PRIORITIZED EXPLORATION STRATEGY BASED ON INVASION PERCOLATION GUIDANCE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MURAT KARAHAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2009

Approval of the thesis:

PRIORITIZED EXPLORATION STRATEGY BASED ON INVASION PERCOLATION GUIDANCE

submitted by **MURAT KARAHAN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, **Graduate School of Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Aydan M. Erkmen
Supervisor, **Electrical and Electronics Engineering Dept., METU** _____

Prof. Dr. İsmet Erkmen
Co-Supervisor, **Electrical and Electronics Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. Kemal Leblebicioğlu _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Aydan Erkmen _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Mübeccel Demirekler _____
Electrical and Electronics Engineering Dept., METU

Assist.Prof. Dr. Afşar Saranlı _____
Electrical and Electronics Engineering Dept., METU

Assist.Prof. Dr Yiğit Yazıcıoğlu _____
Mechanical Engineering Dept., METU

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work

Name, Last name : Murat Karahan

Signature :

ABSTRACT

PRIORITIZED EXPLORATION STRATEGY BASED ON INVASION PERCOLATION GUIDANCE

Karahan, Murat

M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Aydan Erkmen
Co-Supervisor : Prof. Dr. İsmet Erkmen

December 2009, 110 pages

The major aim in search and rescue using mobile robots is to reach trapped survivors and to support rescue operations through the disaster environments. Our motivation is based on the fact that a search and rescue (SAR) robot can navigate within and penetrate a disaster area only if the area in question possesses connected voids Traversability or penetrability of a disaster area is a primary factor that guides the navigation of a search and rescue (SAR) robot, since it is highly desirable that the robot, without hitting a dead end or getting stuck, keeps its mobility for its primary task of reconnaissance and mapping when searching the highly unstructured environment We propose two novel guided prioritized exploration system: 1) percolation guided methodology where a percolator estimates the existence of connected voids in the upcoming yet unexplored region ahead of the robot so as to increase the efficiency of reconnaissance operation by the superior ability of the percolation guidance in speedy coverage of the area; 2) the hybrid exploration methodology that makes the percolation guided exploration collaborate with entropy based SLAM under a switching control dependent on either priority given to position accuracy or to map accuracy This second methodology has proven to combine the superiority of both methods so that the active SLAM becomes speedy, with high coverage rate of the area as well as accurate in localization.

Keywords: SLAM, Robotic Exploration, Search and Rescue Mobile Robots, Percolation Theory, Active SLAM ,Entropy based methodologies. .

ÖZ

KUŞATMA AMAÇLI SÜZÜLÜM TEMELLI ÖNCELIKLENDIRILMIŞ KEŞIF STRATEJILERI

Karahan, Murat

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Aydan Erkmen
Ortak Tez Yöneticisi : Prof. Dr. İsmet Erkmen

Aralık 2009, 110 pages

Arama kurtarma çalışmalarında mobil robotların kullanılmasındaki temel amaç enkaz altında sıkışan kazazedelere en kısa sürede ulaşabilmek ve gerekli kurtarma işlemlerini gerçekleştirebilmektir. Tezimizin motivasyonu arama kurtarma robotlarının enkaz ortamında ilerlerken birbirine bağlı boşlukları tespit edebilmesi ve bu yönde arama çalışmalarının hızlandırılabilmesi gerçeğine dayanmaktadır. Arama kurtarma robotlarının engellere takılmadan enkaz ortamında mobilitesini koruyarak ilerlemesi istenildiğinden; mobil robotun yönlendirilmesinde ortamda yer alan boşluklar arası geçişlerin tespit edilmesi önemli bir faktördür.Bu kapsamda iki yeni önceliklendirilmiş keşif stratejisi geliştirdik: 1) Süzülüm temelli keşif stratejisi ile robotun hareketi boyunca keşfedilmemiş bölgelerdeki boşlukları tahmin ederek arama çalışmalarının hızlandırılmasıdır 2) Süzülüm temelli ve entropi temelli keşif stratejileri arasında pozisyon ya da haritadaki belirsizliğe göre anahtarlama yapabilen bir hibrit keşif stratejisidir. Hibrit strateji süzülüm ve entropi temelli keşif stratejilerinin güçlü yönlerini alarak; arama faaliyetlerinde üstün ilerleme sağlarken pozisyon bilgisini de korumayı amaçlamaktadır.

Anahtar Kelimeler: Arama Kurtarma Robotları, SLAM, İnsansız Keşif Araçları, Süzülüm Tabanlı Keşif

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1. Introduction

1.1 Motivation

The major aim in urban search and rescue using mobile robots is to reach trapped survivors and to support rescue operations through the disaster environments. Many problems are defined in the literature so as to contribute new approaches for search and rescue mission such as reconnaissance and mapping, rubble removal, structural inspection, medical assessment and intervention, medically sensitive extrication and evacuation of causalities, adaptively shoring unstable rubble, serving as a repeater for wireless communication under trapped zone [1] The focus of this thesis is mainly related to the reconnaissance problem that is a critical issue of search and rescue operations so as to generate a disaster inventory that helps predicting the possibility of survivors to reach them among the cluttered and highly unstructured disaster area that generally includes many entrapments. With this reconnaissance as our main aim, we target to improve mobile robot navigation algorithms so as to maintain considerable performances for time critical search and rescue missions within the irregular disaster areas. Our motivation is based on the fact that a search and rescue (SAR) robot can navigate within and penetrate a disaster area only if the area in question possesses connected voids. Traversability or penetrability of a disaster area is a primary factor that guides the navigation of a search and rescue (SAR) robot, since it is highly desirable that the robot, without hitting a dead end or getting stuck, keeps its mobility for its primary task of reconnaissance and mapping when searching the highly unstructured environment.

1.2 Goals and Objectives

Reconnaissance mission consists of many distinct phases such as recognition of survivor's health conditions, detection of cluttered ruin structure and finding available paths towards trapped survivors. Within the balance of this thesis, we have focused on finding available paths, and exploring ramified voids until reaching trapped survivors. Since search and rescue mobile robots have to explore available paths within unknown areas, they cannot be initially aware of their location with respect to any existing map. For this reason our search and rescue mobile robot uses particle filter based simultaneous mapping and localization (SLAM) algorithm to estimate position and orientation while learning the environmental conditions. Our further objective is also to improve mobile robot mapping and localization performances against time and accuracy criteria. The mobile robot requires sonar and odometry measurements for its SLAM processes. Since those measurements are noisy, obtained SLAM maps and their corresponding available paths to the trapped survivors are only estimates, imprecise at first and cannot be reliable guides for search and rescue robots. We set our first goal to estimate maps more reliably and generate corresponding paths to trapped zones as correctly as possible. In all our goals, processes are done in disaster environments, which are complicated irregular environments that are deprived from any special distinctions that can be used as landmarks for mobile robot localization. Thus our second goal is to overcome this problem using metric partitioning of the environment such as grids, for divide and conquer in SLAM processes. On the other hand, our mobile robots are autonomous and their remote control has an increased complexity within cluttered and labyrinth-wise areas where sensing conditions can yield noisy and imprecise data. Our third goal is therefore to explore the current region gathering sufficient data so as to guide the mobile robot in the unexplored regions based on the knowledge gathered from the explored one. This guidance should be to estimate upcoming voids of unexplored areas, decreasing the risk of getting trapped and thus decreasing the search time.

1.3 Contributed Methodology

Towards our objective and having met the goals previously stated, the robot builds incrementally the map of the disaster area as a spatial "inventory" that provides at each step of the search, evolutionary situation awareness. This situation awareness represented by an in-growing map becomes a reference when deciding on the next move, which is the

direction that should be taken for the most suitable evolution of search at the next step. Guidance to our SAR robot at any instance of time provides this direction which is most optimal for search since it is based on disaster inventory gathered during the previous time steps of navigation until the frontier of the already explored area. Moreover knowledge of the frontier map of the explored area provides an essential clue for where to enter the unexplored area. In this thesis work, we developed a novel SLAM methodology that gives the robot navigation all the properties that aforementioned which encompasses a robot motion guided by the frontier based prediction of upcoming connected voids of the unexplored region that will be explored in the next time steps ahead. More precisely, our contribution in this thesis work is the development of a novel guided SLAM approach that puts preference on the exploration of connected voids within the rubbles of a disaster area that make use of Invasion Percolation Guidance for mobile robots in unstructured disaster areas based on the continuum of obstacles partially seen at the frontiers of explored regions and continuing, into unexplored upcoming areas. Our contribution resides in the methodology of predicting connected voids of the upcoming area to be explored, under the clue of partially seen obstacles at the frontiers of explored areas with an approach relevant to frontier-based explorations. Moreover our frontier based conditioning of a posteriori occurrences of new connected voids uses the fact that every obstacle partially seen at the frontier of the explored domain has a spatial continuity into the unexplored area and makes use of this conditioning in invasion percolation based estimation of upcoming connected voids. By using spatial continuity of obstacles, we propose to estimate available upcoming paths through possible connected voids among the locally detected obstacles. Towards this end, we develop an "invasion percolation estimator" that guides particle filter-based incremental SLAM. Percolation estimator estimates the possibility of spanning voids, conditioned on the obstacle configuration at the frontier of the present explored domain so as to guide mobile robot through the connected voids. As a second major contribution we integrated the percolator with the classical entropy based approach in a switching process to combine advantage of both methodology which are speed, high coverage and localization accuracy , which are all three highly critical for search and rescue.

1.4 Outline of the Thesis

Within the scope of this thesis work, we begin by introducing a literature survey in chapter 2, consisting of the mathematical background spanning Bayesian Filters, Particle

Filters, Particle Filter approaches to SLAM techniques and Entropy based Exploration approaches. We introduce our proposed methodologies as purely applied Percolator based Exploration and Percolator Enhanced Entropy Based Exploration techniques in chapter 3.We also present the detailed algorithms of those proposed methodologies with their block diagrams in chapter 3.Our simulation environment and its tools are given in chapter 4 so as to introduce the experimental instruments to analyze and compare our results obtained using the proposed methodologies with those of the classical approaches. In chapter 5, we interpret and discuss the obtained simulation results together with performance sensitivity analyses. Finally, in chapter 6, we draw necessary conclusions from the performance analyses of our two new proposed percolation added approaches based on the figures of chapter 5. This final chapter also extrapolates into possible future works.

CHAPTER 2

LITERATURE SURVEY

2. Literature Survey

2.1 Relevant Approaches in the Literature

Robot navigation for search and rescue (SAR) in highly irregular and unpredictable natural disaster environments needs to handle different combinations of problems including localization, coverage and motion planning issues many of which are solved by probabilistic or geometrical approaches. While coverage process time and mapping accuracy are critical for SAR explorations, many different exploration techniques exist [5, 6, 7, 8] that need spatially distinctive landmarks in the unknown environment such as wall following [7]. Special landmarks cannot be naturally found or artificially placed in disaster area, therefore exploration techniques with metric representations based on evidence grids [9] such as frontier based approaches [10] are more suitable for mobile robots in cluttered irregular environments. On the other hand, classic coverage based exploration techniques rely on known robot position and therefore cannot deal with the uncertainty about mobile robot position within unknown environments. Using such a methodology under increased position uncertainty of a SAR mobile robot will mislead the mapping phase of unknown environment. However, integrated exploration techniques deal with generating motion commands, while concurrently mapping unexplored regions and disambiguating robot positions. They provide a trade-off between mapping and localization processes so as to optimize time and enhance accuracy during navigation in unknown environments. Makarenko et al. [11] have weighted the costs of map exploration, robot position localization and navigation according to their corresponding utility functions under the trade-off of pose uncertainty and mapping. They have used

integrated exploration technique based on frontier approach using distinguishable special landmarks in the environment [11]. So their algorithm is not appropriate for irregular, unknown environments that cannot be landmarked. But Stanchiss et al. [12] develop an integrated exploration strategy without any landmark assumptions using active loop closing so as to disambiguate position uncertainty and enhance map accuracy. These proposed integrated exploration techniques generally consist of principles based on frontier-based exploration and information gain through entropy minimization strategies. Going beyond the recent integrated exploration approaches [11, 12, 20] the guidance of our SLAM approach, based on the prediction of connected voids leads our mobile robots into unexplored regions without losing their position awareness, tracking the connected voids within this new guided SLAM perspective. We have developed a novel extension to Grid Based Fast SLAM.

2.2 Mathematical Background

In this chapter, we give mathematical backgrounds that are introductory explanations for our proposed methodologies of chapter 3. We introduce the theoretical background of Bayesian filter and its approximation the particle filter. Mathematical framework of the simultaneous localization and adaptive mapping techniques that use particle filters are also presented in this chapter. Classical entropy based exploration methodology is overviewed including its mathematical derivations and its algorithm based representation. Finally, we give the mathematical background of invasion percolation methodology which our proposed methodologies of chapter 3 are based upon.

2.2.1 Simultaneous Localization and Adaptive Mapping

The simultaneous localization and mapping SLAM needs of a SAR robot are primarily, robustness, computational simplicity and speed. In Fast SLAM methodologies, these needs are met by Rao-Blackwellized particle filter approach speeding the computation by a factorization approach. Rao Blackwellized particle filter is a contribution of Murphy et.al [2] using the sequential Monte Carlo sampling approaches [3,4] for Rao Blackwellized particle filter factorization as seen in equation 1 [13, 14]. This factorization enables us to express the posterior probability of map m and robot trajectory $x_{1:t}$ over observation $z_{1:t}$ and odometry measurement data $u_{0:t-1}$ as expressed in the well known relation.

$$p(x_{1:t}, m \mid z_{1:t}, u_{0:t-1}) = p(m \mid x_{1:t}, z_{1:t}) * p\left(x_{1:t} \mid z_{1:t,}, u_{0:t-1}\right) \qquad (1)$$

Here $x_{1:t}$ is the robot trajectory that consists of robot states $x = (\overline{x}, \overline{y}, \overline{\theta})^T$, where $\overline{x}, \overline{y}, \overline{\theta}$ parameters correspond to horizontal, vertical and angular robot positions in 2D environment. u is the Odometry measurement u=$(x_t\ x_{t-\Delta t})^T$ that consists of relative robot positions so as to define robot movements in 2D environments.$z_{1:t}$ is the Proximity sensor (sonar, laser etc.) measurements until time step t. m corresponds to the variable set assigning map specifications. Feature based maps use mean and variance value pairs corresponding to landmark spatial position and uncertainties. On the other hand, metric based maps use binary variable set corresponding to grid based map occupation probabilities for each cell. In the scope of our thesis, we use metric based map representation.

2.2.1.1 Grid Based Fast SLAM Algorithm Outline

Rao-Blackwellized Fast Slam has individual trajectory estimation for each particle and therefore each particle carries individual estimation map according to sensor observations and their individual predicted trajectories. Grid based fast slam technique is outlined in Table 1 below. In that table, Grid Based Fast SLAM parameters are the particle set X_{t-1} that represents the previous time step's robot state belief , the sensor measurements or observations z_t at time step t, the odometry measurements u_t at time step t, the k th particle SLAM map $m_t^{[k]}$ at time step t, the k th particle robot position $x_t^{[k]}$ at time step t and the corresponding k th particle weight $w_t^{[k]}$ at time step t. The Grid Based Fast SLAM returns the particle set X_t for the current robot state belief. The accuracy of the map estimation is known to be mostly dependent upon trajectory estimation which is computed with re-sampling which uses $w_t^{[k]}$ weights proportional to the likelihood of the most recent z_t observation over obtained map and robot trajectory. There are no interactive effects for odometry control in this Grid Based Fast SLAM approach. State sampling processes with respect to motion model and odometry measurements, particle weight computations using measurement model and occupancy grid mapping using inverse measurement model, as expressed in the Sample State Extraction Phase of Table 1 (first row) will be covered in the upcoming subsections.

Table 1: Grid Based Fast SLAM Algorithm

Algorithm Fast SLAM (X_{t-1} , z_t , u_t)	

Sample State Extraction Phase	$\hat{X}_t = X_t = \emptyset$, $\hat{G}_t = \emptyset$
	FOR k = 1 to M do
	// State sampling with respect to motion model and odometry measurements
	$x_t^{[k]} = \text{sample\_motion\_model}(u_t, x_{t-1}^{[k]})$
	// Particle weight computations using measurement model
	$w_t^{[k]} = \text{measuerement\_model}(z_t, x_t^{[k]}, m_{t-1}^{[k]})$
	// Occupancy grid mapping using inverse measurement model
	$m_t^{[k]} = \text{update\_occupancy\_grid\_map}(z_t, x_t^{[k]}, m_{t-1}^{[k]})$
	END

Resampling Process	FOR i = 1 to M do draw i with probability $\alpha\, w_t^{[i]}$ add $\langle x_t^{[i]}, m_t^{[i]} \rangle$ to X_t END
Obtained Particle Set	RETURN X_t

2.2.1.2 Sample Motion Model

Firstly, state sampling processes is realized according to motion representation that models the noise effects on odometry measurements. Especially in slippery floor conditions, wheel movements can cause linear or angular errors within the odometry measurements. Odometry measurement is given by the vector components pair in $u_t = \begin{bmatrix} \overline{x}_{t-1} \\ \overline{x}_t \end{bmatrix}$ .In this pair of states in the vector components of u_t, $\overline{x}_t = \left(\overline{x}', \overline{y}', \overline{\theta}'\right)$ and $\overline{x}_{t-1} = \left(\overline{x} \ , \overline{y} \ , \overline{\theta} \ \right)$ are vectors which entries correspond respectively to the horizontal, vertical and angular displacements with respect to time step between t and t-1. Odometry motion can be modeled with three parameters that consist of initial turn δ_{rot1}, translation δ_{trans} and second rotation δ_{rot2} as is seen in Figure 1. [14]

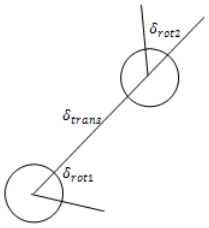

Figure 1 Odometry Motion Model Parameters [14]

According to that motion model representation of Figure 1, we can express the sample_motion_model function used in Table 2 generated from the time differential between sample robot states $\overline{x}_t = \left(\overline{x}', \overline{y}', \overline{\theta}'\right)$ and previous state $\overline{x}_{t-1} = \left(\overline{x}\ , \overline{y}\ , \overline{\theta}\ \right)$ with respect to given odometry measurement u_t and expected noise effects. We can express sample_motion_model with Table 2:

Table 2 Sample Motion Model Odometry

Algorithm sample_motion_model_odometry (u_t, x_{t-1})

// Motion Model Parameters
$\delta_{rot1} = atan2\left(\overline{y}' - \overline{y}\ , \overline{x}' - \overline{x}\ \right) - \overline{\theta}$
$\delta_{trans} = \sqrt{(\overline{x}' - \overline{x}\ )^2 + (\overline{y}' - \overline{y}\ )^2}$
$\delta_{rot2} = \overline{\theta}' - \overline{\theta}\ - \delta_{rot1}$
 	
//Noisy Motion Model Parameters
$\hat{\delta}_{rot1} = \delta_{rot1} - sample(\alpha_1 \lvert\delta_{rot1}\rvert + \alpha_2 \delta_{rot2})$
$\hat{\delta}_{trans} = \delta_{trans} - sample(\alpha_3 \delta_{trans} + \alpha_4 (\lvert\delta_{rot1}\rvert + \lvert\delta_{rot2}\rvert))$
$\hat{\delta}_{rot2} = \delta_{rot2} - sample(\alpha_1 \lvert\delta_{rot2}\rvert + \alpha_2 \delta_{trans})$
 	
//Generated Sample Robot State by Noisy Motion Model Parameters
$x' = x + \hat{\delta}_{trans} \cos(\theta + \hat{\delta}_{rot1})$
$y' = y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$
$\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$

RETURN $\overline{x}_t = \left(\overline{x}', \overline{y}', \overline{\theta}'\right)$

Sample (n) function in Table 2 corresponds to the Gaussian normal distribution, n being the variance. Noise characteristics are tuned using $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ motion model parameters which correspond to respectively the weights of importance given to initial turn δ_{rot1}, translation δ_{trans} and second rotation δ_{rot2} (Figure 1). For example, if we

desire to model the angular noise effects on odometry measurements that arise from slippery wheel; we can increase the weights α_1 *and* α_2 accordingly.

2.2.1.3 Measurement Model

Measurement model determines the particle weights of generated robot states according to obtained sensor observations from the environment. Measurement model [14] is explained according to equations presented in Table 3.

Table 3 Computing Particle Weights by Measurement Model

Algorithm $w_t^{[k]} = \text{measurement\_model}(z_t, x_t^{[k]}, m_{t-1}^{[k]})$
$\text{measurement\_model}(z_t, x_t^{[k]}, m_{t-1}^{[k]}) = P\left(z_t \middle\| x_t^{[k]}, m_{t-1}^{[k]}\right) = \eta\, N\left(z_t,\ z_t^{k*}, \sigma_{hit}^2\right)$ if $0 \le z_t \le z_{max}$ The normalizer can be evaluated as $\eta = \sum N\left(z_t,\ z_t^{k*}, \sigma_{hit}^2\right)^{-1}$; z_t^{k*} is ray casting values obtained from $x_t^{[k]}$ and $m_{t-1}^{[k]}$; $N\left(z_t,\ z_t^{k*}, \sigma_{hit}^2\right)$ *as the univariate normal distribution with mean* z_t^{k*} *and standard deviation* σ_{hit}^2*:* $$N\left(z_t,\ z_t^{k*}, \sigma_{hit}^2\right) = \frac{1}{\sqrt{2\pi\sigma_{hit}^2}}\, e^{-\frac{1}{2}\frac{(z_t - z_t^{k*})^2}{\sigma_{hit}^2}}$$

As is seen from Table 3, measurement modeling gives $P\left(z_t \middle\| x_t^{[k]}, m_{t-1}^{[k]}\right)$ post probability conditioned on robot state $x_t^{[k]}$ at time step t and SLAM map $m_{t-1}^{[k]}$ at time step t-1.Without considering normalizer η, the summation of all conditional probabilities $P\left(z_t \middle\| x_t^{[k]}, m_{t-1}^{[k]}\right)$ of possible ray tracing values according to possible robot states $x_t^{[k]}$ can be greater than 1.Therefore we have to utilize a normalizer factor within the equations of measurement modeling to equate the maximum sum of probabilities to 1.

2.2.1.4 Occupancy Grid Map Update Process

Occupancy grid mapping process generates SLAM maps according to robot trajectory $(x_t^{[k]})$ and sensor observation (z_t) that constitute information for each particle. We can express occupancy grid map (m) as a collection of grid cells $m = \{m_i\}$. Each grid cell m_i has binary occupancy value that specifies is a cell is free or occupied. Those binary values are usually expressed with "1" for occupied grids and "0" for free ones in literature [10]. So we can express grid cell occupation probability with notations such

that $(m_i = 1)\ or\ p(m_i)$.We can project down the posterior map estimation problem $p(m|z_{1:t}, x_{1:t})$ over grid cells using the notation $p(m_i|z_{1:t}, x_{1:t})$ as a separate estimation problem for each grid cell. Since each of these estimation problems on grid cells are binary with static state, we can approximate such factorization as $p(m|z_{1:t}, x_{1:t}) = \prod p(m_i|z_{1:t}, x_{1:t})$. Using this factorization, the estimation of occupancy probability for each grid cell is now a binary estimation problem with static case.[14] So binary Bayes filter can be applied to occupancy grid map posterior probability estimation problem as is seen from Table 4 and the occupancy grid mapping algorithm uses log odds representation as $I_{t,i} = \log \frac{p(m_i|z_{1:t}, x_{1:t})}{1-p(m_i|z_{1:t}, x_{1:t})}$ for time t and i th grid cell on occupancy grid map representation.

Table 4 Occupancy Grid Mapping [14]

Algorithm occupancy_grid_mapping { $\{I_{t-1,i}\}, x_t, z_t$ }

For all cells m_i do
If m_i in perceptual field of z_t then
$I_{t,i} = I_{t-1,i} + inverse\_sensor\_model(m_i, x_t, z_t) - I_0$
Else
$I_{t,i} = I_{t-1,i}$
End If
End For
Return { $I_{t,i}$}

The $inverse\_senor\_model(m_i, x_t, z_t)$ that the occupancy grid mapping uses (Table 4) $\log \frac{p(m_i|z_{1:t}, x_{1:t})}{1-p(m_i|z_{1:t}, x_{1:t})}$ that is calculated as given in Table 5 [14].

Table 5 İnverse Sensor Model Algorithm [14]

Algorithm inverse_sensor_model(m_i, x_t, z_t)
Let x_i, y_i *be the center of grid cell* m_i $r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$ $\varphi = atan2(y_i - y, x_i - x) - \theta$ $k = \min_j \lvert\varphi - \theta_{j,sens}\rvert$ IF $r > \min(z_{max}, z_t^k + \alpha)\ \ or\ \ \lvert\varphi - \theta_{k,sens}\rvert > \frac{\beta}{2}\ then$ Return I_0 IF $z_t^k < z_{max}\ \ and\ \lvert r - z_t^k\rvert < \alpha/2$ Return I_{occ} IF $r \leq z_t^k$ Return I_{free} END IF

The parameters of this calculation are represented on the sensor beam of Figure 2.

