov model (HMM), as seen in the diagram below.

A picture containing text, clock, clipart

Description automatically generated

Figure 1 HMM illustrating the progression of states, measurements, and control (Thrun et al., 2000).

### Belief

A robot's belief is a core idea in probabilistic robotics, which is used to express the robot's internal awareness of the environment's state (Thrun *et al.*, 2000). However, since the state is seldom directly evaluated, the robot must infer its position from the data obtained. Conditional probability distribution functions are used for belief description in probabilistic robotics. The belief distribution of a robot assigns a density score to each potential state hypothesis in relation to the true state.

Belief over state xt at time t is expressed as bel(xt) = p (xt|z1:t, u1:t). It is a posterior likelihood over states that is based on all the previous control and measurement results. This expression gives the posterior likelihood integrating the most recent measurement data zt.

### Unimodality and Multimodality

A location probability distribution is a function that defines the robot's potential locations, with the input being a location and the output being the probability of being there (Zeineh, 2001). A Gaussian distribution with single peak is an example of a unimodal location distribution. A multimodal position distribution, as in a population of Gaussians, has two or more peaks. Figure below depicts two examples of multimodal and unimodal distributions, respectively.

Chart, line chart

Description automatically generated

Figure 2 Gaussian distribution

Chart, line chart

Description automatically generated

Figure 3 Population of Gaussians

### Localization and Filtering

Filtering in localization means the removal of positions that are not possible. Impossible positions are those in which the robot's sensory feedback does not fit the map in that area, such as within an object. Scan matching is a technique for localization (Gutmann & Schlegel, 1996). The initial position is established, and it is assumed that the location error is low (i.e., the true location and location guessed by the technique are very close). Following a move, an exhaustive search of the current guessed position by translating and rotating it produces the next location approximate, which is normally the location with the closest match with the map.

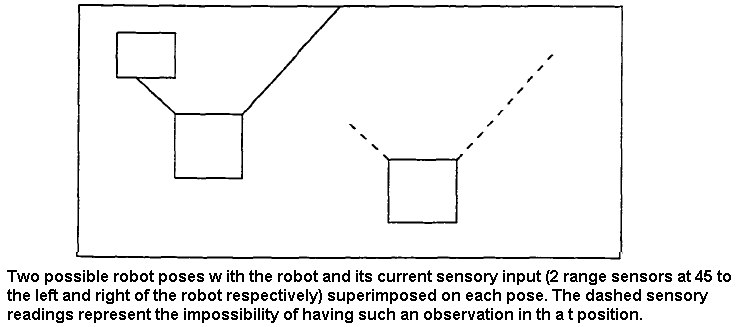


Figure 4 Scan matching

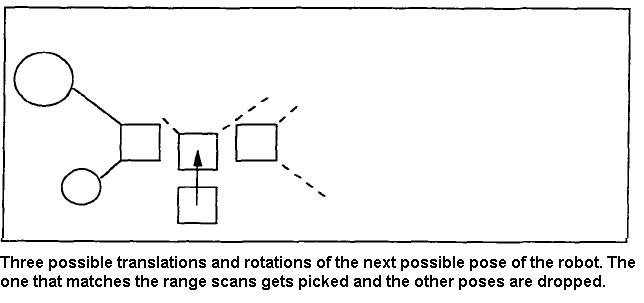


Figure 5 Pose filtering.

### Localization and mapping

For purposes like exploration, mapping and localization are combined. Using Markov localization which is a variant of the Baum-Welch algorithm (Thrun, Burgard, & Fox, 1998) proposes a method for simultaneous localization and mapping. There is no map at the start of this technique, but as the robot explores its surroundings, a map is built up incrementally, and any newly mapped obstacles added to the map are also used when matching range scans of samples against these.

### Bayes Filter

The issue of predicting the state of a dynamic system through observations is addressed by Bayes filtering. In most problems involving mobile robots, the state is presumed to be described by a location in Cartesian x-y space and the orientation of the robot 'Ɵ'. The system's state is defined by a probability distribution based on the data that spans all of space. This posterior t is commonly referred to as the state belief and is represented by:

|  |  |  |
| --- | --- | --- |
|  |  | (1.1) |

xt = state at time t.

dt =observation at time t.

Our understanding of the initial robot pose is represented by the initial belief. The initial assumption is generally believed to be uniformly distributed in the lack of such understanding. In brief, Bayes filters use recursive equations to estimate this emerging distribution. The way in which the recursive equations are implemented to their specific representation of the state space distinguishes each form of Bayes filter from the other.

Markov Localization is a method of estimating a robot's position based on Bayes filters and their recursive equations. The Markov assumption allows both the motion and sensing models to be simplified. The odometry data (a) is differentiated from other sensor observations (o) i.e., data from range scanners in Markov localization. The current belief can be expressed as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (1.2) |

Applying bayes rule to (1.2)

|  |  |  |
| --- | --- | --- |
|  |  | (1.3) |

Using Markov assumption can be reduced to , therefore.

|  |  |  |
| --- | --- | --- |
|  |  | (1.4) |

can be expanded by integrating the state at time t-1.

|  |  |
| --- | --- |
|  | (1.5) |

can be simplified to as the state at t -1 is independent of at-1. We may also use the Markov assumption to reduce to .These transformations give us the below equation:

|  |  |  |
| --- | --- | --- |
|  |  | (1.6) |

μ = normalizing constant.

The prior or the state belief before ot can be expressed as:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | (1.7) |

In the literature of filtering, prior(xt) is called prediction density and posterior(xt) is called the filtering density (Pitt & Shephard, 1999).

## [Scope](http://www.cs.stir.ac.uk/~kjt/research/conformed.html) and Objectives

The aim of this project is to demonstrate the MCL algorithm in action to localize a robot in an unknown environment and to investigate the impact of particle limits on the localization.

The project is carried out in a simulated environment. Real-world limitations like errors in sensor and odometry readings which are observed while localizing a real robot were also incorporated into the simulation to make it more realistic. As the project is run in a simulator the scope of this study is to develop, test and optimize the necessary software for mobile robot localization.

The following are the goals that must be met in order for the project to be completed successfully:

* Investigate the need for mobile robot localization.
* Investigate the most widely used methods for localizing mobile robots.
* Develop an understanding of ROS, gazebo simulator, Rviz tool and TurtleBot.
* Demonstrate the Monte Carlo Localization algorithm in a virtual Gazebo environment using TurtleBot and MATLAB.
* Comparing the localization results for various fixed particle set sizes to evaluate the performance of the MCL algorithm.

## Achievements

The following achievements were accomplished upon successful completion of the project:

* Developed an understanding of the different techniques used to localize mobile robots.
* Gained knowledge of the different variants developed in the MCL algorithm.
* Managed to learn new skills like using Ubuntu, Robot Operating System (ROS) and gazebo environment.
* Developed a methodology by integrating MATLAB and ROS to demonstrate the working of MCL algorithm. The same methodology can be extended to localize a mobile robot in different environments.
* Successfully implemented the MCL algorithm and localized a mobile robot.
* The impact of particle size limits on the accuracy of localization was clearly demonstrated.
* The influence of secondary parameters like the input map, navigation, and the robot environment on the robot localization was also explained.

## Overview of Dissertation

The dissertation work is divided into chapters. The first chapter gives an introduction about the concept of mobile robot localization. This chapter also describes the MCL algorithm and its advantages in brief. The objectives of the project along with several background concepts which are required to get a clear understanding of the working of the MCL algorithm are also described in the first chapter. The second chapter is devoted to a study of the literature on the key research topics, such as types of localization problems, types of localization algorithms and the different variants in MCL algorithm. Chapter 3 outlines the methodology adopted, experimental setup and the software utilized to visualize MCL in action. Chapter 4 presents the observations and results of the simulation. Chapter 5 gives the future scope of the project work and establishes conclusions from the observed results.

# Literature review

## Categories of Localization Problems

According to the conditions of the environment and the initial information that a robot may have about the given map, the mobile robot localization problem can be divided into several forms. We begin with a brief overview of the taxonomy of localization issues.

