APPLICATION OF PARTICLE FILTERING
IN PREDICTABLE DYNAMIC TUBULAR ENVIRONMENTS

By

Karthik Puvvada

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

December, 2013

Nashville, Tennessee

Approved:

D. Mitchell Wilkes, Ph.D.

Richard Alan Peters II, Ph.D.

To my three shining examples of love, sacrifice and happiness

Syaami, Paaru and Priyanka

and

To Advaitham

ACKNOWLEDGEMENTS

This work would not have been possible without the incredible support of my adviser Professor Mitch Wilkes. His encouragement, humor and a burst of positive energy were some of the biggest driving forces for my success in this thesis. I am thankful for his faith on me.

Besides my adviser, I'd like to thank Professor Alan Peters for always being a great inspiration right from my first semester in the fall of 2011. I am also very grateful that Dr. Peters was willing to spend his time to review this thesis even on such a short notice.

I am grateful to all of my Vanderbilt grad school friends and staff who have supported me with wonderful smiles along the corridors and warm hugs every time I needed them. I personally like to mention Linda Koger who has gone out of her way always to help me in many administrative affairs throughout my education at Vanderbilt.

My family has contributed a great deal indirectly to this work by giving me the much needed love and support from India. I thank my ever-loving mom Parvathi for being a role model, my sister Priyanka for her limitless affection, and my grandmother Syamala Devi for teaching me unforgettable lessons in my childhood.

Last but not the least, I would like to thank my friends, Sharath Chandra Kuruganty and Bharat Chand, the closest of all, who helped me during the final stages of my thesis, enriching my self-confidence, discipline and guts.

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

Introduction

Robust localization is essential for successful navigation of autonomous mobile robots in any environment. Advanced techniques such as simultaneous localization and mapping (SLAM) also consider reliable localization as a critical component. The major existing localization algorithms like the Particle Filter, however, are, based on the assumption that the environment is static which does not hold true for some practical applications of mobile robots.

In this thesis, we explore a rather fledgling area of wireless robotic capsule endoscopy where dynamism is an inherent feature of the environment. Since the last decade, intense research has spurred in the field of robotic diagnostic technologies for inspecting the entire gastrointestinal (GI) tract for various diseases, such as obscure gastrointestinal bleeding (OGIB), tumors, cancer, Crohn's disease, and celiac disease. It is envisioned that the next generation of these micro-robots will traverse the GI tract with greater autonomy. In such self-propelling tiny capsule robots, the incorporation of mini-surgical tools and artificial intelligence is anticipated. Capsule robot localization and mapping of the environment inside the human body will be a key essential for successful autonomous navigation. This becomes a similar problem to that of standard mobile robot pose estimation and SLAM, but the difference being the complex dynamic environment that it operates in.

In this work, we report the existing state of affairs of several research activities related to capsule robot technology, technical challenges in such conditions, and finally propose an approach that explores extending the Particle Filtering paradigm to this new type of tubular elastic environment. In our methodology, we describe a restricted version of the dynamism problem, where the map changes with time in a predictable manner. A series of simulations reflecting several situations that can arise in an elastic colon-like environment were demonstrated. Our approach utilizes and adapts to a series of maps that are assumed to be determined a priori based on patterns of expansion and contraction of the colon walls. Experimental results demonstrate that in such scenarios the Particle Filter cannot be directly applied, but with clever changes in the algorithm, robust pose estimation and localization may be achievable.

This thesis is divided into 6 Chapters starting with this first chapter of introduction. Chapter 2 sheds light on the background of the problem we aim to solve in our research, the various motivations and the significance of extending standard mobile robot navigation paradigms into the recent domains like bio-robotics and miniature robot endoscopy.

In Chapter 3, we review the literature on the recent trends in innovation and research of several capsule robots, their state-of-the-affairs and the present challenges faced by scientists in this domain.

Chapter 4 outlines how we approach this problem and our methodology of applying a Particle Filtering technique in colon-like conditions. In this section, we also try to cover the mathematical background in order to prepare readers for the later sections. We start from deriving the basics of Bayes' filtering, which can in many ways be extended to other recursive filters including Particle filters.

In Chapter 5, we present the results of the implementation of Particle Filter in various scenarios in dynamic colon-like environment, and discuss what we learned from analyzing these results. This section covers several experiments that we performed to quantify the impact of dynamism in the pose estimation error and methods to minimize it. We implemented the algorithms in a Matlab environment, and readers can find these codes we used in the Appendix. Additionally, we present the particle cloud images as the robot moves in the dynamic map eventually localizing itself.

Lastly, we close with conclusions and summary in Chapter 6 and discuss possible future research in this promising field of robotics briefly.

Chapter 2

Background & Significance

In the past two decades, mobile robot localization and SLAM have received substantial interest within the AI and robotics community. The localization problem deals with estimating the pose of a robot relative to a fixed map [1] whereas SLAM addresses a more challenging problem of building a map of an environment and concurrently using this map to deduce the location of the robot. The solution to the simultaneous localization and map building (SLAM) problem is, in many respects, a "Holy Grail" of the autonomous vehicle research community. [2] SLAM is of inestimable value in a range of applications where absolute position or precise map information is unobtainable, including, amongst others, autonomous planetary exploration, subsea autonomous vehicles, autonomous air-borne vehicles, and autonomous all-terrain vehicles in tasks such as mining and construction.

The general localization problem also has attracted similar attention and has been the subject of substantial research because reliable position estimation is a key problem in mobile robotics especially in areas such as unmanned vehicle navigation systems and geophysical surveying. Both problems, mobile robot localization and SLAM have been formulated and solved at a theoretical and conceptual level in a number of different forms.

The paradigm of Markov localization has proven to be extremely effective in solving the mobile robot localization problem as it applies probabilistic representations for the robot's location, the outcome of actions, and the robot's observation. Markov localization has been employed by various groups with remarkable success [3, 4, 5, 6, 7]

The most popular algorithms that follows this framework can be classified into two categories: Grid-based Markov localization and Monte Carlo localization. Between the two, Monte Carlo localization (also called Particle Filter localization) has better performance parameters because it has reduced memory usage since its memory usage only depends on the number of particles and does not scale with size of the map, and can integrate measurements at a much higher frequency. Particle filters have been applied with great success to many real world localization and tracking problems, as documented by various chapters in [8].

Similarly, a number of approaches have been proposed to address the SLAM problem out of which, The EKF-SLAM and FastSLAM are the two most widely used solution methods.

FastSLAM algorithm, introduced by Montemerlo et al. [9], marked a fundamental conceptual shift in the design of recursive probabilistic SLAM. FastSLAM, with its basis in recursive Monte Carlo sampling, or particle filtering, was the first to directly represent the nonlinear process model and non-Gaussian pose distribution. FastSLAM quickly becase a highly popular technique because the algorithm decomposes the SLAM problem into a robot localization problem, and a collection of landmark estimation problems that are conditioned on the robot pose estimate. Interestingly, the success of SLAM and Particle Filter localization techniques has not by any means accomplished the much-coveted dream of totally autonomous mobile robot navigation. They both have a common deficiency that we shall discuss in detail throughout this research.

On a holistic level, a striking characteristic of the rich literature on this topic is that virtually all published work on SLAM and Particle Filters assumes that the environment is static. This assumption, called Markov assumption, is commonly made in the mobile robot navigation literature. It postulates that the robot's location is the only state in the environment which systematically affects sensor readings. This is not a trivial but a critical shortcoming, because, this static world assumption is in clear contrast to some robotic environments, which change over time. Thus, at the start of this research, we believed that a worthwhile research goal would be to investigate those applications where a dynamic environment is indispensably inherent and hence a major part of the problem.

Again, dealing with dynamism in a robot's environment is not a new prospect which is evident in the volume of research conducted by various groups discussed in chapter 3. However, upon profound investigation into their research, we discovered that dynamism in the robot's environment that was studied can be broadly classified into 2 types:

a) Hundreds of people moving around robot corrupting the sensor readings[10, 11,12]

b) Changes in the map over the time because a door is opened/closed, or a piece of furniture is displaced.[13]

In this research, we postulate that there is a third kind of dynamism that can arise in several areas of mobile robot navigation that are not explored to their totality as yet.
This is the dynamism inherently caused by the changes in the structure of the environment over a course of time either deterministically or non-deterministically. It was clear that the earlier techniques fail if the map is not static, but we explored areas of application that required newer takes on our standard algorithms.

No one has shed light on this aspect probably because in outdoor/indoor mobile robot navigation, it is hard to imagine such a scenario. But recent studies have shown that fields such as micro-robotics, robotic capsule endoscopy, robotic capsule colonoscopy, and medical robotics in general are increasingly borrowing concepts from mobile robotics. PillCam[14], PillCam Colon2[15] and PillCam SB[16] are some examples of tiny capsule size robots that can propel themselves in the interior of the human gastrointestinal tract to perform early cancer diagnosis. In these applications, the intestinal walls are stretchable and thus the environment in which the micro-robots have to operate is prone to this new kind of dynamism. This avant-garde medical micro-robotics domain is highly promising and we investigated during this research what kind of challenges this new application segment presents to standard mobile robot navigation methodologies and how we can address them.

Again, we believe this is a field that is going to revolutionize medical diagnosis and cancer studies because experimental tests from reputed studies showed that these capsule devices deform intestinal walls ten times less than conventional colonoscopies, even reaching the same objectives. [17] We envision that in the future, there will be very tiny robots in the nano-scale that can swim anywhere inside a living animal to perform diagnosis and even in some cases appropriate treatment. We are anticipating where robotics will achieve their goal of truly miniscule robots.

Upon further investigation into this field, we found that in many ways this scenario of reliable navigation within the gastrointestinal tract becomes exceedingly similar to the earlier discussed SLAM and robot localization problem. But with an added element of dynamism. Here, dynamism is clearly a huge part of the problem and is certainly indispensable. The environment is constantly changing because of respiration of the human subject or other fluid movements inside the tract. This change caused in the structure of the map over a course of time can either be deterministic or non-deterministic.

In most cases, the robot travels along the intestine lumen like a worm. This is very similar to how a mobile robot navigates in an indoor environment. But the biggest difference is here the environment where the robot traverses is elastic and the walls contract and relax periodically or aperiodically. Our research started at this point to investigate where the previous solutions to SLAM and mobile robot localization fail due to this new kind of dynamism in the environment and how to effectively apply these techniques in such a challenging and an obscure environment.

Chapter 3

Literature Review

In the past, several researchers have studied the problem of dynamism in the context of robot localization and SLAM. Many successful approaches have been proposed but there remain a number of open challenges.

As far as localization in non-static environments is concerned, one of the first strategies was described by Crowley [18] who uses a Kalman filter to fit lines on range measurements obtained from sonar sensors. Fox et al. [19], for example, use an entropy gain filter to identify the measurements caused by dynamic objects. Burgard et al. [20] additionally use a distance filter which selects individual measurements based on the difference between their measured and their expected distance. Montemerlo et al. [21] propose a method for tracking people while simultaneously localizing the robot which increases the robustness of the robot pose estimation.

Although these 'filtering of the range data' approaches have proven to be robust in certain dynamic environments, they discard valuable localization information and thus cannot be applicable in the context of our problem.

The problem of dealing with dynamism has also been investigated in the field of simultaneous localization and mapping (SLAM).

Wang and Thorpe [22] employ a feature-based heuristic to identify dynamic objects in range measurements and use the filtered result for localizing the robot and building a map at the same time. HϒahneI et al. [23] use a probabilistic method for tracking people and filter out the corresponding measurements to improve the map building process. Although these filtering approaches have proven to be robust in highly dynamic environments, they discard valuable localization information when the changes in the environment occur with a relatively low frequency.

Like aforementioned, there have been many researches on the detection and tracking of the moving objects like people around the robot. However, they have focused on the tracking of moving objects rather than the location change of dynamic landmarks.

In 2005, Wolf et al. proposed an on-line algorithm for SLAM in dynamic environments [24]. It is capable of differentiating static and dynamic parts of the environment and representing them appropriately on two grid maps with occupancy probability model.

However, it was later identified that performing SLAM with occupancy grids suffers from deteriorating map quality over time.

Mitsou and Tzafestas [25] suggest modifying the occupancy grid structure to maintain a history of sensor values. In a similar vein, some authors [26] presented a novel framework called Dynamic EKF SLAM that separates the landmarks into static and dynamic type. It estimates the optimal robot pose by combining the probability distribution models of the traditional EKF and the individual EKFs of the dynamic landmarks. However, this algorithm needs to cope with real robot platforms in real environments like the one we are discussing in this research.

From investigating all of the research cited above, one can classify the dynamism that that has been studied into two broad categories:

a) Dynamism in the environment due to moving people/vehicles/other robots.
b) Dynamism in the environment caused by re-arrangement of furniture like door/chair.

To a large extent, we can consider these two kinds of dynamism problems solved.
Only a few studies have been published about the third kind of dynamism that we discuss in this research. It is the dynamism that is part of the intestinal environment where it is not the objects/landmarks that change with respect to time, but since the outer walls are elastic, the entire structure is inherently dynamic.

Compared to outdoor and indoor environments, the inside of the human body is a complex environment making engineering design and visualization a formidable task.

There are some authors who have studied somewhat similar environments. Biber and Duckett [27] propose a spatio-temporal map where the environment is represented at multiple time-scales simultaneously. This adaptive map approach aims at generating a consistent representation of the environment. Another idea is about using sets of local maps as representation of the changing environment. Williams et al. [28] create a local submap of the features around the robot and fuse the local maps regularly with a global map. This approach is very useful for our research problem but the key difference is that it assumes the environment to be static and uses the gathered information for constructing globally consistent maps.

In the field of bio-robotics and medical robotics, wireless capsule endoscopy (WCE) has garnered great reputation over the conventional methods of endoscopy as it is a painless, effective, novel, diagnostic technology for inspecting the entire gastrointestinal

(GI) tract for various diseases, such as obscure gastrointestinal bleeding (OGIB), tumors, cancer, Crohn's disease, and celiac disease. [29]

In the field of WCE, rigorous research has been oriented towards the mechanical design aspects and miniaturization of the micro-robot so that it can traverse the GI tract safely and aid in medical diagnosis.

These tiny micro-robots, called capsule robots, are in their nascent stage of commercial application inside a human body. However, there is an increased attention in capsule endoscopy after a series of breakthroughs since 2001.

An Israeli company, Given Imaging Ltd., developed the first commercial disposable capsulated pill named M2A, as illustrated in Figure 3.1. The single capsule design incorporates a light source, a miniature complementary metal oxide semiconductor (CMOS) color camera, a battery, an antenna, and a radio transmitter.

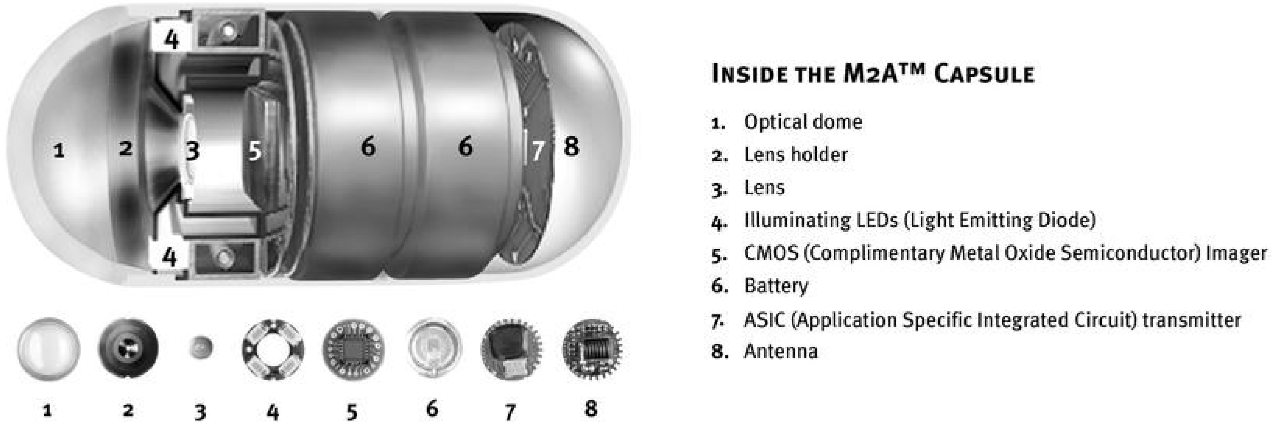

4
1
2
3
5
6
6
7
8
4
INSIDE THE M2A™ CAPSULE
1. Optical dome
2. Lens holder
3. Lens
4. Illuminating LEDs (Light Emitting Diode)
5. CMOS (Complimentary Metal Oxide Semiconductor) Imager
6. Battery
7. ASIC (Application Specific Integrated Circuit) transmitter
8. Antenna
1
2
3
4
5
6
7
8

Figure 3.1, image of M2A

The capsule is the size and shape of a pill typically 20-25 mm in length, 11 mm diameter and contains a tiny camera. After a patient swallows the capsule, it takes pictures of the inside of the gastrointestinal tract like the one shown in Figure 3.2.

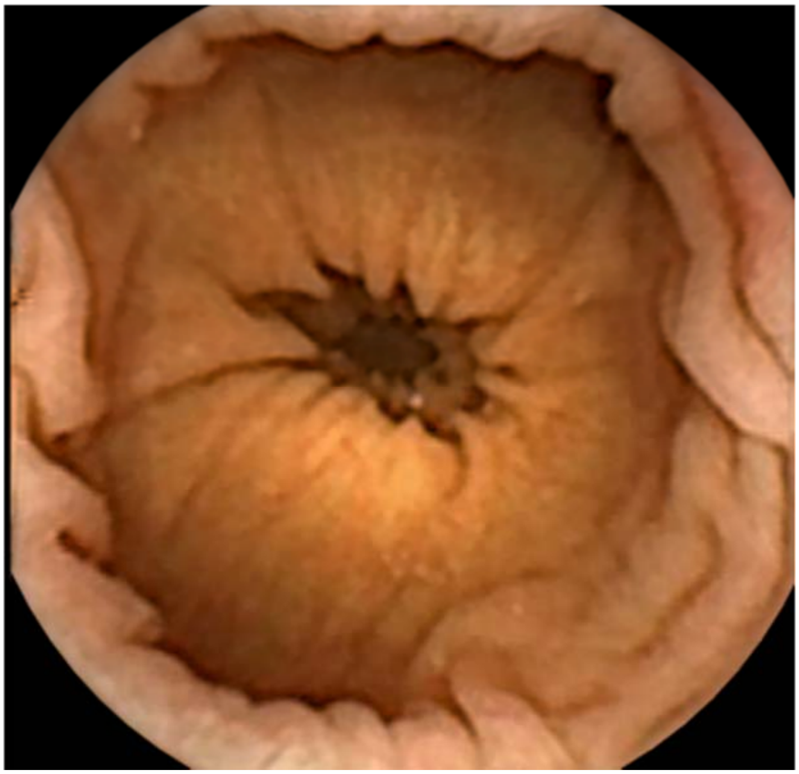


Figure 3.2, Image of the colon acquired by capsule endoscopy

Due to the dissimilarities of different areas of the GI tract, Given Imaging, Ltd. launched its esophagus-specific and colonspecific capsules, that is, PillCam ESo and PillCam Colon, respectively. In addition to Given Imaging Ltd., Jinshan (China), Olympus (Japan), and IntroMedic (Korea) made significant improvements in the performances of their own endoscopic capsules.

Even so, all of the present capsules that are used move through the GI tract by the natural motility of the tract itself. There is no control of the capsules' movements or camera orientation, which limits the accuracy of medical diagnoses. Along with this, the poor image resolution and low frame rate confine the wider application of the capsules for sensitive detection. Overcoming these limitations and developing the next generation of such capsules with enhanced functionality and control is the objective of many research communities in this area.

One promising step in that direction is a microcapsule named MiRO #1, which is being developed by the Intelligent Microsystem Center [30] in Korea. As illustrated in Figure 3.3, it has the ability to advance forward, backward, orientate, stop, and anchor itself onto the colon wall at will.