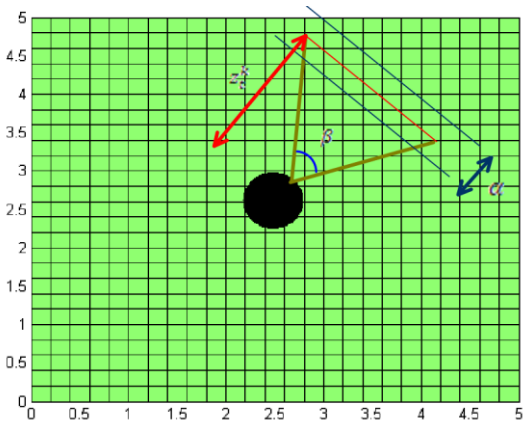

Figure 2 Sensor Beam Representation for Inverse Sensor Model

We can see from the Figure 2 that z_{max} is the maximum sensor range, the beam width β , the possible obstacle width α, z_t^k expresses the sensor proximity reading to obstacle from sensor beam k and (x_i, y_i) represents grid cell centroids location. As is seen from Table 5, grid cells that drop in the region between possible obstacle width correspond to possible void cells while grid cells drop in the region of possible obstacle width correspond to possible obstacle cells. On the other hand, other cells which are outside of the sensor cone expressed in Table 5 are unaffected grid cells from inverse sensor measurement model.

2.2.1.5 Bayes Filter and Its Approximated Derivatives

A belief represents the robot's internal knowledge about its environment. Since the robot cannot measure its pose or state information directly from its environment, this information must be inferred from acquired data from its surrounding. Robot state belief is represented by a distribution that corresponds to the posterior probabilities over state variables conditioned on obtained data from the environment. We can express this belief distribution as:

$$bel(x_t) = p(x_t | z_{1:t}, u_{1:t})$$

More specifically the posterior probability distribution $bel(x_t)$ of robot state variable $x_t$is given as conditioned on past measurements $z_{1:t}$ and control data $u_{1:t}$. Bayes filter algorithm is the most general algorithm to calculate beliefs distributions from measurement and control data based on recursions that computes belief distribution $bel(x_t)$ at the current time step t from belief distribution $bel(x_{t-1})$ at the previous time step with most recent control u_t and measurement data z_t (Table 6).

Table 6 Bayes Filter Recursive Algorithm [14]

Algorithm Bayes_Filter($bel(x_t), u_t, z_t$)

FOR all x_t do
$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1}) dx_{t-1}$
$bel(x_t) = \eta\, p(z_t \mid x_t) \overline{bel}(x_t)$
END FOR

RETURN $bel(x_t)$

Bayes filter has two main steps as prediction and measurement update. In the prediction, step belief $\overline{bel}(x_t)$ over robot state x_t, is computed by the integral of the product of prior belief distribution $bel(x_{t-1})$ and conditional probability distribution $p(x_t|u_t, x_{t-1})$ corresponding to the state transition between x_t and x_{t-1} performed by the control u_t. At the measurement update step, $bel(x_t)$ is obtained as a measurement probability $p(z_t|x_t)$ conditioned over robot state multiplied by the predicted belief distribution $\overline{bel}(x_t)$. The normalization constant η is used for the case that multiplication $p(z_t|x_t)\overline{bel}(x_t)$ violates the probability range. Hence normalization constant η is used as a normalizer to obtain a probability value for $bel(x_t)$.

2.2.1.5.1 Bayes Filter Mathematical Derivation

In this section, we will prove the Bayes filter algorithm using mathematical induction. We will demonstrate how Bayes filter recursive iterations are obtained as a computation of current posterior probability $p(x_t|z_{1:t}, u_{1:t})$ from the belief set $bel(x_{t-1})$ of the previous time step. In the derivation of Bayes filter, the robot state has to be complete information while control actions are chosen at random [14]. Completeness of a robot state means that the knowledge of one step past states, measurements or controls carries no additional information that would help us predicts the future more accurately. Namely, a state x_t is called complete if it is the best predictor of the future state. Temporal processes that fit those completeness conditions are known as Markov Chains. The Markov assumption postulates the past and future data are independent if one knows the current state x_t [14]. At the first step of Bayes derivation; let use the Bayes rule for our target posterior probability conditioned on odometry measurements and proximity sensor observations:

$$p(x_t|z_{1:t}, u_{1:t}) = \frac{p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t})}{p(z_{1:t}|z_{1:t-1}, u_{1:t})}$$
$$= \eta\, p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t}) \quad (2)$$

In this derivation, we assumed that x_t robot state is complete. This means there is no parameter prior to robot state x_t in the stochastic evaluation of future states. Hence past measurements and past control data cannot provide additional information for robot state x_t. So we can derive equation 3 as:

$$P(z_t|x_t, z_{1:t-1}, u_{1:t}) = p(z_t|x_t) \quad (3)$$

Since equation 3 means conditional independence, it leads equation 2 to equation 4

$$p(x_t|z_{1:t}, u_{1:t}) = \eta\ p(z_t|x_t)\ p(x_t|z_{1:t-1}, u_{1:t}) \quad (4)$$

So we can trivially write equation 5 as an expression of robot current belief state

$$bel(x_t) = \eta\, p(z_t|x_t)\, \overline{bel}(x_t) \quad (5)$$

On the other hand, equation 5 corresponds to the measurement update phase of the Bayes filter algorithm in Table 6. Now, we can derive the computation of predicted belief state distribution $\overline{bel}(x_t)$ from the control data and belief state distribution at the previous time step, that is expressed as in equation 6

$$\overline{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t}) = \int p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t})\, p(x_{t-1}|z_{1:t-1}, u_{1:t})\, dx_{t-1} \quad (6)$$

Let use our robot state completeness assumption in equation 6 to prove the Bayes filter prediction step. As is seen from equation 7, we know the x_{t-1} robot state at time $t-1$. So due to the robot state completeness, past measurements $z_{1:t-1}$ and control data $u_{1:t-1}$ do not yield any additional information, thus,

$$p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t) \quad (7)$$

As a result of completeness assumption in equation 7, we can reach equation 8

$$\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t)\, p(x_{t-1}|z_{1:t-1}, u_{1:t-1})\, dx_{t-1} \quad (8)$$

Equation 8 corresponds to the prediction step equations in Bayes filter in Table 6.If we summarize the derivation results, the Bayes filter algorithm computes posterior

probability over robot state x_t conditioned on the past measurements and control data until time step t. On the other hand, the importance of the completeness assumption on the robot state has been seen on the derivation of Bayes filter prediction (equation 8) and measurement update (equation 5) equations.

2.2.1.5.2 Particle Filter and Importance Sampling

The particle filter is one of the most popular nonparametric implementations of Bayes filter algorithm. A particle filter represents the posterior probability of a robot state as a sampled robot state belief distribution. In particle filters, the samples drawn from a posterior probability distribution are called particles and can be listed as is seen in equation 9.

$$X_t = x_t^{[1]}, x_t^{[2]}, \dots., x_t^{[M]} \quad (9)$$

Particles in equation 9 are instantiations of the state at time t. Each particle $x_t^{[n]}$ $1 \leq n \leq M$ corresponds to a hypothesis that expresses what the true state may be at time t. The aim of the particle filter is to estimate the belief $bel(x_t)$ in the Bayes filter which algorithm is provided in Table 6 using the set of particles X_t. The generic particle filter algorithm is known to be as given in Table 7. As is seen from Table 7, the input of the particle filter algorithm is the particle set X_{t-1}, control data u_t and measurement data z_t. The particle filter algorithm firstly constitutes a temporary particle set $\hat{X}_t$ corresponding to the predicted state $\overline{bel}(X_t)$ of the Bayes Filter algorithm explained in Table 6. A hypothetical robot state $x_t^{[m]}$ for time step t is generated from the m th particle within the particle set X_{t-1} labeled as particle $x_{t-1}^{[m]}$ by control u_t. After that this sampled robot position is generated, the particle filter algorithm assigns importance factors $w_t^{[m]}$ to those robot positions $x_t^{[m]}$ particles by incorporating the most recent measurement data z_t. The Importance factor $w_t^{[m]}$ corresponds to the probability of generated robot position particle $x_t^{[m]}$ with respect to measurement data z_t as expressed by $w_t^{[m]} = p\left(z_t \middle| x_t^{[m]}\right)$ in line 4 of Table 7. If importance factors are interpreted as particle weights then we can interpret that weighted set of particles approximately represents the posterior belief distribution $bel(X_t)$. Resampling or importance sampling process are processed between line 7 and line 10 in Table 7 so as to draw M particles from the temporary particle set $\hat{X}_t$ according to importance weights $w_t^{[m]}$. Importance sampling

process transforms the temporary particle set $\hat{X}_t$ into a new particle set X_t of same size using particle displacements. Since particles set is re-sampled according to most reliable particles, there exist particle duplications within the re sampled particle set X_t. As a consequence of importance sampling process, some particles with lower importance weights are removed from the temporary particle set $\widehat{X}_t$. Consequently, the particle set obtained at the end of re sampling process is approximately distributed with respect to posterior $bel(X_t)$.

Table 7 Particle Filter Algorithm [14]

Algorithm Particle Filter(X_{t-1}, u_t, z_t)
1. $\hat{X}_t = X_t = \emptyset$
2. FOR m=1 to M do
3. $sample\ x_t^{[m]} \sim p\left(x_t \middle\| u_t, x_{t-1}^{[m]}\right)$
4. $w_t^{[m]} = p\left(z_t \middle\| x_t^{[m]}\right)$
5. $\hat{X}_t = \hat{X}_t + < x_t^{[m]}, w_t^{[m]} >$
6. END FOR
7. FOR m=1 to M do // Importing Sampling Phase
8. $draw\ i\ with\ probability\ \propto\ w_t^{[i]}$
9. $add\ x_t^{[i]}\ to\ X_t$
10. END FOR
11. RETURN X_t

Importance sampling process corresponds to the measurement update step of the Bayes filter algorithm that implements measurement update using $bel(X_t) = \eta\, p\left(z_t \middle\| x_t^{[m]}\right) \overline{bel}(X_t)$. Importance sampling process yields particles within the temporary particle set $\hat{X}_t$ that are distributed according to the target posterior $bel(X_t)$ In some particle filter algorithms, resampling process is not applied to the generated particle set in line 3 so as to maintain diversity of particle set. Particle weights are initialized by 1 and updated multiplicatively as given in the following equation:

$$w_t^{[m]} = p\left(z_t \middle\| x_t^{[m]}\right) w_{t-1}^{[m]}$$

In this case, the particle filter approximates a posterior distribution that is similar to the previous one that the importance sampling has used. Since many particles converge to

regions with low posterior probabilities, resampling process is necessary as an adaptive mechanism that refocuses the particle set to the higher posterior probability regions so as to use computational resources more effectively. Let us now present the mathematical background of the particle filter outlined in Table 7. In order to derive the particle filter mathematically, we consider particles as a sequence of state samples:

$$x_{0:t}^{[m]} = x_0^{[m]}, x_1^{[m]}, x_2^{[m]}, \dots, x_t^{[m]} \quad (10)$$

The particle filter computes the posterior probability over all state sequences of equation 10; the posterior probability is expressed as in equation 11, instead of the expression $bel(x_t) = p(x_t|u_{1:t}, z_{1:t})$ of the Bayes filter approach.

$$bel(x_{0:t}) = p(x_{0:t}|u_{1:t}, z_{1:t}) \quad (11)$$

Using Bayesian rule we can derive equation 12 from equation 11;

$$p(x_{0:t}|z_{1:t}, u_{1:t}) = \eta\, p(z_t|x_{0:t}, z_{1:t-1}, u_{1:t})\, p(x_{0:t}|z_{1:t-1}, u_{1:t}) \quad (12)$$

Let use the robot state completeness assumption to reduce equation 12 as;

$$\begin{aligned} p(x_{0:t}|z_{1:t}, u_{1:t}) &= \eta\, p(z_t|x_t)\, p(x_{0:t}|z_{1:t-1}, u_{1:t}) \\ &= \eta\, p(z_t|x_t)\, p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t})\, p(x_{0:t-1}|z_{1:t-1}, u_{1:t}) \end{aligned}$$

$$p(x_{0:t}|z_{1:t}, u_{1:t}) = \eta\, p(z_t|x_t)\, p(x_t|x_{t-1}, u_t)\, p(x_{0:t-1}|z_{1:t-1}, u_{1:t-1}) \quad (13)$$

Let us use mathematical induction in this step by assuming that the initial particle set is obtained by sampling $p(x_0)$ and that the particle set at time t-1 is distributed with respect to $bel(x_{0:t-1})$. Thus we can derive proposed distribution in equation 14 for $x_{0:t-1}^{[m]}$ the m th particle sample generation process that correspond to line 3 of particle filter algorithm in Table 7

$$p(x_t|x_{t-1}, u_t)\, bel(x_{0:t-1}) = p(x_t|x_{t-1}, u_t)\, p(x_{0:t-1}|z_{0:t-1}, u_{0:t-1}) \quad (14)$$

Particle weights are expressed as the equation below:

$$w_t^{[m]} = \frac{target\ distribution}{proposal\ distribution} = \eta \frac{p(z_t|x_t)\, p(x_t|x_{t-1}, u_t)\, p(x_{0:t-1}|z_{1:t-1}, u_{1:t-1})}{p(x_t|x_{t-1}, u_t)\, p(x_{0:t-1}|z_{0:t-1}, u_{0:t-1})} = \eta\, p(z_t|x_t)$$

By multiplying the proposed distribution $p(x_t|x_{t-1}, u_t)\, p(x_{0:t-1}|z_{0:t-1}, u_{0:t-1})$ with particle weights $w_t^{[m]}$, the resulting particle distribution is approximated by the posterior probability of belief function $bel(x_{0:t})$ as given in equation 15

$$\eta\, w_t^{[m]} p(x_t|x_{t-1}, u_t)\, p(x_{0:t-1}|z_{0:t-1}, u_{0:t-1}) = bel(x_{0:t}) \qquad (15)$$

2.2.1.5.3 Importance Sampling: Mathematical Derivation

In this section we provide the mathematical background of the importance sampling process that is used for particle filter posterior probability approximations. Let us assume that we are computing an expectation over a probability density function f with given samples generated with respect to different density function g. Let assume that we are also interested in expectation $E_f[I(x\epsilon A)]$ for $x\epsilon\, A$. In order to express the expectation $E_f[I(x\epsilon A)]$ for $x\epsilon\, A$ using expectation over g instead of expectation over f which is not directly available base on chain rule as follows [14]:

$$E_f[I(x\epsilon A)] = \int f(x)\, I(x\epsilon A) dx = \int \frac{f(x)}{g(x)}\, g(x)\, I(x\epsilon A)\, dx \qquad (16)$$

Thus we can derive equation in 17 according to equation 16 to express the expected value for function f using expected value for function g:

$$E_f[I(x\epsilon A)] = E_g[w(x)\, I(x\epsilon A)] \;\; where\; w(x) = \frac{f(x)}{g(x)} \qquad (17)$$

In equation 17, $w(x) = \frac{f(x)}{g(x)}$ weights correspond to the mismatch between two distributions f and g. $f\; and\; g$ should have similar monotonicity such as $f(x) > 0 \rightarrow g(x) > 0$ for equation 17 to hold. Importance sampling process in particle filter is based on the transformation expressed in 17. As we recall from particle filter explanations in the previous section, the targeted distribution $bel(X_t)$ is approximated from the proposed distribution $\overline{bel}(X_t)$ using importance sampling. We should note that the finite number of counted particles $x^{[m]}$ that fall into interval A converges to the integral of g under A as expressed in equation 18

$$\frac{1}{M}\sum_{m=1}^{M} I(\, x^{[m]} \epsilon\, A) \rightarrow \int_A g(x)\, dx \qquad (18)$$

Let use $w^{[m]} = \frac{f(x^{[m]})}{g(x^{[m]})}$ importance weights to offset the difference between f and g density functions then equation 19 arises:

$$\left[\sum_{m=1}^{M} w^{[m]}\right]^{-1} \sum_{m=1}^{M} I(x^{[m]} \epsilon\, A)\, w^{[m]} \rightarrow \int_A f(x)dx \quad is\ normalizer \qquad (19)$$

$\left[\sum_{m=1}^{M} w^{[m]}\right]^{-1}$is the normalizer in equation 19 for all importance weights. It is seen that we have reached weighted particles that converge to density f from a set of sampled particles according to the density g under mild conditions. In similar conditions, particles in particle set X_{t-1} are distributed with respect to $bel(X_{t-1})$ and density g in equation 18 can be matched to the product distribution as expressed by $p(x_t|u_t, x_{t-1})bel(X_{t-1})$.

2.2.1.5.4 Low Variance Sampler Algorithm

During resampling processes, displacement of lower weighted particles could cause particle deprivation as a result of decreased diversity among particles within the particle set. Since decreasing diversity of particles increase particle filter variance as an estimator, obtained estimations could be faulty as time passes by. In order to prevent the sudden decrease of particle diversity, we use low variance sampler algorithm in the resampling phase of the particle filter between lines 7 and 10 in Table 7. Low variance sampler algorithm has been shown in Table 8. Low variance sampler takes the particle set X_t and particle weights W_t (importance or resampling weights) as input parameters and returns re-sampled particle set as output parameter $\overline{X}_t$.M is the number of particles within the given particle set X_t.At the first step, a random number r is generated from the interval $0< r < M^{-1}$. Then the first particle weight value is assigned to the c parameter in the third line. Thus the sampler algorithm starts the looping process in line 6 to setting the particle weight values based on the U variable that starts to be computed from r value with stepping constant interval M^{-1} . In each step, the sampler algorithm compares the reached value U with the accumulation of particle weights c in order to decide particle inclusion within the re-sampled particle set $\overline{X}_t$.

Table 8 Algorithm Low Variance Sampler [14]

Algorithm Low Variance Sampler(X_t, W_t)
1. $\overline{X}_t = \emptyset$
2. $r = rand(0, M^{-1})$
3. $c = w_t^{[1]}$
4. $i = 1$
5. FOR m=1 to M do
6. $U = r + (m-1)\ M^{-1}$
7. WHILE $U > c$
8. $i = i + 1$
9. $c = c + w_t^{[i]}$
10. END WHILE
11. $add\ x_t^{[i]}\ to\ \overline{X}_t$
12. END FOR
13. RETURN $\overline{X}_t$

Until the accumulation of particle weight values reaches the step value U, the low variance sampler algorithm omits the lowest weighted particles based on regular comparisons at each step instead of randomly driven cases. Thus the diversity of particles in the re-sampled particle set $\overline{X}_t$ is protected by distinct selection intervals M^{-1} at consecutive steps.

2.3 Active SLAM Techniques

2.3.1 General

Active SLAM techniques provide odometry commands based on the decision making performed using the observations from the explored environment. Active SLAM guides two types of actions; first ones are exploitation actions that localize the robot and the second ones are exploration actions that help the mobile robot in reaching unexplored regions. Entropy based Fast SLAM techniques [11, 20] are optimized active slam methodologies that utilize a trade off mechanism between exploitation and exploration actions.

2.3.2 Entropy Based Exploration Strategy

Entropy based exploration method aims to gain maximum information during robot navigation. This gain information is necessary for robot localization and map exploration. In SLAM approaches we have to decrease localization uncertainties that tend to build up while exploring new regions. Entropy based exploration methods choose optimal action from a proposed set of alternative actions in order to obtain the maximum information gain about robot position and its environment. We can express information gain as the decrement of robot position and environment map uncertainties, where uncertainty changes are computed using Shannon Entropy. First of all, we have to state the uncertainty values related with robot position and map based on the acquired sensory data and given motion commands. As is seen from equation 20 derived in [20]; Rao-Blackwellized (equation 1 in section 2.2.1) factorization is used so as to obtain the total Shannon Entropy as the summation of position and map individual entropies. So we can compute the total entropy value related with robot uncertainty as a summation of position and map entropy values that correspond respectively to position and map uncertainties as:

$$H\big(p(m, x_{1:t}|d_t)\big) \approx H\big(p(x_{1:t}|d_t)\big) + \sum_{i=1}^{\#Particles} w_t^{[i]} H\left(p\left(m^i \middle| x_{1:t}^{[i]}, d_t\right)\right) \qquad (20)$$

Where $H\big(p(m, x_{1:t}|d_t)\big)$ is the entropy value that corresponds to total uncertainty about robot and its environment . $H\big(p(x_{1:t}|d_t)\big)$ is the entropy value that corresponds to the robot position uncertainty and $\sum_{i=1}^{\#Particles} w_t^{[i]} H\left(p\left(m^i \middle| x_{1:t}^{[i]}, d_t\right)\right)$ is the entropy value that corresponds to map uncertainty. Here $w_t^{[i]}$ is the i th particle weight at time t; m^i,the i th particle SLAM map at time t; z_t ,the sensor measurements at time step t ; $u_{1:t-1}$, motion commands until time t-1; and d_t corresponds to $\{u_{1:t-1}, z_t\}$. The computations of individual position and map entropies are given in equations 21 through 25. Since $m^{[i,j]}$ values are binary and considered independent, we can state the total map entropy using Binary Shannon Entropy as shown in equation 21 below:

$$\sum_i \sum_j -p_{(i,j)} \log p_{(i,j)} - (1 - p_{(i,j)}) \log(1 - p_{(i,j)}) \qquad (21)$$

Here $p_{(i,j)}$ is the occupation probability of grid cell $m^{[i,j]}$. Where $m^{[i,j]}$ is the (i, j) th grid cell on the grid based SLAM map. $H\left(p(x_{1:t}|d_t)\right)$ is the expression for Position Entropy Value which is approximated as equation 22

$$H\left(p(x_{1:t}|d_t)\right) \approx H\left(p(x_t|d_t)\right) \qquad (22)$$

This approximation in equation 22 can be done because of the completeness of robot state variable $x_t.\left(p(x_t|d_t)\right)$ state can also be expressed from previous states and measurements in equation 23 :

$$p(x_t|d_t) = \eta\, p(z_t|x_t)\; p\left(x_t|x_{t-1,}u_t\right) p(x_{0:t-1}|z_{1:t-1}, u_{1:t-1}) \qquad (23)$$

Since we generate samples x_t with respect to the probability distribution function as shown in equation 24, $p\left(x_t|x_{t-1,}u_t\right)$ corresponds to the state transition probability and $p(x_{0:t-1}|z_{1:t-1}, u_{1:t-1})$ states robot state belief distribution at previous time step. Since previous belief distribution is represented by a particle set, in a similar way, $p\left(x_t|x_{t-1,}u_t\right)$ is also stated with a generated sample state set instead of a parametric type of expression.

$$x_t \sim\; p\left(x_t|x_{t-1,}u_t\right) p(x_{0:t-1}|z_{1:t-1}, u_{1:t-1}) \qquad (24)$$

Since those generated sample states are selected randomly but weighted according to the measurement update step, we can approximate $p(x_t|d_t)$ probability value of equation 23 with an iteratively multiplied particle weights as shown in equation 25.We consider that the initial belief set is distributed properly and that the iteratively multiplied particle weight until any time step corresponds to the previous particle set distribution. Hence:

$$p(x_t|d_t) \approx \eta\, p(z_t|x_t) \qquad (25)$$

In accordance with equation 25, we obtain approximately the robot position entropy expression as shown in equation 26. Such an approximation is considerable since the uncertainty amount for the robot position can be approximately estimated from the uncertainty amount of the particle set distribution that corresponds to the robot state belief distribution in particle filter implementations.

$$H\left(p(x_{1:t}|d_t)\right) \approx H\left(\eta\, p(z_t|x_t)\right) \qquad (26)$$

In this equation, u_t is the odometry measurement at time step t; $z_{1:t}$ are the proximity sensor measurements until time step t; $x_{1:t}$ are the robot states until time step t; d_t corresponds to the $\{u_{1:t-1}, z_t\}$ parameters. For map entropy computations, it is significant to reach those uncertain occupancy grid map cells within the SLAM map during action selection process so as to maximize information gain from the SLAM map. Mapping unknown regions is more effective for gaining map information than mapping obstacle or void cells for entropy based exploration strategy. After unknown regions are explored, the robot localization uncertainty is reduced according to recently explored regions. For this reason, obstacle or void cells will be more informative on the recovery of robot position awareness. The trade-off between precision on localization and that of the map has to be dealt with all along the whole mobile robot exploration process. Since position and map entropies are derived as shown in equations 21 and 26 [20] with considered assumptions, we can state the information gain computation as in equation 27 below:

$$I(\hat{z}, a_t) = H\big(p(m_t, x_{1:t}|d_t)\big) - H\big(p(m_t, x_{1:t}, \hat{x}|d_t, a_t, \hat{z})\big) \qquad (27)$$

Where, a_t is the proposed action at time step t; $\hat{x}$ is the hypothetical robot position at time step t+1;$\hat{z}$ is the obtained virtual measurements using ray tracing for time step t+1;m_t is the SLAM map at time step t; $x_{1:t}$ is the robot states until time step t; d_t corresponds to the $\{u_{1:t-1}, z_t\}$ parameters. In order to realize equation 27, we iterate the SLAM computations one step forward using proposed odometry action alternatives and compare their effects on the position and map uncertainties. On the other hand, sonar measurements not being available at the unreal mobile robot next step; we generate ray tracing measurements on the obtained SLAM maps that correspond to proximity measurements at the next time step. Those proximity measurements obtained from ray tracing values are called as virtual measurements ($\hat{z}$) as stated in equation 27.As a consequence of information gain function of equation 27, we can state our mobile robot entropy based exploration strategy as in Table 9. According to Table 9; the entropy based exploration methodology takes the particle set that corresponds to the current belief distribution of mobile robot X and returns a proposed action u as odometry command in SLAM processes. On the other hand the inner loop of entropy based exploration algorithm repeats the Fast SLAM processes $X'_t = \text{Fast SLAM}\,(X_{t-1}, z_t, u_t)$ with simulated virtual measurements $z \sim p(z|x')$ and generates the sample robot state $x' \sim p(x'|u, x)$. So for every proposed action, the entropy minimization is

computed as denoted by $H_{X'}(x')$ and the trade-off between cost $r(x,u)$ and entropy minimization $H_{X'}(x')$ is provided as $p_u = p_u + r(x,u) - \alpha H_{X'}(x')$

Table 9 : Entropy Based Exploration Algorithm

Algorithm Entropy Based Exploration (X)
Set $p_u = 0$ for all actions u
FOR i = 1 to N do
sample $x \sim X(x)$
FOR all control actions u do
sample $x' \sim p(x'\|u,x)$
sample $z \sim p(z\|x')$
X'_t = Fast SLAM (X_{t-1}, z_t, u_t)
$p_u = p_u + r(x,u) - \alpha H_{X'}(x',u)$
END For
END For
RETURN $\max_u p_u$

.In our thesis work, we did not considered $r(x,u)$ in entropy based exploration implementations since we have mostly focused on information gain based exploration guidance. Thus our update equation of p_u is as: $p_u = p_u - \alpha H_{X'}(x')$ meaning that action selection through the alternatives is realized according to a tradeoff between information gain and cost in performing that action.