The various forms of initial awareness of a robot are used to categorize the first class of localization problems. Two types of localizations have been extensively researched in this class: local and global localization. The first type is the easiest since the robot's initial state is known, and the challenge is then to accommodate for noise in the odometry readings of the robot. It is worth noting that the noise is normally presumed to be low, and the variability is localised and limited to the region closest to the robot's true position. A unimodal PDF, such as Gaussian, can be used to model the uncertainty in local localization. The case of global localization is more complicated since the robot does not know its initial pose and must figure it out on its own. The kidnapped robot problem is a version of the global localization in which the robot is kidnapped during the process and moved somewhere in its world without being warned. It's even more complicated because even the most advanced localization techniques can occasionally struggle to locate. As a result, fully autonomous robots are evaluated based on their potential to recover from errors.

The property of environment is used to categorise the second class. It is possible to have a static or dynamic environment. The term "static world" refers to an environment in which only the robot moves, and all other objects are stationary. On the other hand, in a dynamic environment, the position of robot as well as the objects change with respect to time. People, pets, and movable furniture are examples of these changes. Because of the burden of coping with evolving states in the dynamic system, localization is more difficult.

The third class is divided into two categories based on whether the localization algorithm governs the robot's motion: active localization and passive localization. The algorithm in passive localization only watches the robot operate and does not monitor the motion to aid in the localization. Normally, the robot moves at random. The algorithm often regulates the robot motion in active localization to achieve the aim of reducing localization errors. Active localization methods clearly outperform passive methods in terms of producing better performance.

Localization problems are divided into two categories based on the number of robots participating: single-robot and multi-robot localization. In recent years, the former has received the most attention. Since a single robot is involved in this situation, and all data is incorporated into only one robot network, there are no communication issues. More than one robot is involved in multi-robot localization. The difficulty arises in determining how to portray different belief systems and the correspondence that occurs between them.

## Localization Algorithms

As previously stated, localization is the first phase in almost all robotic navigation systems. To address the numerous localization problems, particularly for single robot localization, several probabilistic methods have been created. Current methods for localizing mobile robots can be differentiated by the way they describe the robot’s state space.

### Kalman filter-based techniques.

Kalman filters are used in most of the earlier robot localization applications (Kalman, 1960). The majority of these methods are based on the premise that the robot's location uncertainty can be expressed by a unimodal Gaussian distribution. Over the robot's location, sensor readings are often supposed to map to Gaussian-shaped distributions. Kalman filters have highly efficient update rules that can be seen to be suitable for these assumptions (Peter S. Maybeck, 1979). For keeping track of the robot's location, Kalman filter-based techniques have proven to be reliable and precise (Leonard & Durrant-Whyte, 1992). These strategies, on the other hand, do not account for multi-modal probability distributions, which are common during global localization. In theory, Kalman filter localization approaches usually require that the robot's starting location be identified. Kalman filters often depend on sensor models that produce measurements with Gaussian uncertainty, which is mostly unrealistic.

### Topological Markov localization

In most of the robot localization problems, the probability distribution function of estimated state of robot is non gaussian and hence Kalman filter based approaches do not yield reliable results. Alternative techniques have used richer strategies to describe uncertainty to overcome these limitations, going beyond the assumption of Gaussian probability density function required for vanilla Kalman filter. The form of discretization used for the interpretation of the state space will largely distinguish these different methods. Markov localization based on landmarks is used for hallway navigation in (Cassandra, Kaelbling, & Kurien, 1996) and in (NOURBAKHSH, POWERS, & BIRCHFIELD, 1995b), and the state space is arranged and as per the environment's coarse topography. The location estimates' precision is limited by the state representation's coarse resolution. Topological methods usually only offer a rough idea of where the robot is.

### Grid-based Markov localization.

Grid-based methods perform numerical integration over an equally spaced point grid to represent non-Gaussian and multimodal probability densities at a fine resolution (Burgard, W. *et al.*, Dec 31, 1996). This includes discretizing the relevant section of the state space and using it as the foundation for state space density estimation, such as by a piecewise constant function. The spatial resolution of these grids for indoor localization is typically within 10cm and 40 cm, and the angular resolution is typically 5° (Fox, 2003). Grid-based methods are efficient, but they have a high computational requirement and require a pre-fixed state space's size and resolution. Furthermore, the resolution and, as a result, the accuracy with which they will define the state must be determined ahead of time. Since not all measurements can be processed in real time, important data about the state is lost. Some of these issues have been addressed in a study by (Burgard, W. *et al.*, 1998) by using oct-trees to achieve a flexible resolution representation of the state space. This has the benefit of maximizing memory and computation use to where it is required, as well as overcoming the limitations imposed by fixed resolutions.

### Multi-hypothesis Tracking (MHT)

MHT methods use Gaussian mixtures to describe the belief state (Arras, Castellanos, & Siegwart, 2002) (Austin & Jensfelt, 2000). An extended Kalman filter is usually used to track each Gaussian. These methods can address the problem of global localization because of their ability to represent multi-modal beliefs. These approaches still depend on the same assumptions as the Kalman filters because each hypothesis is traced using a Kalman filter. MHT methods, on the other hand, have been extremely resistant to deviations of these assumptions in practice. MHT methods need sophisticated heuristics in addition to plain Kalman filtering to solve the issue of data association and decide when to add or remove hypotheses (COX, 1993).

### Particle filters

The particle filter is a filter that represents posteriors using a finite number of samples (Thrun *et al.*, 2000). A potential robot pose is described by each particle drawn as per the posterior distribution. Sampling, importance weighting, and resampling are the three stages in particle filters. The particle set (xt-1), latest measurement(zt) and control data (ut) are the inputs to the filter. A theoretical sample set Xt[n] is created during the first step depending on the state transition likelihood p(ut, Xt-1[n]) for time period t. The probability model of sensor measurement p(zt| Xt[n]) is used to determine the importance weights wt[n] for each particle. The measurement zt is incorporated into the particle collection using these importance weights. It then draws N particles with replacement from the temporary set Xt[n] in the final stage. The importance weight of a particle determines the likelihood of drawing it. The size of the resulting sample set is identical to that of the actual sample set. As the resampling phase focuses the particles to locations with large posterior probability, the greater the number of particles that appear in that location, the more probable that region is. representing the actual state. By iteratively evaluating the three steps. the particle filter focuses the computational resources to locations in the state space where they are required the most.

Text, letter

Description automatically generated

Figure 6 Particle filter algorithm (Thrun et al., 2000)

Graphical user interface, text, chat or text message

Description automatically generated

Figure 7 Properties of different localization algorithms (Fox, 2003)

Table

Description automatically generated

Figure 8 Comparison of different localization algorithms

## Monte Carlo Localization (MCL)

The sensor model and state change model in particle filters, refer to the measurement model and motion model in MCL. The figure below shows the steps involved in the MCL algorithm. A sample collection of N total particles is used in the MCL algorithm to describe the posterior belief bel(xt). The prior particle collection (Xt-1), measurement data (zt), control data (ut), and the provided environment map (m) are all used as inputs to the MCL algorithm. The particle sets and Xt are both initialized on line 1. Line 3 samples from the motion model for each particle, and line 4 measures the particle's importance weight from the measurement model. The resampling step spans lines 7 through 10. The algorithm draws N samples from the temporary set with replacement. The importance weight of each sample determines the likelihood of drawing it. Return the posterior sample collection Xt, which includes the samples with highest importance weights. If MCL algorithm is completed successfully, the majority of particles are distributed in a small area that reflects the robot's location; however, the simple MCL algorithm lacks a stop condition, so the recursive algorithm will continue to run throughout the robot's lifetime.

Text, letter

Description automatically generated

Figure 9 MCL algorithm (Thrun et al., 2000)

Though effective MCL algorithm has its own drawbacks. To address these drawbacks researchers have developed several variants of MCL which are discussed below

### MCL with sensor noise injection

To overcome the "bad performance with accurate sensors" effect, the basic MCL algorithm is supplemented with sensors embedded with fake noise (Fox *et al.*, 1999), (Dellaert *et al.*, 1999). This algorithm is not mathematically rigorous (localizes a robot approximately). This variant assumes that for localization, a constant threshold is good enough. However, in order for this to operate, a significant amount of noise must be introduced into the sensors, reducing the accuracy.