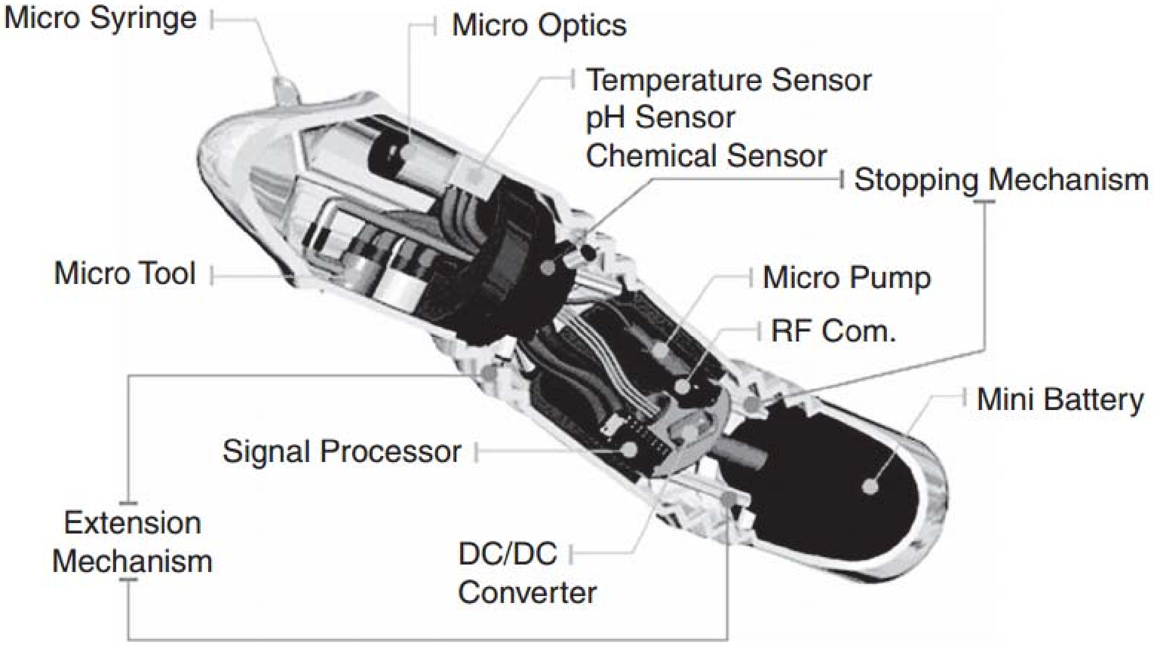


Figure 3.3, Image of the MIRO#1 micro-robot

Only 10 mm in diameter and 20 mm in length, it is equipped with a micromanipulator arm that is able to perform therapeutic procedures like taking tissue samples and administering an injection. The images are transmitted wirelessly, and it is also integrated with a position tracking device. A transmitted radio-frequency signal is typically used to accurately estimate the location of the capsule and to track it in real time inside the body and gastrointestinal tract.

Several researchers [31] argue that a better technique would be to inflate the colon, and then make the capsule should enter the tract, so that it can detect abnormalities in the colon with greater effectiveness. With the implementation of a micromanipulator arm and the ability to control the capsule's movement and orientation at will, they predict that this would be a huge advantage when evaluating the small intestine.

They further envision that Capsule endoscopy will reach a stage where both diagnosis and therapeutic procedure can be performed simultaneously. In the future, it would not be as hard as it is now to design an active robot that can move on its own and against the natural peristalsis of the gastrointestinal (GI) tract. This would be a great advantage to the community of medical diagnosis.

We aim to push the boundaries in this particular context. We believe that an active self-propelling micro-robot that is capable of navigating through the intestines is certainly possible in the coming decades. In this research, we explore how far the standard navigation techniques that are widely popular in mobile robotics in outdoor and indoor environments can be applied in this avant-garde and challenging environment.

Chapter 4

Our Methodology

With this research, we focus on making it clear to the robotics community on what assumptions to make, what set of challenges to consider and what approaches to use while adapting the standard algorithms like particle filtering in such avant-garde application domains. Our performance results in this pursuit reveal that although it is certainly not directly feasible to account for all new challenges this new tubular environment presents, a vast majority of them can be addressed by following a set of simplification techniques and appropriate assumptions. We hope this research is a starting step to many more things to do.

In Chapter 4, we described the status of the research in capsule robot endoscopy and the challenges that arise in operating in dynamic intestinal environments. In this chapter, we dive into the details of such environments and how they can be modelled for a feasible implementation of standard robot localization algorithms. We further discuss the methodology that we have used to attack this problem in the most effective yet computationally tractable manner.

4.1 Our Environment Model

In this particular research, we focus our attention on the human colon environment that is traditionally examined by a procedure called "Colonoscopy". Harvard School of Public Health (HSPH) states that colonoscopies could prevent 40% of colorectal cancers. [32] With this insight in mind, several scientific centres like CODIR[33] are exceedingly working towards building next generation robots that can swim/submerge in the colon environment and perform crucial tasks like visual examinations and biopsy. Based on the suggestions of various researches who are working on Pill Colonoscopy [34], we have identified that there was a great need for models that experimented with the colon.

Due to these compelling factors we selected the human colon to be our organ of interest in this thesis. Below is the actual shape of a human colon which has several divisions as illustrated in Figure 4.1.

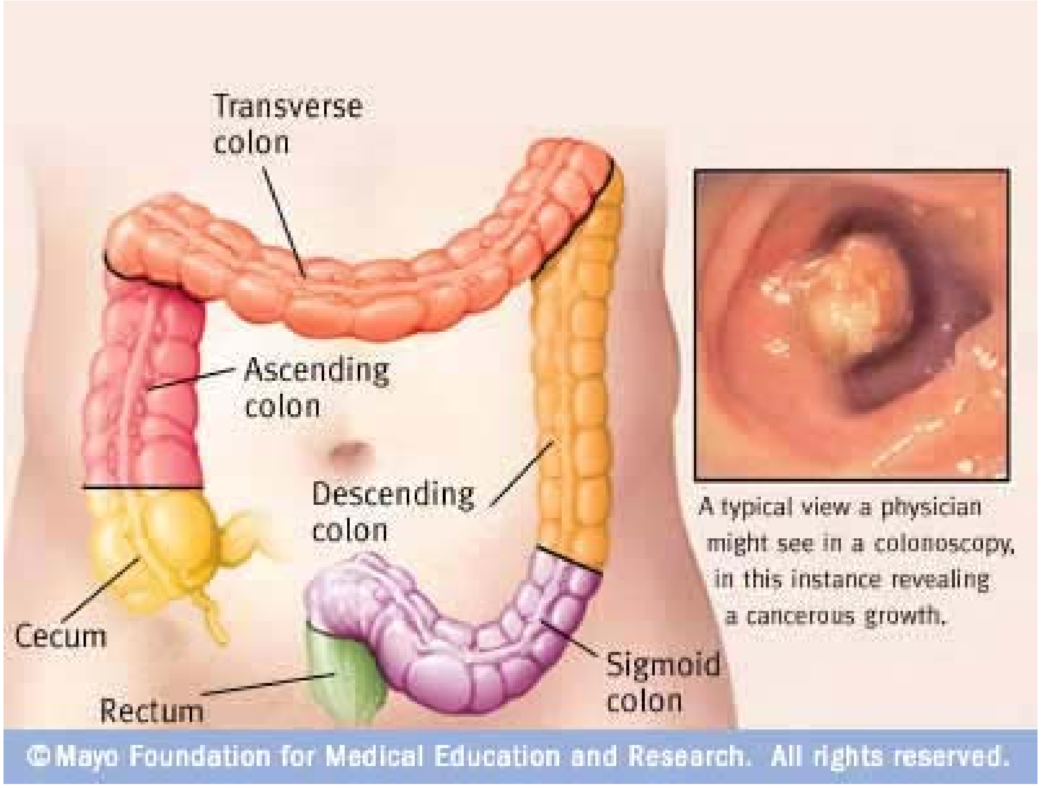

Transverse colon
Ascending colon
Descending colon
Cecum
Rectum
Sigmoid colon
A typical view a physician might see in a colonoscopy, in this instance revealing a cancerous growth.
©Mayo Foundation for Medical Education and Research. All rights reserved.

Figure 4.1: Human colon and its various sections [Mayo]

Figure 4.2 shows the outline of a segment of colon that we have chosen to build our model after. It is called **"Transverse Colon"** and has several ridges like other parts of the colon.

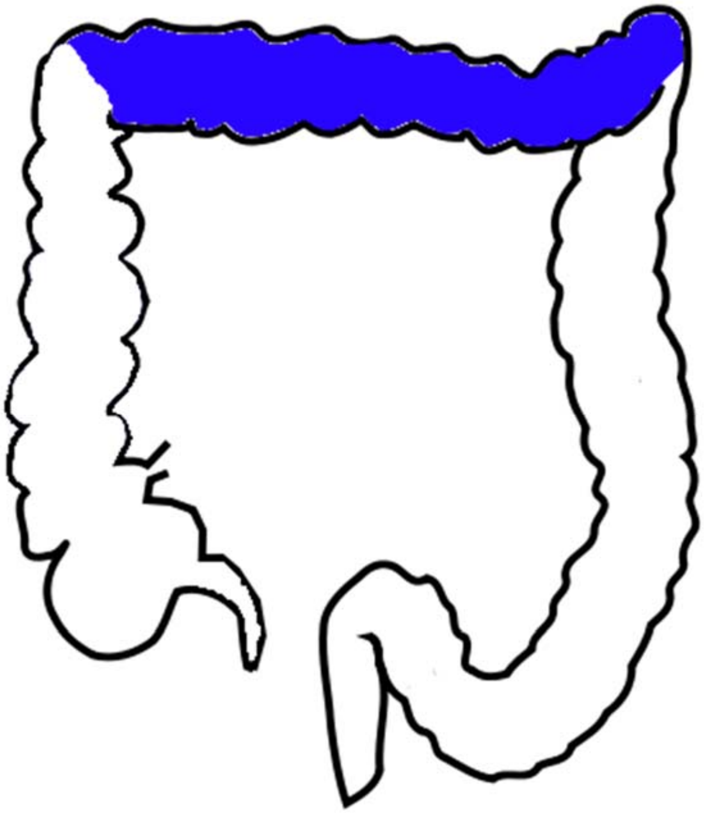


Figure 4.2: Transverse Colon (area filled with blue)

The transverse colon, the longest and most movable part of the colon, passes with a downward convexity from the right hypochondrium region across the abdomen, opposite the confines of the epigastric and umbilical zones, into the left hypochondrium region, where it curves sharply on itself beneath the lower end of the spleen, forming the splenic or left colic flexure. The right colic flexure is adjacent to the liver. In its course, it describes an arch, the concavity of which is directed backward and a little upward; toward its splenic end there is often an abrupt U-shaped curve which may descend lower than the main curve. [35]This specific part of the colon was interesting to us because it provides a better navigation path without too many turns and bends for the capsule robot.

Inspired by the design of several capsule endoscopy robots like PillCam Colon2 shown in Figure 4.3, we modelled our actual robot simulator to be extremely small so that it can operate within the transverse colon's space with great freedom. It is further assumed that the micro-robot has active navigation capabilities like advancing forward or backward or in any given direction. The details of the robot model will be discussed in later parts of this chapter.

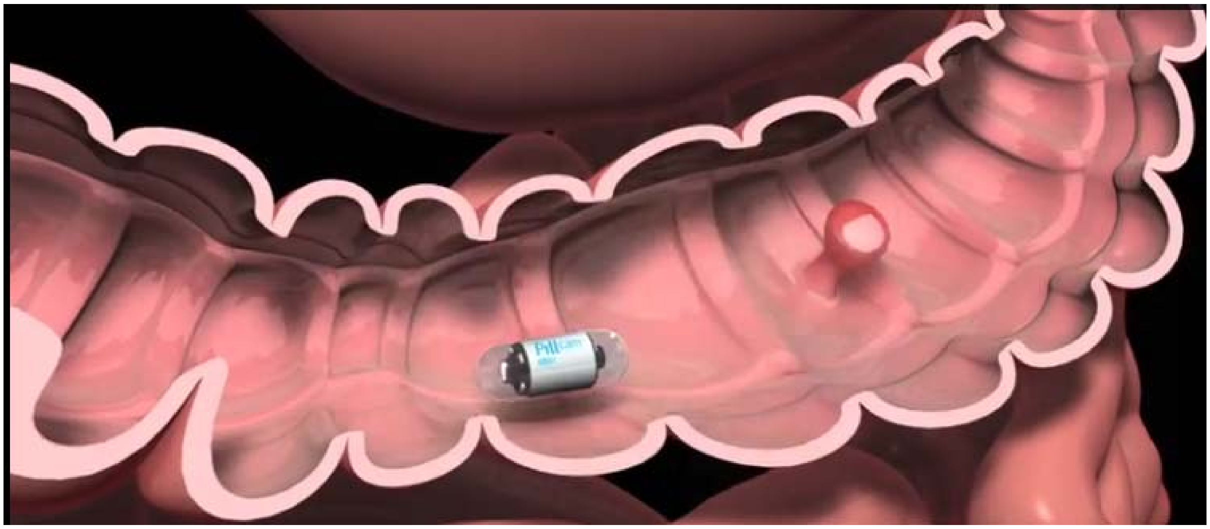


Figure 4.3: Image from a 3D simulation of PillCam Colon2 moving in transverse colon

Dimensionally speaking, the total human colon averages 150 cms (60 inches) in length. The transverse colon is usually over 18 inches (45 cm) in length and extends across the upper abdomen to the splenic flexure.

From a robotic mapping point of view, the general situation inside the human transverse colon is extremely complex with millions of features and mucous layers making it a high dimensional problem. But colonoscopic procedures followed by medical examiners make use of several cleansing apparatus and they also inflate the colon using CO2 or Water to maximum capacity. This is sometimes called Robotic HydroColonoscopy as suggested in [36]. This provides greater freedom for the robot to navigate inside and for the camera equipment mounted on the robot to function effectively.

Since, the level of complexity that the navigation inside an internal organ presents is too high dimensional to perform our research at these early stages, for simplicity, we have considered building a 2D model of the organ. We built a 2D outline map of **“Traverse Colon”** using Adobe Photoshop as shown in Figure 4.4. This map of the colon environment thus serves as the global map that is used in several localization problems in autonomous mobile robotics. It is to be observed that the transverse colon has various ridges and bends and is uneven throughout its structure. This is another deviation from a traditional map used in localization problems where the structure is more uniform and rigid.

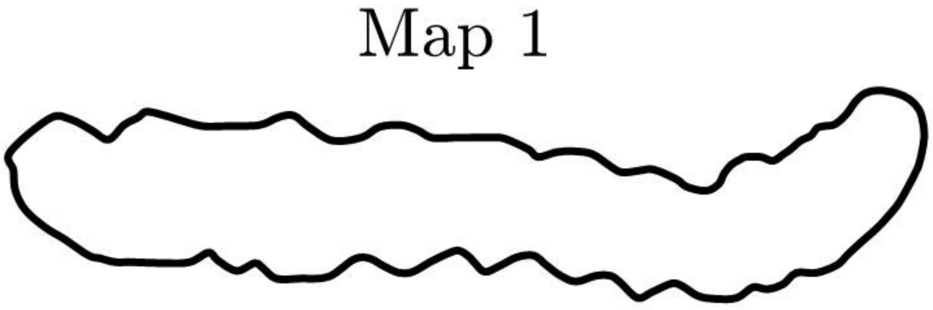


Figure 4.4: One of the transverse colon maps used in our algorithm

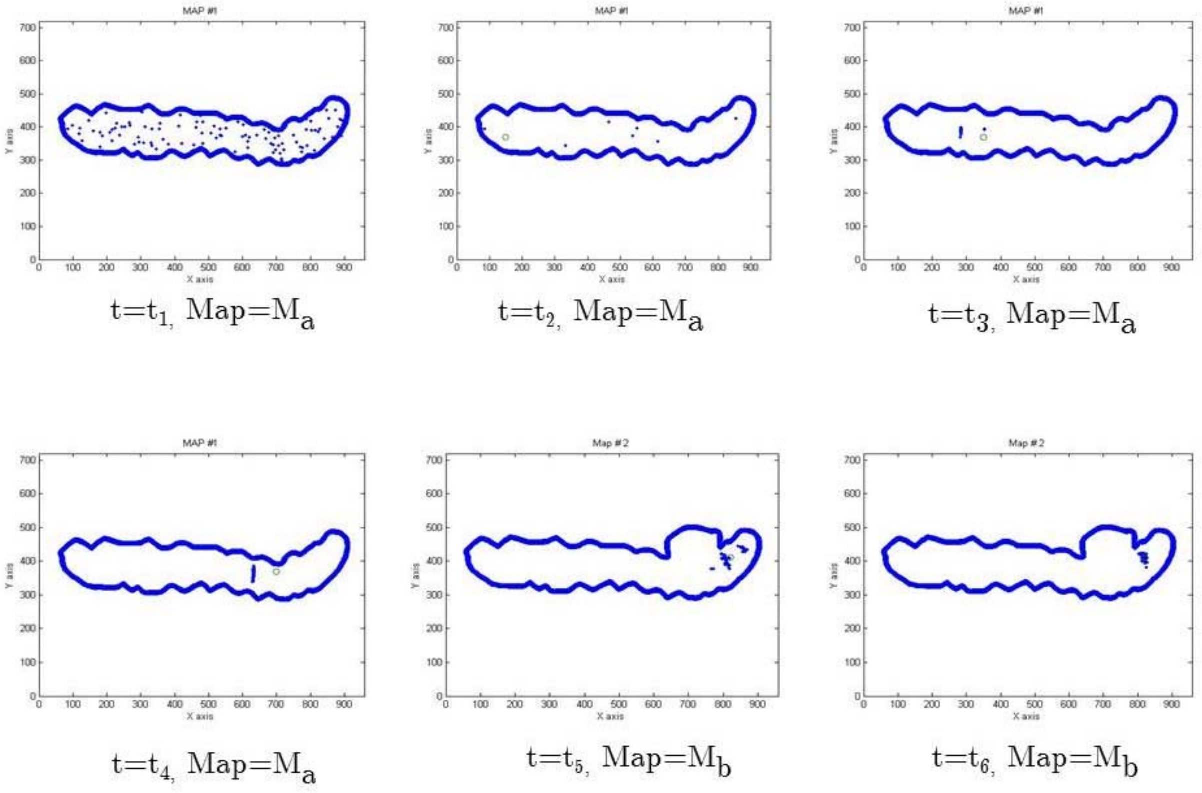
The ends of the transverse colon extend themselves into other fixtures in reality but in the scope of this research, we assume that they are closed so that the colon segment looks like a closed tube as depicted in Figure 4.4.

Putting these simplifications together, the problem now becomes exceedingly similar to a conventional mobile robot localization in a given map. We know that this localization problem, in other words, robot pose estimation, can be addressed using a Sequential Monte Carlo algorithm.

4.2 Dynamism in the Environment

In this research, we investigate a restricted version of the dynamic environment problem. To account for the inherent dynamism present in an intestinal environment such as the transverse colon, we have utilized a series of maps. Dynamism arises due to the change in the state of the environment. This is caused by natural respiration patterns of the examinee resulting in contraction and expansion of the outer layers of the colon. Thus, the structure of the transverse colon changes with time. The extent to which the change in the map can happen is an unknown parameter as of current state of the colonoscopic research and is beyond the scope of this research. However, it is assumed that it will be possible in the future for advanced medical diagnosticians to predict the accurate change in the map by observing thousands of examinee images over extended periods of time. In the light of such assumption, we consider that we have a way by which the present shape of the transverse colon is estimable at any given instant of time. Ofcourse, in reality we expect there may still be some random perturbations in the map, but for our initial work we assume a predictable model. In our model, a series of maps are built on the top of one another with gradual changes in certain regions to simulate the dynamism described above.

Figure 4.5 shows the maps M_a and M_b at six time instants t_1, t_2, t_3, until t_4 respectively.



$t=t_1$, Map=M_a $t=t_2$, Map=M_a $t=t_3$, Map=M_a

$t=t_4$, Map=M_a $t=t_5$, Map=M_b $t=t_6$, Map=M_b

Figure 4.5: Maps with changes in a region over the time t_1 to t_6

As in the standard mobile-robot localization problem, here the map is always available. Due to the dynamism in the structure of the environment, the map varies with time. This is a unique constraint that pertains only to such tubular intestinal environments. But since we assume that it changes deterministically, the standard Particle Filtering algorithm can still be applied as discussed below.