2.4 Invasion Percolation Model

Percolation theory has been of a particular interest to us since it is the study of connected objects in applications fluid flow in porous media [15,16], molecular connectivity of water, diffusion of different materials, city growth [17].Percolation is also important for oil extraction since oil exists as connected cluster underneath the earth. One characteristic that has caught researchers' attention is minimum path in percolation among connected clusters (spanning voids) which provides a conditional probability that is related, in oil research, to the time elapsed between injection of water until the extraction of the first bit of oil [15, 21].Invasion percolation models the diffusion of one fluid (invader) into other

fluid (defender) within a porous medium. Invasion percolation theory differs from ordinary classical percolation in that it automatically finds the critical points of the system that determine existence of spanning clusters [15, 19]. Furthermore the primary principle of invasion percolation is to diffuse along a path of least resistance [19]. Consequently invasion percolator finds the site which has lowest resistance porosity among all already invaded adjacent sites and then invades it. Invasion depends on the threshold pressure of each pore which is defined as an occupation probability. Lattice representation used for site percolation is also an integrated part of invasion percolation as that each lattice site becomes equipped with invasion probability similar to percolation probability. The mathematical model of the invasion percolation has been summarized in [15]. As stated in there, $X(e): e \in E$, is an independent random variable indexed by the set of all edges E in the d-dimensional lattice $Z^d, d \geq 2$, having uniform distribution on the interval [0, 1].In our situation; we can illustrate a 2 dimensional lattice whose edges are indexed by random variables $X(e)$ as in Figure 3.

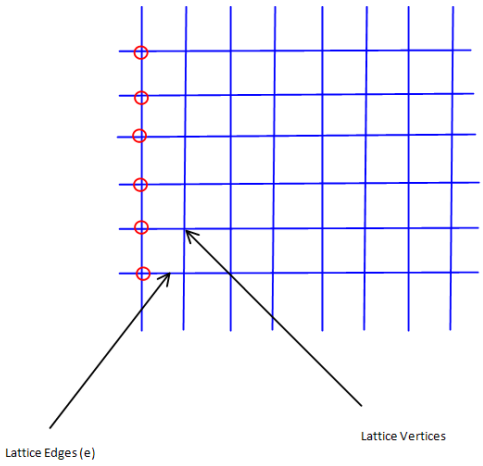

Figure 3 2 Dimensional Lattice Structure

Figure 3 shows lattice edges and vertices of a 2 dimensional array. Invasion percolation algorithm starts its iteration with initially given graph and a sequence of random connected sub graphs of the array is constructed in an iterative way. We can illustrate that process by representing sample iteration on the 2 dimensional arrays. For this, consider C_0 corresponding to our initial sub graph represented by red marked vertices on the lattice in Figure 3. In the scope of invasion percolation, our iteratively expanding sub graph corresponds to the invader matter while given array structure corresponds to the

defender one. Those vertices of initially given sub graph select their neighboring edges (e) that have minimum assigned random variable ($X(e)$). Neighboring edges are defined as edges that have one of their vertices connected with the iterated sub graph while others are outside of that sub graph. Neighboring edges are represented in Figure 4 with red marked arrows for the C_0 sub graph.

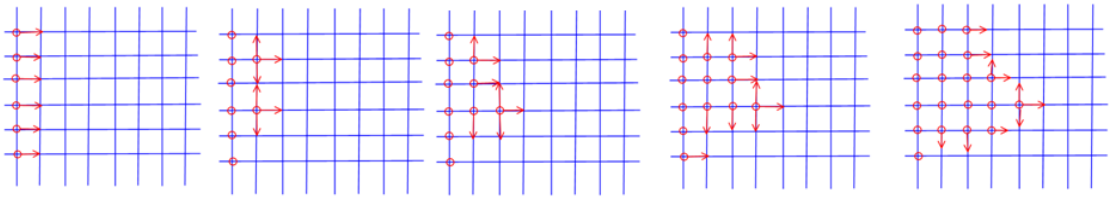

Figure 4 Neighbor Edge Iterations

Invasion percolation model adds an edge (e) to the initially given sub graph according to its assigned random variable ($X(e)$) that has the minimum value among the other neighboring edges. In brief edge e_{i+1} is chosen from the outer edge boundary of C_i that minimizes $X(e_{i+1})$ to expand initially given sub graph at each iteration step as seen from Figure 4. Red arrows corresponds to the edges while randomly assigned $X(e)$ are determined which neighbor edge will be selected. As is seen from the iterations of Figure 4, each perimeter vertices of sub graph has been shown with added red arrow corresponding to the neighbor edges between perimeter and neighbor vertices. On the other hand, those random variables $(X(e))$ assigned to the edges can be modeled as porosity values for the grid based site lattice structures [19].Thus we handle those invader and defender representations so as to model the continuity of possible void and obstacles represented by grid based SLAM maps based on adjustment of porosities. Due to the occupation probabilities of grid cells, their porosity levels are assigned according to our analogy for modeling continuity of voids and obstacles on SLAM map as introduced in chapter 3.

CHAPTER 3

METHODOLOGY

3. Methodology

3.1 Most Relevant Methodologies at the Foundation of Our Approach

We consider our mobile robot exploration problem as an invasion of voids over occupancy grid cells of the environment being explored under the defense of obstacle cells (defender). Our primary aim is to obtain spanning probabilities of the voids in order to be able to choose, under this guidance, the most suitable regions to explore next, in the upcoming unknown regions. Percolation theory has been of a particular interest to us to achieve this objective [15] having found strong applications to fluid flow in porous media [15] , molecular connectivity of water , diffusion of a different materials, city growth [17]. Percolation is also important for oil extraction since oil exists as connected clusters beneath the earth. One characteristics that has caught researchers attention is minimum path in percolation among connected clusters (spanning voids) which provides a conditional probability that is related , in oil recovery , to the time elapsed between injection of water until the extraction of the first lit of oil [21].However in our exploration problem , spanning voids of unexplored area are not known explicitly and it is nonsense to measure time passed in our exploration problem since our mobile robot does not know where to go in our unknown environment. Furthermore we cannot obtain any percolation probability of our unexplored medium before exploration due to excessively high unknowns. Therefore the classical percolation methodology has been proven non applicable for predictive SLAM in search and rescue robotics. On the other hand, gradient percolation [15] such as modeling diffusion in variably porous material is of high relevance to our objective. Among those, invasion percolation that models the

percolating invader into a material occupied by defender molecules depending upon its porosity degrees is highly appropriate for our mobile robot exploration in unknown irregular environments. This diffusion in materials modeled by invasion percolation inspired us into using it to estimate the "diffusion" of possible paths within unknown regions based on "extrapolation" of obstacles detected at the vicinity of frontiers of the explored region as a continuum into the unexplored region. Porosity term has been used for defender resistance to traversability of invader [15, 19]. To represent traversability between the defender and invader, we assign porosity degrees to the lattice sites of the percolation environment. The invasion model proposed by Wilkinson and Willemsen [19] is based on assigning the threshold pressures of pores to site occupation probabilities. Invasion percolation expands iteratively by porosity minimization on sites of outer boundary. These iteratively added sites trace the percolating path of the invader over defender molecules. In our approach we model the spatial distributions of obstacles and voids using the displacement of defender and invader fluids within the porous medium. We represent voids by invader fluid and obstacles by defender fluid. The key motivation behind using invasion percolation in our approach is the continuity of percolating invader expanding through neighboring sites with lowest porosity degrees. Thus, invasion percolation ramifications continue through neighboring sites with lower degree porosity while other ramifications coming across higher porosity, die out. Since occupancy grid cell representation is also similar to the site lattice structure of reference [19], we have also related occupation probabilities of occupancy grid cells with the porosity degrees of the lattice sites. We adjust these porosity degrees of unexplored sites so as to impose the continuity of previously detected obstacles to trace possible spanning voids into the unexplored areas. Our percolator modification augments the porosity degrees at obstacle sites and decreases it for free sites. Iteratively added neighbor sites that have lowest porosity degrees represent the expansion of invader percolation. This builds a representation of the frontier of the explored area such that is indented by obstacles that plunge into the unexplored region. Thus, we can model the indentations of spatially connected spanning voids by increasing the porosity degrees at the vicinity areas of obstacles. Our robot is guided in this way, by upcoming spanning voids avoiding the possible continuities of obstacles in the unexplored region, just detected at the boundary of the explored region. We have generated two novel systems in this thesis work, which can be easily explained by the block diagrams that we outlined in Figure 5 and Figure 9 The first one is the Percolator Based Exploration architecture of a Grid Based Fast

SLAM guided by a percolator module as seen in Figure 5.This approach will be detailed in section 3.2 going through each of its modules one by one in the subsections. The second one is a hybrid system that we name the Percolation Enhanced Entropy Based exploration (Figure 9) where the guidance of SLAM switches between the entropy based one when precision in localization is highly required , and the percolator based one when map precision in highly demanded. This hybrid system is detailed in section 3.3.

3.2 Percolator Based Exploration with Grid Based Fast SLAM

The simultaneous localization and mapping needs of a SAR robot are primarily, robustness, computational simplicity and speed. These needs are met by the Rao-Blackwellized particle filter approach. We adapt and modify here, the Fast SLAM technique [13] which is an efficient extension of Rao-Blackwellized particle filter approach to simultaneous localization and mapping problem. Rao-Blackwellized particle filter provides robustness and simplicity to SLAM with a factorization assumption. There are no interactive effects for odometry control in the classical approach. However our invasion percolator realizes this active control for the mobile robot by providing our active feedback to the classical SLAM. Thus our percolator guided grid based fast SLAM follows the orientation advised by the percolator so as to reach larger probable connected voids rather than wandering without guidance among rubbles or getting entrapped with small isolated dead-end voids. Our percolator based exploration technique is outlined in Table 10 below:

Table 10 : Percolator Based Exploration with Grid Based Fast SLAM Algorithm

Algorithm of *Percolator Based Exploration with Grid Based Fast SLAM* (X_{t-1})	
Apply Grid Based Fast SLAM Algorithm	X_t = Fast SLAM (X_{t-1}, z_t, u_t)
Extract Map and Trajectory Information	$(m_t^{[k]}, x_{t-1}^{[k]}) = Extract\_Map\_\&\_Trajectory\_Info(X_t)$
Percolator Controller Guidance (module 7 in Figure 9) or system of Figure 5	$u_{t+1} = Invasion\_Percolator\_Controller(m_t^{[k]}, x_{t-1}^{[k]})$

Where, X_{t-1} is the particle set at time step t-1; $w_t^{[k]}$ is the k th particle weight at time step t; $m_t^{[k]}$ is the k th particle's SLAM map at time step t; $x_t^{[k]}$ is the k th particle's

hypothetical robot position at time step t; z_t is the proximity sensor measurements at time step t; $u_{odometry\_command}$ is the given odometry guidance for the robot navigation.

3.2.1 Invasion Percolator Guidance Infrastructure

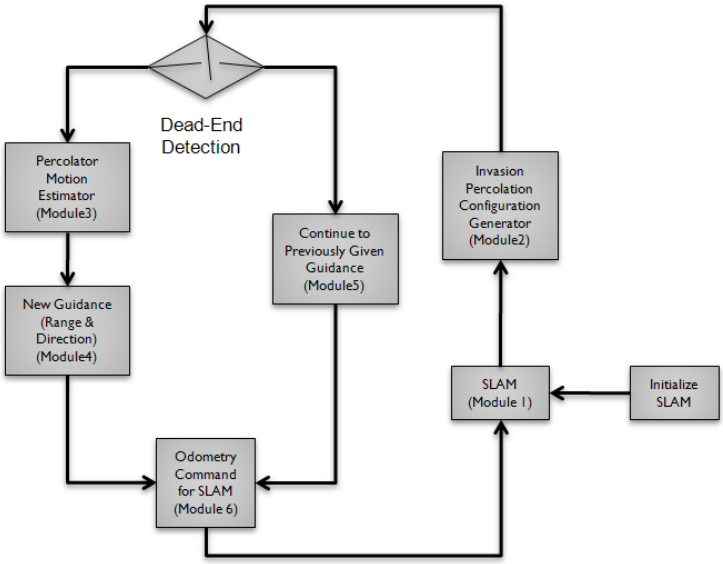

Figure 5 Outline of the Invasion Percolator Guidance Module (Module 7 of Figure 9)

The Invasion Percolator guidance module has the basic infrastructure as seen from Figure 5. We initialize percolator based exploration with the classical Fast SLAM (module 1) overviewed in chapter 2 section 2.21.1 .that provides an initial map according to the sensor readings obtained from the environment. This initially found SLAM map is input to the Invasion Percolation Configuration Generator (module 2) that will be introduced in section 3.2.2. Invasion Percolation Configuration Generator generates the percolator map according to given SLAM maps. The generated percolator maps are analyzed for dead ends over generated percolator map. If no dead ends are encountered on the percolator map then, the mobile robot continues its navigation through the connected voids in the previous same direction as seen in module 5. Otherwise , Percolator Motion Estimator (module 3), as explained in section 3.4.2, estimates available spanning directions and a new guidance is selected at the most spanning path direction (module 4). Within module 6, we construct odometry displacement vector as explained in chapter 2 section 2.2.1.2 from those coordinate couples consisting of robot position coordinates and available spanning path destination coordinates which are obtained from module 4.

3.2.2 Invasion Percolation Configuration Generator (module 2 of Figure 5)

The invasion percolator estimates percolating voids and detects obstacles by obtaining spanning probabilities of voids through the unexplored regions. For this reason the classical invasion percolation process mentioned in section 2.4 needs modifications to make it sensitive to obstacles. Thus, we modify the porosity distribution in our approach by increasing the porosity degrees of neighboring sites of the previously detected obstacles. So that invasion of the voids mostly spread over the unexplored region, while avoiding previously detected obstacles. This is the key idea for efficient and fast reconnaissance that predicts upcoming voids and obstacles based on the assumption of their continuity from the frontier of the already explored area. Occupancy grid cells correspond to the sites of the lattice structure; the porosity of unknown sites neighboring the occupied sites within the locally explored region has high values while the porosity of unknown sites neighboring the free sites has low porosity. So invasion percolator generates a ramification (path) conditioned on these given porosity degrees. Invasion Percolation Configuration Generator generates different ramifications for each different weighted particle. Invasion percolation ramification has been derived as a guidance map from particle's SLAM map by the Invasion Percolator Configuration Generator algorithm as explained in Table 11 above. Invasion percolation configuration generator algorithm M takes occupancy grid based SLAM maps as input parameter represented by three occupancy matrices that have occupation probabilities according to each grid cell of occupancy grid map. The three matrices that are the cluster, perimeter and porosity matrices, have the same size based on the given occupancy grid map. Cluster matrix is an array where the invaded cells take the value of 1 while invasion free ones are assigned 0 values. Porosity matrix arranges the transitions possibilities of invasion expansion on the Cluster array. Porosity matrix has been formed between line 6-14 in table 15.Entries of the Porosity matrix take the value of 1 for obstacles that has occupation probability values $M(I,J)$ greater than 0.5 where I,J is the occupancy grid index on the map. While -1 value is assigned to Porosity entries that have occupation probability $M(I,J)$ value lower than 0.5, thus corresponding to void cells. This classification is performed to fit a guidance that avoid obstacles and lead to void regions. We use Perimeter matrix to control invasion through the Cluster array. We can explain the mechanism with Figure 6 below:

Table 11 Invasion Percolator Configuration Generator Algorithm

Invasion Percolator Configuration Generator Algorithm (M)
1. $Cluster = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}_{RowSize_M x ColSize_M}$ //Initialize Cluster Matrix
2. $Perimeter = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}_{RowSize_M x ColSize_M}$ //Initialize Perimeter Matrix
3. $Porosity = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}_{RowSize_M x ColSize_M}$ //Initialize Porosity Matrix
4. //Construct Porosity Matrices
5. FOR $I = 1\ to\ RowSize_M$
6. FOR $J = 1\ to\ ColSize_M$
7. IF $M(I,J) > 0.5$
8. $Porosity(I,J) = 1$ //For obstacles
9. ELSE
10. $Porosity(I,J) = -1$ //For Voids
11. END
12. END
13. END
14. $Perimeter\left(:, round\left(\frac{ColSize_M}{2}\right)\right) = 1_{RowSize_M}$
15. WHILE ($(y,x)\ not\ exist$)
16. $(y,x) = Find_{(y,x)}(\ Perimeter(y,x) = 1)$
17. $(y,x) = min_{(y,x)}(Porosity(y,x))$
18. $Cluster(y,x) = 1$
19. $Perimeter(y,x) = 0$
20. //Update Perimeter and Cluster Matrices
21. IF $(y < ColSize_M)$,
22. $Perimeter(y+1,x) = Reverse(Cluster(y+1,x))$;
23. END
24. IF $(y > 1)$,
25. $Perimeter(y-1,x) = Reverse(Cluster(y-1,x))$;
26. END
27. IF $(x < RowSize_M)$,
28. $Perimeter(y,x+1) = Reverse(Cluster(y,x+1))$;
29. END
30. IF$(x > 1)$,
31. $Perimeter(y,x-1) = Reverse(Cluster(y,x-1))$;
32. END
33. END WHILE
34. RETURN Spanning Grid Cells = CLUSTER

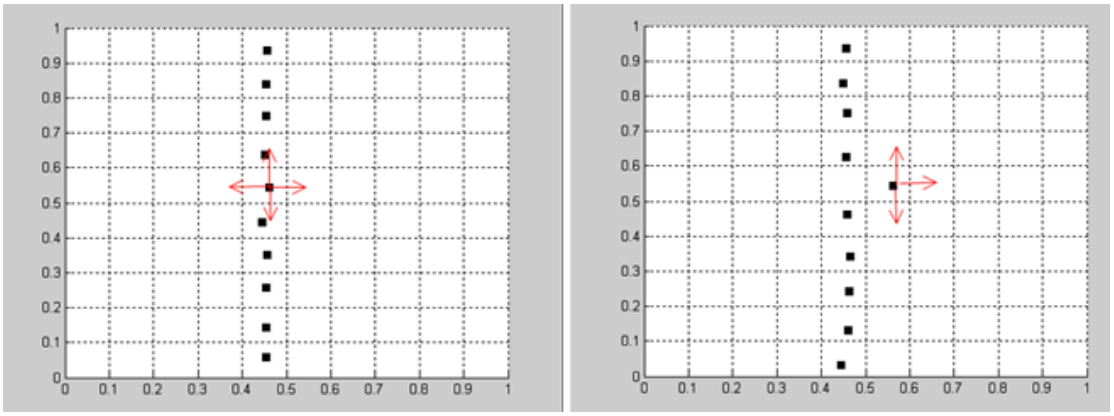

Figure 6 Invasion Percolation Perimeter Matrix Transition

Consider an array as seen in Figure 6 where, empty cells correspond to the 0 valued elements of Perimeter matrix while black dotted cells correspond to the 1 valued entries of the same matrix. As is seen from the red arrows on the array, the invasion percolator generator algorithm seeks neighboring cells that have minimum porosity value. To realize this step as seen in Figure 6 , we have to find first the perimeter cells in line 17 and then find the neighbor cells of that perimeter cell that has minimum porosity in line 18. We know that Porosity matrix has 0 values for occupancy grid map cells corresponding to unknown region with $M(I,J) = 0.5$; the values of -1 for occupancy grid map cells corresponding to void cells $M(I,J) < 0.5$ and 1 for occupancy grid map cells corresponding to obstacle cells $M(I,J) > 0.5$. Thus perimeter cells that correspond to void or unknown region in Perimeter matrix are selected as cells of occupancy grid cell values $M(I,J) < 0.5$ that have not been invaded yet but having neighboring cluster cells invaded as shown in Figure 7. Thus our invasion percolation will expand through the void or unknown regions avoiding obstacles. At the last step, invasion generator algorithm updates Perimeter and Cluster matrices between the lines 21-33. Cluster matrix entries that correspond to the invaded neighbor cells then reverse their values from 0 to 1. On the other hand, Perimeter matrix values corresponding to the invaded cell are assigned as new perimeter cells at the lines through 21-33. Hence this Cluster matrix is called a Spanning Grid Cell matrix in our invasion percolator guidance estimator computations. Spanning Grid Cell Matrix is a two dimensional matrix that holds the invasion situations of the grid cells with respect to occupancy grid map cell indices. 1 valued Spanning Grid Cell Matrix entry corresponds to the invaded regions while 0 valued regions correspond to the invasion-free regions. Invaded regions correspond to the

voids or void continuities while invasion free regions correspond to the obstacles or obstacle continuities. In occupancy grid mapping representation of particle SLAM maps as shown in Figure 7 (left) , voids correspond to the occupation probability values lower than 0.5 while possible obstacle cells have occupation values higher than 0.5.

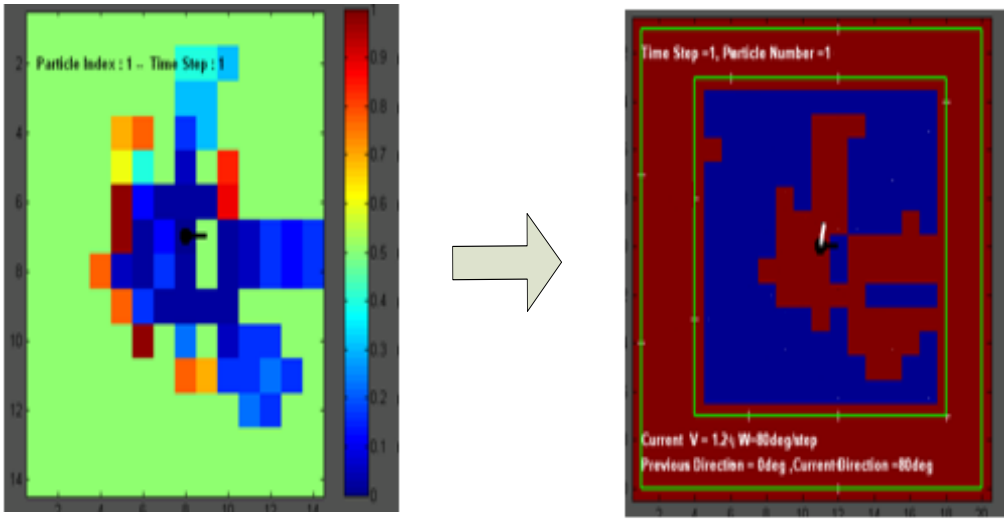

Figure 7 Invasion Percolator Configuration Map Generation (right) from given SLAM Map (left)

On the right of Figure 7, an invasion percolator configuration map is obtained for a particle's occupancy grid map. As is seen from the right part of Figure 7; blue colored cells correspond to the invasion free cells while red ones are invaded. White colored arrow on percolator map in Figure 7 (right) corresponds to mobile robot next direction, while black one corresponds to its current bearing in Figure 7 (right).If mobile robot's measurements were noise-free then we could consider only one Spanning Grid Cell matrix so as to determine robot orientation and maximum span range. But we have to generate more invasion percolation configurations for different particles because of robot's noisy odometry and sonar measurements. Thus for each Spanning Grid Cell matrix corresponding to a particle set, the Invader Motion Estimator module needs to estimate the spanning range and orientation.