### Adaptive Monte Carlo Localization

For the case of global localization, an adaptive sample size has been shown to be more robust than a static sample size (Dellaert *et al.*, 1999), (Fox *et al.*, 1999). When the error level is high, AMCL localizes robot at the cost of higher memory usage and recursive computation cycles. The AMCL algorithm for global localization is defined in pseudo-code as follows:

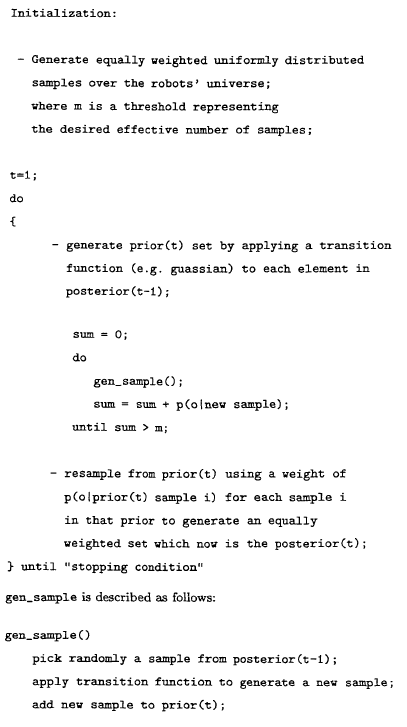


Figure 10 AMCL algorithm (Zeineh, 2001)

### MCL with Mixture proposal

This approach was developed to address the issue of "bad performance with accurate sensors" by taking samples from the map so that they fit the sensor readings at that time point, and then weighting the samples using the transition model (Thrun *et al.*, 2000). This is essentially the inverse of an MCL phase, which draws samples from the prior and estimates its weight according to the sensor model. The reverse method was combined with the MCL's initial sampling procedure, rather than being used alone. As seen in (Thrun *et al.*, 2000), this mix still reflected the true posterior and outperformed the original MCL with noisy sensor readings. However, this technique required the development of a sensory map that can be easily sampled before it could be used.

The original MCL's cause of error was the discretization of the position rather than the sensory model, which made positions that were similar to the real robot location but beyond the range of a reliable sensor model appear less probable. Because of the discretization of the robot's position, this could happen frequently. By introducing noise to the sensor model, the algorithm was made more receptive to this, effectively addressing one issue by creating another.

### MCL with Gradient Ascent

The gradients of p(o|s) and p(s|a,s') with respect to ‘s’ are used in MCL with gradient ascent. Although this variant was less robust than the EM (Baum Welsh) for generating sampling maps, it was the basis for a mapping algorithm that can map huge environments (Thrun, Burgard, & Fox, 2000). For certain environments, this approach can produce “near global maxima”, but as the distance between the samples grows, it can also produce local minima.

# Methodology

MATLAB (for executing MCL algorithm), ROS (for controlling robot and map building using RViz tool ), and Gazebo simulator are used to demonstrate the algorithm's functionality. The robot and its environment are simulated using a TurtleBot robot simulator (readily accessible from Gazebo). TurtleBot is a replica of a real robot that sees with a Kinect sensor (Shikhman, ). Then, we can use an existing map of the environment or by using the Rviz app, we create a map that takes into account the robot's motion as well as the information obtained from its sensors. MATLAB is used to transform the resultant map into a 2D binary occupancy grid. Each cell in the binary occupancy grid has a value that represents whether the cell is occupied (denoted by true or 1) or empty (denoted by false or 0). The ROS toolbox in MATLAB is used to establish a connection between ROS and MATLAB and to send and receive the data to the TurtleBot. The Navigation toolbox in MATLAB is used to run simulation, execute MCL algorithm and visualize the results.

To begin, turn on the robot. The “gmapping\_demo” node is then initialised in ROS and configured. Next the "keyboard teleop" node is launched in ROS which helps to drive the robot manually. The robot begins to move around the room in response to inputs from the keyboard. Keep an eye on the robot's pace when it's moving. If the robot moves too quickly the environment will not be mapped satisfactorily. The robot should move along walls or other obstacles to ensure that it maps the environment in detail (Xiaoyu *et al.*, Jul 2018). Finally, the robot returns to its original location in order to obtain all of the information about the environment After mapping, the robot measures the odometry and the laser scan data, which is then compared to the map publisher's results. On obtaining the comparison results, MCL processes the data and gives the results in the form of an array. Using this array information, a particle cloud is created in MATLAB which represents the probable robot locations. A dispersed point cloud indicates high uncertainty in robot’s pose, whereas a concentrated point cloud indicates a low uncertainty in robot’s pose.

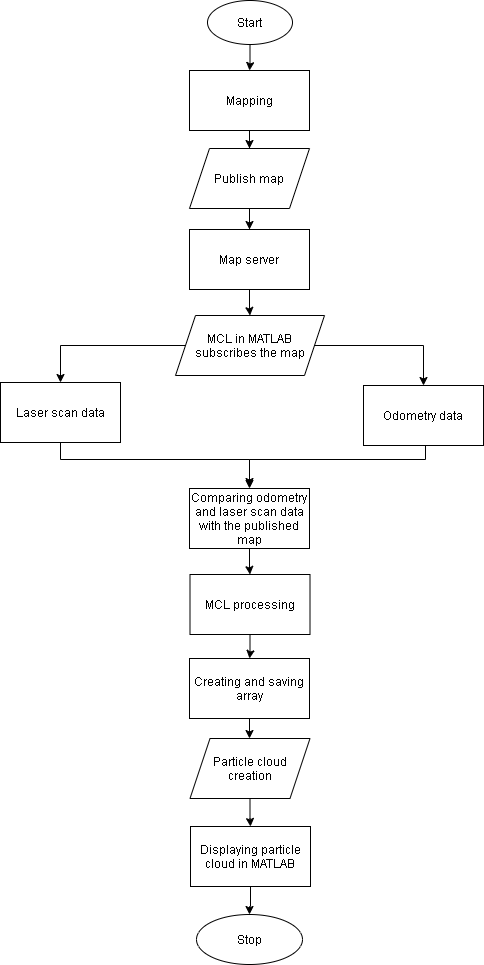


Figure 11 System design

## Experimental setup

Experiments are conducted using a PC (host computer) and a ROS robot (slave computer). To finish the setup, they both connect to the same wireless network. The ROS robot is managed by the PC issuing commands. The robot builds a map using a laser radar sensor during the experiment.

### Robot Operating System (ROS)

ROS is a meta-OS for robots and is open source. It is a versatile platform for developing robot apps. It is a collection of conventions, tools, and libraries aimed at making the task of programming complex and reliable robot actions on a variety of robotic platforms easier. It includes package management, implementation of widely used features, low-level device control, communication of messages between processes, and hardware abstraction, as well as other services which are expected from an OS. It also comes with libraries and tools for designing, writing, and executing code on several computers. ROS was constructed from the get-go to promote "collaborative robotics" (ROS.org, ).

### Connecting MATLAB with TurtleBot

The experiment involves localizing a turtle bot in a virtual office environment. First, we open a simulated TurtleBot which is inside an office building in a virtual machine to launch the Gazebo Office World from the desktop. Next in the MATLAB on the host computer, we initialize ROS global node in MATLAB and connect to the ROS master in the virtual machine through its IP address (The MathWorks, a).

A picture containing sky, light, LEGO, outdoor

Description automatically generated

Figure 12 Simulated office environment.

### Load the Map of the Simulation World

In Gazebo, load a binary map of the office location. TurtleBot is driven around the office environment to build the map. The map is made up of ground truth poses from the ros topic gazebo/model states and range-readings from Kinect.

Diagram

Description automatically generated

Figure 13 Map of the office environment

### TurtleBot Motion model and the Laser Sensor model setup in MATLAB

TurtleBot can be represented as a robot with a differential drive, with odometry data used to estimate its motion. The Noise property describes the degree of uncertainty in the rotational and linear motion of a robot. While using odometry measurements for particle propagation, increasing the value of “odometryModel.Noise” property would allow for further spread. TurtleBot's sensor is a virtual range finder based on Kinect readings. By matching the measurements of the end points of the range finder to the binary occupancy grid of the environment, the likelihood field method is used to calculate the probability of observing a range of measurements. The likelihood of perceiving an observation is high if the occupied points match the end points in the occupancy grid. To obtain better test performance, the sensor model should be calibrated to fit the real sensor property. The “SensorLimits” property specifies the sensor readings maximum and minimum ranges. The Map property specifies the occupancy grid of the environment that will be used to compute the probability area. The propery “rangeFinderModel.SensorPose” is set to the fixed camera's coordinate transform with respect to the base of the robot. This is used to convert laser measurements to the TurtleBot's base frame from the camera frame. ROS subscribers are created to get TurtleBot’s odometry and sensor readings. ROS publisher is created to send velocity commands to TurtleBot (The MathWorks, ).