4.3 Particle Filter Theory

For estimating the pose of the robot relative to the environment, our localization framework relies on a standard particle filter paradigm. The reason we choose Particle

Filtering is because of its central presence in both the localization and SLAM solutions. In most variants of the mobile robot localization problem, particle filters have been consistently found to outperform alternative techniques, including parametric probabilistic techniques such as the Kalman filter and more traditional. The Particle Filter has been implemented with as few as 50 samples [37] on robots with extremely limited processing and highly inaccurate actuation, such as the soccer playing AIBO robotic shown in Figure 4.6.

Figure 4.6: Particle filters have been used successfully for on-board localization of soccer-playing Aibo robots with as few as 50 particles.

In Particle Filter localization, we are interested in estimating the state of the robot at the current time-step k, given knowledge about the initial state and all measurements Z^k=$\{z_k, i = 1 \ldots k\}$ up to the current time. Typically, we will work with a 3-dimensional state vector X = [x, y, θ], i.e. the position and orientation of the robot.

This estimation problem is an instance of the Bayesian filtering problem, where we are interested in constructing the posterior density $p(X_k \mid Z^k)$ of the current state conditioned on all measurements. In the Bayesian approach, this probability density function (PDF) is taken to represent all the knowledge we possess about the state X_k, and from it we can estimate the current position.

The key idea of particle filters is to represent the posterior by a set of weighted samples or particles, where each particle corresponds to a potential pose of the robot.

Summarizing, to localize the robot we need to recursively compute the density $p(X_k \mid Z^k)$ at each time-step.

4.3.1 Particle Filter Theory

Particle filters inherit a lot of their mathematical explanation from Bayes filters. For the sake of consistency, here we briefly derive the basics, beginning with Bayes filters.

Bayes filters address the problem of estimating the state X of a dynamical system from sensor measurements. For example, in mobile robot localization the dynamical system is a mobile robot and its environment, the state is the robot's pose therein (often specified by a position in a two dimensional Cartesian space and the robot's heading direction θ), and measurements may include range measurements, camera images, and odometry readings.

Odometers are sensors that measure the revolution of a robot's wheels. As such they convey information about the change of state. Even though odometers are sensors, we will treat odometry as control data, since they measure the effect of a control action.

Control data is denoted by u_t. The variable u_t will always correspond to the change of state in the time interval (*t-1;t*]. As before, we will denote the sequence of control data by $u_{t1:t2}$, for $t1 \leq t2$:

$u_{t1:t2}$= u_{t1}, u_{t+1}, u_{t+2}, ...u_{t2}

Bayes filters assume that the environment is Markov, that is, past and future data are (conditionally) independent if one knows the current state.

The key idea of Bayes filtering is to estimate the posterior probability density over the state space conditioned on the data. In the robotics and AI literature, this posterior is typically called the **belief.** Throughout this chapter, we will use the following notation:

$$Bel(x_t) = p(x_t|d_{0\ldots t})$$

Here x denotes the state, x_t is the state at time t, and $d_{0\ldots t}$ denotes the data starting at time 0 up to time t.

For mobile robots, we primarily have two types of data: *perceptual/sensor data* such as sonar/laser range measurements and *odometry data* or *controls* which carries information about robot motion.

Denoting the sensor measurements by z and the motion control by u we have,

$$Bel(x_t) = p\big(x_t \mid z_{t,}u_{t-1}, z_{t-1,}u_{t-2}, z_{t-2,}u_{t-3} \ldots\ldots u_{0,}z_0\big) \qquad (2.1)$$

Without loss of generality, we assume that observations and actions occur in an alternating sequence. Note that the most recent perception in Bel(x_t) is z_t whereas the most recent controls/odometry reading is u_{t-1}.

Bayes filters estimate the belief recursively. The initial belief characterizes the initial knowledge about the system state. In the absence of such knowledge, like in global localization, it is typically initialized by a uniform distribution over the state space.

To derive a recursive update equation, we observe that Expression 2.1 can be transformed by Bayes rule to:

$$Bel(x_t) = \frac{p(z_t \mid x_{t,}u_{t-1} \ldots..z_0)\, p(x_t \mid u_{t-1} \ldots..z_0)}{p(z_t \mid u_{t-1} \ldots..z_0)}$$

$$= \frac{p(z_t \mid x_{t,}u_{t-1} \ldots..z_0)\, p(x_t \mid u_{t-1} \ldots..z_0)}{p(z_t \mid u_{t-1}, d_{0....t-1})}$$

(2.2)

The Markov assumption states that measurements z_t are conditionally independent of past measurements and odometry readings given knowledge of the state x_t:

$$p(z_t \mid x_{t,}u_{t-1} \ldots..z_0) = p(z_t \mid x_{t,})$$

This allows us to conveniently simplify Equation (2.2):

$$Bel(x_t) = \frac{p(z_t \mid x_t)\, p(x_t \mid u_{t-1} \ldots..z_0)}{p(z_t \mid u_{t-1}, d_{0....t-1})}$$

To obtain our final recursive form, we now have to integrate out the pose x_{t-1}, at time t -1, which yields

$$= \frac{p(z_t \mid x_t)}{p(z_t \mid u_{t-1}, d_{0....t-1})} \int p(x_t \mid x_{t-1}, u_{t-1}, \ldots..z_0)\, p(x_{t-1} \mid u_{t-1}, \ldots..z_0) dx_{t-1}$$

The Markov assumption also implies that given knowledge of x_{t-1} and u_{t-1}, the state x_t is conditionally independent of past measurements $z_1, z_{2,} z_3 \ldots. z_{t-1}$ and odometry readings $u_1, u_{2,} u_3 \ldots. u_{t-2}$, upto time t-2, that is:

$$p(x_t \mid x_{t-1}, u_{t-1}, \ldots..z_0) = p(x_t \mid x_{t-1}, u_{t-1})$$

Using the definition of the belief *Bel*, we obtain a recursive estimator known as Bayes filter:

$$Bel(x_t) = \frac{p(z_t \mid x_t)}{p(z_t \mid u_{t-1}, d_{0....t-1})} \int p(x_t \mid x_{t-1}, u_{t-1})\, Bel(x_{t-1}) dx_{t-1}$$

$$Bel(x_t) = \eta\, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_{t-1})\, Bel(x_{t-1}) dx_{t-1} \qquad (2.3)$$

where η is a normalizing constant. This equation is of central importance, as it is the basis for our Particle Filter algorithm described below.

4.3.2 Models of Robot Motion and Perception

In the context of mobile robot localization, Bayes filters are also known as Markov localization. To implement Markov localization, one needs to know three distributions:

a) The initial belief $Bel(x_0)$ (e.g., uniform),

b) The next state probability $p(x_t \mid x_{t-1}, u_{t-1})$ called the motion model

c) The perceptual likelihood $p(z_t \mid x_t)$ called the perceptual model.

The specific shape of these probabilities depends on the robot's odometry, and the type of sensors used for localization. Both of these models are time-invariant: we will henceforth omit the time index t.

4.3.3 Our Motion Model

In the methodology of this thesis, we have chosen a specific motion model which is the result of combining conventional robot kinematics with two independent zero-mean random variables, one of which models noise in rotation, and one models translational noise. The values depend on the odometry of the wheels and are usually pre-determined. Since we are just performing simulations, we assume a standard Gaussian with zero mean and a standard deviation of 0.1 for both rotational and translational noise. The model is easily coded using MATLAB. The idea is to simulate the movement of a capsule robot in the given map. In this localization approach, initially, the pose of the robot is given, i.e. the state X_0 is known. From here, using the sensor readings from the perception model, and our likelihood pdf, we make the micro-robot move in the map so that we can recover the pose over a series of prediction and updation stages. The robot simulator as described above was built using MATLAB code and helps us move anywhere in X-Y Cartesian co-ordinate system with a certain heading direction, θ.

4.3.4 Our Perception Model

The perceptual model $p(z \mid x)$ usually depends on the specific sensor. In our case, however, we assume the micro-robot is equipped with 5 range finders for perception placed at θ=0^0, 30^0, 60^0, 300^0, 330^0 with respect to horizontal (x) axis. Such sensors measure the distance of the robot to nearby obstacles using sound or structured laser light. We do not delve into the details of which of them would be practically applicable, however, we believe laser range finders are more accurate but the trade-off is their size and cost. We believe in the future, there will be a possibility where the micro-robots will be equipped with high-precision sonar/laser range finders or possibly modulated Infra-red sensors without worrying about size and cost factors.

The specific density $p(z \mid x)$ is computed in two stages. First, the measurement in an ideal, noise-free environment is calculated. For laser range finders, this is easily done using ray-tracing in a geometric map of the environment, such as the one shown in Figure 4.7.

Second, the desired density $p(z \mid x)$ is obtained as a mixture of random variables, composed of one that models the event of getting the correct reading additive with small Gaussian-distributed measurement noise, one for receiving a max-range reading (which occurs frequently), and one that models random noise and is exponentially distributed.

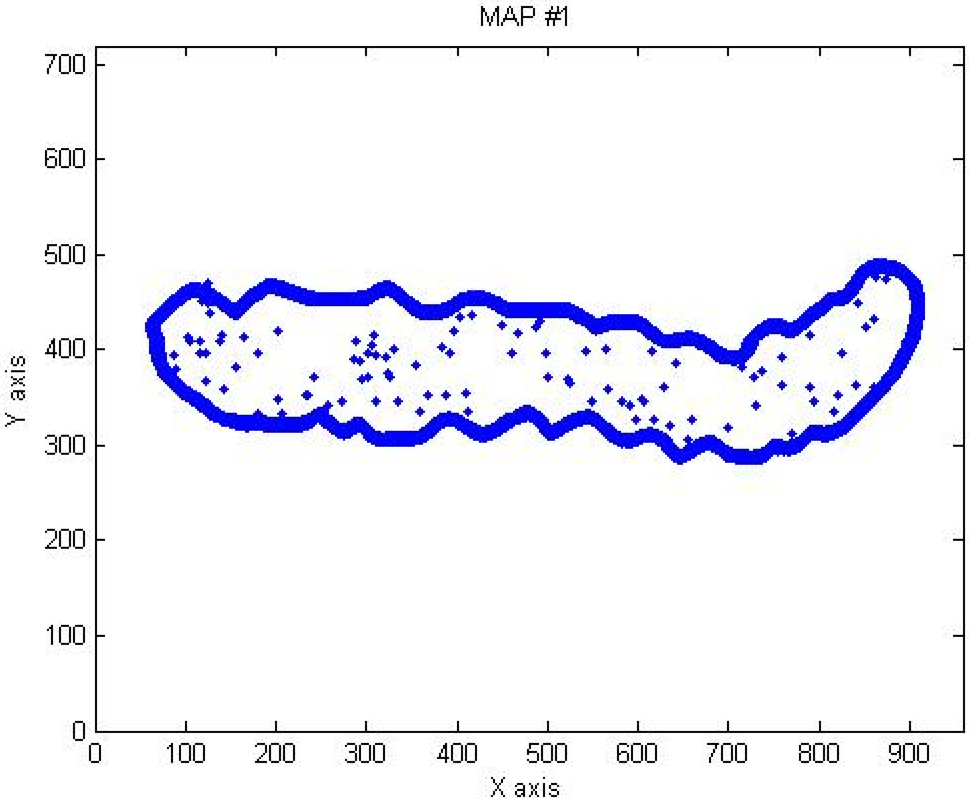


Figure 4.7

4.3.5 Implementation as Particle Filter

If the state space is continuous, as is the case in mobile robot localization, implementing the belief update equation (2.3) is not a trivial matter, particularly if one is concerned about efficiency. The idea of a particle filter algorithm is to represent the belief $Bel(x)$ by a set of *m* weighted samples distributed according to $Bel(x)$.

$$Bel(x) = \{x^{(i)}, p^{(i)}\}_{\,i=1,2,\ldots m}$$

Here each $x^{(i)}$ is a sample (a state), and $p^{(i)}$ are non-negative numerical factors called importance factors, which sum up to one. As the name suggests, the importance factors determine the weight (=importance) of each sample.

In global mobile robot localization, the initial belief is a set of poses drawn according to a uniform distribution over the robot's universe, annotated by the uniform importance factor 1/m. For the sake of simplicity, we assume the initial position is known. It is the same pose (x co-ordinate, y co-ordinate and heading direction) we assign our robot motion model to start from. We call it $X_0 = [x_0, y_0, \theta_0]$.

The recursive update is realized in three steps, computing the expression in (2.3), from *the right to the left*,

1. Sample a state x_{t-1} from $Bel(x_{t-1})$, by drawing a random $x_{t-1}^{(i)}$ from the sample set representing $Bel(x_{t-1})$ according to the (discrete) distribution defined through the importance factors $p_{t-1}^{(i)}$.

2. Use the sample $x_{t-1}^{(i)}$ and the action u_{t-1} to sample $x_t^{(j)}$ from the distribution $p(x_t \mid x_{t-1}, u_{t-1})$. The predictive density of $x_t^{(j)}$ is now given by the $p(x_t \mid u_{t-1}, x_{t-1})\ Bel(x_{t-1})$.

3. Finally, weigh the sample $x_t^{(j)}$ by the (non-normalized) importance factor $p\left(z_t \,\middle|\, x_t^{(j)}\right)$, the likelihood of the sample $x_t^{(j)}$, given the measurement z_t.

After the generation of 'm' samples, the new importance factors are normalized so that they sum up to 1 (hence define a probability distribution). The reader should quickly see that this procedure in fact implements Eq (2.3) using an (approximate) sample based representation.

The various versions of particle filters proposed in the literature can be regarded as special cases of this general Sequential Importance Sampling algorithm. These special cases can be derived from the SIS algorithm by an appropriate choice of importance sampling density and/or modification of the resampling step.

Below are the three particle filters proposed in the literature and we know these are all derived from the SIS algorithm.

a) sampling importance resampling (SIR) filter;
b) auxiliary sampling importance resampling (ASIR) filter;
c) regularized particle filter (RPF).

Particularly in this thesis, we are interested in the first type, 'sampling importance resampling (SIR) filter.

Sampling Importance Resampling Filter: The SIR filter proposed in [38] is a Monte Carlo method that can be applied to recursive Bayesian filtering problems. The assumptions required to use the SIR filter are not very binding.

The weights given by the proportionality in the equation above are normalized before the resampling stage. As the importance sampling density for the SIR filter is independent of measurement, the state space is explored without any knowledge of the observations. However, the SIR method does have the advantage that the importance weights are easily evaluated and that the importance density can be easily sampled.

Obviously, our algorithm constitutes just this one possible implementation of the particle filtering idea: other sampling schemes exist that further reduce variance. Further below, it will be convenient to notice that in this version of particle filter, the proposal distribution for approximating Bel(x) via importance sampling is given by,

$$q := p(x_t \mid x_{t-1}, u_{t-1})\, Bel(x_{t-1}) \tag{2.4}$$

which is used to approximate the desired posterior,

$$\frac{p(z_t \mid x_t)\, p(x_t \mid u_{t-1}, x_{t-1}) Bel(x_{t-1})}{p(z_t \mid d_{0....t-1}, u_{t-1})} \tag{2.5}$$

Consequently, the importance factors are given by the quotient

$$[p(x_t \mid x_{t-1}, u_{t-1})\, Bel(x_{t-1})]^{-1} \frac{p(z_t \mid x_t)\, p(x_t \mid u_{t-1}, x_{t-1}) Bel(x_{t-1})}{p(z_t \mid d_{0....t-1}, u_{t-1})}$$

thus the quotient, $q \propto p(z_t \mid x_t)$ (2.6)

To update the weights, $w_{k-1}^{(i)}$, we note that according importance sampling, the weights of the particles at time k should be as follows:

$$w_k^{(i)} \propto \frac{p\left(x_{0:t}^{(i)} \,\middle|\, z_{1:t}^{(i)}\right)}{q\left(x_{0:t}^{(i)} \,\middle|\, z_{1:t}^{(i)}\right)}$$

We will not give the full derivation here, but it turns out that it is possible to express the above equation recursively in terms of $w_{k-1}^{(i)}$

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(z_t \mid x_t^{(i)})p\left(x_t^i \mid x_{t-1}^{(i)}\right)}{q\left(x_t^i \mid x_{0:t-1}^{(i)}, z_{1:k}\right)}$$

To put it simply, in the resample step, a new set of particles is chosen so that each particle survives in proportion to its weight. The weighted cloud of particles turns into the somewhat more condensed and smoother cloud of unweighted particles. Highly unlikely particles at the fringe are not chosen, and the highly likely particles near the center of the cloud are replicated so that the high-probability region has a high density, correctly representing p(x), our posterior distribution.

Refer to the MATLAB code attached in Appendix to see how this algorithm was realized.

4.4 Our adaptations

We assume that the observations, $z_t()$ that are required as per our Particle Filtering technique are obtained from a sonar range scanner mounted on the robot and that each observation z_t consists of a set of range measurements.

To evaluate the likelihood $p(z_t \mid x_t)$ of an observation z_t given the pose x_t of the robot and a reference map m, we use the likelihood fields model. In this model, the individual range measurements of the observation z_t are assumed to be independent of each other and the likelihood of each one is computed based on the distance between the endpoint of the range measurement and its closest wall in the map m. We use ray-tracing to determine the actual distance of the micro-robot from the nearest wall in the geometric map. First, we verify that a point actually is inside a map. We do this by diving the X-Y plane into 4 zones at θ= 90, 180, 270, 360. We trace the line of θ=0 until it hits a wall in the map. This process repeats for all the 4 sides described above. Only if a particle's rays hit all the 4 sides, it is considered within the map and thus a valid one. Otherwise, it is discarded in the initial distribution step. This is because we assumed that the map is a closed tube structure.

Once the range value is calculated for every sensor using ray-tracing for the actual robot, we believe this value to be the 'true value' denoted by the range vector $z_t()$.

$$z_t() = (z_1,\ z_2,\ z_3,\ z_4,\ z_5)$$

Where $z_1,\ z_2,$ etc., are the range values for the respective sensors.

The range that pertains to the particles is denoted by $\overline{z_t}$. We then assign weight to each particle by its closeness to the micro-robot. That is, by the closeness of its sensor readings $\overline{z_t}$ to that of the actual robot's.(z_t) The weights are correspondingly assigned following the SIR algorithm, which is to say, that the particles with 'lesser' deviation in their range vector $\overline{z_t}$ from that of the actual values will receive higher weights than others. The weights are distributed over all the particles according to a Gaussian distribution.

Subsequent motion commands, u_t move the robot in the map in a given path resulting in the particles to follow suit. The particle cloud finally converges to the actual pose of the robot is a few steps. The performance metrics and results are discussed in Chapter 5.

Chapter 5

Simulations & Results

This chapter presents several experiments performed on our Particle Filter in order to validate the different dynamic situations that might arise while operating in the colon environment. As mentioned earlier in Chapter 4, in our approach, we model various operating conditions on deterministically changing maps. That is, localization is done not on one single map but over a series of maps which vary with time. (t_1, t_2, etc.)The experiments have been conducted in the MATLAB environment.

In a non-dynamic environment, at time t = t_1, t_2...t_6 the map that the mobile robot uses to localize itself remains the same. Whereas in a dynamic environment like the colon, a consistent change in the map shape is an inherent problem. Since the beginning of our methodology, we have assumed that there could be a way in the future by which we can pre-determine the exact map at every time instant. In other words, we believe we will be equipped to know at what time instant the map change occurs.

For this implementation, we have used two similar maps with change in only a certain region to make it more noticeable. As in the case of a real life situation of colon expansion, the map begins with being in a steady shape at time t_1 but it bulges out as it reaches the time instant t_5.