3.2.3 Percolator Motion Estimator (module 3 of Figure 5)

Percolator Motion Estimator provides the average span range and robot orientation from each particle's invasion configuration. The Spanning Grid Cells matrix represents the invasion configuration of each particle according to their individual hypothetical robot positions and SLAM maps. Since our odometry and sensor measurements are noisy, we

cannot trust any particle's orientation command or maximum span distance information. Thus we have to take weighted averages of these hypothetical spanning orientation commands and maximum span distance values according to the Percolator Motion Estimator algorithm as outlined below in Table 12.

Table 12 : Percolator Motion Estimator Algorithm

Algorithm Percolator Motion Estimator (Spanning Grid Cells Matrix Collection)	
Part 1	1. FOR $ParticleIndex = 1\ to\ Particle\ Number$ 2. Sensor Based Cells =Extract Sensor Based Cells(Spanning Grid Cells Matrix Collection) 3. FOR $Sensor\ Direction = 1\ to\ Sensor\ Number$ 4. FOR $Cell\ Index = 1\ to\ Total\ Cell\ Number - 1$ 5. IF($SensorDirectionBasedCells(Sensor\ Direction, Cell_{Index}) > SensorDirectionBasedCells(Sensor\ Direction, Cell_{Index})$) 6. $Counted\_Spanning\_Cell\_Number = CellIndex$; 7. ELSE 8. BREAK; 9. END 10. END 11. $Spanning\_Range(Sensor\ Direction) = Counted\_Spanning\_Cell\_Number * Cell\_Interval$; 12. END 13. END 14. $SpanningDirection(ParticleIndex) = \max_{SensorDirection} Spanning\_Range$ (Sensor Direction) 15. $SpanningRange(ParticleIndex) = SpanningRange(SpanningDirection(ParticleIndex))$ 16. END
Part 2	17. $AverageOrientation = \sum_{ParticleIndex=1}^{Particle\ Number} ParticleWeights(ParticleIndex) * SpanningOrientations(ParticleIndex)$ 18. $AverageSpanRange = \sum_{ParticleIndex=1}^{Particle\ Number} ParticleWeights(ParticleIndex) * MaxSpanningRange(ParticleIndex)$
	19. RETURN $AverageOrientation, AverageSpanRange$

We can explain Percolator Motion Estimator in two parts as shown in Table 12. Firstly in part 1; our estimator takes Spanning Grid Cell Matrix Collection as an initially given parameter. Spanning Grid Cell Matrix Collection contains Spanning Grid Cell Matrix for each particle. Thus we realize the spanning range estimations on this Spanning Grid Cell

Matrix through the sensor directions relative to the hypothetical robot bearing. We form an array consisting of grid cells through the mobile robot sensor directions, as called Sensor Direction Based Cells, using line 2. Sensor Based Direction Cells array is a two dimensional matrix that holds the invasion situations of Spanning Grid Cells Matrix through the sensor direction at the particle robot positions. Sensor Direction Based Cells array contains invasion situations of grid cells through the sensor directions within the Spanning Grid Cells Matrix

.

Figure 8 Sensor Direction Based Grid Cells are represented with black and white markers on the Spanning Grid Cell Matrix representation

In Figure 8, Sensor Direction Based Cells are seen as white and black dots on the array that is represented by the Spanning Grid Cell Matrix. Sensor Direction Based Cells contains invasion situations according to the grid cells addresses that can be seen on Figure 8 as black and white markers. Black markers correspond to the available grid cells invaded by voids taking a value of 1 in the Spanning Grid Cell Matrix, while white markers correspond to the non available grid cells through the Sensor Based Grid Cells array that corresponds to the 0 valued grid cells in the Spanning Grid Cell Matrix. Our percolator motion estimator algorithm counts the available grid cells that correspond to the invaded cells within the Sensor Direction Based Cells array that is obtained from Spanning Grid Cells Matrix values, according to the sensor directions. Percolator motion

estimator algorithm checks the invasion situations through the Sensor Direction Based Cells array for the sensor direction orientation with an if statement in line 5, which detects the invasion situation that changes from 1 valued situation to 0 valued situation on Spanning Grid Cell Matrix ,that is to say from an invaded phase to an invasion free phase. So such a transition ends the spanning range and our algorithm counts the available grid cells until dead-end using line 5.For the dead-end phase, line 8 breaks the loop process for counting available grid cells. In lines 14 and 15; most spanning sensor direction and its range value are found and saved as Spanning Direction and Spanning Range vectors for each particle. Part 2 computes the expected spanning directions and spanning range values with respect to particle weights as is seen in lines 17 and 18. As a consequence of these processes, each particle has individually spanning direction and spanning range according to its particle robot position and SLAM map. Those spanning specifications differ from each other according to measurement noises. Therefore we compute the average values of those spanning range and orientation values according to particle weight values so as to obtain guidance feedbacks that are closer to true values for mobile robot exploration.

3.3 Percolator Enhanced Entropy Based Exploration

3.3.1 Motivation

Our proposed percolator based exploration methodology in section 3.2 is developed to reach connected voids with an increased efficiency according to classical exploration techniques. Percolator realizes that objective by estimating the continuity of voids beyond the explored region. Thus percolator guided mobile robot mostly explores unknown regions rather than revisiting navigated sides. That type of navigation strategy increases the map information but decreases the localization precision. Furthermore, obtained maps of the explored areas also cannot be reliable due to the lower localization precision. For this reason, percolator can mislead mobile robot as a consequence of localization corruption. We proposed a hybrid solution to solve localization problem of the purely applied percolator exploration. In the frame of that hybrid solution, we have supported percolator’s localization lankness by entropy based switching mechanism.

3.3.2 General Architecture

In this system either the switch always favors the percolator guidance and we have the novel system introduced in section 3.2 which is the Percolator based exploration with Grid Based Fast Slam or the switch undergo its switching process with proposed guidance by Entropy based exploration and we have the novel system introduced in this section. The Entropy based Fast SLAM (modules 1 and 8 of Figure 8) classically chooses optimized trajectory from the given action set so as to explore unknown regions beyond the frontiers or exploit within the known regions for localization. Entropy based Fast SLAM realizes its decision making process by comparing entropy decrements for the selected action alternatives. In thesis work, the proposed action set consists of directional displacements between robot position and frontier grid centroids within the grid based SLAM map. On the other hand, the switching operator, firstly, determines the most effective action alternative for entropy minimization from a given proposed action set. Then, the selected action is investigated according to its effect on position and map entropy decrements. If selected action alternative causes more entropy change for map information then, switching favors the use of invasion percolator guidance instead of entropy based proposed action alternative that was restricted to the frontiers of obtained environment map. As a consequence of our approach, we have been able to increase exploration performance of information gain based Fast SLAM technique using invasion percolation guidance instead of classical entropy based technique. Our robot starts SLAM from a given initial direction (module 1 of Figure 9), but is periodically controlled by the percolation estimation of connected voids (module 7 of Figure 9). Module 7 of Figure 9 is the exact module 2 of Figure 5 and is the system already detailed in section 3.2. Module 1 has been given in section 2.2.1 of chapter 2. A proposed action set for entropy evaluations (module 9) consists of computed displacements by ray tracing to the frontier cells on obtained SLAM maps. Then Module 9 computes information gain values corresponding to those alternative actions. Module 10 finds the action alternative that causes maximum information gain. Our switching mechanisms detect if map precision or position precision is required by analyzing the selected action alternative offered by module 10. Switching mechanism decides to continue with selected action alternative if localization precision is required and switches to percolator exploration technique (module 7) if map information is dominant within the total uncertainty.

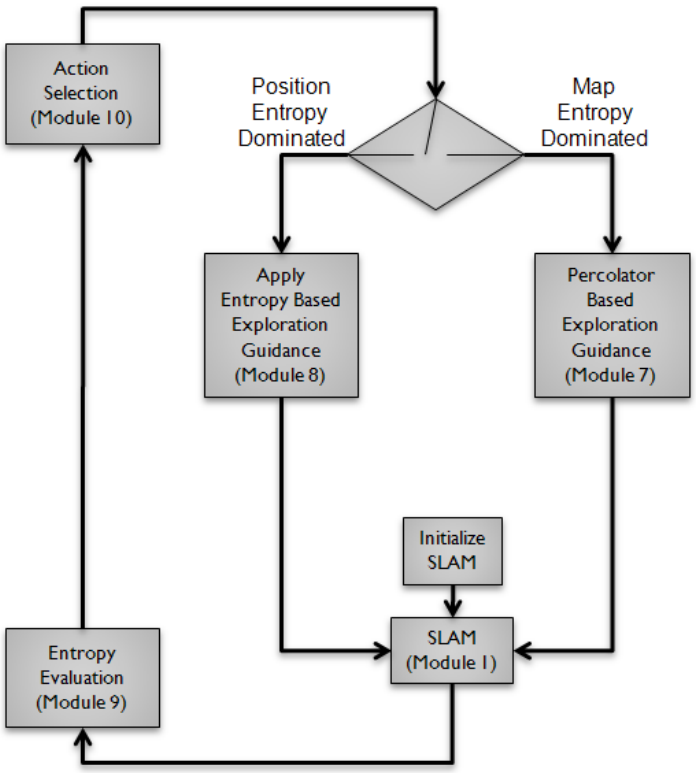

Figure 9 Percolation Enhanced Entropy Based exploration

Entropy based exploration technique runs under selected action set based on entropy minimization; our invasion percolator contributes in our enhanced approach to this active control for the mobile robot by providing active feedback to the classical slam and generating a predictive action into the unexplored region. The grid based Fast SLAM then works under this generated action and its outcome is evaluated in terms of entropy which them is given to the switch function for decision making of which action to take next. Thus our percolator guided grid based fast SLAM follows the orientation advised by the percolator (section 3.2) so as to reach larger probable connected voids rather than wandering without guidance among rubbles or getting entrapped with small isolated dead-end voids. But it is known that exploration actions are not solely adequate for mobile robot navigations. Exploitation actions are also necessary so as to protect its localization knowledge. Invasion Percolator (in section 3.2) mechanism provides exploration actions as a consequence of estimating connected voids beyond the frontiers of explored terrains. But invasion percolator cannot re localize mobile robot position preserving its localization precision. We have seen some dead-end crashes through the

percolator guided robot trajectory during our simulations since adequate position precision cannot be maintained. Thus we can collaborate with the entropy based exploration methodology at the utilization of percolator guidance whenever localization precision is required besides map precision. Our invasion percolator contributes to the entropy based exploration methodology (section 2.3) at the selection of exploration actions. Percolator enhanced entropy based exploration has action set that consists of exploitation actions for localizations and exploration action for map building precision based on connected voids. Hence we intend to use exploitation actions if the position entropy minimization is required (left hand after switch) while using exploration action for the map entropy minimization necessities based on percolator (right hand after switch). We set switching operation that drives the decision making process between exploitation and exploration actions. We can summarize our switching operation within the percolator enhanced entropy based exploration by the investigation methodology on the chosen proposed action. Percolator enhanced approach investigates the chosen action by entropy based exploration whether exploitation or exploration action is. If chosen action by the end of entropy minimization process is detected as exploitation action, then it is applied to mobile robot for localization. Otherwise, chosen action by the end of entropy minimization process is exploration action. In this case our percolator breaks into the process and determines new exploration action instead of proposed one by the end of entropy based exploration methodology. Thus we have more enhanced exploration action alternatives so as to find out connected voids estimated by our invasion percolator through the unknown terrain. But simultaneously, position awareness is provided for required situations. Our percolator enhanced entropy based exploration technique is outlined in Table 13 below:

Table 13 Percolator Enhanced Entropy Based Exploration

Algorithm of Percolator Enhanced Entropy Based Exploration (X_t)
$u_{optimized\_proposed\_action}$ = Entropy Based Exploration (X_t) ;
[$E_{position}$ E_{map}]=Recall Entropy Changes($u_{optimized\_proposed\_action}$);
IF ($E_{position} \geq E_{map}$) then
return $u_{optimized\_proposed\_action}$
Else IF ($E_{map} \geq E_{position}$) then
$u_{enhanced\_exploration\_action} = \text{Invasion\_Percolator\_Controller}(m_t^{[k]}, x_{t-1}^{[k]})$
return $u_{enhanced\_exploration\_action}$
End IF

In Table 13, entropy based exploration methodology (module 8) introduced in the mathematical background of chapter 2 section 2.3, is driven at the first step according to the current mobile robot state belief distribution X_t and gives optimized action $u_{enhanced\ exploration\ action}$ from the proposed action set. On the other hand, proposed action set constitutes of possible directional displacements to the frontier neighboring cells of mobile robot position. While entropy minimization values are computed according to the principles of section 2.3; position entropy and map entropy changes for the proposed action set are saved in memory. Entropy based exploration compares total entropy changes that consist of map and position entropy changes; then chooses action index that has the maximum total entropy change value among the proposed actions. Then we recall the position and map entropy changes of selected proposed action index by Recall Entropy Changes method in Table 13. Hence we can compare position entropy change $E_{position}$ and map entropy change E_{map} values so as to realize the switching operation between entropy based exploration actions and invasion percolator guided exploration actions. If percolator guidance is switched according to the map entropy minimization mode; then invasion percolator provides percolating exploration action $u_{enhanced\ exploration\ action}$ according to the particle maps $m_t^{[k]}$ and trajectories $x_{t-1}^{[k]}$.

CHAPTER 4

SIMULATION ENVIRONMENT

4. Simulation Environment

4.1 Overview to Simulation Environment

4.1.1 General Architecture

In order to test our novel methodology, we developed a simulation environment in MATLAB as a simulator of a mobile robot navigating within an unstructured area. We modeled a 2D search area where the robot is equipped with online sensors acquiring data modeled as ray tracing techniques. Similar to sensor measurements, the simulator also obtains odometry measurement from modeled robots movements which are generated based on time steps using robot coordinates on 2D search environment. In our experimental simulation runs, we also added a noise factor to the odometry measurements in order to analyze the capabilities of our probabilistic approach to noisy robot measurements. We have added noises at the given percentage of odometry motions. We added linear error with a given percentage e_L to odometry linear displacement at each time step and added angular error with a given percentage e_A to odometry angular displacement at each time step. We will discuss the effects of e_L and e_A odometry error percentages over SLAM methodologies by simulating them within our simulator. All computations results about mobile robot measurements and probabilistic localization and mapping algorithms are regularly recorded according to time steps and particles of the applied particle filter runs. In this chapter we will introduce the details of the simulator for which the general architecture is given in Figure 10. Within the scope of the given architecture in Figure 10, we constitute a generic GUI for different type of algorithm implementations. That GUI is shown in Figure 11 and its entries are initial simulation

specifications as is seen from Figure 10. Our simulator runs real time simulation with animated sensor readings and robot movements. A simulator environment is also initially given with GUI entries to the simulator to realize robot's simulation.

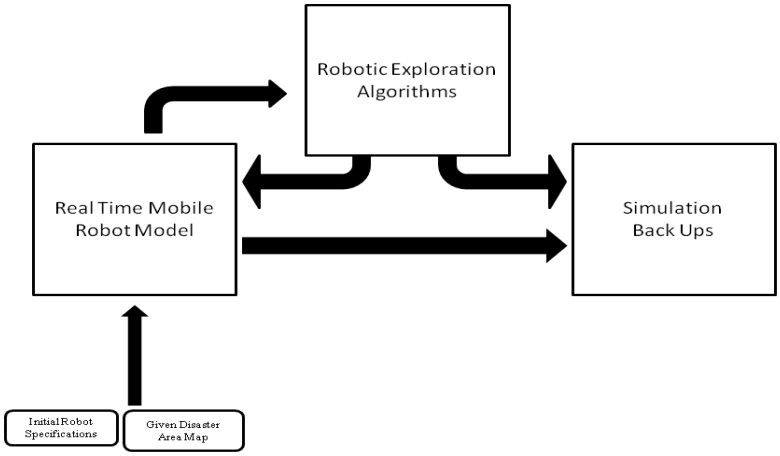

Figure 10: Simulator Environment Abstraction

Meanwhile real time mobile robot realizes a simulation environment with given SLAM parameters; robotic exploration algorithms perform their controls on the running robot simulations. As a consequence of that intervention, obtained results such as SLAM maps, particle resampling displacements, particle weight values, particle robot trajectories, particle percolator maps, entropy computations and their related action alternatives on SLAM maps, ray tracing values for particle filter measurement updates, obtained proximity sensor readings and mobile robot odometry measurements are saved to SimBackUp (Simulation Back Up as seen in Figure 10) structure. After simulation steps are completed, we can easily investigate and interpret those recorded simulation parameters according to their time steps and particle set specifications such as particle number, particle weight, re-sampling displacements. Our simulator analysis tools contain required instruments to plot related specifications of investigated parameters that listed above. We introduce those analysis instruments through the coming subsections.

4.2 Simulation Environment: User Interface Specifications

The mobile robot exploration simulator user interface consists of interactive text box groups and menus for user tailored test configuration arrangements. An outline of our

simulator user interface control module is shown in Figure 11 below. As is seen from Figure 11, text box groups are classified as Simulation Parameters, Motion Model Parameters, Mapping Parameters, Load Sim Back Up, Simulation Analyses Results, Robot Odometry Driver Step and Motion Model Tester.

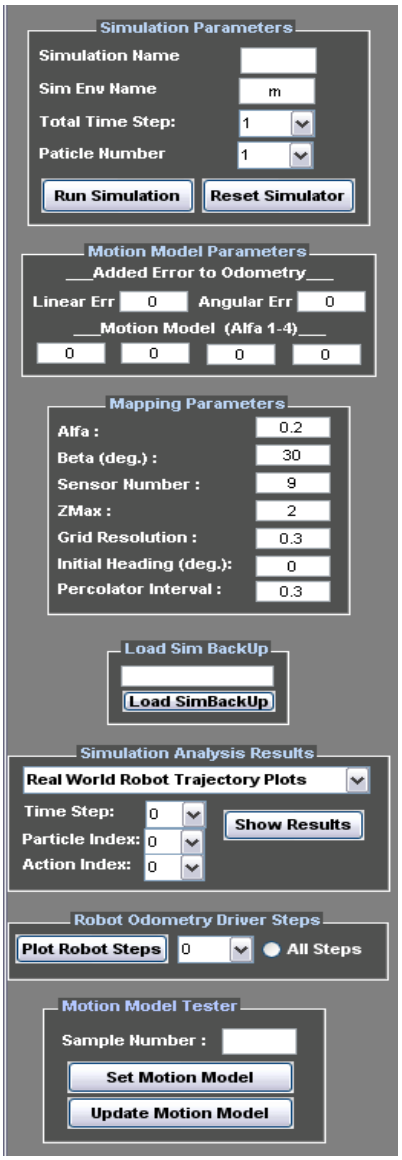

Figure 11 Our Simulator Control Module

Our simulations are configured with those experiment parameters. Those text box parameters are discussed in the next sub sections. We introduce Simulation Parameters, Motion Model Parameters, and Mapping Parameters in subsection 4.2.1, 4.2.2 and 4.2.3

respectively. Simulator analysis instruments given with Load Sim Back Up, Robot Odometry Driver Step, Motion Model Tester and Simulation Analysis Results groups in Figure 11 are handled in subsections 4.2.4, 4.2.5, 4.2.6 and 4.2.7 respectively. On the other hand, we also introduce simulator menu at the subsection 4.3.Interactive text boxes on the user interface in Figure 11 are useful in determining the mobile robot specifications and the parameters of the exploration algorithm. As is seen from Figure 11, there exist interactive text boxes such as Simulation Parameters, Motion Model Parameters, and Mapping Parameters so as to determine simulation's initial parameters. We have already explained the details of text box groups related with initial simulator parameters in section. We also introduce text box groups about simulator analysis instruments. We outline those analysis box groups as Load Sim Back Up, Robot Odometry Driver Steps, Motion Model Tester and Simulation Analysis Combo Box.

4.2.1 Simulation Parameters Text Box Group

First of all, we assign a name to our experiment in order to record all simulation results with chosen experiment specifications with that name. We can also choose our simulation environment from "Sim Env Name" text box. Simulation environments are prepared before running the experiment and recorded with a special name to recall its parameters using that special given name. On the other hand; we can determine the total time steps of the experiments and the particle filter particle numbers with "Total Time Step" and "Particle Number" text boxes.

4.2.2 Motion Model Parameters Text Box Group

Mobile robots that are used for search and rescue operations in real world have noisy odometry measurements. Angular and linear errors can be encountered within their odometry measurements. In this part of simulator text box groups, we determine the generated noise specifications for our robot odometry within simulator. "Linear Err" and "Angular Err" text boxes determine the linear and angular cumulative error percentages of odometry measurements. On the other hand; "Alfa 1-4" text boxes correspond to the Alfa parameters of classical odometry motion model to estimate noisy odometry data.

4.2.3 Mapping Parameters Text Box Group

Mapping text box parameters are used for grid based SLAM mapping using Inverse Measurement Model technique. As is seen from Figure 11, mapping parameters consist

of Alfa (possible width of the obstacle within grid based area.), Beta (angular width of each sonar sensor), and Sensor Number, ZMax (maximum sonar range), Grid Resolution (grid cell sizes for grid based mapping), Initial Heading (mobile robot initial bearing) and Percolator Interval (step size for percolator estimator).

4.2.4 Load SimBackUp Text Box Group

Our simulator saves all our experiment results and specifications with an assigned name in "Simulation Name" text box within Simulation Parameters Text Box Group. So we can easily analyze any past experimental results according to different criteria such as particle and time steps. After the desired experiment has been selected from the text box embedded in Load SimBackUp text box group, the experiment can be run and all its records from the run can be loaded in Load SimBackUp text box for being analyzed later.

4.2.5 Robot Odometry Driver Steps Text Box Group

In this text box group; we investigate robot odometry motion steps within the simulation environment. A previously saved simulation run can be reloaded from load SimBackUp text box, and the selected experimental motion steps can be shown according to time steps.

4.2.6 Motion Model Tester Text Box Group

Percolator based and percolator enhanced entropy based SLAM use particle filter in order to disambiguate mobile robot localization. During SLAM, alternative robot positions are generated according to the motion model explained in earlier sections for each particle at each odometry command. To understand the effects of motion model we have to compare location from the noisy odometry and those generated by the motion model. "Sample Number" text box determines the number of samples generated by the motion model. On the other hand, the motion model tester assigns the odometry motion model specifications of the motion model text box group.

4.2.7 Simulation Analysis Combo Box

Simulation results are saved within the Sim Back Up mat files in MATLAB environment that consist of configuration and performance parameters of realized experiments. Configuration parameters are explained in sections 4.2.1,4.2.2,4.2.3. The analysis of each experiment should be done according to its particle specifications and time steps. Hence

we develop suitable simulation analysis tools in our simulator in accordance with particle parameters and time steps. Simulation analysis issues such as True Robot Trajectory and Position Uncertainty Analysis, Map Coverage Analysis, Motion Model Analysis, Particle Memory Analysis, Proximity Sensor Analysis, Ray Tracing Analysis, SLAM Map Analysis, Percolator Map Analysis, and Proposed Action Set Analysis are represented by selecting Simulation Analysis Results combo box options. We select analysis type from a combo box to investigate recorded experiment results as seen from Figure 12.

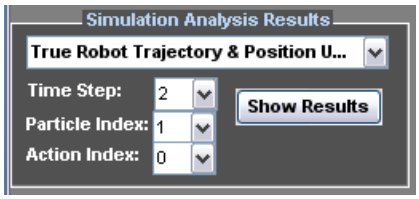

Figure 12 Simulation Analysis Results Part

4.2.7.1 Robot Trajectory & Position Uncertainty Analysis Option

We can represent mobile robot trajectory through the simulation environment matrix and position uncertainty according to time steps as is seen in Figure 13.