### Initialization and configuration of AMCL object in MATLAB

In MATLAB, the “monteCarloLocalization” command generates an MCL system object. The lidar and pose data are fed into a “monteCarloLocalization” object. The MCL object transforms the lidar data from one reference frame to another as per the specified “SensorModel.SensorPose” property. By integrating the odometry readings, the input pose is calculated. The particles are modified, and the algorithm calculates an updated state approximation from the particle filter, if the change in robot's pose is higher than the specified value of “UpdateThresholds” property.(The MathWorks, b). The following method is used to update the samples:

1. The samples are updated depending on the transition in pose and the “MotionModel” property that has been defined.
2. The probability of obtaining the sensor readings for each sample is used to determine the weights of the samples. The sensor model defined in "SensorModel" property is used to calculate these probability weights.
3. The samples are redrawn from the posterior distribution using the specified "ResamplingInterval" property, and the samples with low weightage are removed. A resampling interval of two, for example, implies that the samples are redrawn after alternative updates.

The covariance and approximate pose, as well as the value of the property "isUpdated," are the MCL object's outputs. The covariance and mean of the cluster of samples with highest weight are used to estimate the robot's state. The output pose i.e. the “SensorModel.Map” property, is displayed in the map's coordinate frame. The state estimate will be changed if the shift in pose is higher than either of the update thresholds, and “isUpdated” is true. If not, “isUpdated” will return false, and the state estimate will remain unchanged.

The upper and lower bounds on the number of samples produced in the resampling phase are described by the property “amcl.ParticleLimits.” Allowing more samples to be produced can increase the likelihood of accurately localizing the robot, but it has a negative effect on computation efficiency, and samples can take longer to converge or even fail. In comparison to localization with an idea of the initial pose, global localization may require considerably more samples. Even if the robot only has an approximate idea of its initial pose, this additional knowledge will help AMCL in achieving faster localization with less samples. In the absence of any initial pose approximation AMCL may still attempt to perform localization without knowing robot's initial location. MCL algorithm starts by assuming that the robot has an equal chance of being somewhere in the open space of the office, and then produces uniformly distributed samples in the office space. As a result, relative to localization with an initial position approximate, global localization problem needs substantially higher number of samples. The MCL object's "amcl.GlobalLocalization" property is set to true to allow AMCL's global localization function. (The MathWorks, ).

### TurtleBot navigation

TurtleBot with Vector Field Histograms (VFH) was used in this project to avoid obstacles during robot navigation. The robot wanders in the environment by moving forward until it encounters obstacles. When attempting to move forward, the "controllerVFH" object calculates steering angles required to avoid obstacles. Using VFH, the MATLAB controllerVFH object allows the vehicle to stay clear off objects based on the readings of the range sensor (The Mathworks, ). The VFH object determines a collision free steering path based on the specified target direction to move and laser scanner readings.

The VFH+ algorithm is used to determine an obstacle-free path. Firstly, the algorithm creates a polar histogram for obstacle positions using the angles and ranges from laser scans. The input histogram limits are then utilized to create a binary histogram that shows the free and occupied paths. In the final step, the algorithm calculates a masked histogram from the previously generated binary histogram based on the vehicle's minimum turning radius.

Based on the feasible driving paths and the available space, the algorithm selects several steering paths. The cost of various potential paths is calculated using a cost function which takes into account the weights referring to the current, previous, and target directions. The object then presents a cost-effective, object-free path. The user may use the obstacle-free path to send commands to the robot to drive in that direction.

When the robot moves around the map, the MCL algorithm updates the particles using the sensor and odometry readings at specified time level.

# Experimental results

To evaluate the performance of MCL algorithm and estimate the effect of particle limits on the localization several simulations were carried using the TurtleBot in gazebo environment. During the study, it was observed that MCL is easy to understand and implement in comparison to other localization algorithms. Even though the algorithm was not applied on a real robot in this study, many real-world conditions like noise in sensor readings and odometry readings due to vehicle slips are modelled into the simulated TurtleBot in order to achieve reliable results. The MCL algorithm applied in this study uses an adaptive threshold mechanism that calculates and updates the number of particles each time the robot moves.

Note: As MCL is a probabilistic algorithm, the simulation result of each trial was slightly different from the other. Hence to confirm the observed results each experiment was carried out several times.

To study the effect of number of particles on localization several experiments were conducted by varying two parameters: the maximum number of particles (call it HP) at the start of the algorithm and the minimum number of particles (call it LP) retained as the algorithm progresses. It was observed that starting the algorithm with a high value of HP (between 5000 to 6000) resulted in the accurate localization of the robot.

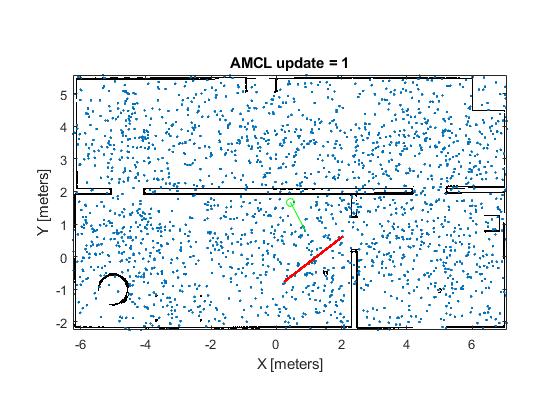


Figure 14 Particle distribution when the localization algorithm was initialized with 5000 particles.