Here is the map at the time instants t_1 and t_5 as described in Figure 5.1:

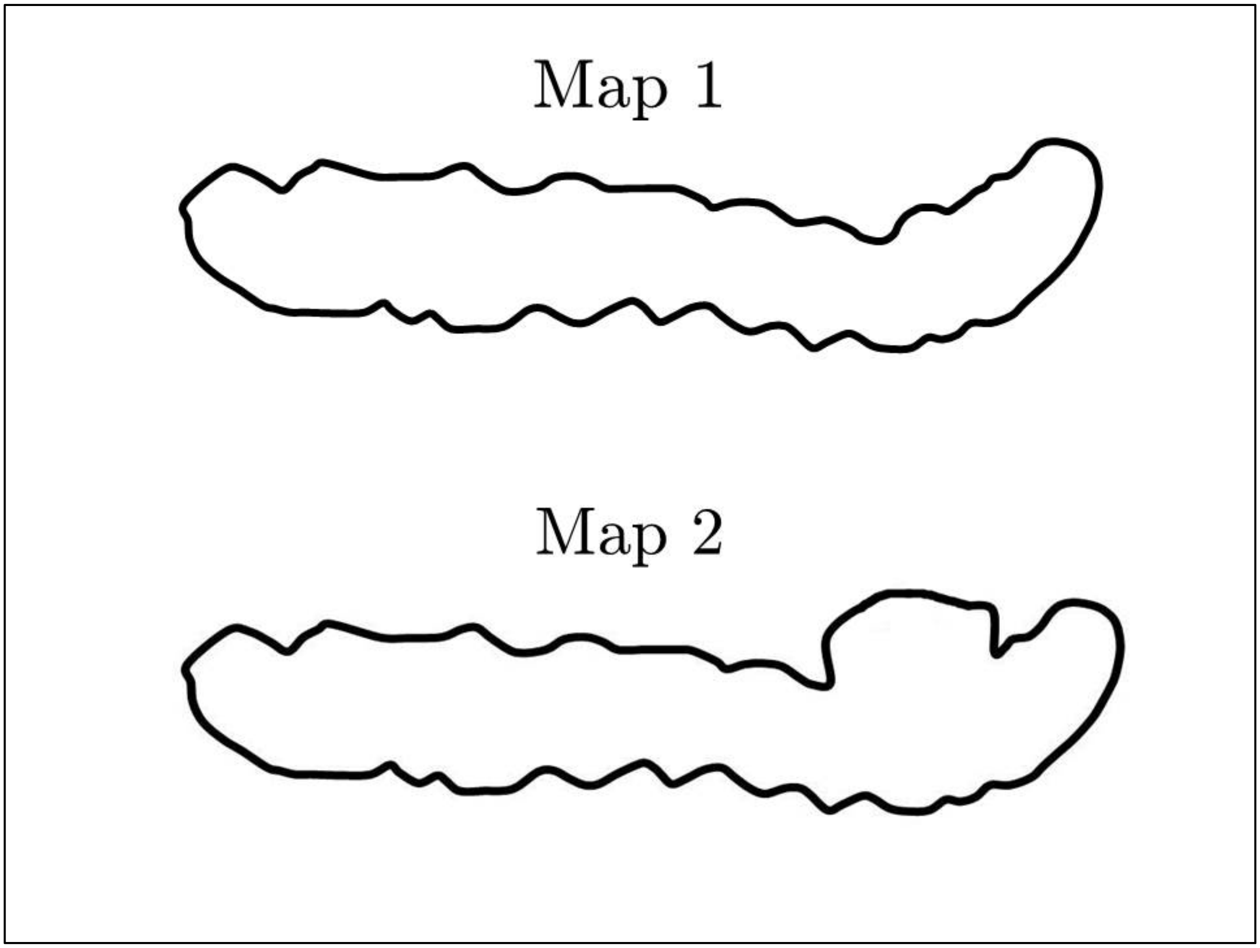
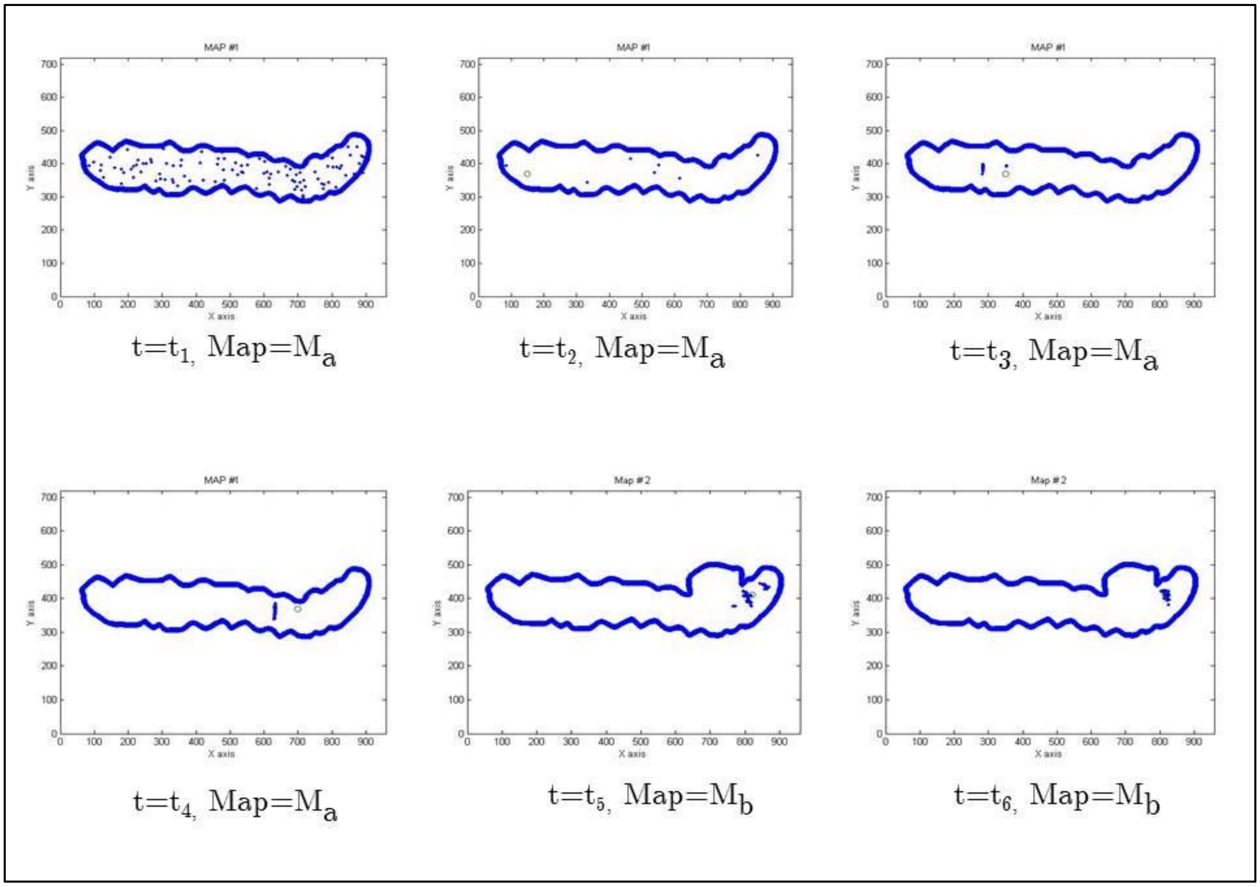


Figure 5.1 Outline of the colon maps used, M1 and M2.

We begin our Particle Filter execution by distributing the particles (samples) throughout the map uniformly. Each particle as discussed in previous chapters represents a valid robot pose, which in Estimation Theory is called a 'state'. The colon map here is a 2D outline and hence we operate in X-Y coordinate system. The robot pose contains information about the position of the robot i.e. its (x, y) location and its heading direction with respect to horizontal (x) axis.

The particle filter ran smoothly and the particles were sampled and resampled in cycles finally resulting in all of them converging very close to the actual robot pose. The actual robot pose is (820, 410, 1.5708) in the X-Y coordinate system.

A series of images in Figure 5.2 demonstrates one of our trails where the map changes from M_a to M_b, yet the filter successfully estimates the robot pose with good precision.



$t=t_1$, Map=M_a $t=t_2$, Map=M_a $t=t_3$, Map=M_a

$t=t_4$, Map=M_a $t=t_5$, Map=M_b $t=t_6$, Map=M_b

Figure 5.2 Time line of Particle Filter localization in a colon map with N=100 particles

Quantitative results of the above implementation are presented in Table 4.3. These results are obtained after simulating several times exactly the same setup (for instance atleast 10 times), and then computing the mean values of all simulations.

No.of particles	Error in x	Error in y	Error in θ
20	5.1440 %	3.5810 %	16.8173 %
100	1.20841 %	1.89962 %	7.7777 %
500	0.8027 %	1.5505 %	4.9009 %
1000	0.6113 %	1.1954 %	3.6445 %
5000	0.8449 %	1.3645 %	3.9973 %

Table 5.1

Table 5.1 suggests that the performance of our algorithm was enhanced by increasing the number of particles as is the expected case in any typical PF implementation technique. However, as far as time consumption is concerned, it seems clear that PF is slightly computationally taxing because of the presence of higher number of particles requiring longer times at each step, mainly due to the time spent in computing the weights.

A performance comparison chart is depicted below in Figure 5.3 that outlines the impact of higher particle size on the accuracy of the final robot pose estimated.

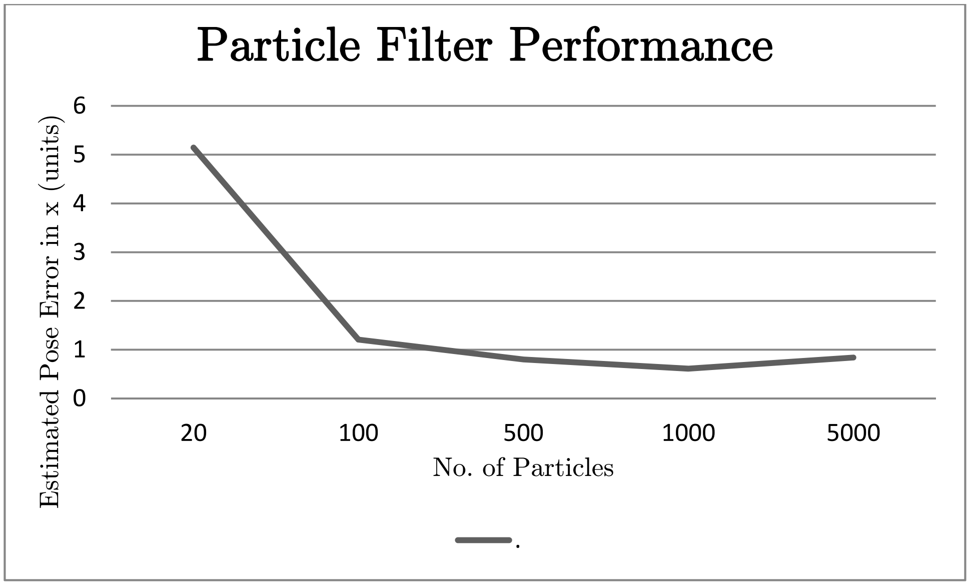


Figure 5.3 Performance comparison of our particle filter by varying the number of particles.

Additionally, we have characterized several interesting operating conditions that might arise while navigating inside a transverse colon into a set of cases which are presented below. Various worst-case scenarios, for example the change of maps in regions that we couldn't anticipate, were tested in the experiments

The experiments are classified into several cases, like Case1, Case2, etc., with each of them representing one unique scenario while operating under our assumed conditions inside the colon. Uniformly, each case assumes a particle set size of 100 particles. The experimental cases presented in Table 5.2 helped us analyze the performance metrics of our Particle Filter in each situation with respect to that of its implementation with a static map.

Case 1:

When the map is static, i.e. there is no dynamism in the colon environment. This case stands as a reference and is identical to any typical Particle Filter implementation in static environments.

Case 2:

When the map changes, i.e. there is dynamism in the colon environment but our Particle Filter model still follows the static assumption.

Case 3:

When the map changes, i.e. there is dynamism in the colon environment and our Particle Filter model adapts itself to the map change. (assuming predictability of maps a priori)

Case 4:

When the map doesn't change, i.e. there is no dynamism in the colon environment but our Particle Filter model incorrectly makes a map change.

Case 5:
When the map changes, i.e. there is dynamism in the colon environment but it changes unpredictably as compared to the pattern our Particle Filter model has assumed, i.e., the PF model changes to a new map, but that map is incorrect.

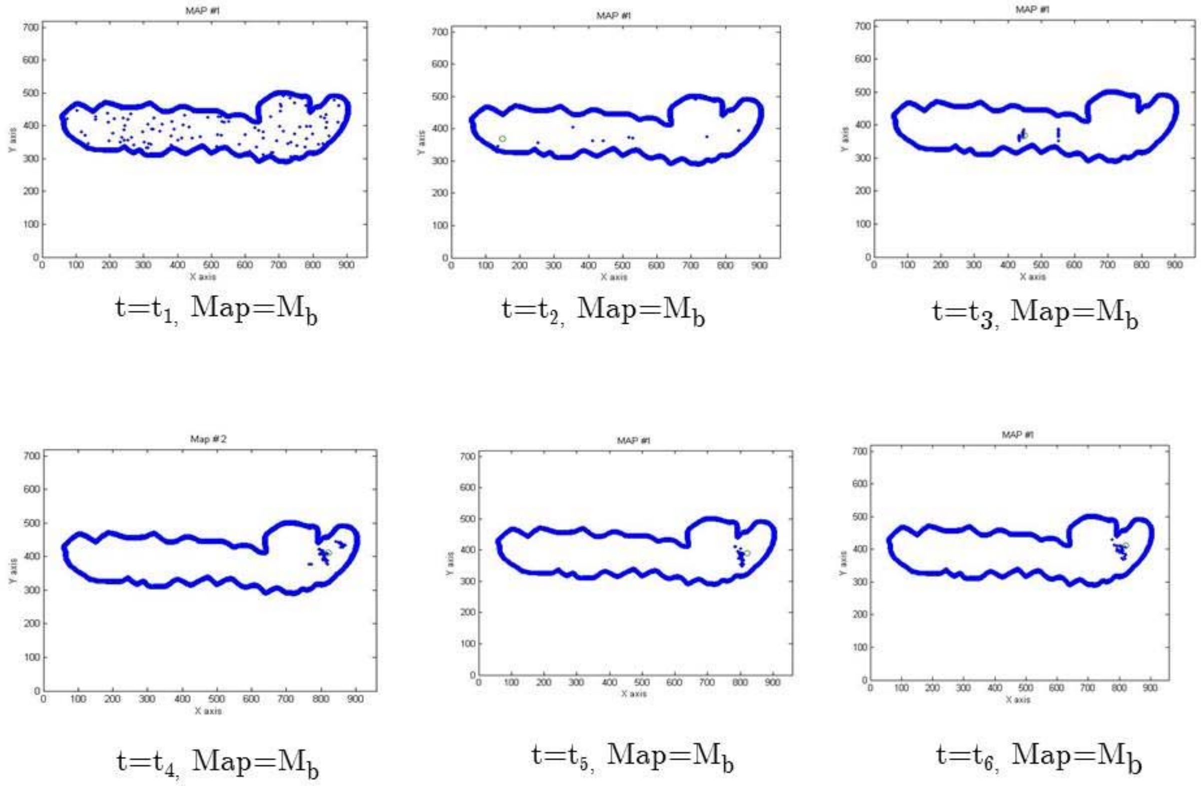
Table 5.2

Case 1:

As described in Table 5.2, the first scenario deals with a non-dynamic situation. Understandably, here the map won't change with time and will simply remain constant. This case stands as a reference and is identical to a typical Particle Filter implementation in static environments. [outdoor or indoor environments]

A series of images in Figure 5.4 demonstrates one of our Case-1 type trails where the map stays fixed at M_b. The filter with 100 particles successfully estimates the robot pose with great precision because of the conditions being highly static.

Case 1



$t=t_1$, Map=M_b $t=t_2$, Map=M_b $t=t_3$, Map=M_b

$t=t_4$, Map=M_b $t=t_5$, Map=M_b $t=t_6$, Map=M_b

Figure 5.4 Time line of Particle Filter localization in Case 1 with N=100 particles

The qualitative results estimate the error in robot pose to be minimal in this scenario. Infact, it is the least amount of error when compared to the rest of the cases and this can be attributed to a lack of perturbations in the environment.

Figure 5.5 is a bar graph that displays the robot pose error (x, y, θ) in percentage for our Particle Filter localization in Case 1.

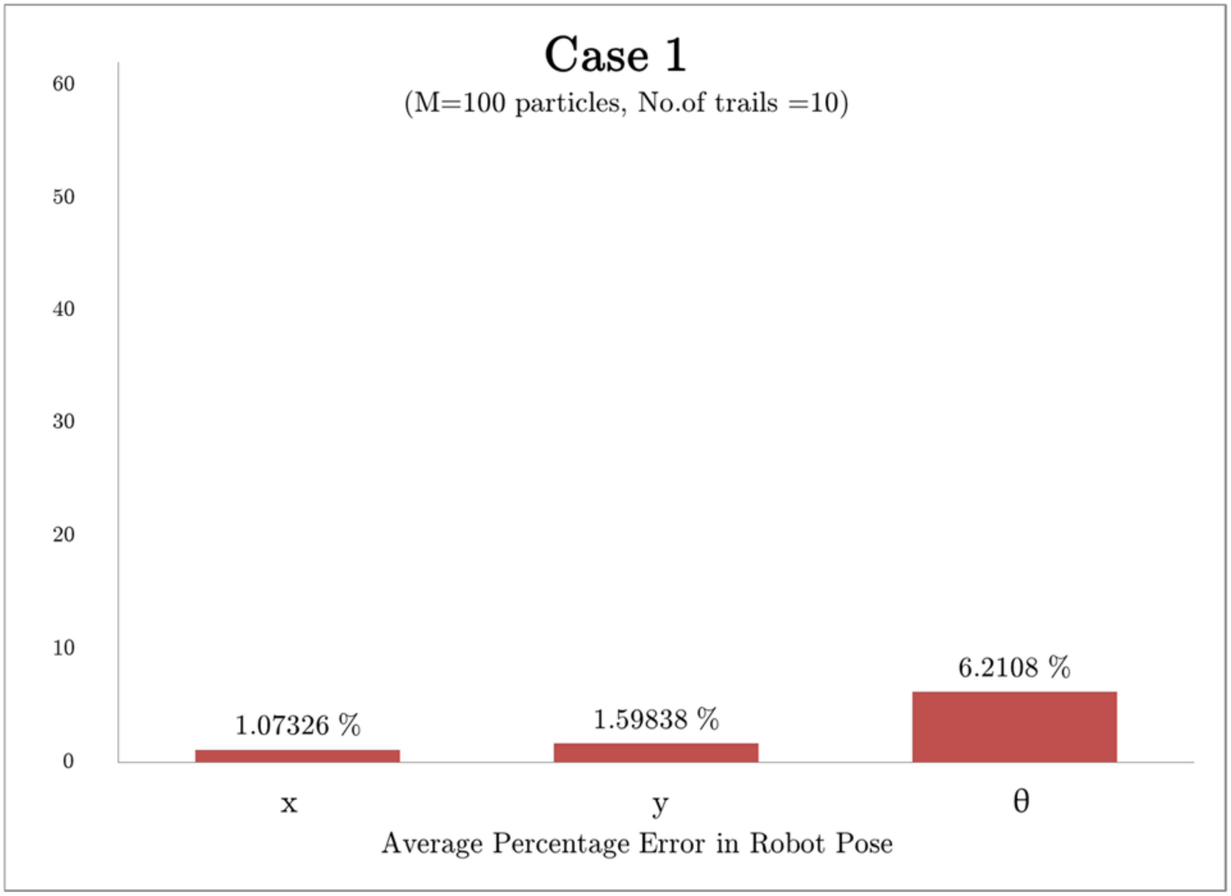


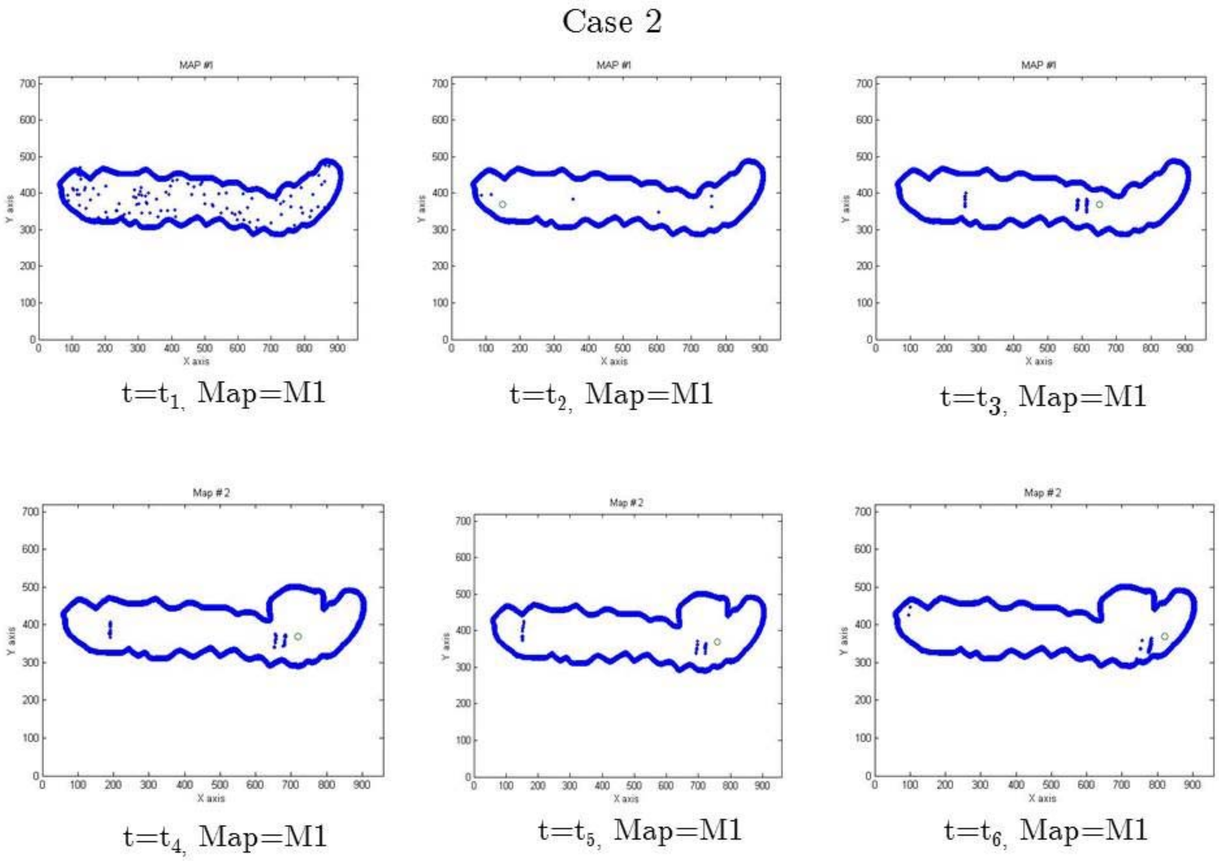
Figure 5.5 Mean of the error percentage of robot pose in Case 1 (with atleast 10 trials)

Case 2:

As described in Table 5.2, the second scenario deals with a dynamic situation. Here, the colon environment undergoes a change in a certain region of the map. Thus the actual map of the colon changes from M_a to M_b between t_1 and t_6.