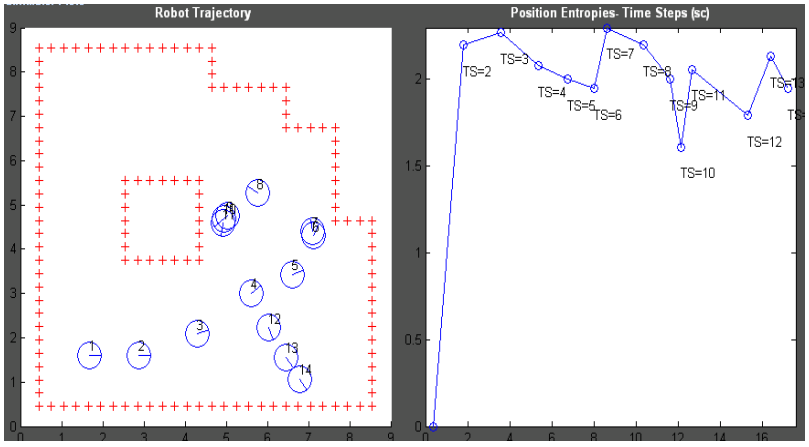

Figure 13 : Robot Trajectory (left side) and Position Entropies (right side)

Robot odometry measurements have noise and their localization errors are corrected by particle filter based algorithms. But we can clearly see true positions through the

simulation environment matrix with labeled time steps. On the other hand; we can investigate position uncertainties according to those mobile robot positions. We have utilized MATLAB plot tools to obtain true robot position trajectory in Figure 13 (left) and position entropy change levels with respect to time in Figure 13 (right). We record the robot position values through the simulation within the SimBackUp which is a mat file that is one of the file types of MATLAB. We also record computed entropy levels with that SimBackUp mat file through the simulation running. Red marked points correspond to the plotted values of initially given environment matrix to model simulation environment. To obtain environment matrix, we firstly draw our simulation environment in two colored gray scale bitmap. We paint obstacles black colored and voids white colored using a paint program. Then we track over those two colored maps according to their bit values representing colors. We take the perimeter bits of the obstacles on the map represented with black colors on two colored bitmaps so as to constitute the simulation environment matrix that indicates the obstacles of the simulation environment. We can see those perimeter points that indicate obstacles, extracted from the bitmap in Figure 13 (right).

4.2.7.2 Map Coverage Analysis Option

We can analyze coverage of explored area based on utilized exploration technique at each time step using map coverage option within simulation analysis results part. Map coverage with respect to time steps and related robot positions can be seen with this analysis option as for which an example is seen in Figure 14.

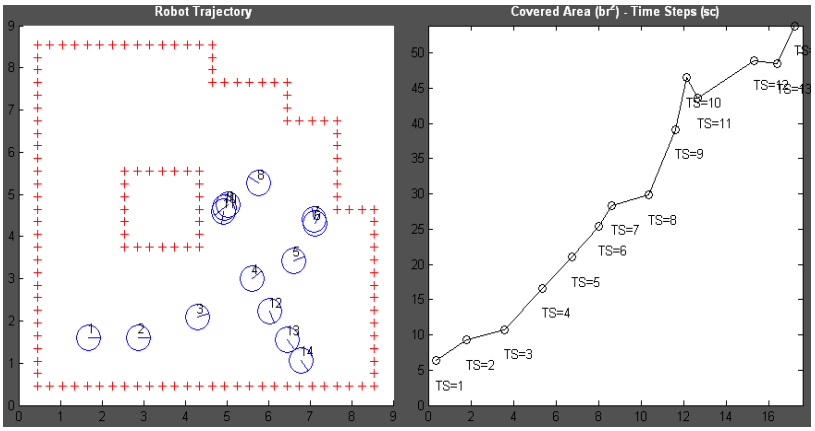

Figure 14 : True Robot Positions and Map Coverage with respect to Time Steps

We have explained how the environment matrix and simulated environment represented is shown in Figure 14 (left) in section 4.2.7.1. On the other hand, we record coverage level values according to each time step through the simulation runs. We obtained coverage levels by counting the grid cell numbers used in SLAM map and then multiplying that number of void grid cells with grid cell area that are used in that SLAM map. We reach a unit area of grid cell by computing the square of the grid cell size. Therefore after we obtained those coverage values according to SLAM maps, we then record those values in SimBackUp mat file. To analyze coverage levels with respect to time steps, we recall SimBackUp mat file and extract the value of coverage levels according their time steps. Then we trivially plot those coverage levels corresponding to time steps using MATLAB plot function as seen in Figure 14 (right).

4.2.7.3 Motion Model Analysis Option

Since robot odometry measurements are noisy; we have to use motion model in order to generate samples for possible localizations. Particle filter assigns those generated possible localizations with respect to ray casting correction on SLAM maps. We can see the effect of our motion model on noisy odometry measurement in Figure 15.

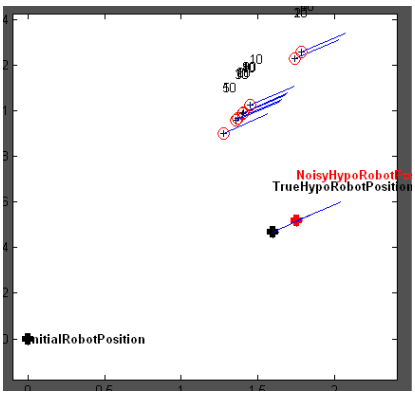

Figure 15: Noisy, Noise-Free Odometry Measurements and Generated Motion Model Samples

As is seen from Figure 15; red and black colored markers represent true and noisy robot positions while circled positions correspond to the generated localization samples according to previous noisy robot position. There exists linear error within the odometry measurements in Figure 15; that can be seen from the position difference of red and black colored markers with respect to initial robot position marker. On the other hand; the initial robot position marker is always placed at the origin as an offset to the noisy robot position and true robot position markers so as to represent noisy and true odometry

measurements respectively. We generate Figure 15 using MATLAB plot tools which plot our given sample positions, true odometry positions and noisy odometry position coordinates with 2D representation. We record the particle sample positions proposed by motion model, true odometry positions obtained by computed guidance values from exploration methods and noisy odometry positions obtained by adding initially defined error values within the SimBackUp mat file which is a type of file in MATLAB. Since we recorded those values with respect to their time steps and particle number values, we can easily figure noisy odometry position, true odometry position and generated particle sample positions with extract from SimBackUp and show on 2D frame using MATLAB plot function.

4.2.7.4 Particle Memory Analysis Option

This simulation analysis option represents the particle weights at each time step and particle situations through the resampling processes on a notepad file as seen in Figure 16

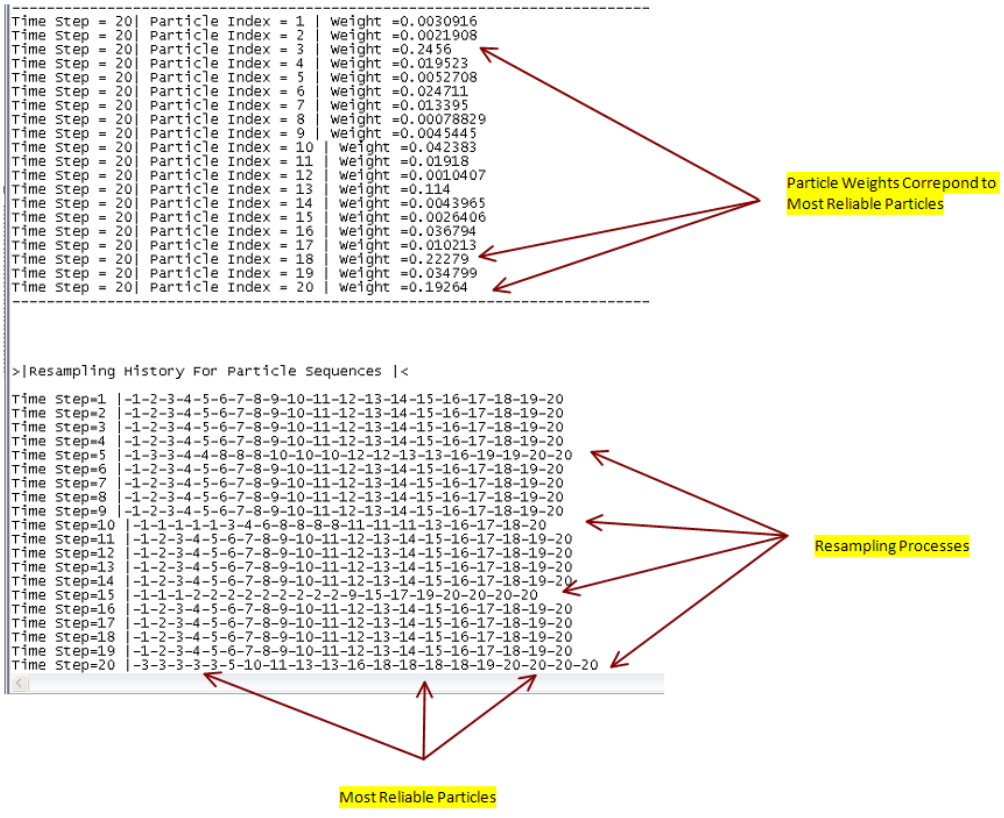

Figure 16: Particle Weights (left side) and Re-sampling Situations (right side)

At the bottom of the particle weight history; the particle set resampling process can be is seen at each time step according to saved SimBackUp records during the simulation.

4.2.7.5 Proximity Sensor Analysis Option

Proximity sensor scans are modeled with ray tracing technique on simulation environment matrix. There exist three types of ray tracing operation between mobile robot position and obstacles within the simulation environment as left side tracing, inner side tracing and right side tracing according to the mobile robot sensor direction. So similar to the real proximity sensors; nearest ray tracing corresponds to sonar sensor measurements. On the other hand; there exist a number of sonar sensors determined by the Simulation Parameters text box.

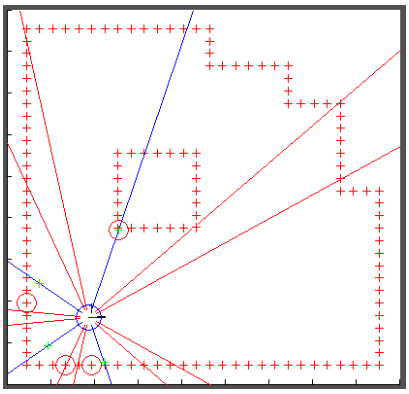

Figure 17: Sonar Sensor Measurements Representation on Simulation Environment Matrix

Figure 17; represent sonar sensor directions colored in blue while red colored lines represent the boundary of sonar waves. On the other hand; red colored circles represents nearest obstacles that sonar sensors read. In MATLAB red colored markers on Figure 17 are saved within the environment matrix which explained in section 4.2.7.1. According to extracted environment matrix from SimBackUp at each time step, we compute ray tracing values so as to simulate proximity measurements within our experiments. We utilize ray tracing to red marked obstacle points within the sonar cones that represented with red colored lines in Figure 17. We compute the minimum ray tracing value which is represented by blue colored line in Figure 17. Red circled places on Figure 17 correspond to the sonar reading points as consequence of the ray tracing. We use line function of MATLAB to plot those ray tracings on the 2D plot diagram as seen in Figure 17.

4.2.7.6 SLAM Map Analysis Option

Simultaneous localization and mapping technique is used for both entropy and percolator based exploration methodologies. Our mobile robot uses inverse measurement model in order to construct grid based map from sonar sensor measurements. Those SLAM maps are used for localization and guidance for exploration algorithm in our simulator. Our simulator represents SLAM map that are bounded with sonar measurements and also represents the totally accumulated SLAM maps at each time step. SLAM map at each time step. Bounded SLAM maps are used for localization and guidance instead of total SLAM maps so as to decrease computation burden.

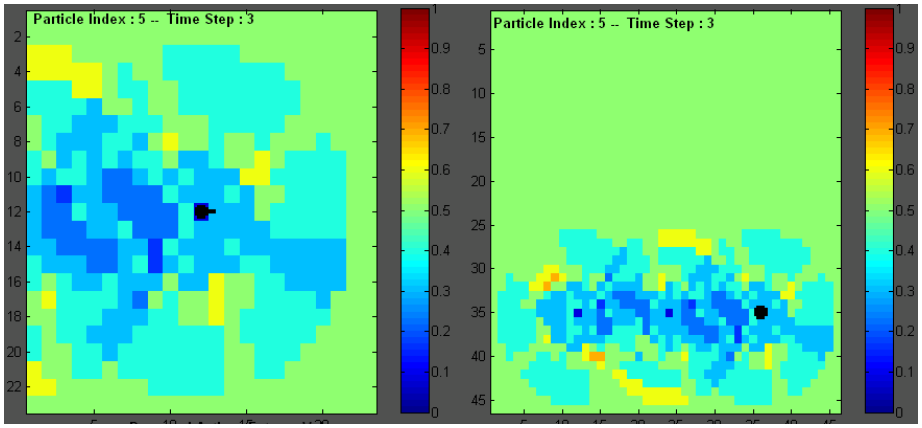

Figure 18: Bounded (left side) and Total SLAM (right side) Maps

As is seen from Figure 18, bounded and total accumulated SLAM maps are represented according to particle index and time step values. While regions bearing a color near to blue represent possible voids; regions colored by colors near to red represent possible obstacles in Figure 18. The green colored regions represent unexplored or uncertain parts in SLAM maps. We utilize imagesc function of MATLAB to represent those SLAM maps in Figure 18. MATLAB imagesc function converts two dimensional matrices into colored image maps as seen in Figure 18. We recorded SLAM map occupation probabilities as a 2D matrix which corresponds to total or bounded SLAM maps within the SimBackUp mat file according to each time step and each particle individually. Then we recall those occupation probability matrices from the SimBackUp mat file according to the desired time step and particle index and then, those occupation probabilities are converted to image maps as seen in Figure 18 by MATLAB imagesc function. Converted image maps consist of grid cells that are placed on image map corresponding to elements of occupation probability matrix. Grid cells are placed on the image map according to its

corresponding row and column element indices while their color is assigned according to the value of occupation probability matrix value. Their colors are assigned according to the color scale between 0 and 1. Occupation probabilities correspond to the existence probability of obstacle within any grid cell. We can see those color scales easily at the right sides of image maps representing SLAM maps in Figure 18.According to that color scale, occupation probability matrix elements that are less than 0.5 are represented by blue colored grid cells while other matrix elements that are greater than 0.5 are represented by colors which near to red. Green colored grid cells are also assigned to occupation probability matrix elements that have 0.5 occupation probability, namely uncertain regions.

4.2.7.7 Ray Tracing Analysis Option

Since mobile robot odometry measurements are noisy, our mobile robot needs to correct its localization according to obtained SLAM maps. This correction step take place within the particle filter based SLAM technique by assigning weights to the generated alternative localization samples. This correction step consists of ray tracing operations between each particle's robot position and possible SLAM map obstacles. Then those ray casting values are compared to sonar sensor measurements. We can see the comparison in Figure 19.

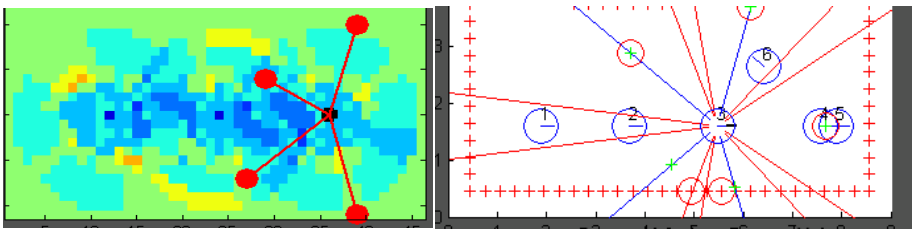

Figure 19: SLAM Map (left side) and Sonar Measurement (right side)

As is seen from Figure 19, ray tracing operation can be seen on SLAM map corresponding to the sonar sensor measurements on third step of mobile robot trajectory. It is seen that there is no large amount of mobile robot localization error in Figure 19. We have explained how we generated SLAM maps in section 4.2.7.6 Figure 19 (left) and sonar measurement in section 4.2.7.5 in Figure 19 (right). We utilize ray tracing operation on SLAM map as seen with red colored lines and red points in Figure 19 (left). We realize ray tracing operation in a similar way with proximity measurement representation as seen in Figure 19 (right) as explained in section 4.2.7.5. But this time

for SLAM map ray tracing, we use the coordinate values of grid cells on image map of SLAM map representation. In a similar way to measurement computation in section 4.2.7.6, we constitute cones and select minimum obstacle grid cell coordinate to find out ray tracing distance. We have shown those ray tracing distances with red colored lines in Figure 19 using MATLAB line function which takes two coordinate and then combine them by a line.

4.2.7.8 Percolator Map Analysis Option

Since each particle proposes an alternative to the SLAM map and localization correction; their percolator guidance as a continuity of explored regions has different accuracy levels according to their particle weights. So each particle has individual void prediction that has been derived from its SLAM map using our invasion percolator algorithm in section 3.2 in chapter 3. We can investigate particle percolator map guidance of each particle from Particle Percolator Map option of simulation analysis results part as is seen in Figure 20.

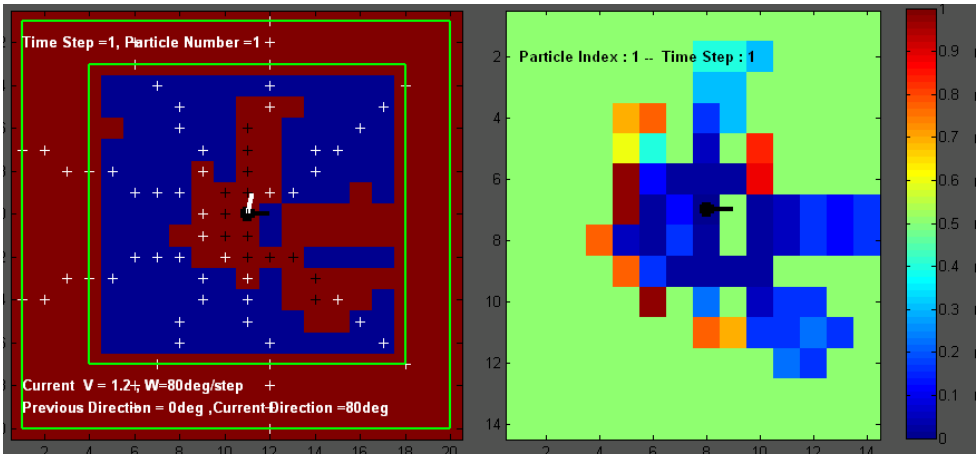

Figure 20: Particle Percolator Map and Bounded SLAM Map

We utilize MATLAB imagesc function which converts 2D matrix to an image map so as to obtain percolator maps as seen in Figure 20 (left). We have explained insection 4.2.7.6 how that imagesc function converts 2D occupation probability matrix to an image map for SLAM map representation as seen in Figure 20 (right). For percolator map in Figure 20, we convert 2D spanning probability matrix to image map. Spanning probability matrix is explained in chapter 3 section 3.2.2. But we can state that Spanning Probability Matrix has 0 or 1 valued elements that correspond to available or unavailable regions for robot motion. So MATLAB imagesc function assigned 0 valued elements to red colored

grid cells while 1 valued element to blue colored as seen from Figure 20. Black and white arrows on Figure 20 correspond to the current and next motion directions that are represented with MATLAB line function. On the other hand, white and black colored markers on the Figure 20 correspond respectively to the available or unavailable paths.

4.2.7.9 Proposed Actions Analysis Option

We can investigate proposed action alternatives that are used for guidance determination during entropy based exploration experiments. Entropy based exploration algorithm in our simulator determines the most suitable action alternative in order to increase information gain. Information gain is consisting of decrement of robot position and map uncertainties. We can measure information gain using Shannon entropy. So proposed action option can represent the information gain amount related with map and robot position for selected proposed action alternative as is seen Figure 21.

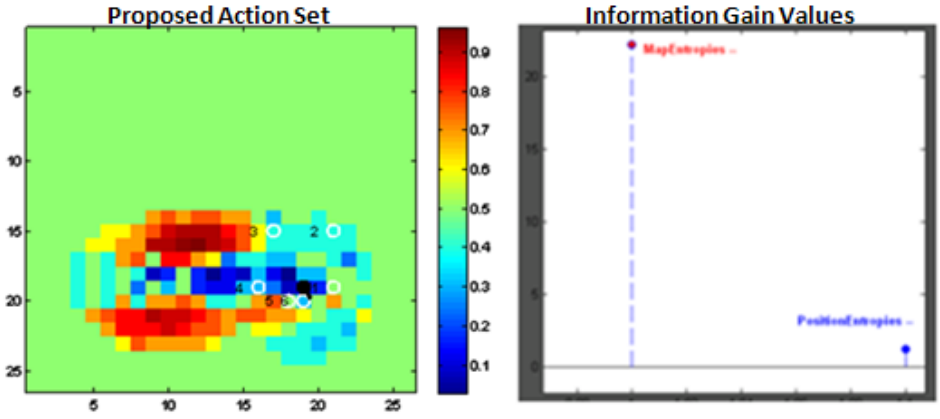

Figure 21 : Proposed Action Displacements Represented with White Colored Circles on SLAM Map (left) and Proposed Action Information Gain Values with respect to Map and Position Uncertainties (right).

As is seen from Figure 21, we obtain map and position information gain values for each proposed action alternatives. We compute those action alternatives according to obtained proximity measurements and then compute their entropy change values as is seen from (right) Figure 21. We have generated proposed action set representation in Figure 21(left) using imagesc and plot functions that are utilized for SLAM map generation and proposed action goal points on that SLAM map. We record the robot position and its bearing according to each time step and particle within the SimBackUp mat file. On the other hand, we also record the proposed action goal points that are generated by sonar

measurement within the SimBackUp mat file. Using imagesc MATLAB function as explained we firstly generate SLAM map for chosen particle and time step to analyze as explained in section 4.2.7.6, than we represent robot position and bearing using plot and line MATLAB function that determine its position and bearing respectively as seen in bold black point with its black arrow in Figure 21 (left). Over those SLAM map and robot representations, we plot odometry goal points using MATLAB plot function on Figure 21(left). On the other hand, we also record computed position and map information gain values for each action alternative and each time step within the SimBackUp mat file. So we use MATLAB stem function to plot those positions and map information gain values in Figure 21 (right). We plot map information gain levels with red colored points and position information levels with blue colored points in Figure 21(right)

4.3 Simulator Menu Options

Simulator menu options consist of Simulator Configuration Menu, Experiment Results Menu, Select Simulation Type Menu and Simulation Environment Menu. Those menu options are used to represent experiment results in previously defined file formats or save their initial run configuration parameters so as to repeat another experiment with same parameters.

4.3.1 Simulator Configurations Menu

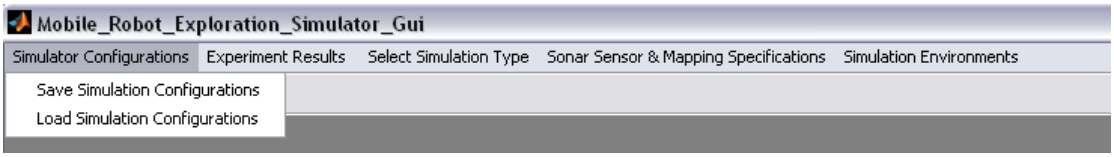

Figure 22 : Simulator Configurations Menu

Each experiment has specified experimental configurations related with robot and simulator environment which consist of mapping parameters and simulation parameters text box groups entries as is seen in Figure 11 in section 4.2.We save those simulation parameters as test configuration structure that is assigned by predefined configuration name. Thus those simulation parameters placed in user text boxes in Figure 11 are reloaded by selecting previously saved simulation configuration file from Load Simulator Configuration menu as is seen from Figure 22. Using memorized parameters of previous experiments, in our simulator; we can simulate experiments with changing specified

parameters such as mapping or odometry motion model parameters while others are fixed. Hence we can easily compare percolator based exploration, entropy based exploration and percolator enhanced entropy based exploration methodologies using similar sensitive parameters and initial conditions on experiment scenarios.

4.3.1.1.1 Experiment Results Menu

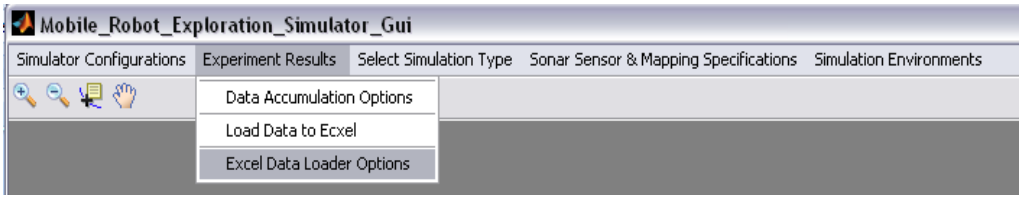

Figure 23 : Experiment Results Menu

Our simulator keeps configuration parameters which are simulation name, total time step, particle number, simulation environment, alfa obstacle width, cone width, sensor number, grid cell size interval, initial robot heading, motion model parameters , injected noise parameters and their experimental performance parameters which are the total coverage area, position entropy and elapsed time. Those configuration and performance parameters can be displayed within a Microsoft Excel file as is seen in Figure 24.