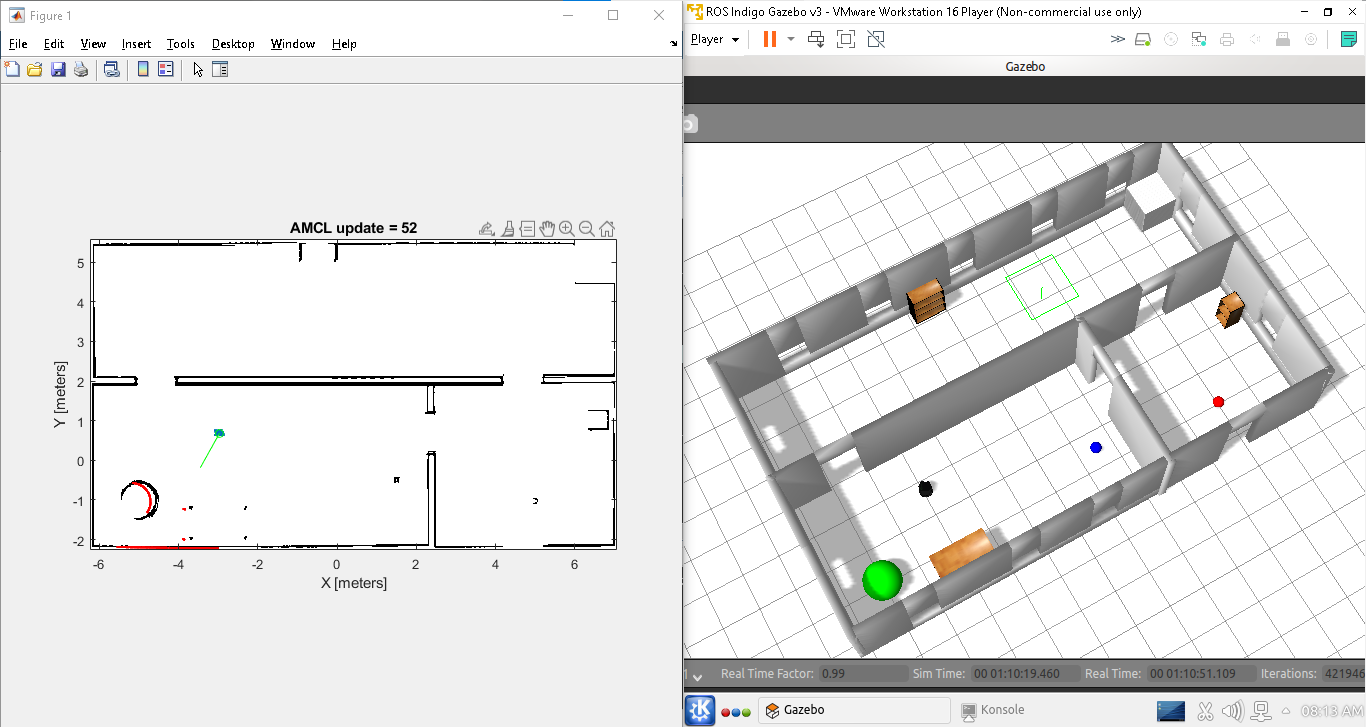


Figure 15 Accurate localization with 5500 particles

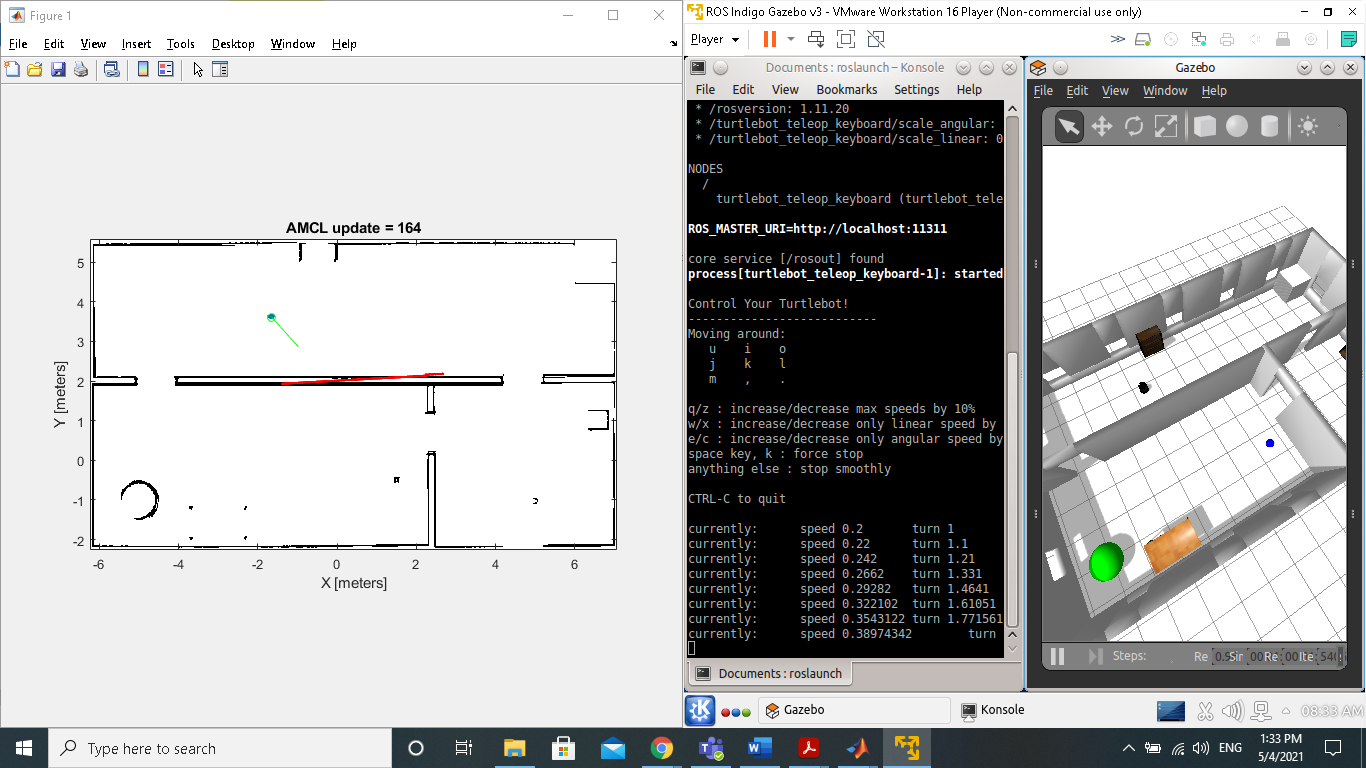


Figure 16 Accurate localization with 5000 particles

The HP value was then gradually decreased (to 3000), this resulted in a decrease in the localization accuracy. The robot position shown by the algorithm and the true robot position did not match, they were offset. This was more severe when the robot was moving through the doors. When the robot was passing through the door the algorithm was showing as if the robot was passing through the wall.

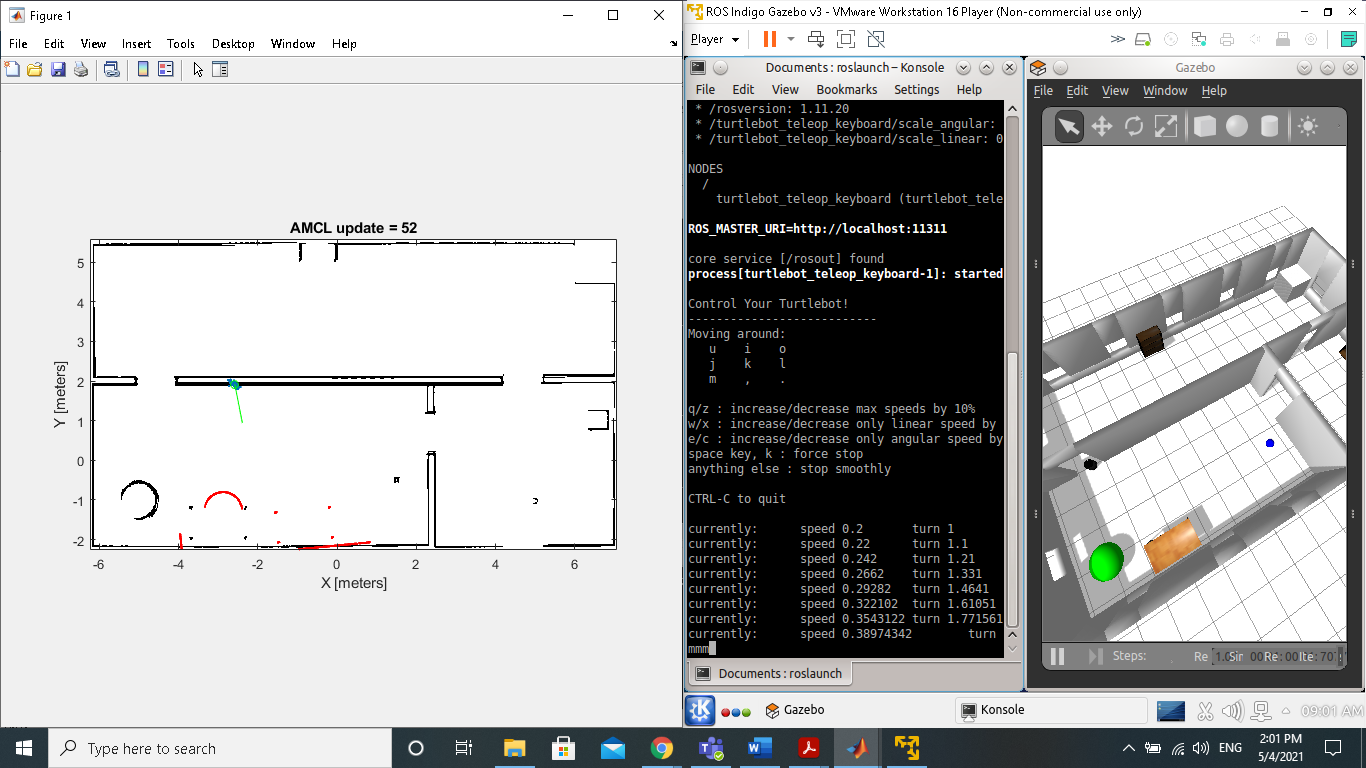


Figure 17 Error in localization when HP is initialized with a low value (4000 particles).

On further reducing the HP value (between 1000 to 3000) there was an even greater error between the true and predicted robot location. Sometimes the robot was not even localized when initially a smaller number of particles were generated near the robot’s true location.