However, in this case, we want to demonstrate how the change in the map impacts the regular robot localization. Thus, our model in this case will not change its reference map making this a static particle filter implementation but with the environment becoming dynamic. Such a filter will quickly become erroneous and the final estimation will be affected severely. This was reflected exactly in the robot pose error calculations in Figure 5.6.

A series of images in Figure 5.6 demonstrates one of our Case-2 type trails where the map changes from M_a to become M_b at time instant t_5. Our filter with 100 particles estimates the robot pose with substantially reduced precision because of the conditions being highly dynamic.



Case 2

$t=t_{1,}$ Map=M1

$t=t_{2,}$ Map=M1

$t=t_{3,}$ Map=M1

$t=t_{4,}$ Map=M1

$t=t_{5,}$ Map=M1

$t=t_{6,}$ Map=M1

Figure 5.6 Time line of Particle Filter localization in Case 2 with N=100 particles

The qualitative results estimate the error in robot pose to be quite high in this scenario. Infact, it is the highest amount of error when compared to the rest of the cases and this can be attributed to the induced dynamism in the environment and to the fact that our model hasn't adapted itself to this change in the map.

Figure 5.7 is a bar graph that displays the robot pose error (x, y, θ) in percentage for the micro-robot localization using our Particle Filter in Case 2.

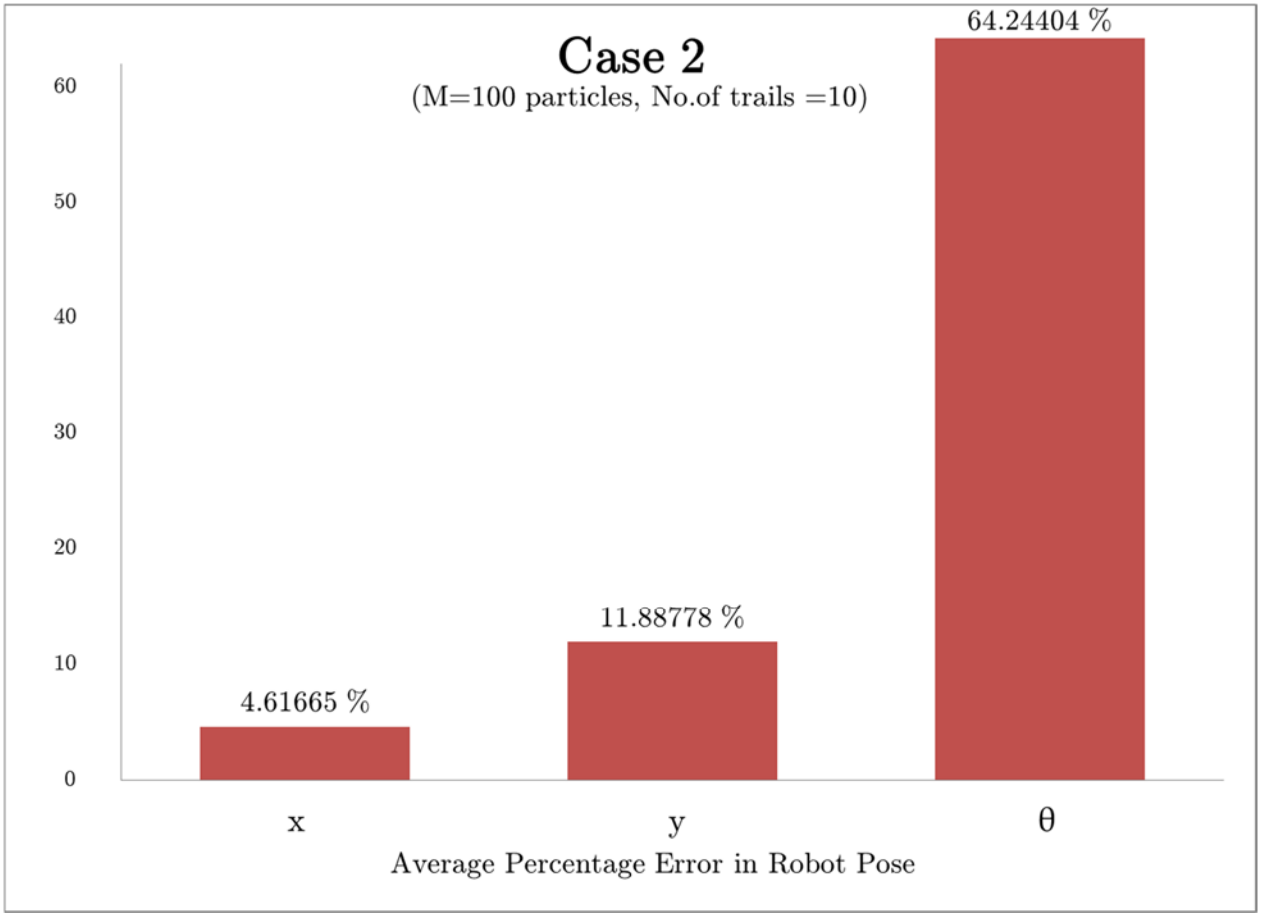


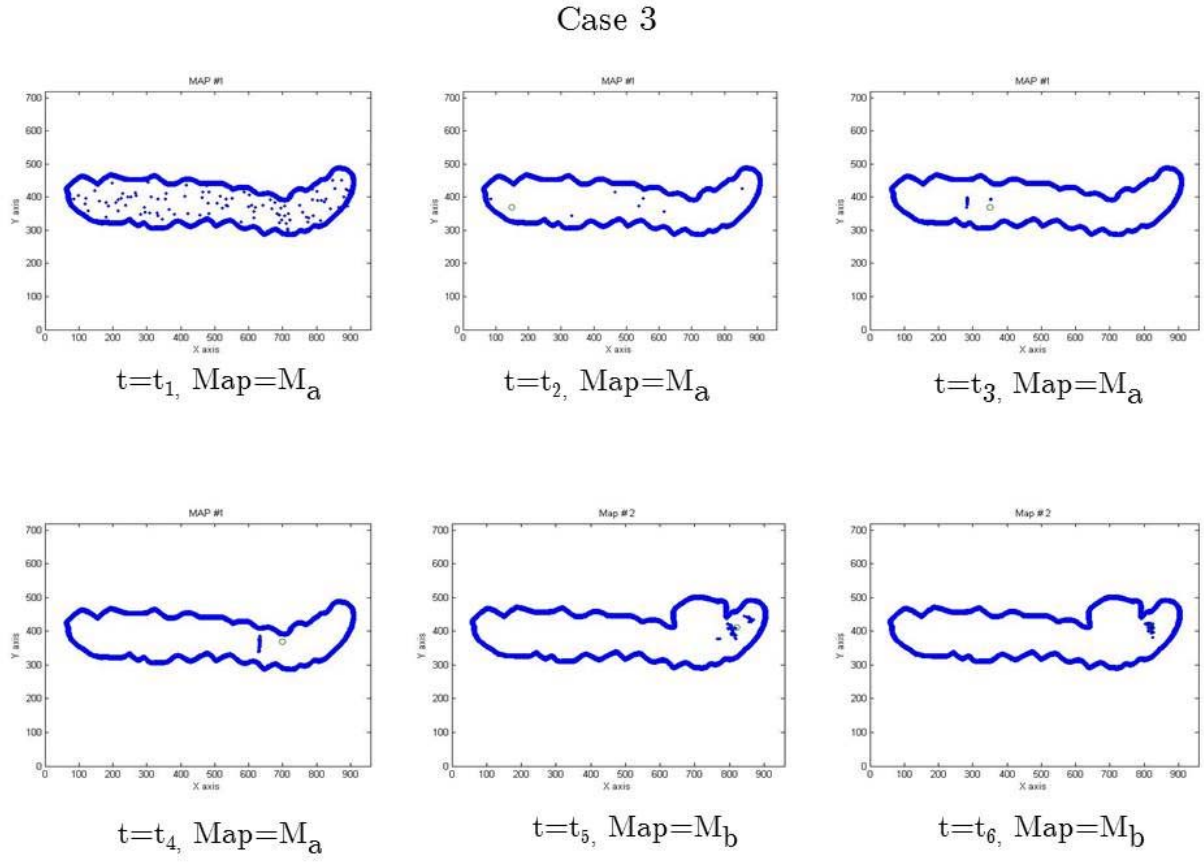
Figure 5.7 Mean of the error percentage of robot pose in Case 2 (with atleast 10 trials)

Case 3:

As described in Table 5.2, the third case is our major experiment since it outlines the scenario for which our Particle Filter was originally conceived for. Case 3 handles the dynamic environment by making the model adapt itself when the colon undergoes a change in a certain region by updating the map to the new map. The actual map of the colon changes from M_a to M_b at time instant t_5 and our model updates this change in the algorithm.

Our Particle filter performed impressively in this bringing the error due to dynamism to very low values.

A series of images in Figure 5.8 demonstrates one of our Case-3 trails where the map changes from M_a to become M_b at time instant t_5 and our filter updates its reference map. Our filter with 100 particles estimates the robot pose with substantially improved precision as compared to the previous case.


Case 3

$t=t_{1,}$ Map=M_a $t=t_{2,}$ Map=M_a $t=t_{3,}$ Map=M_a

$t=t_{4,}$ Map=M_a $t=t_{5,}$ Map=M_b $t=t_{6,}$ Map=M_b

Figure 5.8 Time line of Particle Filter localization in Case 3 with N=100 particles

The qualitative results for this case estimate the error in robot pose to be comparable to that of Case 1, i.e., the ideal conditions. Infact, it is the least amount of error when compared to the rest of the dynamic cases and this can be attributed to the how our model has updated its algorithm and used the most current map.

Figure 5.9 is a bar graph that displays the robot pose error (x, y, θ) in percentage for the micro-robot localization using our Particle Filter in Case 3.

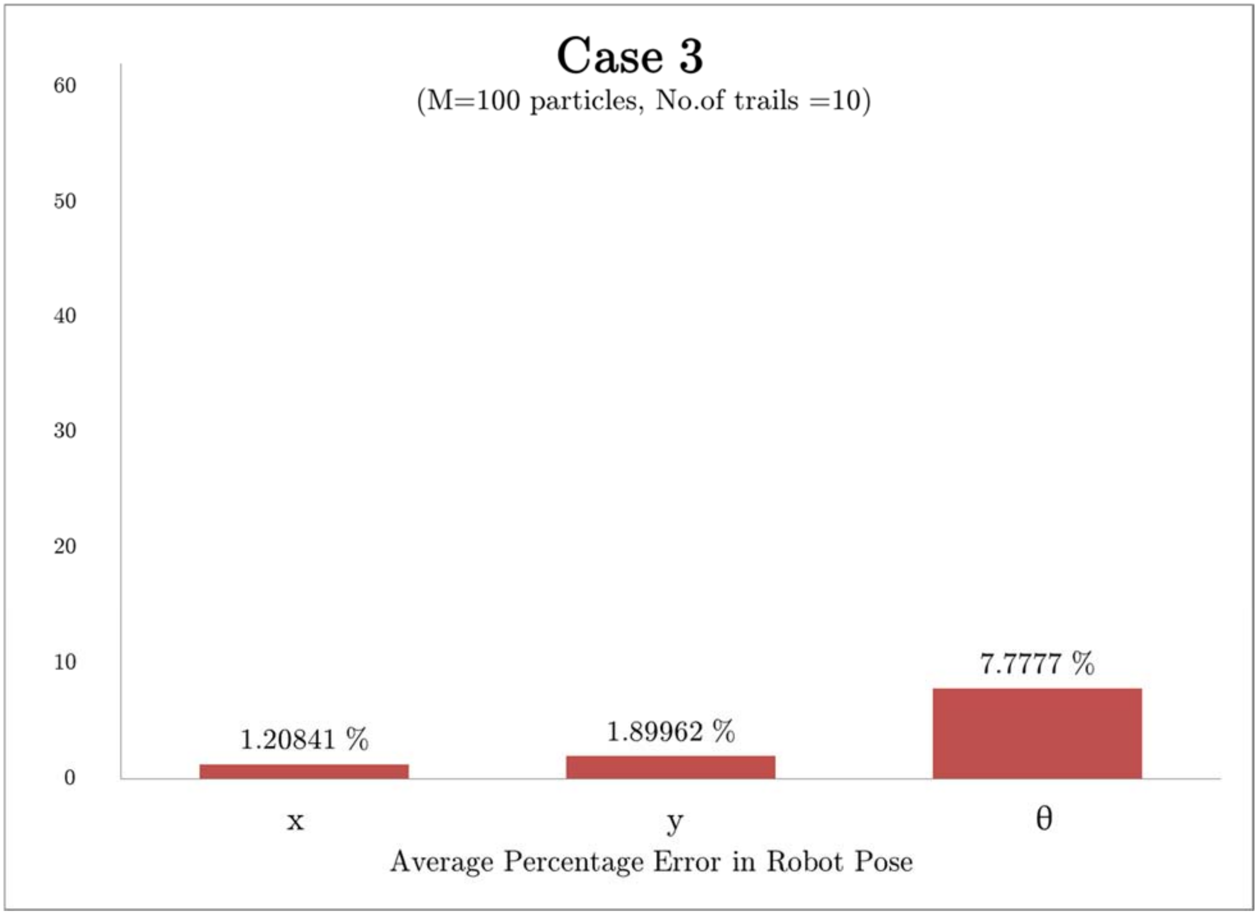


Figure 5.9 Mean of the error percentage of robot pose in Case 3 (with atleast 10 trials)

Case 4:

This is the scenario where our PF algorithm incorrectly assumes a map change but in reality the map hasn't changed. So, the expectation was to check if localization would be erroneous in such a situation, and if it did, how bad was the error.

We have run 10 trials with 100 particles implementing this scenario by changing the base map which our model uses to update from M_a to M_b at time instant t5. However, it is to be noted that in this trial, in reality, the base map of the colon environment will not change as depicted in Figure 5.10. So, this is an intentional experiment to estimate the error in a worst-case scenario.

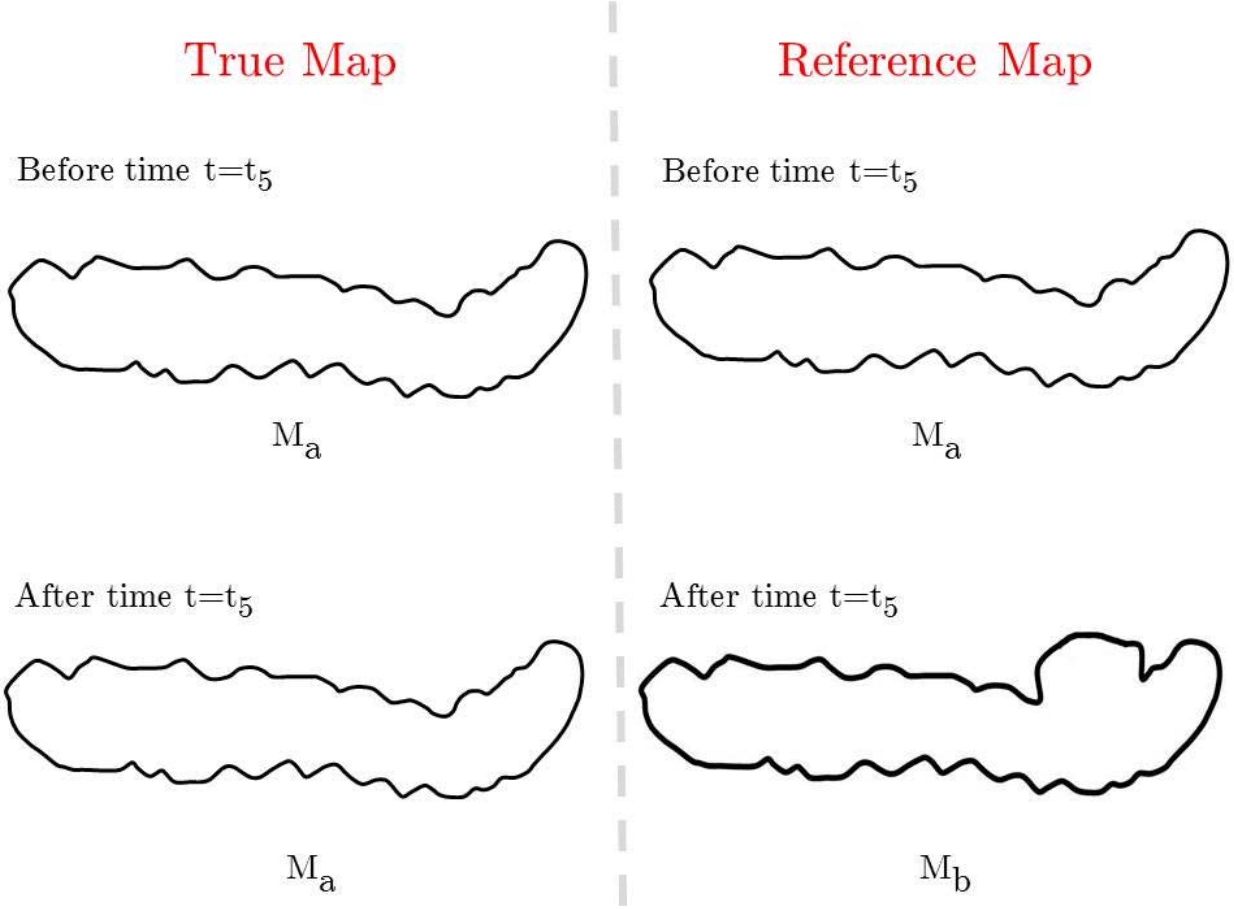
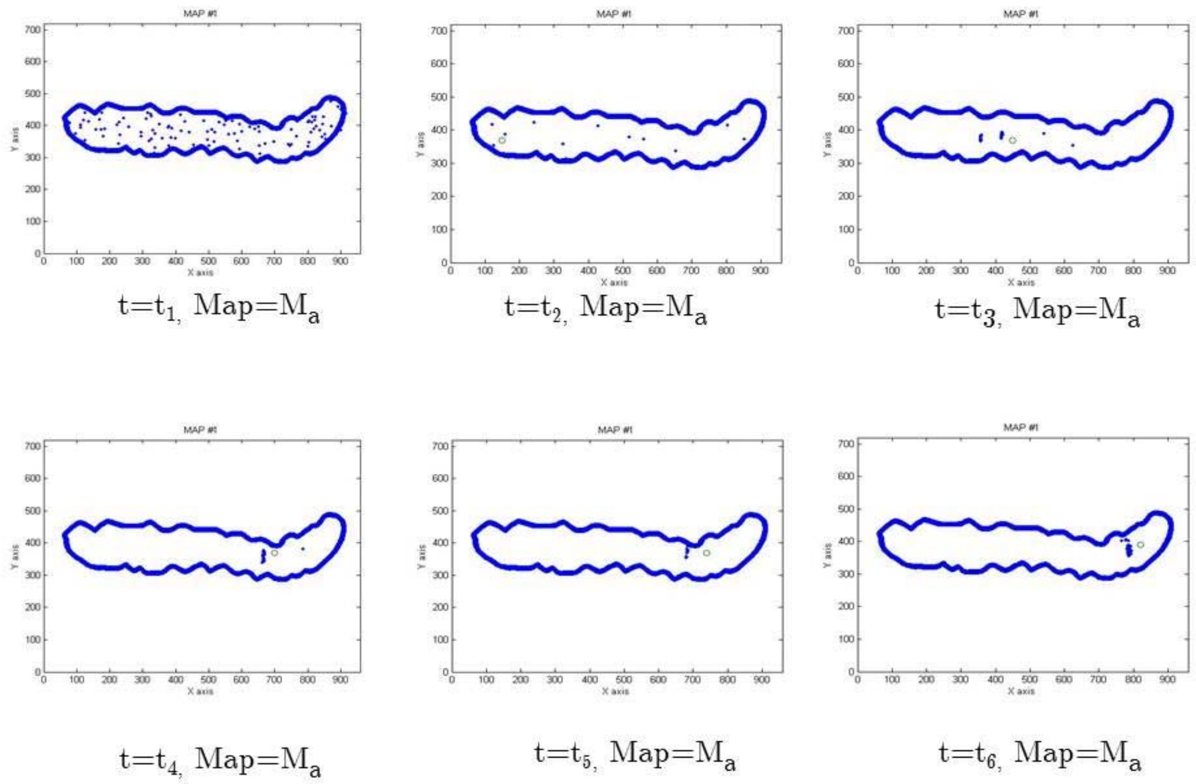