Simulation Name	Environment Na	TotalTS	ParticleNumber	Alfa	ConeInterval	GridCellSize	LinearErr	AngularErr	ZMax	SensorNumber	TotalCoverage	PositionEntropy	ElapsedTime
E_22	m2	14	10	0,4	60	0,2	0,1	0	2	5	34,8	1,098612289	296,4938428
E_10	m2	14	10	0,4	60	0,3	0,1	0	2	5	42,75	1,098612289	99,68051413
E_12	m2	14	10	0,7	60	0,5	0,1	0	2	5	41	2,197224577	44,44174242
E_15	m2	14	10	1	60	0,6	0,1	0	2	5	31,32	2,079441542	45,62188528
E_21	m2	14	10	1	60	0,7	0,1	0	2	5	29,89	1,945910149	37,93283083
E_17	m2	14	10	2	60	1	0,1	0	2	5	14	1,609437912	26,7296153

Figure 24 : Configuration and Performance Parameters Microsoft Excel File Representation

In order to display configuration and parameters as an Excel file, Excel Data Loader Options menu is used. Excel Data Loader Options menu can be selected from pull down menu on the Experiment Results menu in Figure 23.After this menu is selected, Excel Data Loader Options GUI opens up as seen in Figure 25.We can select the experiment type and its configurations and performance parameter options from the opened up GUI in Figure 25. In this specific GUI of Figure 25 ; motion model parameters have not been selected to thus cannot be seen .We save those performance parameters with their test configuration parameters within our simulator buffer using Data Accumulation Options sub-menu in Figure 23 and using popup menu in Figure 25. Data Accumulation Options

menu works as a toggle switch that open up or reset the parameter buffer. “Load Data To Excel” sub menu loads all accumulated experimental configuration and performance parameters to excel file. In addition to these parameter buffering operations, all parameters related with different experiments types such as Percolator Based Fast SLAM, Entropy Based Fast SLAM.etc are saved in the form of different Excel files

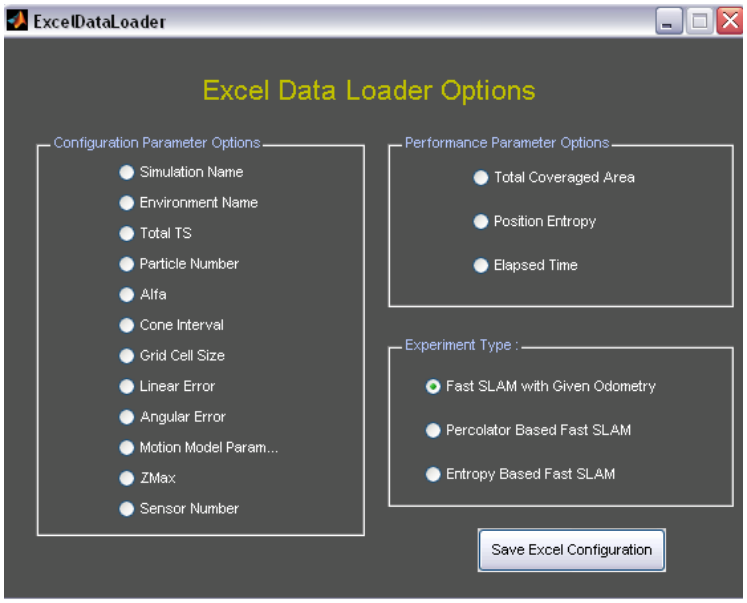

Figure 25 : Excel Data Loader Options GUI

4.3.1.1.2 Select Simulation Type Menu

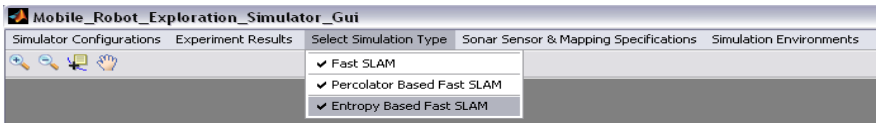

Figure 26 : Select Simulation Type Menu

We can choose four types of simulator modes which consist of Fast SLAM Simulation mode, Purely Applied Percolator Based Exploration Simulation mode, Entropy Based Exploration model and Percolator Enhanced Entropy Based Exploration Simulation mode is seen on Figure 26. Mobile robot exploration algorithms running behind the GUI and related analysis tools with chosen exploration algorithms are determined with this Select Simulation Type menu of Figure 26

4.3.1.1.3 Simulation Environments Menu

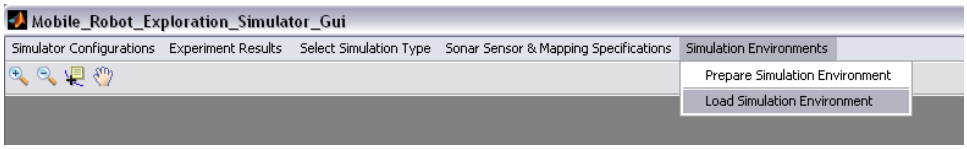

Figure 27 : Simulation Environments Menu

We can prepare and load simulation environments using Simulation Environments menu as is seen in Figure 27.Our simulator converts given bitmaps to the 2 dimensional simulator environment matrices in order to model disaster environments for search and rescue mobile robots. We can see a sample simulation environment matrix that converted from a given bitmap in Figure 28.

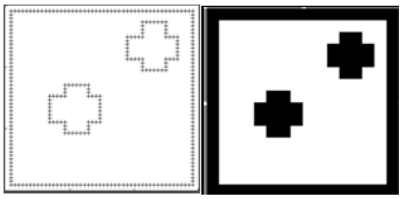

Figure 28 : Simulation Environment Matrix (left side) and Related Bitmap (right side)

Our simulator realizes mobile robot movements and its sonar sensor measurements within the simulator environment matrix in Figure 28.We can convert given bitmaps to the simulation environment matrices with “Prepare Simulation Environment” menu and load those simulation environment matrices within a mat file structure to the simulation environment archives of our simulator. After these steps; we can choose any loaded simulation environment matrix from Sim Env Name text box. While disaster environments are represented with the simulation environment matrices; mobile robots are represented as a circle with a specified radius value and a direction arrow as seen in Figure 29.

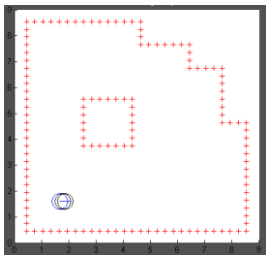

Figure 29 : Mobile Robot Representation with Given Radius and Its Direction Arrow

CHAPTER 5

EXPERIMENT RESULTS

5. Experiment Results

5.1 Introduction to Experiment Results

Our objective in this thesis is to navigate through connected voids exploring highly unstructured environment without getting trapped in dead-ends before reaching enough coverage of the area in question. Since disaster regions have highly irregular labyrinth conditions; mobile robot exploration of the area is a complicated mission that includes significant challenges in terms of correct localization about its position awareness and handling map uncertainties over time and space. Classical approaches that handle precision in localization and map uncertainties are mainly entropy based approaches. Thus, we aim not only at comparing our percolator based exploration methods of sections 3.2 and 3.3 with entropy based technique by their time and coverage performances using our simulator but also integrate percolator with entropy based techniques. We run our comparison on robot localization and map coverage progressions between entropy based and percolator based exploration methodologies depending upon time and coverage specifications. From experimental results we see that our invasion percolator methodology that reach connected voids through the unknown region beyond the explored frontiers promises better coverage performance at the initial steps while the entropy based methodology maintains more reliable robot trajectory through the exploration of unknown regions. Entropy based exploration methodology owes this reliable trajectory to better localization during the exploration process. SLAM maps obtained from proximity sensor readings are faulty due to deficient localization information. Since percolator based exploration guidance focusing on reaching unknown

regions via connected voids reduces that localization awareness, percolator guidance will be adversely affected by localization uncertainties. However we will see from our experimental results, that our proposed percolator enhanced entropy based methodology introduced in section 3.3 is able to compensate the localization deficiencies encountered during the exploration process that is the main disadvantage of in percolator guidance methodology applied alone in section 3.2.In this chapter ,we run comparisons between percolator based exploration ,percolator enhanced entropy based exploration and classical entropy based exploration techniques in order to better enhance the critical contribution of a percolator as an estimator in the performance of an active SLAM performance. Based on this comparison, this chapter will unveil the value of contribution we made in this thesis work. We also conduct in this chapter a sensitivity analysis of performance to its parameters, since the reliability of the exploration guidance also depends upon mapping parameters mentioned in section 2.1.2.3, this guidance being obtained according to SLAM maps, Proposed action set given for entropy based exploration in section 2.2 and percolator probability computations in section 3.3 are evaluated SLAM map parameters. Therefore we evaluated performance sensitivity through experiment trials based on different grid cell sizes; different possible obstacle width α and different sensor beam width β and mapping parameters in section 2.1.2.3.We also evaluated sensitivity to noise in the odometry measurements. According to section 2.1.2.1; we saw that noise effects on odometry measurements and depend on 4 parameters $\alpha_1, \alpha_2, \alpha_3 \, and \, \alpha_4$ according to the error rates on the odometry measurements of linear and angular robot movements. We run our comparisons considering the same initial conditions.

5.2 Introduction to Rating Framework for Simulation Results

Fast SLAM performance responses are in this section analyzed according to injected noise types and amounts, and to odometry measurement data. How Particle filter implementation compensates noisy odometry measurements with respect to obtained proximity sensor measurements has been investigated in this section. And we discuss here particle filter implementation as passive Fast SLAM utilization with given odometry commands. But exploration experiment discussions in this section are related with how Fast SLAM odometry commands are derived according to obtained SLAM maps in the previous time steps. Hence we introduce our fundamental approach to sensitivity analysis of exploration experiments. Rating criteria of compared exploration methodologies through the sensitivity analyses are determined according to their coverage performance

and accuracy levels of their map and position information. In Figure 30, we have outlined the philosophy of the exploration performance analysis according to given and outcome parameters. Coverage of unknown environment depends upon implemented exploration techniques, particle SLAM maps and particle hypothetical robot position used on exploration guidance computations.

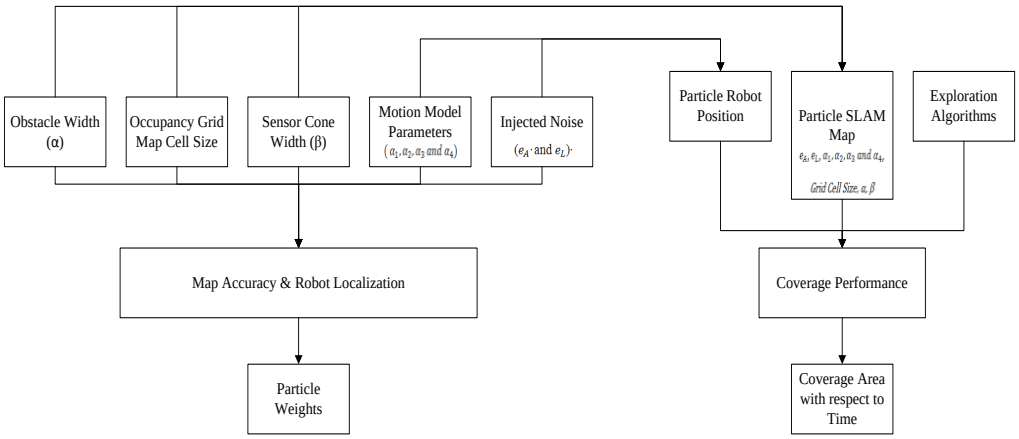

Figure 30 Exploration Experiments Analysis Philosophy

We compare coverage performances of exploration techniques by matching their elapsed time for their coverage. Exploration speed also determines the coverage performance. Map Accuracy and Robot Localization are other criteria of performance analysis. Map accuracy and Robot Localization accuracies are associated with obstacle width α, occupancy grid map cell size and sensor cone width β (section 2.2.1.4) and parameters required for localization consisting of motion model parameters $\alpha_1, \alpha_2, \alpha_3 \, and \, \alpha_4$. In addition to those parameters, injected noise parameters which are e_A and e_L also critical in performance while analyzing exploration experiment results due to map and localization accuracies, particle weights specifications such as particle set variance, most reliable particle weight value are considered. We also compare in this chapter sensitivity factors of required parameters such as possible obstacle width, sensor cone width, occupancy grid map cell size and motion model parameters to SLAM map accuracy. Injected noise parameters e_A and e_L are used to assess sensitivity of implemented exploration algorithm against added noises to odometry measurements. For sensitivity analysis of map and localization accuracies or coverage performance against noisy measurements, we have fixed the mapping and motion model parameters while changing linear e_L or angular e_A cumulative error rates to observe the robustness of driven exploration algorithm to injected noise disturbance levels. For ease of comparative analysis we developed a coverage performance summary table as the one shown for

coverage performance in Table 14 and for map and localization accuracies in Table 15. All comparisons will be done by comparing data in those standardized tables.

Table 14 Data Table for Coverage Sensitivity Analysis

Initial Simulation Parameters									
Total Time Step									
Particle Number									
Sensor Number									
Maximum Proximity Sensor Range									
Sensitivity Analysis of Coverage Performance									
Experiment Name	**α Possible Obstacle Width**	**Occupancy Grid Map Cell Size**	**Sensor Cone Width**	**Motion Model Parameters**				**Exploration Methodology**	**Coverage Area(br^2)**
				α_1	**α_2**	**α_3**	**α_4**		

Table 15 Data Table for Map Accuracy and Localization Sensitivity Analysis

Initial Simulation Parameters												
Total Time Step												
Particle Number												
Sensor Number												
Maximum Proximity Sensor Range												
Sensitivity Analysis of Map Accuracy and Localization Performance												
Experiment Name	**α Possible Obstacle Width**	**Occupancy Grid Map Cell Size**	**Sensor Cone Width**	**Motion Model Parameters**				**Injected Noise Parameter**		**Exploration Methodology**	**Most Reliable Particle Weight Value**	**Particle Set Variance(Position Entropy Level)**
				α_1	**α_2**	**α_3**	**α_4**	**e_L**	**e_A**			

Particle set variance corresponds to the reliability of particle filter as an estimator through the simulated experiment. On the other hand, the most reliable particle means particle that has the greatest particle weight value within the evaluated particle set through simulated experiment. Sensitivity analyses outcomes as particle weights and particle set variance are represented in Table 15. Similar to the data table template in Table 14, experiments are exposed to odometry measurement noise expressed with injected linear

$e_L$and angular noise e_A parameters. Since we have completed defining our experiment rating framework on simulation results using sensitivity and robustness data table templates, we can compare our proposed exploration algorithms according to that experiment rating framework. We will run exploration simulations for entropy based exploration (section 5.4), purely percolator based exploration (section 5.5) and percolator enhanced entropy based exploration (section 5.6) methodologies. While performing those experimental runs, we will extract sensitivity and robustness analyses data as is defined in data table templates represented in Table 14 and Table 15.Finally, we will compare the significant sensitivity and robustness analyses results of performed experiments at the end of chapter 5. Before starting the exploration experiments, we will investigate sensitivity analysis on Fast SLAM implementation as explained in section 5.3. Coverage performance is meaningless for Fast SLAM implementation since we have provided odometry command before simulation. Hence we realize sensitivity analyses for only map and localization accuracy in the experiments of section 5.3 and then, in subsequent sections, we will show the effects of these sensitivities in the performance of our two novel methodologies (section 5.5 through 5.7)

5.3 Fast SLAM Performance Analysis

As we recall from section 5.3.1 sensitivity parameters consist of obstacle width α; occupancy grid map cell size, sensor cone width; motion model parameters. From those sensitivity parameters, obstacle width α, occupancy grid map cell size, sensor cone width are related with SLAM mapping while motion model parameters are related with localization process. As we explained in section 2.1.2, obstacle width; occupancy grid map cell size, sensor cone width parameters define the scanning of proximity sensor on SLAM map in order to assign occupation probability values to the grid cells of SLAM map. We illustrate that relationship between those sensitivity parameters and mapping process. As is seen from the Figure 31, occupancy grid map cell size corresponds to the SLAM map representation grid cell sizes and especially, length of the cell edges. On the other hand, sensor cone width corresponds to the cone angle shown in Figure 31, while obstacle width correspond to the obstacle cells that are indicated by the vicinity of ray traced proximity sensor point on the SLAM map.

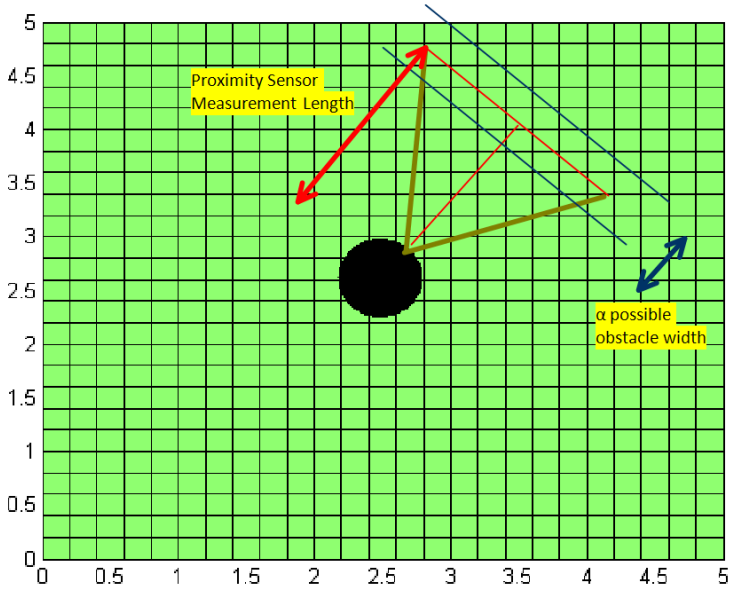

Figure 31 : Occupancy Grid Map for SLAM Map Representation with 0.2 Sized Grid Cell Edges

Inverse sensor measurement model decides on the grid cell occupation probabilities if they drop inside of the cone. As explained in section 2.2.1.4, grid cells whose centroids drop in the vicinity of obstacle region will be assigned occupation probability greater than 0.5 values to indicate those cells as probable obstacle cells. Cells of regions until obstacle regions inside of the cone will be assigned as void cells. In this point, it is important to fit grid cells centroids with the sensor cone. If we choose appropriate obstacle width and sensor cone width according to chosen grid cell sizes, our sensor cone will hardly match grid cell centroids to decide if they are void or obstacle cells. To illustrate this situation we can look into the Figure 32 given below. As is noticed from the Figure 32, grid centroids that drop in the possible width region have dramatically decreased due to inappropriate selection of obstacle width compared to the chosen grid cell size. On the other hand, number of grid cell centroids that drop inside the proximity sensor cone can be increased by widening the sensor cone angle (sensor cone width) and obstacle width. Another sensitivity parameter related with map accuracy and robot position awareness is motion model configuration. As explained in section 2.2.1.2, motion model parameters consist of $\alpha_1, \alpha_2, \alpha_3, \alpha_4$.Those parameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are weighted according to estimated linear or angular deviations from noisy odometry measurement data in our experiments. We can illustrate linear deviation with 0.1 percentage of true displacement in Figure 33.With a similar percentage rate for true angular displacement, we can also represent angular error effects on odometry measurements in Figure 34.

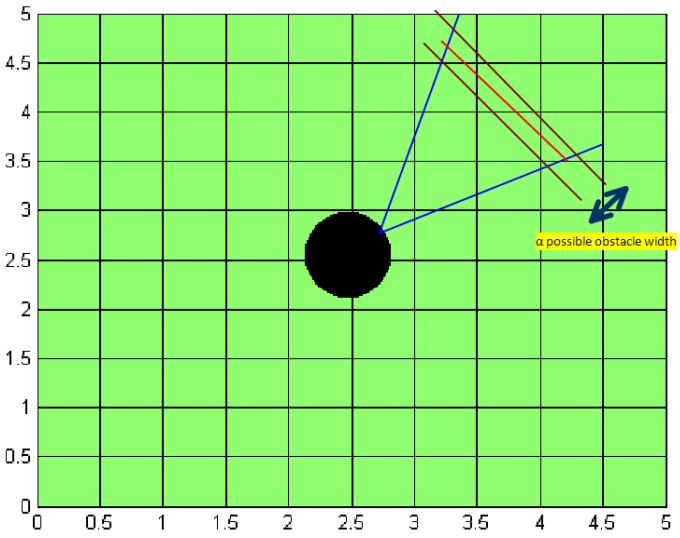

Figure 32 Occupancy Grid Map for SLAM Map Representation with 0.5 Sized Grid Cell Edges

Since red marked displacements represent noisy odometry measurements with respect to the same origin for true odometry measurements shown with black marked displacements in Figure 33 and Figure 34, we have generated samples to estimate true odometry measurements.

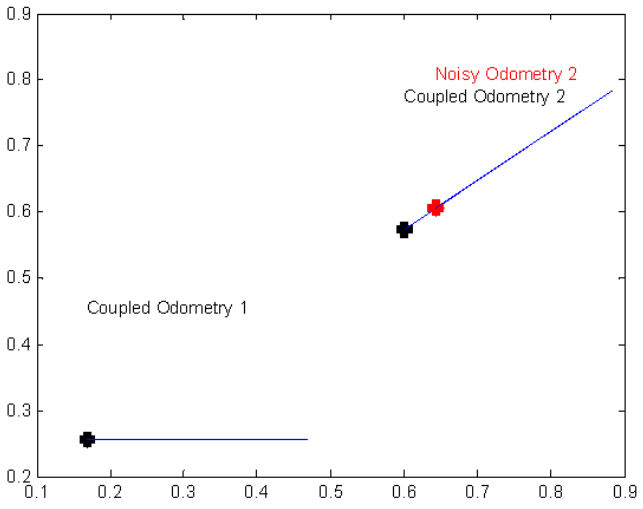

Figure 33 Linear Error Effect on Odometry Measurement

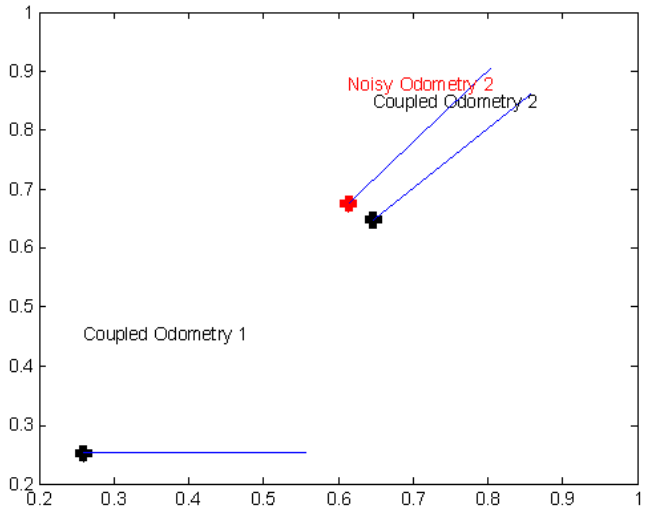

Figure 34 Angular Error Effect on Odometry Measurement

Those samples are generated with respect to motion model parameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ so as to estimate true odometry displacements from noisy odometry measurements. Angular or linear errors can exist with different weights in odometry measurement faults. Hence we utilize those motion model parameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ to weight generated samples with respect to expected error model of obtained odometry measurements. We have motion model configuration that consist of parameter values: $\alpha_1 = 0$, $\alpha_2 = 0$, α_3=0.1, $\alpha_4 = 0.1$ for sample generation in Figure 35.