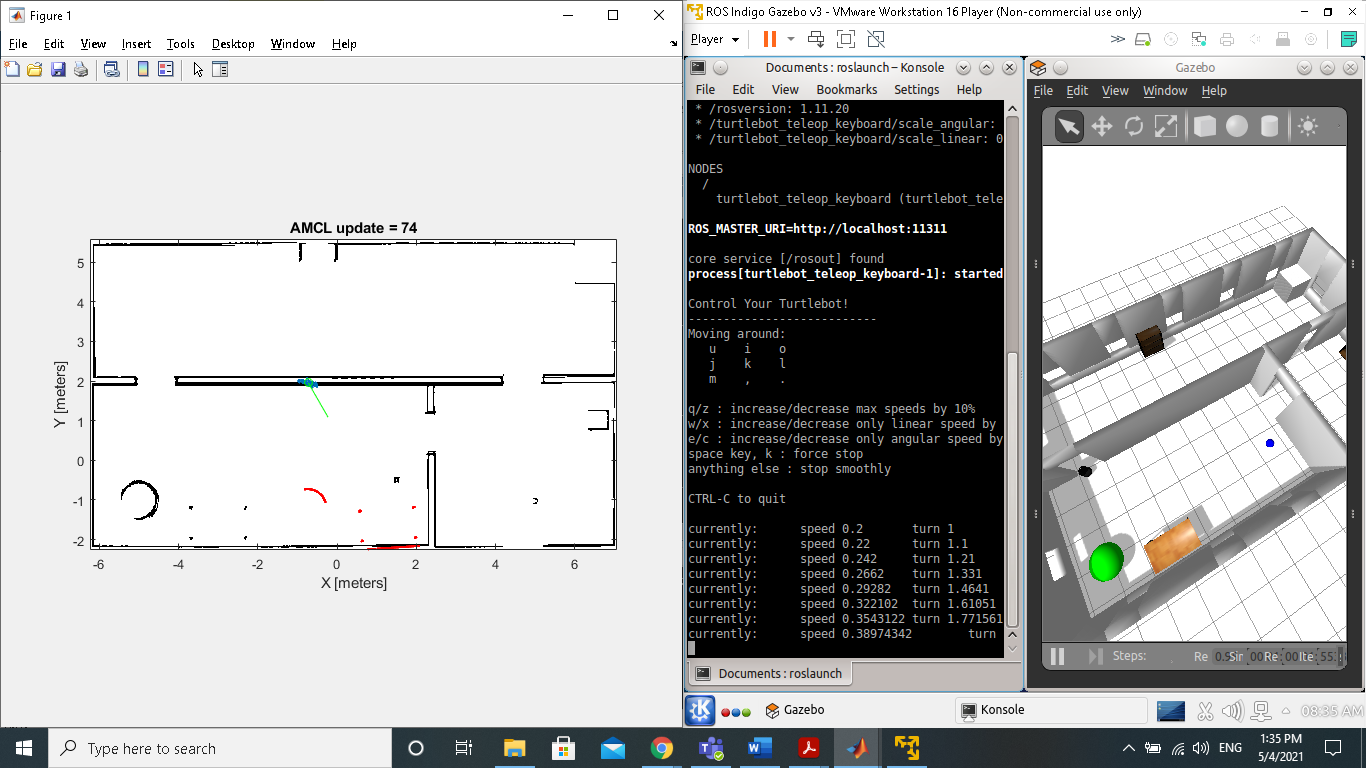


Figure 18 Increased error in localization on further reducing HP value (2500 particles)

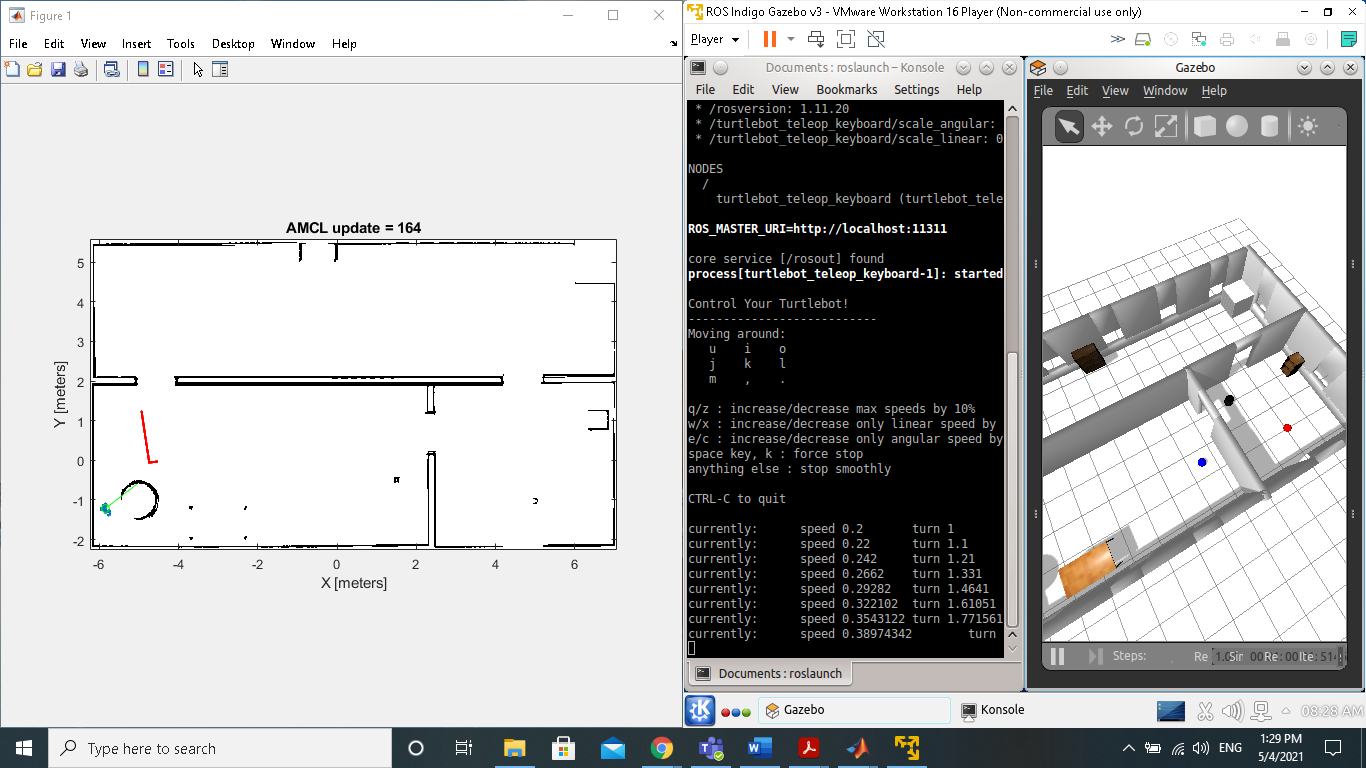


Figure 19 Robot failed to localize when HP value was initialized to 1000.

It was further observed that the LP value played a key role in constantly tracking the robot after localization. LP value is particularly important in case of symmetric environments. As the robot moves through the environment the particles are redistributed in the form of clusters. Each cluster represents a possible robot location. In symmetric environments elements such as objects, humans and doors help the robot to determine its location. If LP is initialized with a high value, then more clusters are formed throughout the map and as soon as a robot identifies a distinctive feature of the environment all the particles will be resampled to appear near the cluster which is close to the distinctive feature. If LP is initialized to a low value, only a few clusters are created and then there is a high probability that because of symmetry the cluster corresponding to the true location of the robot may be eliminated even before the algorithm detects any distinctive feature of the environment.