Figure 5.10 Comparison of true map and reference maps at the time instant t_5 in Case 4

The execution was quite challenging as the program would crash multiple times because of lack of high weighted particles. After several unsuccessful trials, we could acquire some performance data.

Our Particle filter performed very poorly as was expected because of the improper updation of the base map. A series of images in Figure 5.11 demonstrates one of our Case-4 trails where the map remains constant at time instant t_5 but our filter updates its reference map to M_b.

Case 4



$t=t_{1,}$ Map=M_a

$t=t_{2,}$ Map=M_a

$t=t_{3,}$ Map=M_a

$t=t_{4,}$ Map=M_a

$t=t_{5,}$ Map=M_a

$t=t_{6,}$ Map=M_a

Figure 5.11 Time line of Particle Filter localization in Case 4 with N=100 particles

The qualitative results for this case estimate the error in robot pose to be pretty high compared to all the cases. However, this is intuitive and can be attributed to the fact that we have updated our model to change the base map to M_b while in reality the base map hasn't changed.

Figure 5.12 is a bar graph that displays the robot pose error (x, y, θ) in percentage for the micro-robot localization using our Particle Filter in Case 4.

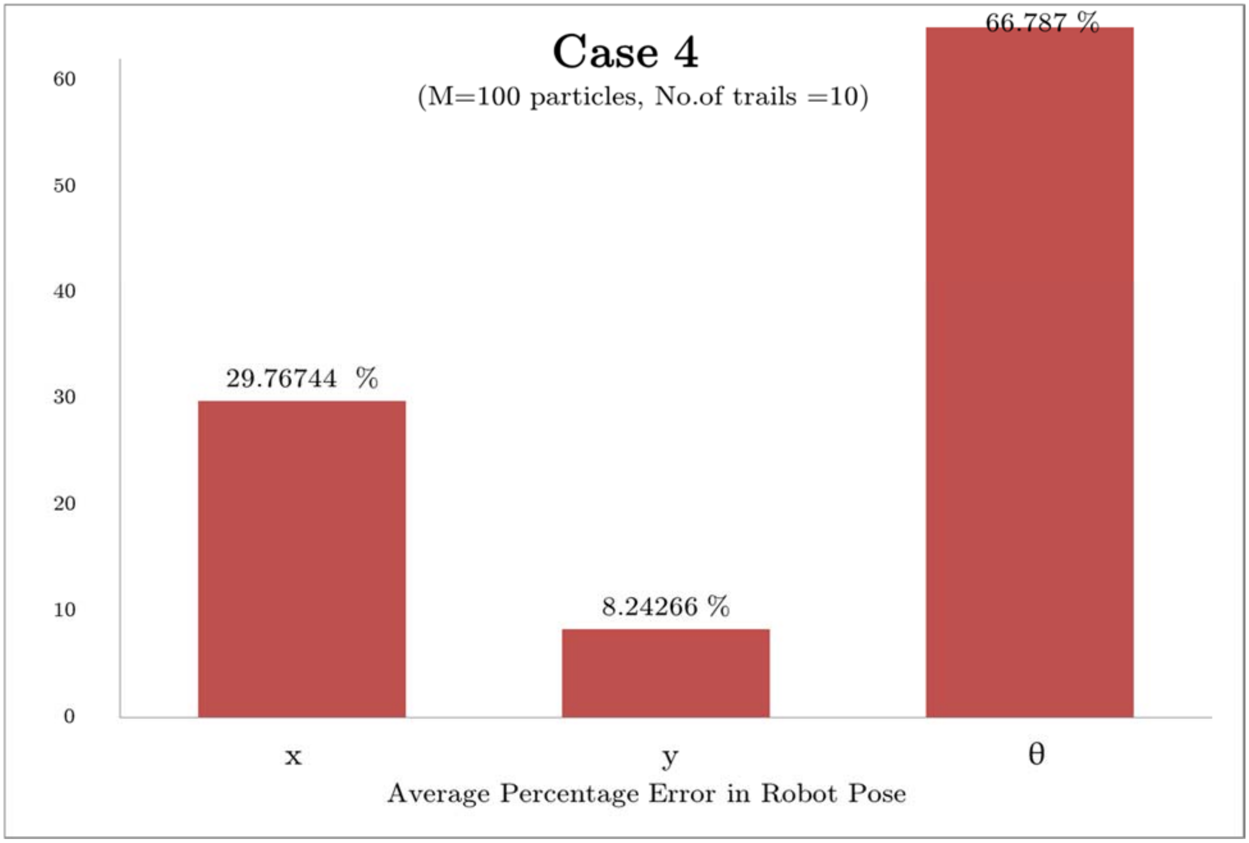


Figure 5.12 Mean of the error percentage of robot pose in Case 4 (with atleast 10 trials)

Case 5:

This is the scenario where our PF algorithm updates the base map to a certain new map whereas in reality the original map has changed to a completely unpredicted map.

This scenario challenges our assumption as the map changes in a manner that was not determined a priori. Expectedly, the error in the robot pose estimation is quite huge and this proves that in this adaptation of standard particle filter technique, the knowledge of map sequence is vital for successful localization.

Map changes in Case 5 are depicted in Figure 5.13. We have run 10 trials with 100 particles implementing this scenario by changing the base map M_a to M_c at time instant t_5. At the same time instant, we made our model update itself to map M_b.

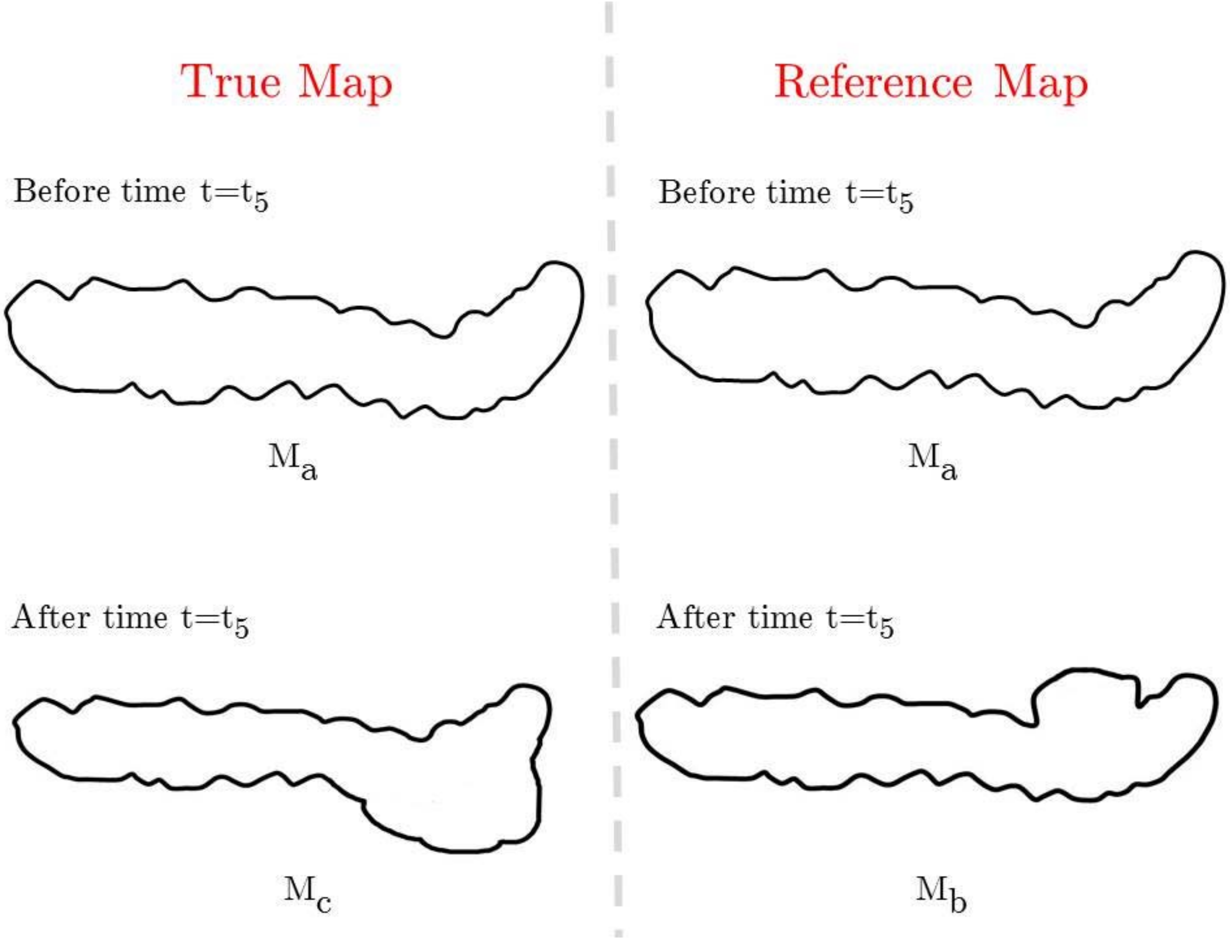
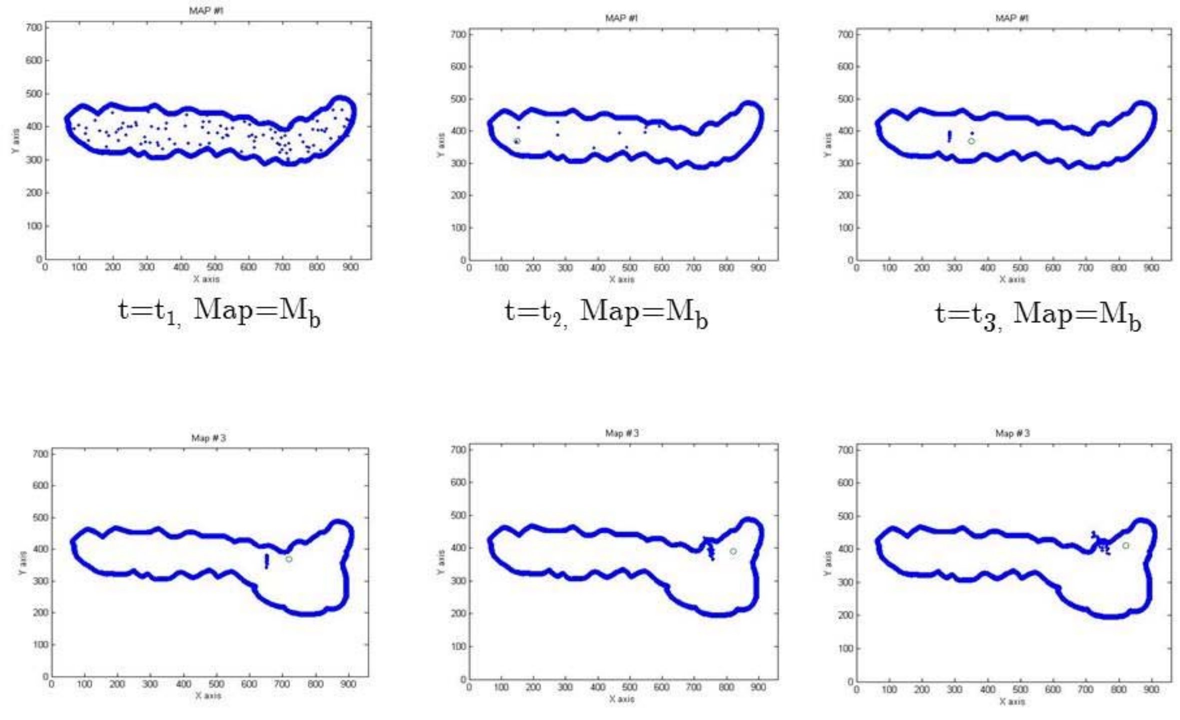


Figure 5.13 Comparison of true map and reference maps at the time instant t_5 in Case 5

The experiment was to check if localization would be erroneous, and if it did, how bad was the error. The execution was quite challenging as the program would crash multiple times because of lack of enough high weighted particles. After several trials, we could acquire some performance data.

Our Particle filter performed poorly as was expected because of the improper updation of the base map. A series of images in Figure 5.14 demonstrates one of our Case-5 trials.

Case 5



$t=t_{1,}$ Map=M_b $t=t_{2,}$ Map=M_b $t=t_{3,}$ Map=M_b

$t=t_{4,}$ Map=M_c $t=t_{5,}$ Map=M_c $t=t_{6,}$ Map=M_C

Figure 5.14 Time line of Particle Filter localization in Case 5 with N=100 particles

The qualitative results for this case estimate the error in robot pose to be pretty high than that of Case 3. After time t_5, our filter used a wrong map as reference for its sensor measurements and assignment of particle weights. Thus the pose estimation is bound to become erroneous in this case.

Figure 5.15 is a bar graph that displays the robot pose error (x, y, θ) in percentage for the micro-robot localization using our Particle Filter in Case 5.

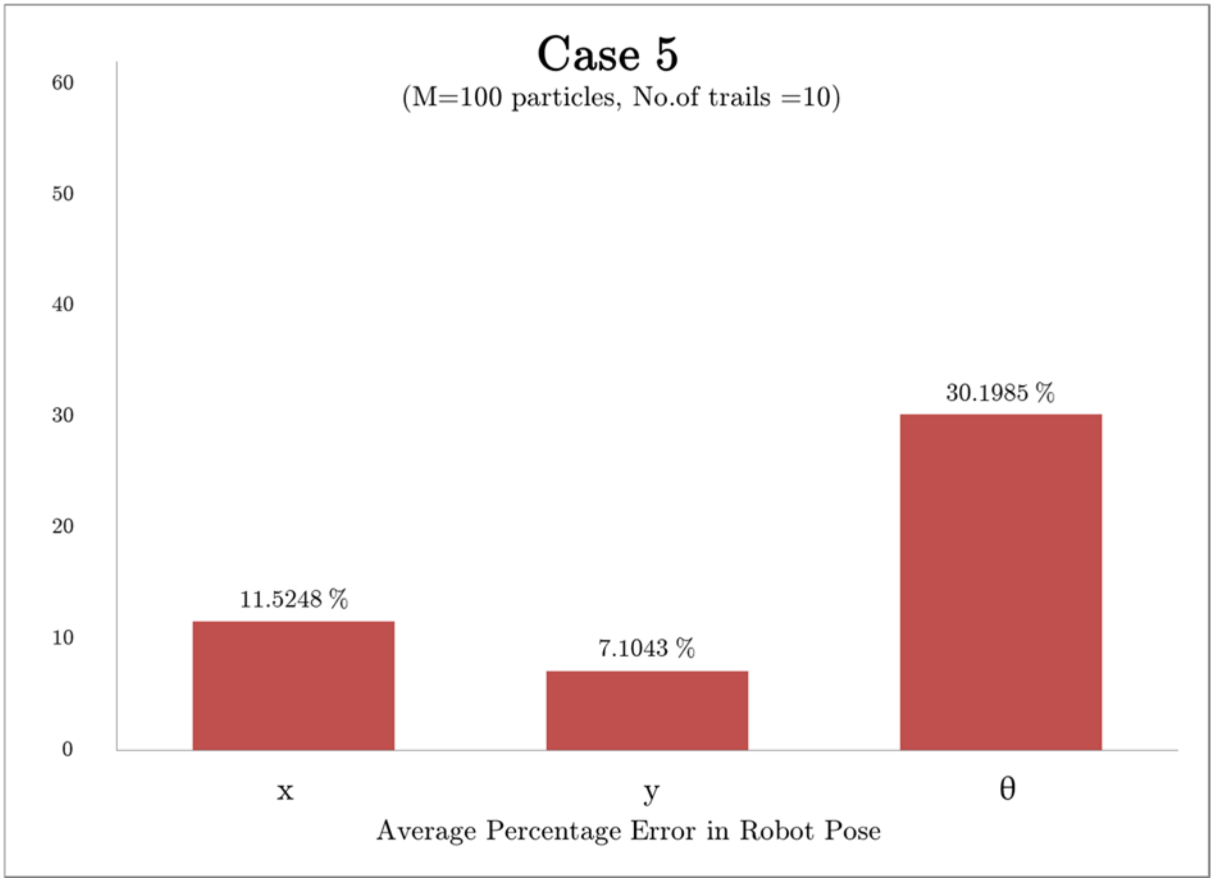


Figure 5.15 Mean of the error percentage of robot pose in Case 5(with atleast 10 trials)

We also performed a separate exercise to record the percentage increase in the error of the robot pose, i.e., (x, y and θ) with incremental changes in the area of the map.

Particle Filter localization is generally robust as long as the true map of the environment stays in its original shape. But, in the case of dynamic environments like that of human colon, it is quite common to encounter expansion/contraction patterns. Thus, the map area either increases relative to the initial value when the colon expands or decreases when the colon contracts.

We were interested in studying to what extent this increase/decrease of map area relative to the original map will affect the sensor readings and thereby the estimated pose.

In order to study that, we've modeled the right most end of the colon map as a circle and varied the radius to increase the local map area as illustrated in Figure 5.16.