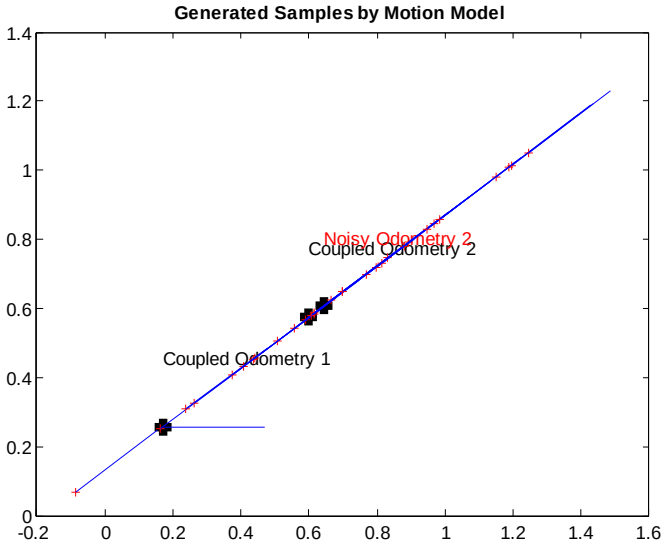

Figure 35 Generated Samples by Motion Model Configuration

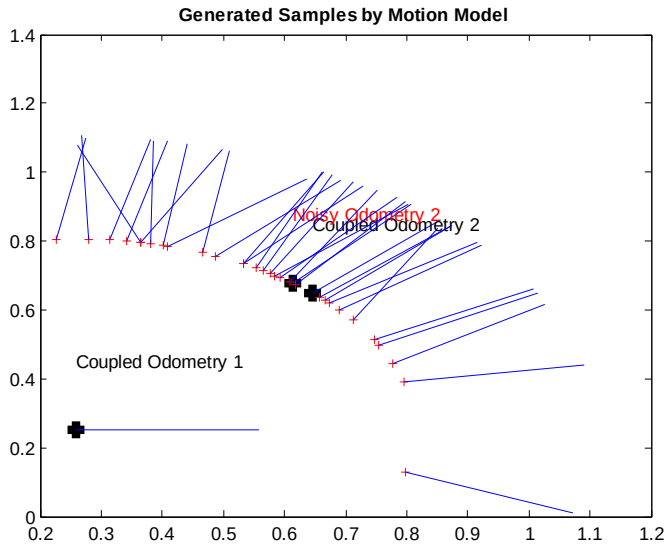

Figure 36 Generated Samples by Motion Model Configuration

In Figure 36, we generated samples for angular based odometry measurement error motion model configuration with respect to parameter values of $\alpha_1 = 0.1$, $\alpha_2 = 0.1$, α_3=0, $\alpha_4 = 0$.We can start Fast SLAM sensitivity analyses according to those given introductory information in this section. Table 16 shows the sensitivities of Fast SLAM map accuracy and robot position awareness performance to changing sensitivity parameters which are obstacle width α, occupancy grid map cell size and sensor cone width over the fixed motion model configuration and noise distribution. These sensitivities demonstrate expected reactions on the particle set specifications which are particle set variance and particle weight values. As particle set variance decreases estimated position or map information certainty increases respectively as explained in chapter 2 section 2.2.1.5.4. Particle weight values correspond to achievement of localization with respect to obtained sensor readings as explained in chapter 2 in section 2.2.1.3 We have realized all Fast SLAM trials using a particle filter having 20 particles. We have implemented that particle filter in all our trials in Table 16 with 6 proximity sensors that have 2br maximum sensing ranges through 20 time steps. We have applied different scenarios through the Table 16 experiments. At the first glance, we change obstacle width α through the FS_Exp_1, FS_Exp_2 and FS_Exp_3 experiments while other parameters are fixed. We have seen that increasing obstacle width changes the localization level of mobile robot. FS_Exp_1 , FS_Exp_2 and FS_Exp_3 have individual

particle variances as 2.94, 2.83 and 2.89. Particle set variance means the certainty of estimated robot position. While particle set variance decreases, position certainty increases.

Table 16 Sensitivity Analyses for Fast SLAM Map Accuracy and Robot Position Awareness Performance

Initial Simulation Parameters												
Total Time Step										**20**		
Particle Number										**20**		
Sensor Number										**6**		
Maximum Proximity Sensor Range										**2br**		
Sensitivity Analysis of Map Accuracy and Position Awareness Performance												
Experiment Name	**α Possible Obstacle Width**	**Occupancy Grid Map Cell Size**	**Sensor Cone Width**	**Motion Model Parameters**				**Injected Noise Parameter**		**Exploration Methodology**	**Most Reliable Particle Weight Value**	**Particle Set Variance(Position Entropy Level)**
				α_1	**α_2**	**α_3**	**α_4**	**e_L**	**e_A**			
FS_Exp_1	0.2	0.3	30	0	0	0.1	0.1.	0.1	0	Fast SLAM	0.055	2.94
FS_Exp_2	0.4	0.3	30	0	0	0.1	0.1	0.1	0	Fast SLAM	0.062	2.83
FS_Exp_3	0.6	0.3	30	0	0	0.1	0.1	0.1	0	Fast SLAM	0.059	2.89
FS_Exp_4	0.6	0.3	40	0	0	0.1	0.1	0.1	0	Fast SLAM	0.089	2.83
FS_Exp_5	0.6	0.3	60	0	0	0.1	0.1	0.1	0	Fast SLAM	0.11	2.303
FS_Exp_6	1	0.3	60	0	0	0.1	0.1	0.1	0	Fast SLAM	0.17	2.5
FS_Exp_7	1.5	0.3	60	0	0	0.1	0.1	0.1	0	Fast SLAM	0.35	1.94
FS_Exp_8	1.5	0.5	60	0	0	0.1	0.1	0.1	0	Fast SLAM	0.27	1.94
FS_Exp_9	1.5	0.7	60	0	0	0.1	0.1	0.1	0	Fast SLAM	0.21	2.3
FS_Exp_10	1.5	0.5	60	0.1	0.1	0	0	0	0.1	Fast SLAM	0.35	2.303
FS_Exp_11	1.5	0.5	60	0.1	0.1	0.1	0.1	0	0.1	Fast SLAM	0.2	2.303
FS_Exp_12	1.5	0.5	60	0.2	0.2	0.2	0.2	0	0.1	Fast SLAM	0.26	1.94
FS_Exp_13	1.5	0.5	60	0.1	0.1	0.5	0.5	0	0.1	Fast SLAM	0.24	2.197

On the other hand, particle weights correspond to the comparison between ray tracing values obtained from SLAM map and sensor reading values. Hence increasing particle weight corresponds to the consistency between robot position information and sensed environment. According to those facts, FS_Exp_2 possesses to the best localization compared to FS_Exp_1 and FS_Exp_3. This means that increasing obstacle width cannot directly enhance localization. We have increased sensor cone width through the FS_Exp_4 and FS_Exp_5, and changed sensor cone width from 40 degree to 60 degree. As a consequence of increased sonar cone we can assign more grid cells on the SLAM map in accordance with inverse sensor measurement model in (chapter 2 section 2.2.1.4). Hence our mapping capability has spanned over wider region within the SLAM map. Since we have mapped wider region we can compare more specifications on the map and

reach more certain localization levels. As noticed from Table 16, FS_Exp_5 has particle variance 2.303 that is less than FS_Exp_4 particle variance which is 2.809. So FS_Exp_5 has more certain localization results with respect to FS_Exp_4. But wider regions are also tuned with obstacle width. Exaggerated obstacle width can also deteriorated mobile robot localization. As noticed from FS_Exp_6, we have increased obstacle width parameter as 1 and fixed other FS_Exp_5 parameters. So our particle variance has increased to 2.5 in FS_Exp_6 meaning that our certainty in localization decreased. We have also changed grid cell size values to see their effects on localization through FS_Exp_6 to FS_Exp_9 experiments. We have changed grid cell size values from 0.3 to 0.7 and it is seen that coarser grid cell sizes leads to deteriorated localization as noticed from FS_Exp_9 particle set variance as 2.3 while FS_Exp_7 and FS_Exp_8 with finer grid cells has particle set variance as 1.904. Finally, we have tested motion model parameter effects on localization performance. We have applied different motion model parameters with injected angular noise as 0.1 percentages at each mobile robot step. We have tuned sampling variances corresponding to the α_1 and α_2 values 0.1 in FS_Exp_10. Those sampling variances correspond to the angular deviation samplings as seen from Figure 36 and as explained in chapter 2 section 2.2.1.2. In FS_Exp_11 we have also tuned α_3 and α_4 for linear sample deviations as seen from Figure 35(explained in chapter 2 section 2.2.1.2) besides of the α_1 and α_2 values that are set to 0.1. According to their particle set variance values which mean the certainty of their position localizations, FS_Exp_11 and FS_Exp_10 have similar particle set variances but FS_Exp_10 has particle weight value that is greater than FS_Exp_11 particle weight value. We can explain that fact with the existence of constant sample numbers where sampling was done according to those motion model sampling variances. Since FS_Exp_10 is only generated according to the angular deviation while FS_Exp_11 is generated according to both linear and angular deviations, FS_Exp_10 has more appropriate samples for angular based odometry errors. So FS_Exp_10 has greater particle weight (value of 0.35) while FS_Exp_12 has smaller valued ones (value of 0.2). On the other hand, FS_Exp_12 motion model parameters are tuned with larger deviation variances (value of 0.2) according to FS_Exp_10 and FS_Exp_11. As a consequence, larger deviation leads to larger error probabilities and FS_Exp_12 has particle set variance level as 1.94 which is smaller than FS_Exp_10 and FS_Exp_11 particle set variance values as 2.303. But similar to the FS_Exp_11, increased probability variations leads to lower particle weights in FS_exp_12 as 0.2. But our priority to determine localization is particle set variance.

We firstly compare particle set variances of those trials. Then we compare their particle weights if the trials have similar particle set variance. This priority is due to the fact that particle set variance means the certainty of our estimator to predict true robot states, while particle weight correspond to which sample is more correct within the particle set. The particle set variance is a low value then, localization is successfully achieved on that particle set. So we can infer that FS_Exp_12 has better localization performance compared to FS_Exp_10 and FS_Exp_11 due to their particle set variance levels. At the last trial in FS_Exp_13, we have increased linear sampling generation variance by increasing the corresponding variables α_3 and α_4, and inFS_Exp_12 we decreased angular sampling generation variance by decreasing the corresponding variables α_1 and α_2. We have set α_1 and α_2 to values 0.1 while α_3 and α_4 variables are assigned values of 0.5 in FS_Exp_13 Since we increased the possibility of linear deviation sample generations according to angular based ones, we have lowered the localization performance as noticed from lowered particle set variance FS_Exp_13 (as value of 2.197). That situation of FS_Exp_13 is similar to the redundant sample generation for linear based deviations in FS_Exp_11.As noticed from Table 16, FS_Exp_11 has lowered localization certainty attested by its particle set variance value, compared to FS_Exp_10. In a similar way, FS_Exp_13 has lowered localization performance with increased linear generated error samples. But FS_Exp_13 has smaller variance than FS_Exp_10 since its sampling variances under motion model parameters which are $\alpha_1 = 0.1$, $\alpha_2 = 0.1$, α_3=0.5, $\alpha_4 = 0.5$, are greater than those of FS_Exp_13 under motion model parameters : $\alpha_1 = 0.1$, $\alpha_2 = 0.1$, α_3=0.1 $\alpha_4 = 0.1$. At the next step in this section we will illustrate and discuss the given experiments in Table 16 over their obtained SLAM maps. Thus we will observe the effect of localization on the SLAM map generation process. We can see the effect of obstacle width α parameter on the SLAM maps of related experiments as shown in Figure 37 and Figure 38. As is seen from Figure 37 and Figure 38, increasing α enhances sensing obstacles within the SLAM maps, this is seen by the existence of more reddish colors on the maps as α is increased It is important to tune obstacle width α parameters with respect to grid cell size. More irregular environments including narrow passages require finer grid cells sizes and thinned possible obstacles width α , while coarser grid cell sizes and wider obstacle widths can be considered for environments including voids and wide passages for the robot to get through. As is seen from the tunnels that are circled at the top side of the SLAM maps represented in Figure 37,Figure 38 and Figure 39, using the obstacle width α as value of

1.5 with grid cell size 0.5 within experiment 8 opposes the tunnel traversability on SLAM map compared to the tunnels in Figure 37 and Figure 38 with finer grid cell sizes as 0.3 with more thinner obstacle widths α as 0.6 and 1 respectively.

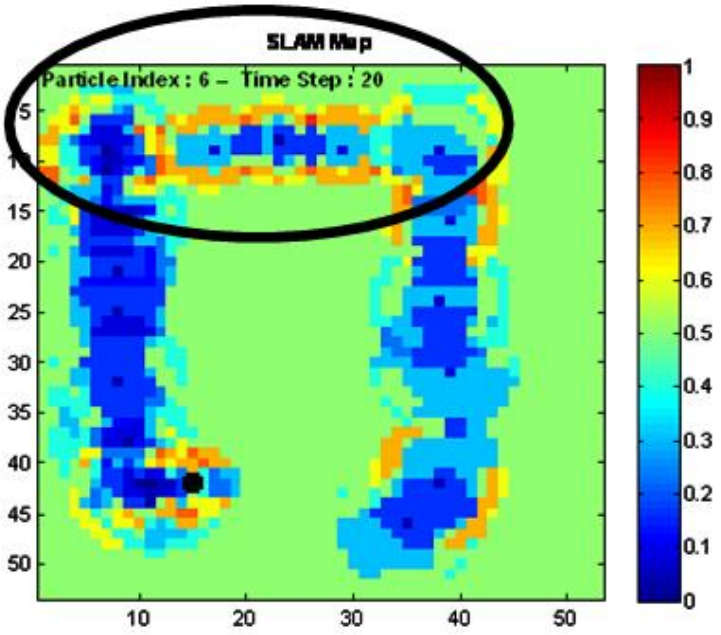

Figure 37 Most Reliable Particle SLAM Map in FS_Exp_5 Experiment with α=0.6 and grid cell size= 0.3

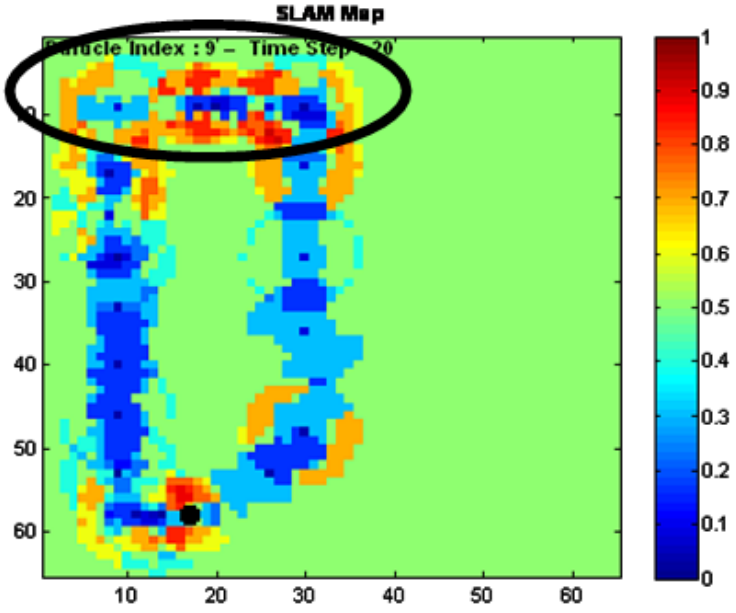

Figure 38 Most Reliable Particle SLAM Map in FS_Exp_6 Experiment with α=1 and grid cell size= 0.3

As a consequence of that coarser grid cell size selection with thick obstacle widths, we cannot see exact path consisting of connected voids in Figure 39.

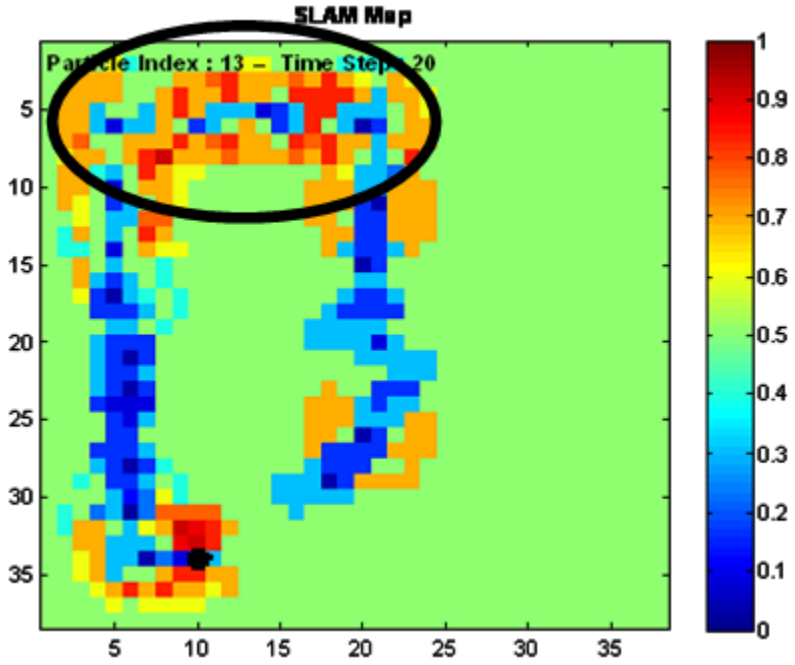

Figure 39 Most Reliable Particle SLAM Map in FS_Exp_8 Experiment with α=1.5 and grid cell size= 0.5

As we compare the tunnel regions of FS_Exp_8 with FS_Exp_5 and FS_Exp_6 of Figure 37, Figure 38 and Figure 39 respectively all zoomed in and summarized in Figure 40. The tunnel represented by connected voids consisting of void cells represented with blue colored cells are narrowed by obstacle cells represented with colors near to red color through the FS_Exp_5 to FS_Exp_8 as grid cell size and obstacle width value are increased. So it is hard to distinguish connected voids from obstacles in FS_Exp_8 SLAM map when compared to FS_Exp_6 and FS_Exp_5 with finer grid cell sizes and thinner obstacle widths. FS_Exp_5 in Figure 40 allows traversability of the tunnel that was blocked within other two experiments. On the other hand, we have injected linear errors to odometry measurements for experiments FS_Exp1 till FS_Exp_9.In order to see the difference between linear based and angular based odometry error effects on SLAM maps, we have realized Fast SLAM experiments with angular error injected in odometry measurement from experiment 10 until experiment 13. At the next step we investigate the linear and angular error affects on the generation process of SLAM maps. As noticed from Table 16, we have realized experiment according to those linear e_L and angular e_A

based errors. Firstly, we can recall the relationship between particle's robot positions and generated SLAM maps.

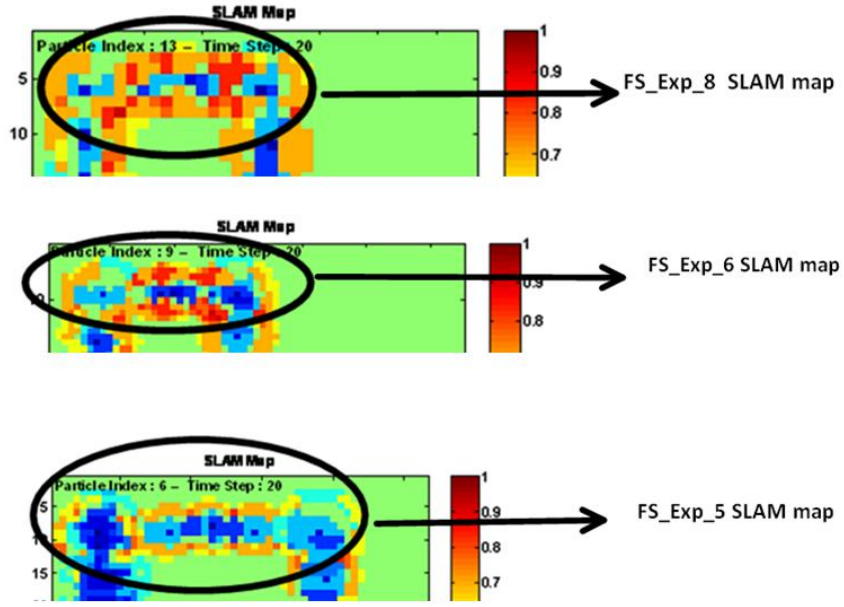

Figure 40 Tunnel Comparisons of FS_Exp_5, FS_Exp_6 and FS_Exp_8

As seen from Figure 41, particle robot positions are shown on the SLAM map so that each particle robot position has its individual bearing and coordinates on its SLAM map.

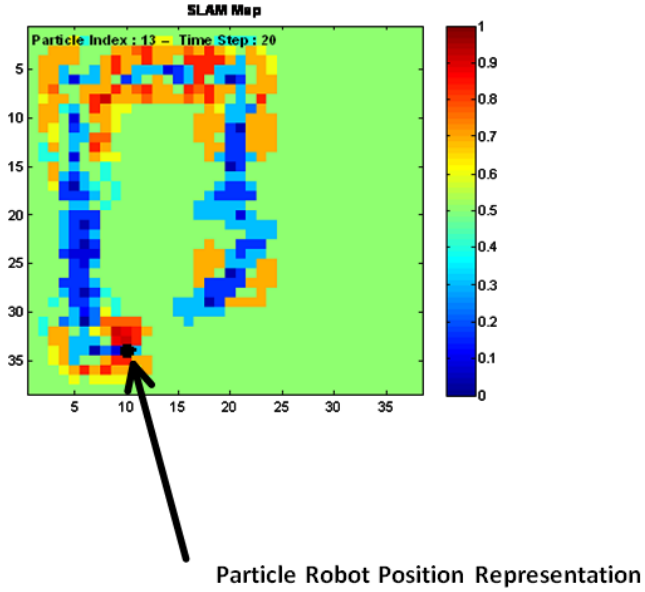

Figure 41 Particle Robot Position Representation

Furthermore, particles generate their SLAM maps according to their particle robot position coordinates on the SLAM map. As explained in chapter 2 section 2.2.1.4, grid cells that drop in the sonar cones are assigned as voids cells and obstacle cells. In this concept, our sonar cone directions are determined within the SLAM map according to the particle robot bearing. Thus obstacles or voids that are seen on Figure 41 with reddish and blue toned colors are assigned on the SLAM map according to their particle robot position bearing and coordinate values. So our SLAM map will be shaped with respect to robot coordinates and bearing selections since robot position is chosen as origin point in inverse sensor mapping cone as explained in chapter 2 section 2.2.1.4. We can illustrate that fact with generated SLAM maps in Figure 43 where obstacle grid cells are represented with reddish colors and void grid cells with blue toned colors as a consequence of navigation through the environment seen in Figure 42. Red marked points correspond to the obstacles in Figure 42.

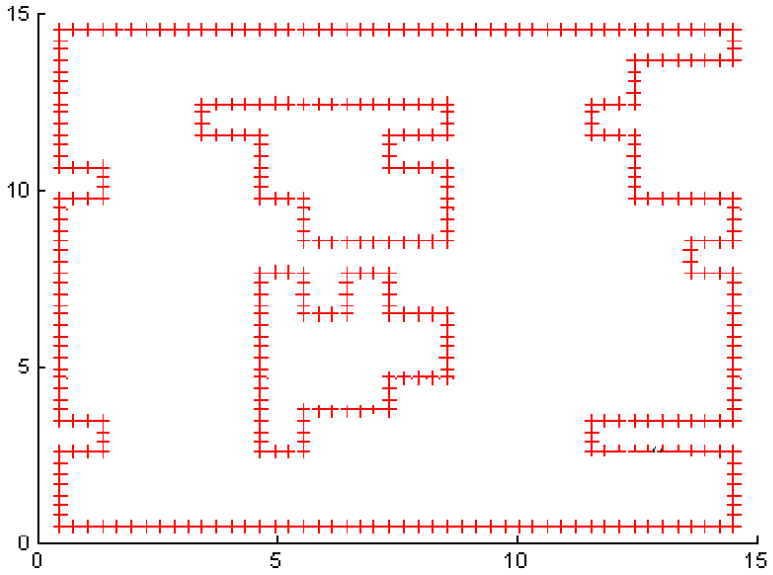

Figure 42 Simulation Environment for Linear and Angular Error Comparisons

We have started our mobile robots with same bearings and same particle robot position bearings. Furthermore, their particle robot position bearings and true robot position bearings are chosen as 0 degree. While FS_Exp_13 is under angular errors of 0.1 percentages in the true odometry commands at each step, FS_Exp_8 is under linear errors of 0.1 percentages in its true odometry displacements. As a consequence of that error differences, SLAM map orientations also change according to the fact that explained with

Figure 41 which explains why SLAM map generation process is dependent on particle robot position coordinate and bearings.

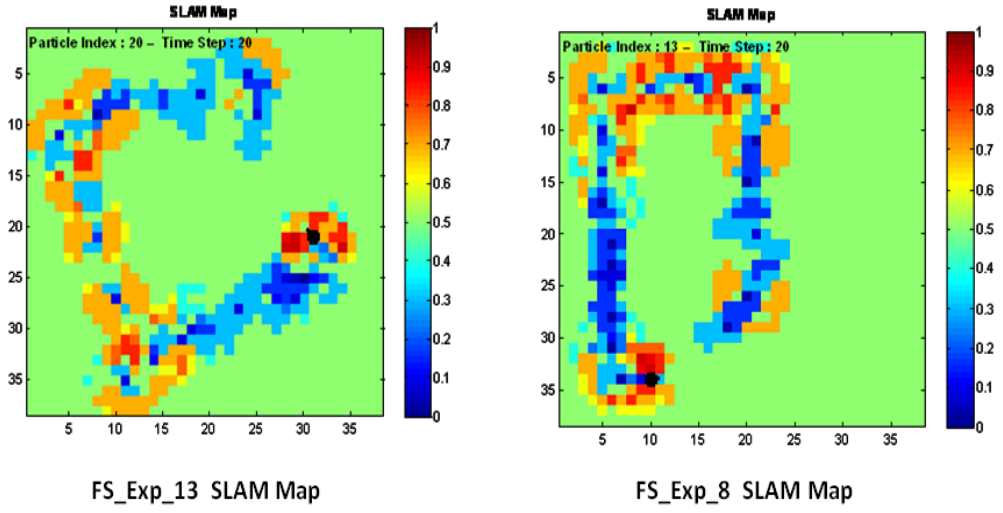

Figure 43 Particle SLAM Maps According to Particle Robot Positions with Different Bearings

We see from Figure 43 that FS_Exp_13 SLAM map rotates 135 degree counter clockwise according to the voids and obstacle placements of FS_Exp_8 which is a map at the true angle consistent with real world map as seen in Figure 42. Figure 43 shows us how particle robot position affects obtained SLAM maps with respect to their bearing and position coordinates. Thus we have shown how the robot position accuracy i.e. its bearing affects the obtained SLAM map as illustrated and explained with Figure 43. We conclude from this relationship that localization of mobile robot can also affect the SLAM generation process.