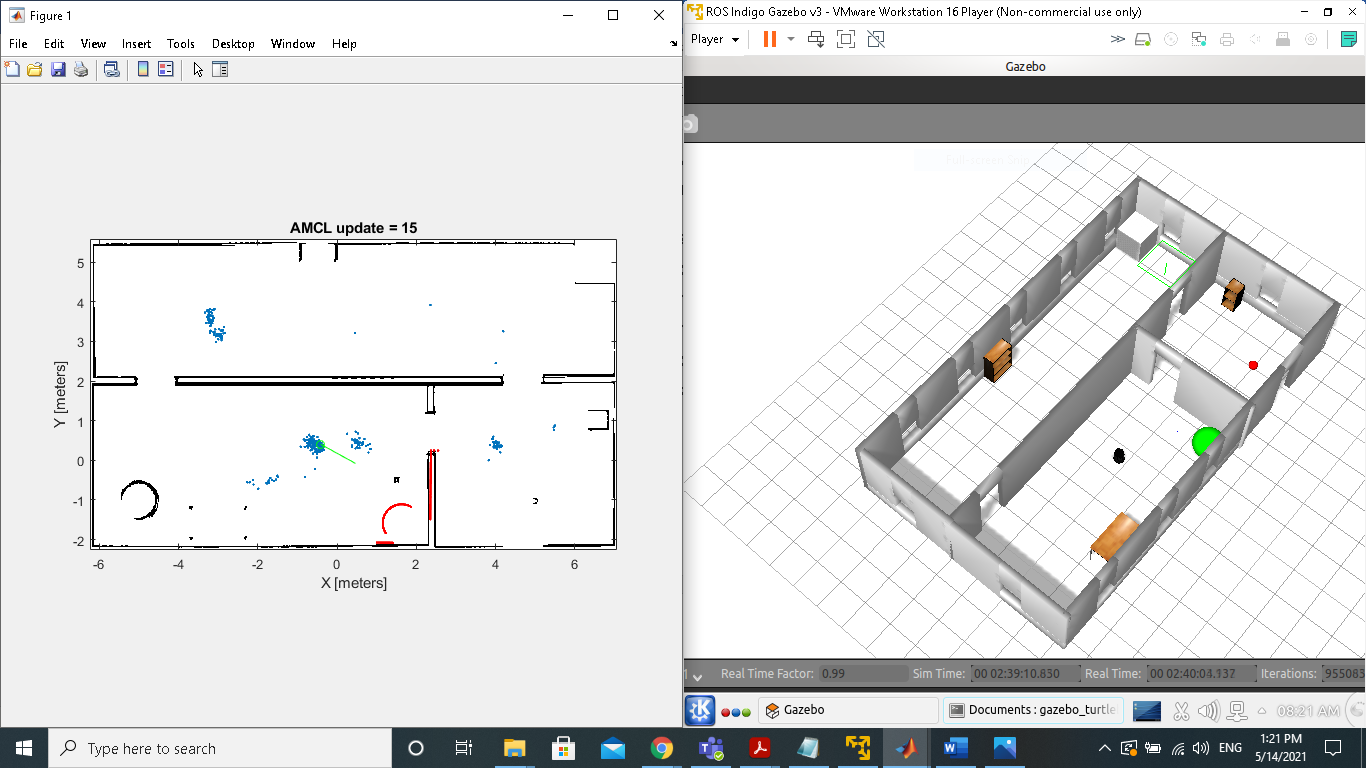


Figure 20 Particle clusters created at probable robot locations.

Chart, line chart

Description automatically generated

Figure 21 Comparison of localization error for different particle sizes

Next the effect of another important parameter the input map is studied. The robot localization was quick and efficient when a detailed map of the environment was provided. The input map contained a combination of sections with detailed and vague mapping of the environment. The robot localization was quick in the locations of the map which contained the details of the objects present in the environment. Hence a well detailed map will be useful in situations where accurate (symmetrical locations) and speedy (as in rescue robots) localization is required.

The office environment has three sections labelled 1, 2, 3. The impact of environment was also studied, and it was seen that the localization time was low, and accuracy was high in the sections of the office map with more features (like objects, doors, room corners etc). By matching the measurements of the end points of the range finder to the binary occupancy grid of the environment, the likelihood field method is used to calculate the probability of observing a range of measurements. Hence theoretically if the environment has stationary and regular shaped objects (like square shaped cabinets, rectangular bookshelves etc.) the likelihood field method can easily compare and match the features when compared to dynamic environments with irregular shaped objects (like plants).

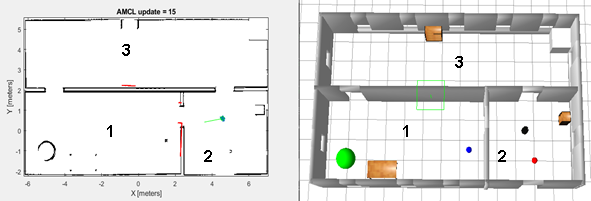


Figure 22 Robot localized by 15th update in the area of map with more features.

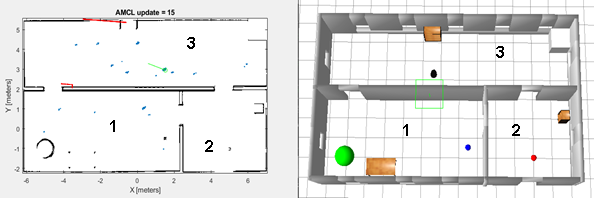


Figure 23 Robot not localized by 15th update in the area of map with less features.

Chart, line chart

Description automatically generated

Figure 24 Comparison of localization error for different zones.

Finally, the impact of robot navigation on the localization was studied. Two navigation schemes: manual and automatic were used to drive the TurtleBot. In both the schemes it was noticed that on driving the robot in a straight line it takes longer for localization. On the other hand, the MCL algorithm was able to achieve speedy localization when driving the robot in circles or in a zig zag pattern. This can be explained as starting at the same point, as the robot travels in circles or in a zig zag path it scans a larger area of the map in comparison to straight line motion at the same time. This phenomenon is particularly useful while navigating in environments which do not have many features. The VFH navigation algorithm used in the study drives the robot in a straight line until it detects an object and only then makes the robot to turn to avoid a collision. But manually controlling the robot gives us greater navigational freedom. Hence, manually driving the robot resulted in a faster localization.

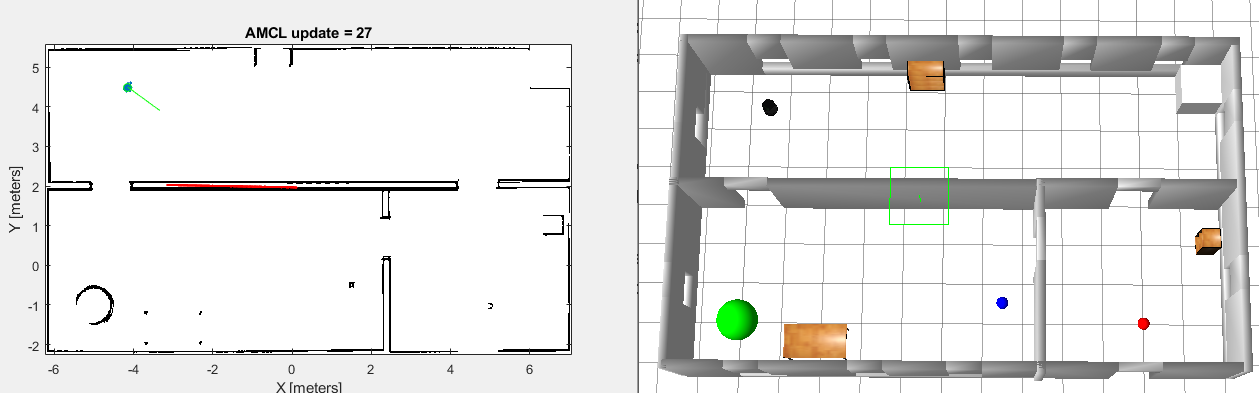


Figure 25 Robot driven in circles localized in the 27th update.

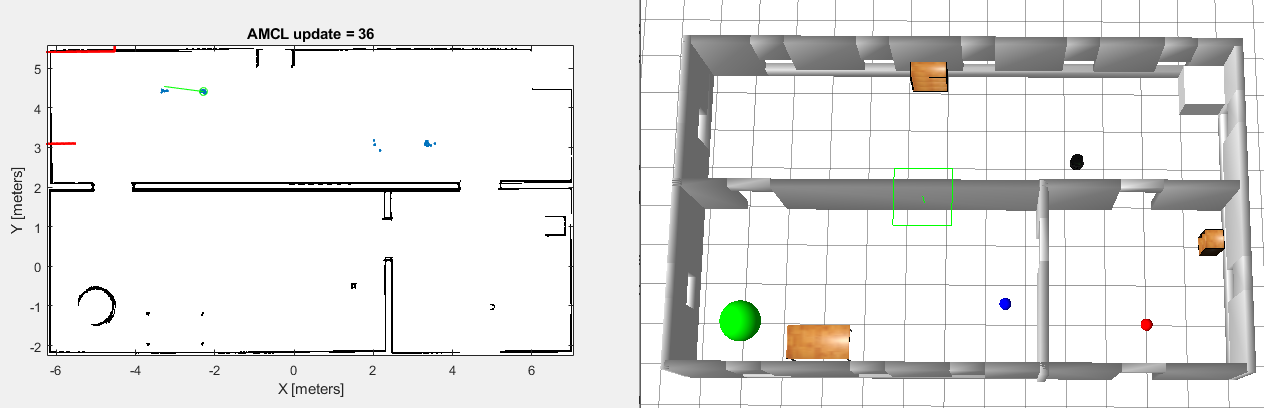


Figure 26 Robot driven in a straight line not localized even after 36th update.