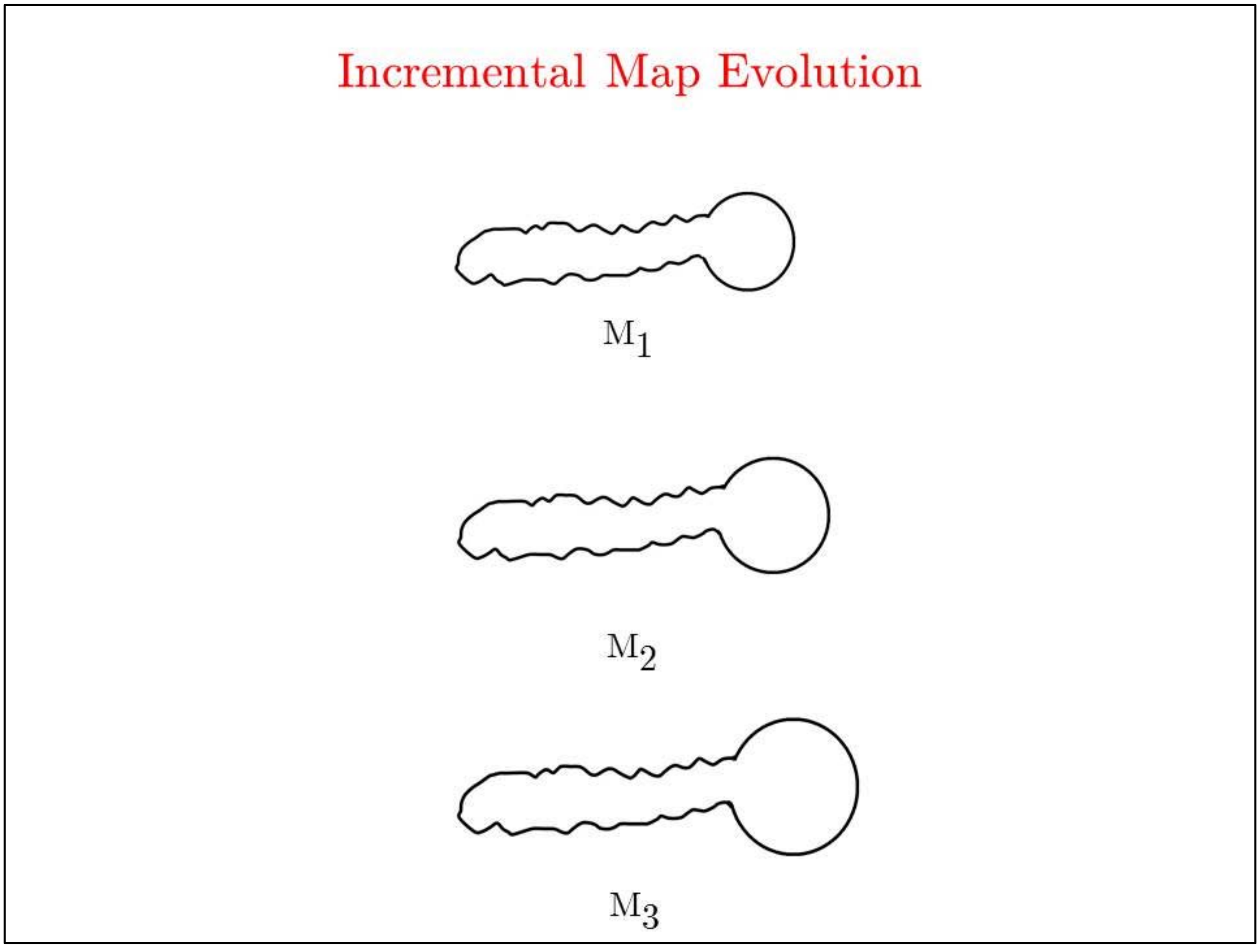


Figure 5.16 Sequence of maps M_1, M_2, M_3

There is no way to determine the enclosed area of the entire map as it is a completely uneven and unstructured outline. So, we have focused our interest in the increase/decrease of the local area of the circle and to measure its impact on the error.

In each of these trials, our Particle Filter model is referenced to a steady map M_1 and the map at time instant $t=t_5$ changes to a new map with a larger area circle like M_2. In the next trial, we retained the initial map M_1 just like before, yet at time t=t5, we made our model change its reference map to now an even larger area circle, M_3. This procedure was repeated over 10 trials and findings are summarized in Table 5.3. One can infer from the table that the larger the change in map, the higher the error.

Map	Area of the circle	Local Area	Error in x error	Error in y error	Error in θ error
M_1	38.4845	A1	0.1690	1.0182	4.1656
M_2	50.2655	A1 + 30.6123 %	1.2715	1.3064	6.2652
M_3	63.6173	A1 + 63.6175 %	2.6470	3.5147	5.8579

Table 5.3 Impact of the magnitude of map change on the robot pose estimation error.

The qualitative and quantitative results presented so far analyzed various operating conditions that a micro-robot might face in a colon environment.

The results of the simulations were encouraging as the greatest error in Case 3 was under 1.8% for estimating the location of the robot (x,y) and under 7.8% while estimating its heading direction θ. Case 3 is the hallmark scenario of our research where we could successfully estimate the robot pose despite the map changes, partly because we could anticipate them ahead of time and adapt our filter. The slightly higher percentage error in θ can be attributed to the non-uniform nature of the walls that the sensors are facing. The resolution of the map could also have contributed to the overall error, so availability of high resolution map would help significantly. The filter performance can also be improved by increasing the particle set size but again slight caution has to be exercised as Particle Filter generally loses its robustness after a certain size of particle set.

Chapter 6

Conclusion

The fields of bio-robotics and capsule robotic endoscopy are increasingly discovering the potential of micro-robots often the size of a pill or even smaller which are currently able to navigate inside the human body especially in the gastro-intestinal tract acting as powerful diagnostic tools. In the future, it is anticipated that these micro-robots will be equipped with AI and enhanced sensory capacities that will enable them to perform unprecedented surgical and diagnostic tasks.

This thesis was aimed at exploring the possibility of applying a standard robot navigation technique widely popular in the robotics literature called Particle Filter Localization based on Bayesian probabilistic state estimation. Particle Filter localization also called as Monte Carlo Localization (MCL) was performed in such tubular intestinal environment in this research where there is a new-kind of dynamism that hasn't been studied so rigorously so far.

In order to demonstrate its implementation potential in such an environment, the Monte Carlo Localization filter was built in a Matlab environment and several simulations were performed under a few assumed conditions. The dynamic transverse colon environment was partially modelled by a map sequence that enables our localization algorithm to adapt itself to the most recent maps at different time instants. The key assumptions underlining this research are the predictability of the map change and the availability of colon maps a priori. A series of maps represent the change in the structure of the colon and our model adapts to these changes by updating its reference map at particular time instants.

The performance of our Particle Filter in various operating conditions was analysed and tabulated in Chapter 5. The robot pose, i.e., its (X, Y) location along with its heading direction θ were estimated using the filter since this is a 2-D implementation. The filter performed appreciatively in some dynamic cases where the mean error in robot pose was 1.208%, 1.899%, 7.777% in x, y and θ values respectively.

Table 5.1 discusses the improvement of the performance of our Particle Filter as we increase the number of particles sampled. The best performance was observed for implementations with neither too little nor too many particles. This is a common trend even in general Particle Filter implementation in outdoor/indoor environments owing to the fact that there should be atleast reasonable number of particles with weights to make the further resampling stages possible. Too less of a particle size will crash the model for the lack of high-weighted particles. Too high of a particle size will risk confining the belief to one weight value and the other weights could be dismissed resulting in a possible fatal error.

Later, a set of interesting experiments were conducted using our filter model and the underlining map. We ran the filter in non-dynamic conditions first to benchmark the observed error as a reference. Keeping this error in perspective, and knowing that these conditions are idealistic in a colon environment, we designed several intuitive experiments. Divided into a list of cases as depicted in Table 5.2, these experiments span the broad spectrum of possibilities of dynamism in the colon environment.

Case 1 was implementation of our Particle Filter in static, idealistic conditions and almost resembles the classic Particle Filter implementation in known environments. The mean error in robot pose we observed was expectedly the least. Case 2 dealt with a scenario when the map undergoes change in a certain region, emulating a bulge-out of a real colon and our model doesn't notice it. Results showed that the filter performs very poorly in such a case because of the altered sensor readings coming off the walls. Table 6.1 presents the noticeable difference in the pose error percentages for Case 1 and Case 2.

Scenario	Error in x	Error in y	Error in θ
Case 1	1.0732 %	1.5973 %	6.2108 %
Case 2	4.6165 %	11.8877 %	64.244 %
Case 3	1.208%	1.89962 %	7.7777 %
Case 4	29.7674 %	8.2426 %	66.7870 %
Case 5	11.5448 %	7.1043 %	30.1985 %

Table 6.1 Performance comparison of multiple scenarios that might arise while estimating the pose of a micro-robot in a colon environment

Case 3 is the most interesting case for us because it presents the most relevant scenario for the implementation of our Particle Filter. It is about making necessary adaptations to the filter so that it handles the dynamism in the environment by updating the reference map to the newest map. The results reveal that this scenario has been impressively handled as the estimated pose error fell to much lower values compared to the Case 2 and almost comparable to the ideal situation of Case 1. Table 6.1 sheds light on the success of our approach.

Case 4 and Case 5 are worst-case scenarios and challenge some of our key assumptions. Particularly, Case 4 describes the situation where the map doesn't change to a new map but our model incorrectly updates itself to the newest map. Expectedly, since the reference map has changed, the sensor readings are all off by several points resulting in a huge error. Case 5 presents another interesting situation where the map change pattern in reality doesn't match with our predicted map sequence. The pose error is thus a function of the randomness of the new map but if we focus on varying the shape of a certain region only, the error still was significant. It is therefore predicted that the more random and incoherent the new maps are relative to the original map, the larger the error would be.

We close our experiments with a final case where we chose to study the impact of the magnitude of map change on the estimation of robot pose. To do this, we've modelled a part of the colon as a circle and kept varying its radius to change the local area incrementally. The trend revealed a positive correlation between the magnitudes of change in map to the estimated pose error.

Future Work:

This research discussed the possibility of applying Particle Filtering techniques in emerging environments of mobile robot application like that of the human transverse colon for diagnostic/surgical purposes. In this work, we described the current state of the art methodologies that are being used in robotic capsule endoscopy. However, we believe it is just a scratch on the surface of the endless potential that these emerging fields contain in them. Even from a solution point of view, there are more open questions to be answered. We assumed we have a 2-D map of the transverse colon. In the future, as the robotic mapping techniques extend themselves into the fields of capsule robot navigation; it will be beneficial to construct 3D maps with richer features. Additionally, further research could lead to estimate the feasibility of simultaneously localizing and mapping (performing SLAM) inside the human body with greater precision. SLAM techniques rely on dividing the environment into a set of landmarks that are distinguishable from each other, so it would be interesting to see if the camera technology mounted on the capsule/micro-robots can transmit such high resolution images. At this point, extensive research is going on in miniaturizing the mobile robots and enabling them to swim inside a human body by groups at Nano Robotics Lab (CMU-RI)[39] and STORM Lab at Vanderbilt University [40]. Moving beyond bio-robotics, it is possible to envision applications for the dynamic particle filters in rare and unusual environments with stretchable walls like tents/canopies. Further research may even be directed towards studying underwater situations which mimic this kind of elastic dynamism. On a whole, we hope robotics and AI becomes seamless across new platforms like medical technology and will be able to attract newer approaches that can revolutionize healthcare and make our world increasingly healthier.

References

[1] Borenstein, Johann, Liqiang Feng, and H. R. Everett. Navigating mobile robots: systems and techniques. AK Peters, Ltd., 1996.

[2] Durrant-Whyte, Hugh, and Tim Bailey. ”Simultaneous localization and mapping: part I.” Robotics & Automation Magazine, IEEE 13, no. 2 (2006): 99-110.

[3] Cassandra, Anthony R., Leslie Pack Kaelbling, and James A. Kurien. ”Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation.” In Intelligent Robots and Systems’ 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on, vol. 2, pp. 963-972. IEEE, 1996.

[4] Thrun, Sebastian, Dieter Fox, Wolfram Burgard, and Frank Dellaert. ”Robust Monte Carlo localization for mobile robots.” Artificial intelligence 128, no. 1 (2001): 99-141.

[5] Nourbakhsh, Illah, Rob Powers, and Stan Birchfield. ”DERVISH an office-navigating robot.” AI magazine 16, no. 2 (1995): 53.

[6] Burgard, Wolfram, Dieter Fox, Daniel Hennig, and Timo Schmidt. ”Estimating the absolute position of a mobile robot using position probability grids.” In Proceedings of the national conference on artificial intelligence, pp. 896-901. 1996.

[7] Koenig, Sven, and Reid G. Simmons. ”Passive distance learning for robot navigation.” In ICML, pp. 266-274. 1996.

[8] Thrun, Sebastian. ”Particle filters in robotics.” In Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence, pp. 511-518. Morgan Kaufmann Publishers Inc., 2002.

[9] Montemerlo, Michael, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. ”FastSLAM: A factored solution to the simultaneous localization and mapping problem.” In AAAI/IAAI, pp. 593-598. 2002.

[10] Hahnel, Dirk, Dirk Schulz, and Wolfram Burgard. ”Map building with mobile robots in populated environments.” In Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on, vol. 1, pp. 496-501. IEEE, 2002.

[11] Wang, Chieh-Chih, and Chuck Thorpe. ”Simultaneous localization and mapping with detection and tracking of moving objects.” In Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on, vol. 3, pp. 2918-2924. IEEE, 2002.

[12] Montemerlo, Michael, Sebastian Thrun, and William Whittaker. ”Conditional particle filters for simultaneous mobile robot localization and people-tracking.” In Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on, vol. 1, pp. 695-701. IEEE, 2002.

[13] Stachniss, Cyrill, and Wolfram Burgard. ”Mobile robot mapping and localization in non-static environments.” In Proceedings of the National Conference on Artificial Intelligence, vol. 20, no. 3, p. 1324. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[14] Capsule Endoscopy. Retrieved from http://www.givenimaging.com/en-int/Innovative-Solutions/Capsule-Endoscopy/Pages/default.aspx

[15] Pillcam Colon. Retrieved from http://www.givenimaging.com/en-int/Innovative-Solutions/Capsule-Endoscopy/Pillcam-COLON/Pages/default.aspx

[16] Pillcam SB. Retrieved from http://www.givenimaging.com/en-us/Innovative-Solutions/Capsule-Endoscopy/Pillcam-SB/Pages/default.aspx

[17] Project Vector. Retrieved from http://www.vector-project.com/consortium/era.html

[18] Crowley, James L. ”World modeling and position estimation for a mobile robot using ultrasonic ranging.” In Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on, pp. 674-680. IEEE, 1989.

[19 Burgard, Wolfram, and Sebastian Thrun. ”Markov Localization for Mobile Robots in Dynamic Environments Dieter Fox dFox@ cs. cmu. edu Computer Science Department and Robotics Institute Carnegie Mellon University.” Journal of Artificial Intelligence Research 11 (1999): 391-427.

[20] Hahnel, Dirk, Dirk Schulz, and Wolfram Burgard. ”Map building with mobile robots in populated environments.” In Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on, vol. 1, pp. 496-501. IEEE, 2002.

[21] Montemerlo, Michael, Sebastian Thrun, and William Whittaker. ”Conditional particle filters for simultaneous mobile robot localization and people-tracking.” In Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on, vol. 1, pp. 695-701. IEEE, 2002.

[22] Wang, Chieh-Chih, and Chuck Thorpe. ”Simultaneous localization and mapping with detection and tracking of moving objects.” In Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on, vol. 3, pp. 2918-2924. IEEE, 2002.

[23] Hahnel, Dirk, Dirk Schulz, and Wolfram Burgard. ”Map building with mobile robots in populated environments.” In Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on, vol. 1, pp. 496-501. IEEE, 2002.

[24] Wolf, Denis, and Gaurav S. Sukhatme. ”Online simultaneous localization and mapping in dynamic environments.” In Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on, vol. 2, pp. 1301-1307. IEEE, 2004.

[25] Mitsou, Nikos C., and Costas S. Tzafestas. ”Temporal occupancy grid for mobile robot dynamic environment mapping.” In Control & Automation, 2007. MED’07. Mediterranean Conference on, pp. 1-8. IEEE, 2007.

[26] Martinez-Cantin, Ruben, and José A. Castellanos. ”Bounding uncertainty in EKF-SLAM: The robocentric local approach.” In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pp. 430-435. IEEE, 2006.

[27] Biber, Peter, and Tom Duckett. ”Dynamic Maps for Long-Term Operation of Mobile Service Robots.” In Robotics: science and systems, pp. 17-24. 2005.

[28] Williams, Stefan B., Gamini Dissanayake, and Hugh Durrant-Whyte. ”Efficient simultaneous localisation and mapping using local submaps.” In Proc. 2001 Australian Conf. on Robotics and Automation, Sydney, pp. 14-15. 2001.

[29] Basar, Md Rubel, F. Malek, Khairudi M. Juni, M. Shaharom Idris, and M. Iskandar M. Saleh. ”Ingestible Wireless Capsule Technology: A Review of Development and Future Indication.” International Journal of Antennas and Propagation 2012 (2012).

[30] Kim, Tae Song, Si Young Song, Han Jung, Jinseok Kim, and Eui-Sung Yoon. ”Micro capsule endoscope for gastro intestinal tract.” In Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, pp. 2823-2826. IEEE, 2007.

[31] Kassim, Irwan, Louis Phee, Wan S. Ng, Feng Gong, Paolo Dario, and Charles A. Mosse. ”Locomotion techniques for robotic colonoscopy.” Engineering in Medicine and Biology Magazine, IEEE 25, no. 3 (2006): 49-56.

[32] Harward Universsity Press Coverage. Retrieved from http://news.harvard.edu/gazette/story/2013/09/40-prevention-rate-for-colorectal-cancers/

[33] Colonic Disease Investigation by Robot Hydro-colonoscopy (CODIR) Retrieved from http://www.imsat.org/CODIR.htm.

[34] Obstein, Keith L., and Pietro Valdastri. ”Advanced endoscopic technologies for colorectal cancer screening.” World journal of gastroenterology: WJG 19, no. 4 (2013): 431.

[35] Transverse Colon. Retrieved from http://en.wikipedia.org/wiki/Transverse˙colon

[36] Manivannan, Siyamalan,Ruixuan Wang, Emanuele Trucco, and Adrian Hood. ”Automatic normal-abnormal video frame classification for colonoscopy.” In Biomedical Imaging (ISBI), 2013 IEEE 10th International Symposium on, pp. 644-647. IEEE, 2013.

[37] Thrun, Sebastian. ”Particle filters in robotics.” In Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence, pp. 511-518. Morgan Kaufmann Publishers Inc., 2002.

[38] Carpenter, James, Peter Clifford, and Paul Fearnhead. ”Improved particle filter for nonlinear problems.” IEE Proceedings-Radar, Sonar and Navigation 146, no. 1 (1999): 2-7.

[39] Nano Robotics Lab at Carnegie Mellon University, Robotics Institute, Retrieved from http://nanolab.me.cmu.edu/projects/capsules/index.shtml

[40] Toennies, Jenna L., Giuseppe Tortora, Massimiliano Simi, Pietro Valdastri, and R. J. Webster. ”Swallowable medical devices for diagnosis and surgery: the state of the art.” Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 224, no. 7 (2010): 1397-1414.

Appendix

Matlab Code:

This is the implementation code for the particle filter with multiple maps written in MATLAB.

myPFmap.m

```
%I1=imread(’1c.jpg’);
I1=imread(’newcolonbit1.bmp’);
J1= rgb2gray(I1);


I2=imread(’newcolonbit2.bmp’);
J2= rgb2gray(I2);


I3=imread(’newcolonbit3.bmp’);
J3= rgb2gray(I3);

[nrows,ncols]= size(J1);
xmax=ncols-1; %xmax=1056
ymax=nrows-1; %ymax=720

%black˙i and black˙j are the row and col number of the black pixels
%in the gray image matrix (J)

[black˙i,black˙j,v] = find(J1==0);



%And also make all rows of J as i and columns as j??????
%J(i,j)=J;

%plot(j,ymax-i,’.’);
%Plotting only the black pixels on a X, Y axis
plot(black˙j-1,ymax-black˙i+1,’.’);
title(’MAP #1’);

axis([0 xmax 0 ymax])
xlabel(’X axis’);
ylabel(’Y axis’);
```

myPFmap2.m

```
[
[nrows2,ncols2]= size(J2);
xmax=ncols2-1; %xmax=639
ymax=nrows2-1; %ymax=479

%black˙i and black˙j are the row and col number of the black pixels
%in the gray image matrix (J)

[black˙i2,black˙j2,v2] = find(J2==0);



%Plotting only the black pixels on a X, Y axis
plot(black˙j2-1,ymax-black˙i2+1,'.');
title('Map # 2');

axis([0 xmax 0 ymax])
xlabel('X axis');
ylabel('Y axis');
```

myPF.m

```
clear all close all clc
%----------------------------------
%Step1
figure(1)
%Step2
myPFmap

%Step3
hold on
%Step4
J=J1;
Jmodel=J1;
X = myPFinitdist(xmax, ymax, J);  % Initial Distribution Done
%Step5
plot(X(1,:), X(2,:), '.')
%----------------------------------

%Setting the Initial  Position & Orientation of the robot
x0 = 150;
y0 = 370;
theta0 = 0;

%Measuring the 5 sensor distances to the walls from the Inital Postion
z = myPFsensorfunction(x0, y0, theta0, xmax, ymax, J);

%Calculating the weights of each particle
myPFweightfunction;
w;
if(w==0)
```

```
    disp('Hey weight func screwed up, all weights are zero')
end

%Plotting the particles that will remain after resampling
figure(2)
myPFmap
hold on
X = myPFresample(X, w);
X0 = X;
plot(X0(1,:), X0(2,:), '.',x0, y0,'o')


%%
% Motion 1.
rho = 100;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
% Moving all particles accordingly to the new position
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X1 = X;
%Moving the actual robot (here the simulator) to the new postition
x1 = myPFmotion([x0 y0 theta0]', rho, deltatheta, 0, 0);
%Measuring the 5 sensor distances to the walls from the this Postion
z = myPFsensorfunction(x1(1), x1(2), x1(3), xmax, ymax, J );
%Allocating weights to the nearest particles
myPFweightfunction
%----------------------------------
figure(3)
myPFmap
hold on
X = myPFresample(X, w); %resampling
plot(X(1,:), X(2,:), '.',x1(1),x1(2),'o') %plotting the remaining particles
%----------------------------------


%% Motion 2.
rho = 100;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
% Moving all particles accordingly to the new position
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X2 = X;
%Moving the actual robot (here the simulator) to the new postition
x2 = myPFmotion([x1(1) x1(2) x1(3)]', rho, deltatheta, 0, 0);
%Measuring the 5 sensor distances to the walls from the this Postion
z = myPFsensorfunction(x2(1), x2(2), x2(3), xmax, ymax, J );
%Allocating weights to the nearest particles
myPFweightfunction
%----------------------------------
figure(4)
myPFmap
hold on
```

```
X=myPFresample(X,w); %resampling
plot(X(1,:), X(2,:), ’.’,x2(1),x2(2),’o’)  %plotting the remaining particles
%-----------------------------------


%% Motion 3.
rho = 100;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X3 = X;
x3 = myPFmotion([x2(1) x2(2) x2(3)]’, rho, deltatheta, 0, 0);
z = myPFsensorfunction(x3(1), x3(2), x3(3), xmax, ymax, J );
myPFweightfunction
%-----------------------------------
figure(5)
myPFmap
hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), ’.’,x3(1),x3(2),’o’)
%-----------------------------------


%% Motion 4.
rho = 100;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X4 = X;
x4 = myPFmotion([x3(1) x3(2) x3(3)]’, rho, deltatheta, 0, 0);
z = myPFsensorfunction(x4(1), x4(2), x4(3), xmax, ymax, J );
myPFweightfunction
%-----------------------------------
figure(6)
myPFmap
hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), ’.’,x4(1),x4(2),’o’)
%-----------------------------------


%% Motion 5.
rho = 50;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X5 = X;
x5 = myPFmotion([x4(1) x4(2) x4(3)]’, rho, deltatheta, 0, 0);
z = myPFsensorfunction(x5(1), x5(2), x5(3), xmax, ymax, J );
myPFweightfunction;
%-----------------------------------
```

```
figure(7)
myPFmap
hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), '.',x5(1),x5(2),'o')
%----------------------------------


%% Motion 6.
rho = 50;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X6 = X;
x6 = myPFmotion([x5(1) x5(2) x5(3)]', rho, deltatheta, 0, 0);
z = myPFsensorfunction(x6(1), x6(2), x6(3), xmax, ymax, J );
myPFweightfunction;
%----------------------------------
figure(8)
myPFmap
hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), '.',x6(1),x6(2),'o')
%----------------------------------


%% Motion 7.
rho = 50;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X7 = X;
x7 = myPFmotion([x6(1) x6(2) x6(3)]', rho, deltatheta, 0, 0);
z = myPFsensorfunction(x7(1), x7(2), x7(3), xmax, ymax, J );
myPFweightfunction;
%----------------------------------
figure(9)
myPFmap
hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), '.',x7(1),x7(2),'o')
%----------------------------------
hold off
J=J2; %True Map changes here
Jmodel = J2;  %Model changes here


%% Motion 8.
rho = 20;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
```

```
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X8 = X;
x8 = myPFmotion([x7(1) x7(2) x7(3)]’, rho, deltatheta, 0, 0);
z = myPFsensorfunction(x8(1), x8(2), x8(3), xmax, ymax, J );
myPFweightfunction;
%---------------------------------
%hold off %HEY DONT FORGET THIS
figure(10)
hold on
myPFmap
hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), ’.’,x8(1),x8(2),’o’)
%---------------------------------


%% Motion 9.
rho = 20;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X9 = X;
x9 = myPFmotion([x8(1) x8(2) x8(3)]’, rho, deltatheta, 0, 0);
z = myPFsensorfunction(x9(1), x9(2), x9(3), xmax, ymax, J );
myPFweightfunction;
%---------------------------------
figure(11)
myPFmap

hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), ’.’,x9(1),x9(2),’o’)
%---------------------------------


%% Motion 10.
rho = 20;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X10 = X;
x10 = myPFmotion([x9(1) x9(2) x9(3)]’, rho, deltatheta, 0, 0);
z = myPFsensorfunction(x10(1), x10(2), x10(3), xmax, ymax, J );
myPFweightfunction;
%---------------------------------
figure(12)
%myPFmap2
myPFmap
hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), ’.’,x10(1),x10(2),’o’)
%---------------------------------
```

```
%% Motion 11.
rho = 10;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X11 = X;
x11 = myPFmotion([x10(1) x10(2) x10(3)]’, rho, deltatheta, 0, 0);
z = myPFsensorfunction(x11(1), x11(2), x11(3), xmax, ymax, J );
myPFweightfunction;
%----------------------------------
figure(13)
%myPFmap2
myPFmap
hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), ’.’,x11(1),x11(2),’o’)
%----------------------------------
% hold off
% J=J1;
%Jmodel = J3;

%% Motion 12.
rho = 20;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X12 = X;
x12 = myPFmotion([x11(1) x11(2) x11(3)]’, rho, deltatheta, 0, 0);
z = myPFsensorfunction(x12(1), x12(2), x12(3), xmax, ymax, J );
myPFweightfunction;
%----------------------------------
hold off
figure(14)
myPFmap
hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), ’.’,x12(1),x12(2),’o’)
%----------------------------------


%% Motion 13.
rho = 30;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X13 = X;
x13 = myPFmotion([x12(1) x12(2) x12(3)]’, rho, deltatheta, 0, 0);
z = myPFsensorfunction(x13(1), x13(2), x13(3), xmax, ymax, J );
myPFweightfunction;
```

```
%----------------------------------
figure(15)
myPFmap
hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), '.',x13(1),x13(2),'o')
%----------------------------------


%% Motion 14.
rho = 20;
deltatheta = pi/2;
sigmatrans = 0.1;
sigmarot = 0.1;
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X14 = X;
x14 = myPFmotion([x13(1) x13(2) x13(3)]', rho, deltatheta, 0, 0);
z = myPFsensorfunction(x14(1), x14(2), x14(3), xmax, ymax, J );
% HERE IS THE CHANGE
myPFweightfunction;
%----------------------------------
figure(16)
myPFmap
hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), '.',x14(1),x14(2),'o')
%----------------------------------


%% Motion 15.
rho = 20;
deltatheta = 0;
sigmatrans = 0.1;
sigmarot = 0.1;
X = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot);
X15 = X;
x15 = myPFmotion([x14(1) x14(2) x14(3)]', rho, deltatheta, 0, 0);
z = myPFsensorfunction(x15(1), x15(2), x15(3), xmax, ymax, J );
myPFweightfunction;
%----------------------------------
figure(17)
myPFmap
hold on
X=myPFresample(X,w);
plot(X(1,:), X(2,:), '.',x15(1),x15(2),'o')
%----------------------------------


%% End of Particle Filter
disp('End of Particle Filter')
[xxx,III]=max(w);
disp('The predicted robot position is')
predicted˙position=X(:,III)
disp('The actual robot position is')
```

```
actual˙position=x15
error˙x=(actual˙position(1,1)-predicted˙position(1,1))/actual˙position(1,1)*100;
error˙y=(actual˙position(2,1)-predicted˙position(2,1))/actual˙position(2,1)*100;
error˙theta=(actual˙position(3,1)-predicted˙position(3,1))/actual˙position(3,1)*100;

error˙x= abs(error˙x);
error˙y= abs(error˙y);
error˙theta= abs(error˙theta);

% Error percentage of the implementation
disp('The error percentage is')
error=[error˙x;error˙y;error˙theta]
%% Error was around 1% in x, 3.1% in y, 12% in theta when I used 200 particles.
```

myPFinitdist.m

```
function X =myPFinitdist(xmax, ymax, J)

M=1000; %no of particles
X = [];
for p=1:M

    validity=0;
    while(validity==0)

%  temp= [round((xmax/3-2)*rand(1))+1; round((ymax-100-2)*rand(1))+1];
 temp= [round((xmax-2)*rand(1))+1; round((ymax-2)*rand(1))+1];
validity= myPFvalidposcheck(temp(1), temp(2), xmax, ymax, J);
        %display('this is the validity value')
    end
    X(:,p)=temp;
end
X = [X
    (2*rand(1,M)-1)*pi];
```

myPFvalidposcheck.m

```
function validity= myPFvalidposcheck(x,y,xmax,ymax,J)

%% This function checks if the point is inside, on the colon, or
%% outside the colon!

xmin = 0; ymin = 0;
xtmp = x;
ytmp = y;



%---------Region I---------Moving towards the right side of map----
atwallflag1= 0;
mapedgeflag=0;

while(atwallflag1˜=1 && mapedgeflag˜=1)
```

```
          ytmp = ytmp;
          xtmp = xtmp+1;
        if(xtmp¿=xmax)
mapedgeflag=1; %the point reached the mapedge before hitting wall
        end
        atwallflag1= isfilledinmap(xtmp,ytmp, xmax, ymax, J); %the point is not occupied

end
%atwallflag1

%---------Region II
atwallflag2= 0;
mapedgeflag=0;
xtmp = x;
ytmp = y;
while(atwallflag2˜=1 && mapedgeflag˜=1)
        ytmp = ytmp+1;
        xtmp = xtmp;
        if(ytmp¿=ymax)
            mapedgeflag=1; %the point is outside the colon
        end
        atwallflag2= isfilledinmap(xtmp,ytmp, xmax, ymax, J); %the point is not occupied
end
%atwallflag2

%---------Region III
atwallflag3= 0;
mapedgeflag=0;
xtmp = x;
ytmp = y;
while(atwallflag3˜=1 && mapedgeflag˜=1)
        ytmp = ytmp;
        xtmp = xtmp-1;
        if(xtmp¡=xmin)
            mapedgeflag=1; %the point is outside the colon
        end
        atwallflag3= isfilledinmap(xtmp,ytmp, xmax, ymax, J); %the point is not occupied
end
%atwallflag3

%---------Region IV
atwallflag4= 0;
mapedgeflag=0;
xtmp = x;
ytmp = y;
while(atwallflag4˜=1 && mapedgeflag˜=1)
        ytmp = ytmp-1;
        if(ytmp¡=ymin)
            mapedgeflag=1; %the point is outside the colon
        end
        xtmp = xtmp;
        atwallflag4= isfilledinmap(xtmp,ytmp,xmax, ymax, J); %the point is not occupied
end
```

```
%atwallflag4

   atwallflag = atwallflag1 * atwallflag2 * atwallflag3 * atwallflag4;


% if the new xtmp, ytmp pos are occupied, how to undo the new values?

%--------------WHILE LOOP JUST ENDED---------------

% IF YOU HIT THE WALLS ON FOUR SIDES, THAT MEANS THE PARTICLE IS
VALID.[INSIDE]
validity = atwallflag;

if isfilledinmap(x,y,xmax,ymax,J) == 1
    %disp('this is what is the problem')
    validity = 0;
end


return
%------------------------------------------------------------------------%
%------------------------------------------------------------------------%
```

isfilledinmap.m

```
function atwallflagnew = isfilledinmap(x, y, xmax, ymax, J);

[rowval,colval]=convertxytoij(x,y,ymax);

rowval = round(rowval);
colval = round(colval);

atwallflagnew = 0;
if (rowval ¿ 1)&&(rowval ¡ (ymax+1))&&(colval¿ 1)&&(colval ¡ (xmax+1))
  temp = J( (rowval-1):(rowval+1) , (colval - 1):(colval + 1) );
  temp = sum(sum(255 - temp));
  atwallflagnew = (temp ¿ 0); %this flag is 0 if there are only
  % white spaces around the pixel. Else the flag becomes 1.
end
```

myPFsensorfunction.m

```
function z = myPFsensorfunction(x, y, theta, xmax, ymax,J)
DMAX = 500;
xmin=0;
ymin=0;

%The below are the sensor positions relative to the heading of the robot (theta)
thetaoffsets = [-pi/3 -pi/6 0 pi/6 pi/3]';
%The global sensor angles are calculated below:
sensorangles = theta + thetaoffsets;
%slopes(1), slopes(2), slopes(3), slopes(4), slopes(5)
```

```
slopes = tan(sensorangles); %calculating various 'm' values.
bintercepts = y - slopes*x; %b intercept values for every sensor line
z = ones(size(thetaoffsets))*DMAX;  % making an matrix of 5's for the z value


%STEP 1
%Verify if the co-ordinate is already near a wall first so that you can check the
%isatwall flag. We have with us x, y, theta.

for ii=1:length(sensorangles)
    if (sensorangles(ii) ¡= -pi)
        sensorangles(ii) = sensorangles(ii) + 2*pi;
    end
    if (sensorangles(ii) ¿ pi )
        sensorangles(ii) = sensorangles(ii) - 2*pi;
    end
end

for i=1:length(z)    %REPEAT 5 times
    %i , sensorangles(i)
xtmp = x;
ytmp = y;
atwallflag= 0;
while(atwallflag˜=1)
      %Region I
       atwallflag1 = 0;
   if (sensorangles(i) ¿ -pi/4 &&  sensorangles(i) ¡= pi/4)
       xtmp = xtmp + 1;  %Sensor line moves along +x axis one step at once
       ytmp = slopes(i)*xtmp + bintercepts(i);
       xtmp=round(xtmp);
       ytmp=round(ytmp);
       atwallflag1= isfilledinmap(xtmp,ytmp, xmax, ymax, J);
   end

   %Region II
   atwallflag2 = 0;
   if (sensorangles(i) ¿ pi/4 &&  sensorangles(i) ¡= 3*pi/4)
       ytmp = ytmp + 1;
       if sensorangles(i) ˜= pi/2
         xtmp = (ytmp - bintercepts(i))/slopes(i);
       else
         xtmp = xtmp;
       end
       atwallflag2= isfilledinmap(xtmp,ytmp, xmax, ymax, J);
   end

   %Region III
   atwallflag3 = 0;
   if (sensorangles(i) ¿ 3*pi/4 &&  sensorangles(i) ¡= pi) —— ( sensorangles(i) ¿ -pi &&
sensorangles(i) ¡ -3*pi/4)
       xtmp = xtmp - 1;
       ytmp = slopes(i)*xtmp + bintercepts(i);
       atwallflag3= isfilledinmap(xtmp,ytmp, xmax, ymax, J);
      % pause
   end

   %Region IV
   atwallflag4 = 0;
   if (sensorangles(i) ¿ -3*pi/4 && sensorangles(i) ¡= -pi/4)
       ytmp = ytmp - 1;
       if sensorangles(i) ˜= -pi/2
         xtmp = (ytmp - bintercepts(i))/slopes(i);
       else
         xtmp = xtmp;
       end
       atwallflag4= isfilledinmap(xtmp,ytmp, xmax, ymax, J);
      % pause
   end

   atwallflag = atwallflag1 + atwallflag2 + atwallflag3 + atwallflag4;
   atwallflag = atwallflag ¿= 1;


   if atwallflag == 1
       break
   end
end


%Now we know the xtmp and ytmp values near the walls.
% So, let's calculate the Range between the new and the old points
   if (xtmp ¿= xmin) && (xtmp ¡= xmax) %Boundary checking if the new point is
within the map or not
       range=sqrt((xtmp-x).ˆ2+(ytmp-y).ˆ2);
       if (range ¿= 0) && (range ¡ z(i))
           z(i) = range;
       end
   end
end
end

%------------------------------------------------------------------------%
%------------------------------------------------------------------------%
```

myPFweightfunction.m

```
[nrows, ncols]=size(X);
M = ncols;
weight = zeros(1, M);
sigma = 100;
sigma2 = sigmaˆ2;
for i=1:M
    temp = X(:,i);
    %What is Jmodel, Jtrue?
    %Jtrue is what the robot(simulator) actually sees, ie, from its sensor
    %readings. Where as Jmodel is what the model is using from
    %previous iterations without any change. So basically, it’s
    %erroneous. So finally, the particles should fail following the robot
    %position.
    validity= myPFvalidposcheck(temp(1), temp(2), xmax, ymax, Jmodel);



    if(validity==1)
    zhat = myPFsensorfunction(temp(1), temp(2), temp(3), xmax, ymax,Jmodel);
    w(i) = exp( -0.5 * (norm( zhat - z)ˆ2)/sigma2);

    else
        %disp(’In myPFweightfunction: Validity is’); validity
        %disp(’ In myPFweightfunction: Hey we are in else loop of myPFweight func’)
        w(i)=0;
    end

end
diff=(X(:,1)-X(:,2));
if(diff(1)¡0.5 && diff(2)¡0.5)
    %disp(’Its the same particle chosen 5 times and all weights are ’);
    %w
end
```

myPFmotion.m

```
function xmoved = myPFmotion(X, rho, deltatheta, sigmatrans, sigmarot)

[nrows, ncols]=size(X);
xmoved = X;

for i=1:ncols

xmoved(1,i) = X(1,i) + rho*cos(X(3,i) + deltatheta) + randn(1)*sigmatrans;
xmoved(2,i) = X(2,i) + rho*sin(X(3,i) + deltatheta) + randn(1)*sigmatrans;
xmoved(3,i) = X(3,i) + deltatheta + randn(1)*sigmarot;

end
```

myPFresample.m

```
function Xresample = myPFresample(X, w)

if(max(w)==0)
disp(’ERROR in myPFresample:Weights of all particles is zero.’)
end
```

```
wsum = sum(w);
edges = cumsum(w/wsum);
%%figure(89),plot(edges);
%%figure(90),plot(w);
M = length(w);
R = rand(1,M);
R = sort(R);
Ichosen = zeros(1,M);
i=1; j=1;
while (i <= M)

    %[i j M R(i) edges(j)]
    if (R(i) ¡ edges(j))

        Ichosen(i) = j;
        i = i + 1;
    else
        j = j + 1;
        if (j>M)

            %j=M;
        end
    end
end
Xresample = X(:,Ichosen);


return
```

convertijtoxy.m

```
function [x,y]= convertijtoxy(i,j,ymax)

ymax=479;

y=ymax-i+1;
x=j-1;

end
```

convertxytoij.m

```
function [rowtmp,coltmp]= convertxytoij(x,y,ymax)

rowtmp=ymax-y+1;
coltmp=x+1;

end
```