Chart, line chart

Description automatically generated

Figure 27 Comparison of localization error for different navigation schemes

# Conclusion and future work

## Conclusion

Localization of a mobile robot based on ROS is introduced and analysed in this study. The MCL and VFH algorithms are used to develop and refine localization and navigation schemes respectively for a robot in an indoor environment. On the basis of the ROS robot, a comprehensive localization accuracy test was conducted. The results of the experiments demonstrate that the proposed scheme is practical. It can establish a wireless link between the ROS robot platform and the host computer node, as well as monitor the robot's actions remotely. The ROS-based mobile robot has a number of advantages, including high performance, low cost, and ease of extension. It has high robustness, precision, and real-time efficiency, and can effectively solve the problem of localization for mobile robots in indoor environments.

## Summary and evaluation

In this study a TurtleBot which is a simulation of a mobile robot has been successfully localized by using the Monte Carlo localization algorithm in an office environment in gazebo. The result of this study is a demonstration of the working of MCL algorithm in ROS and the effect of particle sizes on the localization process. Several combinations of particle sets were used and their effect on the localization accuracy was studied. The experiment suggests that starting with fewer particles in global localization results in incorrect localization or the algorithm may even fail to localize the robot. Additionally, retaining fewer particles as the algorithm progresses may result in the algorithm losing track of the robot or the particles converging in false location in case of symmetric environments. Effect of several secondary but important parameters like map, navigation and environment was also studied. It was observed that a detailed map, flexible navigation scheme and environments which are unsymmetrical with a few distinctive landmarks will all help in quick localization of robot.

## Project impact

This project helps in identifying the optimum number of particles required to localize a robot in an environment. Choosing the correct particle limits will ensure the robot localizes with minimal computational requirement. Different applications require different levels of localization accuracy. So, basing on the application, the localization accuracy can be adjusted by changing the maximum value of particle limits. Similarly, if it is known that the environment is complex or highly symmetric then the minimum value of particle limits is increased ensuring accurate robot localization. The project results also describe the effect of input map, environment, and navigation on the localization process. The results suggest that using a detailed map with a flexible navigation scheme, in an environment with distinct landmarks gives the most localization accuracy. Thus, this study gives a basic idea of how to choose a starting point for the robot in an environment and how to navigate the robot to achieve quick localization.

## Future Work

There are some drawbacks of MCL that should be addressed in future work.

**MCL Failure:** In the case of a symmetrical map, or if there are no particles near the true position of the robot, MCL will fail to indicate the correct location. This is known as the deprivation problem. More measurements can be taken in the future, either by incorporating more efficient and precise sensors (camera and range sensors) or by increasing motion control to ensure that the final position is the true robot pose.

**Active localization:** In the current method, robot path setting is purely passive. The navigation of robots is not designed to make the process of localization easier. As a result, future work could concentrate on developing a more robust mode of movement based on previously collected data in order to speed up the robot localization.

**Scalability of robots:** The improved efficiency of this particle-based localization technique can be extended to multi-robot applications in the future, where the particle sets of the various robots can be synchronized if one detects another. The first tests with two robots indicate that when they combine their samples, the robots are able to localize much faster. (Fox et al. 1999).

Due to the difficulty of recognising individual robots, current methods can only tackle cooperative localization between two robots. Current approaches are unable to differentiate which two robots are interacting when more than two robots meet. Applying different labels to each robot and adding cameras will solve this issue. Another significant scalability argument is that the type of status variable of each robot which is updated upon finding landmarks should be changed to an array, so that it can show the status of all feasible pairs of robots in order to handle the situation. And if a pair of robots have already shared their state estimates, they can still share knowledge with the remaining robots before they identify any landmarks.

# References

Andrew, A. M. (1999). Artificial intelligence and mobile robots: *Kybernetes, 28*(2), 243-246.

Arras, K. O., Castellanos, J. A., & Siegwart, R.Feature-based multi-hypothesis localization and tracking for mobile robots using geometric constraints. *, 2* 1371-1377 vol.2. doi:10.1109/ROBOT.2002.1014734

Austin, D. J., & Jensfelt, P.Using multiple gaussian hypotheses to represent probability distributions for mobile robot localization. *, 2* 1036-1041 vol.2. doi:10.1109/ROBOT.2000.844736

Borenstein, J., Everett, H. R., & Feng, L. (1996) *Navigating mobile robots: Systems and techniques,* USA, A. K. Peters, Ltd.

Burgard, W., Derr, A., Fox, D., & Cremers, A. B.Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach. *, 2* 730-735 vol.2. doi:10.1109/IROS.1998.727279

Burgard, W., Fox, D., Hennig, D., & Schmidt, T.Estimating the absolute position of a mobile robot using position probability grids.

Burgard, W., Cremers, A. B., Fox, D., Haehnel, D., & Lakemeyer, G. (1998). *Experiences with an interactive museum tour-guide robot.* ().

Cassandra, A. R., Kaelbling, L. P., & Kurien, J. A.Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. *, 2* 963-972 vol.2. doi:10.1109/IROS.1996.571080

COX, I. J. (1993). A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision, 10*(1), 53-66.

Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999). (1999). Monte carlo localization for mobile robots. Paper presented at the *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. no. 99CH36288C), , 2* 1322-1328.

Endres, H., Feiten, W., & Lawitzky, G.Field test of a navigation system: Autonomous cleaning in supermarkets. *, 2* 1779-1781 vol.2. doi:10.1109/ROBOT.1998.677424

Fox, D. (2003). Adapting the sample size in particle filters through KLD-sampling. *The International Journal of Robotics Research, 22*(12), 985-1003.

Fox, D., Burgard, W., Dellaert, F., & Thrun, S. (1999) *Monte carlo localization: Efficient position estimation for mobile robots,*

Gelb, A. (1974) *Applied optimal estimation,* Cambridge, Mass, M.I.T. Press.

Gutmann, J. -., & Schlegel, C.AMOS: Comparison of scan matching approaches for self-localization in indoor environments. 61-67. doi:10.1109/EURBOT.1996.551882

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering, 82*(1), 35-45.

Koller, D., & Fratkina, R. (1998). (1998). Using learning for approximation in stochastic processes. Paper presented at the *Icml,* 287-295.

Leonard, J. J., & Durrant-Whyte, H. F. (1992). Directed sonar sensing for mobile robot navigation. Retrieved from <https://doi.org/10.1007/978-1-4615-3652-9>

NOURBAKHSH, I., POWERS, R., & BIRCHFIELD, S. (1995a). DERVISH an office-navigating robot. *The AI Magazine, 16*(2), 53-60.

NOURBAKHSH, I., POWERS, R., & BIRCHFIELD, S. (1995b). DERVISH an office-navigating robot. *The AI Magazine, 16*(2), 53-60.

Peter S. Maybeck. (1979) *Stochastic models, estimation and control,* New York, Academic Press.

Pitt, M. K., & Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association, 94*(446), 590-599.

ROS.org.ROS/introduction - ROS wiki. Retrieved from <http://wiki.ros.org/ROS/Introduction>

Rubin, D. B. (1988). Using the SIR algorithm to simulate posterior distributions. *Bayesian Statistics, 3*, 395-402.

Shikhman, M. V.Monte carlo localization for mobile robot. *International School-Conference of Students, Postgraduates and Young Scientists,* 123-126.

The MathWorks, I. (a). Localize TurtleBot using monte carlo localization. Retrieved from <https://uk.mathworks.com/help/nav/ug/localize-turtlebot-using-monte-carlo-localization.html>

The MathWorks, I. (b). monteCarloLocalization. Retrieved from <https://uk.mathworks.com/help/nav/ref/montecarlolocalization-system-object.html>

The Mathworks, I.Avoid obstacles using vector field histogram. Retrieved from <https://uk.mathworks.com/help/nav/ref/controllervfh-system-object.html>

Thrun, S., Burgard, W., & Fox, D. (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots, 5*(3), 253-271.

Thrun, S., Burgard, W., & Fox, D. (2000). (2000). A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. Paper presented at the *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. no. 00CH37065), , 1* 321-328.

Thrun, S., Fox, D., & Burgard, W. (2000). (2000). Monte carlo localization with mixture proposal distribution. Paper presented at the *Aaai/Iaai,* 859-865.

Xiaoyu, W., Caihong, L., Li, S., Ning, Z., & Hao, F.On adaptive monte carlo localization algorithm for the mobile robot based on ROS. 5207-5212. doi:10.23919/ChiCC.2018.8482698

***Zeineh, R. (2001). Adaptive threshold monte carlo localization***