5.4 Entropy Based Exploration Performance Analyses

Before discussing the experimental results of each of our contributed methodologies, we have to analyze the experimental outcomes of classical methodologies in order to be able to compare our contribution with them. Hence we have chosen to provide the performance analysis of the entropy based exploration methodology heavily used in the literature as explained in section 2.3.2. Entropy based exploration approach is the classical methodology, preferred by many practical implementations in real life and is based on the evaluation of entropy changes in position and map information according to a selected action from a proposed action set. The optimal navigation is therefore

maintained through the exploration process by minimizing entropy in localization and in mapping, this optimality provided by comparing entropy changes corresponding to exploration and exploitation actions. Exploration actions obtain decrements in the map information entropy values, in other words reduce the uncertainty in map information. On the other hand, exploitation actions provide decrements in the position entropy levels, in other words reduce the uncertainty in position. Measurement values obtained from proximity sensor directions according to mobile robot bearing reference are considered as available odometry command steps. Then odometry commands are evaluated according to their capability to minimize total entropy level and their performance costs, so that cost of the selected odometry command is considered with weighted entropy change in the objective function explained in section 2.2 as is expressed below:.

$$O(a) = Cost(a) - \gamma H(a)$$

In the equation above $H(a)$ corresponds to the entropy change in the case of performing action a while $Cost(a)$ corresponds to the cost value to perform action a.We considered cost values corresponding to the actions in proportion to the displacement amounts according to their odometry command. On the other hand, parameter γ, weighting entropy change against action costs, is changed in the experimental sensitivity evaluation. In this evaluation, we observe the effects of that trade off. On the other hand, we constitute our proposed action set from obtained sensor readings. We compute odometry displacements between mobile robot and sensed points at the vicinity of mobile robots. We can illustrate that situation within the figures below:

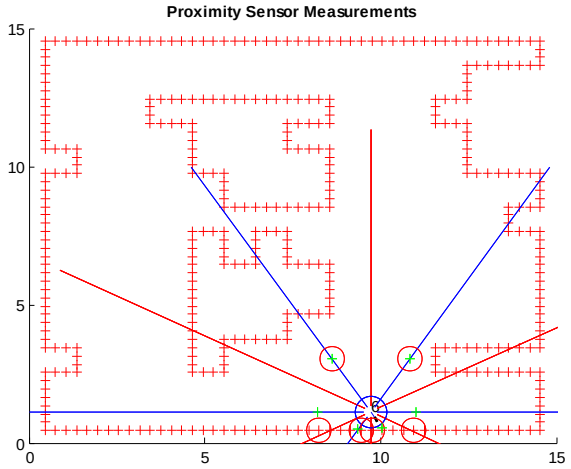

Figure 44 Obtained Measurements using Proximity Sensor Measurements from Simulation Environment

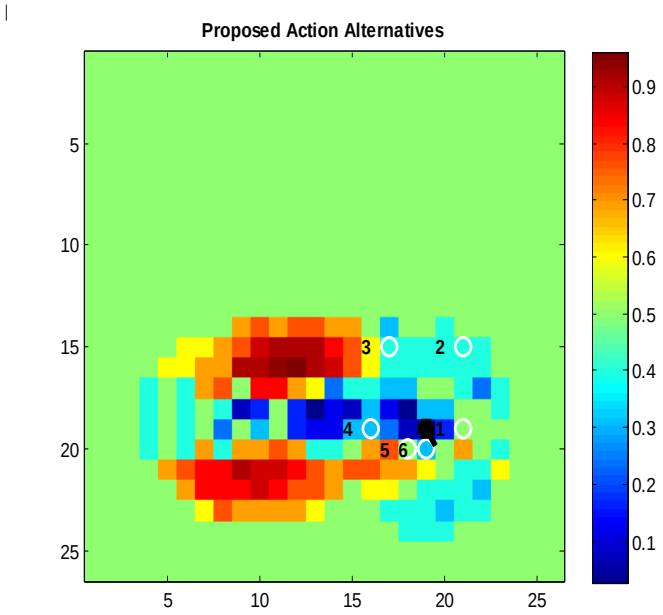

Figure 45 Proposed Actions According to Obtained Measurement Distances as shown in Figure 44

Figure 44 and Figure 45 have been generated according to our proposed action set that we will also make use in the Percolator Enhanced Entropy Based SLAM on the most reliable particle entropy based SLAM map using measurement distances obtained from proximity sensor. As seen from the occupancy grid map represented by Figure 45, grid cells that correspond to the lower occupation probabilities with blue tones are assigned as voids cells while grid cells that correspond to the higher occupation probabilities with red tones are assigned as obstacles. Obtained sensor measurements given in Figure 44 are assigned on the SLAM map with white colored circles on Figure 45. Our entropy based methodology considers its goal points as those white colored circles in Figure 45. Hence an action set for entropy based exploration is prepared according to those goal points with white colored circles on Figure 45.According to those goal points, we select robot position and the goal point position individually to constitute odometry vector which consisting of two coordinate that express displacement between them. As explained in chapter 2 section 2.2.1.2, odometry command vector $u_t = \begin{bmatrix} \overline{x}_{t-1} \\ \overline{x}_t \end{bmatrix}$ consist of two coordinate vectors for next position $\overline{x}_t$ and current position $\overline{x}_{t-1}$. We assign our current robot position to $\overline{x}_{t-1}$ coordinate vector while goal position to $\overline{x}_{t-1}$ vector. So we provide odometry command from the robot position and goal point coordinates. Thus we obtain number of action alternatives as number of goal points. Then, we compute

possible information gains of those proposed actions alternatives according to their effects on particle set during the SLAM process. We compute map and position information entropy changes according to those alternative actions generated in Figure 45. We tried SLAM operation with each alternative action then computes its effect on given particle set with the equations explained in section 2.3.2.Thus we can plot all information gain values corresponding to the alternative actions as seen in Figure 46 below:

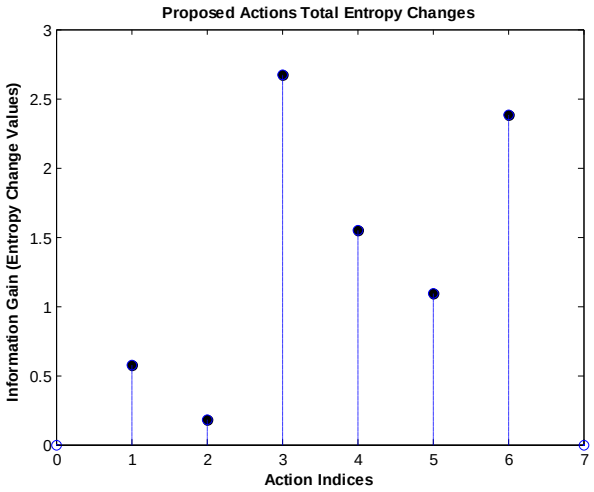

Figure 46 Information Gain Values According to Action Indices

We see from the Figure 46 that horizontal axis correspond to the action indices of action alternatives while vertical axis corresponds to the information gain or entropy change level values. At this step, we have used the implementation tips proposed in reference [19].We iterate the SLAM over the obtained particles' SLAM. Measurements are taken as ray tracing distances to the uncertain grid cells corresponding to the occupation probabilities indicating neither void nor obstacle, because those uncertain grid cells mostly influence the information gain over the SLAM map. We will compare entropy based exploration simulation trials with changing sensitivity parameters such as obstacle width α, occupancy grid cell sizes and motion model sampling parameters. We have initialized all experiments of Table 17 with 20 particles, 9 proximity sensors having maximum proximity sensor range 2br, all experiments running over 10 time steps. We have observed localization performances of entropy based exploration methodology by comparing their most reliable particle weight values and particle set variances. We have

also injected linear odometry measurement at each step with the added 0.1 percentage of odometry motion linear displacements. We have also changed the motion model variances through the experiments.

Table 17 Entropy Based Exploration (EBE) Sensitivity Analyses for Map Accuracy and Robot Position Awareness

Initial Simulation Parameters												
Total Time Step										**10**		
Particle Number										**20**		
Sensor Number										**9**		
Maximum Proximity Sensor Range										**2br**		
Sensitivity Analysis of Map Accuracy and Position Awareness Performance												
Experiment Name	**α Possible Obstacle Width**	**Occupancy Grid Map Cell Size**	**Sensor Cone Width**	**Motion Model Parameters**				**Injected Noise Parameter**		**Exploration Methodology**	**Most Reliable Particle Weight Value**	**Particle Set Variance(Position Entropy Level)**
				α_1	**α_2**	**α_3**	**α_4**	**e_L**	**e_A**			
E_Exp_1	1.2	0.5	40	0	0	0.1	0.1	0.1	0	E B E	0.59	1.79
E_Exp_2	1	0.5	40	0	0	0.1	0.1	0.1	0	E B E	0.27	2.079
E_Exp_3	1.5	0.7	40	0	0	0.1	0.1	0.1	0	E B E	0.71	1.609
E_Exp_4	2	1	40	0	0	0.1	0.1	0.1	0	E B E	0.05	3
E_Exp_5	2	1	40	0	0	0.5	0.5	0.1	0	E B E	0.8	0.6

E_Exp_1 and E_Exp_3 have larger most reliable particle weight value as 0.59 and 0.71 respectively while E_Exp_2 has 0.27 valued most reliable particle weight values. Moreover E_Exp_1 and E_Exp_3 have narrower variances as 1.79 and 1.609 respectively compared to the ones of E_Exp_3 that have 2.079 value. Particle set variance corresponds to the entropy levels of mobile robot localizations through the experiments in Table 17. Particle set variance decreases as the certainty of localization increases. Particle weights also correspond to the consistency of robot position with the true odometry measurements. According to those facts, tuning between grid cell sizes and obstacle widths are more appropriately selected for localization in E_Exp_1 and E_Exp_3 than for E_Exp_2. Because E_Exp_1 and E_Exp_3 have lower entropy levels with respect to E_Exp_2 while E_Exp_1 and E_Exp_3 have higher particle weights with respect to E_Exp_2 particle weight. On the other hand, we see that coarser grid cells cause uncertainty in localization while the obstacle width selection cannot compensate that situation as seen from E_Exp_4 with grid cell size of 1, which has most reliable particle weight as 0.05 but a position entropy level highest among all in the last column carrying the value of 3. On the other hand, progression of localization does not only

depend upon mapping parameters, we also consider motion model parameters to deal with robot localization accuracy using generated particles with a given variance by motion model parameters.

Table 18 Entropy Based Exploration (EBE) Sensitivity Analyses for Coverage Performance

Initial Simulation Parameters									
Total Time Step								**10**	
Particle Number								**20**	
Sensor Number								**9**	
Maximum Proximity Sensor Range								**2br**	
Sensitivity Analyses of Coverage Performance									
Experiment Name	**α Possible Obstacle Width**	**Occupancy Grid Map Cell Size**	**Sensor Cone Width**	**Motion Model Parameters**				**Exploration Methodology**	**Coverage Area(br^2)**
				α_1	**α_2**	**α_3**	**α_4**		
E_Exp_1	1.2	0.5	40	0	0	0.1	0.1	E B E	16 br^2
E_Exp_2	1	0.5	40	0	0	0.1	0.1	E B E	21 br^2
E_Exp_3	1.5	0.7	40	0	0	0.1	0.1	E B E	20 br^2
E_Exp_4	2	1	40	0	0	0.1	0.1	E B E	20 br^2
E_Exp_5	2	1	40	0	0	0.5	0.5	E B E	13 br^2

As seen from E_Exp_5 with a larger motion model sampling variance that is given in motion model parameters columns of Table 17, we have reached most reliable particle weight of 0.8 and particle set variance of 0.6 that is a much better localization performance compared to experiments that has finer grid cell sizes such as E_Exp_1 and E_Exp_3. At the first glance to the analyses of results in Table 17, tuning between possible obstacle width and occupancy grid cell size parameters have been found to be an important issue for localization performance. As seen from E_Exp_1 and E_Exp_3, their obstacle widths in their SLAM maps have provided better ray tracing comparison for localization, while E_Exp_2 has provided less information about the obstacles for ray tracing of localization. As a consequence of experimental results of Table 18, mapping and motion model parameters have not affected dramatically the coverage performance of entropy based exploration guided mobile robot. Since entropy based exploration methodology defines its action set based on sensor readings as explained in Figure 45 in our implementations, mapping sensitivity parameters have not changed coverage performance as affected localization performance. It is noticeable point that while E_Exp_5, E_Exp_1 and E_Exp_3 have better localization performances according to other experiments in Table 17, their covered area are smaller than E_Exp_2 and

E_Exp_4. But E_Exp_3 is not dramatically worse or better in terms of the coverage and localization performance results. So we can infer that E_Exp_3 has successfully realized entropy based trade off among other experiment trials given in tables above. Even though we cannot see any dramatic change within the coverage performance depending upon mapping parameters, we will observe more dramatic coverage performance changes within the next percolator based analyses sections.

5.5 Purely Applied Percolator Based Exploration

In this section, we will investigate the performance of our proposed purely percolator based exploration methodology through the simulated runs. We have outlined the phases of purely percolator based exploration methodology in section 3.2.Since purely applied percolator based exploration methodology leads the mobile robot through connected voids to reach survivors as soon as possible in the context of our thesis, exploration activity dominates the information gain process, when putting the preferences on minimizing uncertainty. Hence our purely applied percolator based approach is a suboptimal technique when compared to the entropy based methodology since localization entropy minimization is not a primary concern besides that of the map entropy. While entropy based exploration algorithms consider both map and position uncertainties during odometry command selecting, the priority in our purely applied percolator approach is based on accuracy of the map so that, Percolator guided actions mostly speed up coverage process over unknown terrains until any detected dead-end. In the mean time, robot position uncertainty increases as mobile robot navigates through unexplored regions. Due to lack of position awareness, SLAM maps accuracy cannot be maintained any longer through the simulation steps. Since robot position accuracy is not priority , purely applied percolator exploration starts to provide guidance in inaccurate directions after multiple runs. In spite of that deficiency, purely applied percolator based technique reduces the computational burden of the mobile robot and speeds up the coverage process using considerable amount of odometry measurements. We will discuss these specifications in sensitivity analyses in Table 19 and Table 20. In the context of Table 19, we will discuss map and localization sensitivities on sensitivity parameters such as obstacle width α, occupancy grid map cell size, sensor cone width, motion model parameters and injected noise parameters. As is seen from the Table 19, we realized 4 percolator based exploration experiments with initial parameters such as total time step set as 10, particle number as 20, sensor number as 9 and maximum proximity sensor

range as 2br. On the other hand, we have applied different scenarios for those experiments of Table 19. We have increased the occupancy grid cell size of SLAM map from 0.5 to 1 as is seen from the P_Exp_1 experiment to P_Exp_4. We have fixed the noise and motion model parameters to see the effect of SLAM parameters on localization accuracy. We interpret the sensitivity results according to particle set variance and most reliable particle weight value. We have started our sensitivity analyses from P_Exp_1 with 0.5 occupancy grid cell size and obstacle widths parameter value α being 1.2 then we have obtained most reliable particle weight as 0.2 and particle set variance as 2.303. At the experiment of P_Exp_2, we have decreased possible obstacle width parameter value to 1 and we encountered that mobile robot localization performance has been enhanced at the end of percolator exploration with most reliable particle weight value as 0.53 and particle set variance as 2.079. Since particle set variance (position entropy level) decreases as particle filter samples get close to the true values. Hence our purely applied exploration methodology has been tried with decreased obstacle width in P_Exp_2 that tuned with 0.5 grid cell size that causes better localization level. The reason depends upon the percolator mechanism. With same grid cell sizes, percolator determines obstacles more intensively as in P_Exp_1. So it can easily predict continuity of those obstacles, but within the lower obstacle width selection, guidance of the percolator worsens due to uncertainty on obstacle regions. For that reason coverage performance of P_Exp_1 is better than P_Exp_2. As seen from Table 20, P_Exp_1 coverage was found to be 40 br^2 while P_Exp_2 was found as 30 br^2. So while P_Exp_1 has navigated to wider area with its intensive obstacle regions that enhance its guidance, P_Exp_2 has navigated at the narrower region according to P_Exp_1 based on weaker guidance because of less intensive obstacle region. So P_Exp_2 was revisiting cells of the environment, enhancing its localization with respect to P_Exp_1. Obstacle width parameter arranges the obstacle width that falls within the sensed distance. Extending obstacle width α narrows the zone of void cells. So our mobile robot starts to map obstacle cells rather than void cells within the obtained SLAM map, since obstacles begin to occupy predominantly grid cells. In that situation, exaggerated obstacle cells mislead localization as is seen in P_Exp_1.On the other hand, selecting coarse grid cell sizes as in P_Exp_3 and P_Exp_4 experiments causes a drop in localization performance of percolator based exploration guided mobile robot. Since coarser grid cell sizes affect SLAM maps so that obtained sensor measurements overlap with voids near obstacles. Then void regions in the simulation environment are detected as obstacles.

Table 19 Pure Percolator based Exploration (PBE) Sensitivity Analyses for Map and Localization Accuracies

Initial Simulation Parameters												
Total Time Step										**10**		
Particle Number										**20**		
Sensor Number										**9**		
Maximum Proximity Sensor Range										**2br**		
Sensitivity Analyses of Map Accuracy and Position Awareness Performance												
Experiment Name	**α Possible Obstacle Width**	**Occupancy Grid Map Cell Size**	**Sensor Cone Width**	**Motion Model Parameters**				**Injected Noise Parameter**		**Exploration Methodology**	**Most Reliable Particle Weight Value**	**Particle Set Variance(Position Entropy Level)**
				α_1	**α_2**	**α_3**	**α_4**	**e_L**	**e_A**			
P_Exp_1	1.2	0.5	40	0	0	0.1	0.1	0.1	0	P B E	0.2	2.303
P_Exp_2	1	0.5	40	0	0	0.1	0.1	0.1	0	P B E	0.53	2.079
P_Exp_3	1.5	0.7	40	0	0	0.1	0.1	0.1	0	P B E	0.18	2.303
P_Exp_4	2	1	40	0	0	0.1	0.1	0.1	0	P B E	0.24	2.303

That faulty situation also decreases the accuracy of ray tracing for localization. Similar to exaggerated obstacle width selection in P_Exp_1, coarser grid cell selection is also corrupting localization accuracy. For this reason we have seen from Table 20 that P_Exp_1 which has larger obstacle width and P_Exp_3 and P_Exp_4 which has coarser grid cell sizes have particle variance namely position entropy level with 2.303 while P_Exp_2 which has appropriate obstacle width and finer grid cell sizes has lower particle variance namely position lower entropy level as 2.079. That means P_Exp_2 has better localization performance than P_Exp_1, P_Exp_3 and P_Exp_4 which have larger obstacle width and coarser grid cell sizes. We will also interpret percolator based exploration coverage sensitivity analyses according to Table 20 experiment results. As is seen from P_Exp_1 to P_Exp_2, finer grid cell size leads the percolator based exploration guided mobile robot to explore more connected voids, while coarser grid cells size prevent such an expansion of robot exploration through the connected voids as seen in P_Exp_3 and P_Exp_4.Because we see from the Table 20 that coverage performances of experiments with coarser grid cells according to the environment obstacles, can cause to extrapolate detected obstacles within the environment. So many tunnels and connected voids are seen as dead-end. For that reason, P_Exp_3 and P_Exp_4 have coarser grid cells and their coverage values are 24 br^2 that are smaller than P_Exp_1 and P_Exp_2 experiments which have finer grid cell sizes. We can illustrate the effect of that tuning of parameters between grid cell size and obstacle width

α parameters with an another percolator based exploration experiment that runs through 20 time steps for obstacle width α parameters values that are set as 0.8 for the first experiment and 0.6 for the second one. We have shown their respective trajectories in Figure 47 and Figure 48.

Table 20 Pure Percolator based Exploration (PBE) Sensitivity Analyses for Coverage Performance

Initial Simulation Parameters									
Total Time Step								**10**	
Particle Number								**20**	
Sensor Number								**9**	
Maximum Proximity Sensor Range								**2br**	
Sensitivity Analyses of Coverage Performance									
Experiment Name	**α Possible Obstacle Width**	**Occupancy Grid Map Cell Size**	**Sensor Cone Width**	**Motion Model Parameters**				**Exploration Methodology**	**Coverage Area(br^2)**
				α_1	**α_2**	**α_3**	**α_4**		
P_Exp_1	1.2	0.5	40	0	0	0.1	0.1	P B E	41 br^2
P_Exp_2	1	0.5	40	0	0	0.1	0.1	P B E	30 br^2
P_Exp_3	1.5	0.7	40	0	0	0.1	0.1	P B E	24 br^2
P_Exp_4	2	1	40	0	0	0.1	0.1	P B E	22 br^2

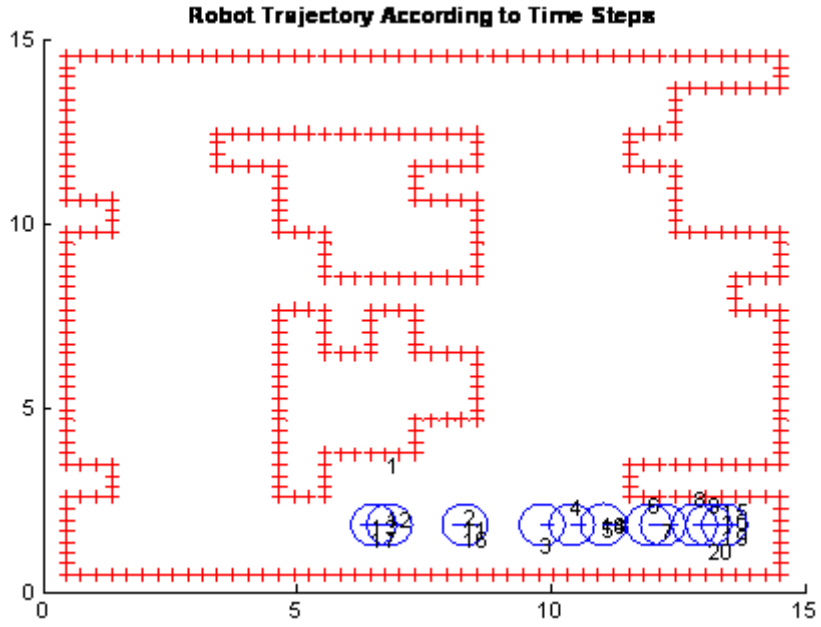

Figure 47 Percolator Based Exploration Experiment with 0.8 Possible Width.

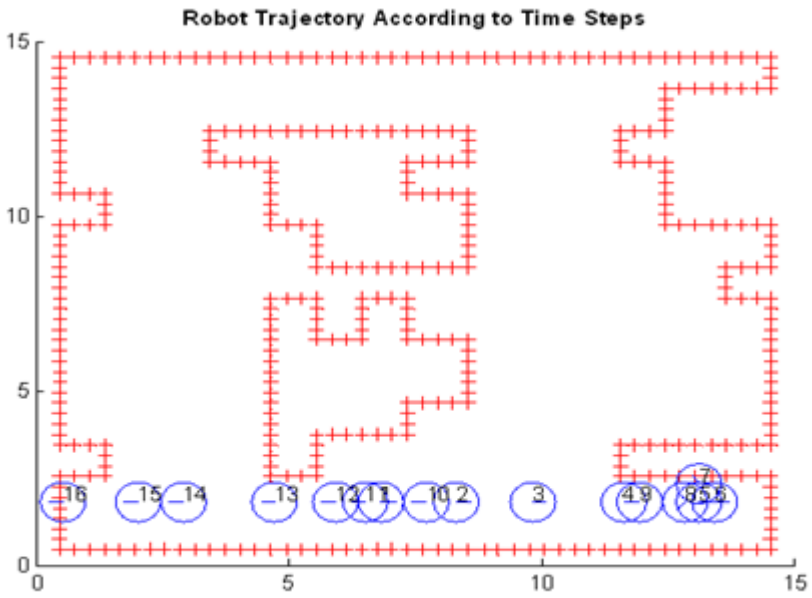

Figure 48 Percolator Based Exploration Experiment with 0.6 Possible Width

As is seen from comparisons of Figure 47 and Figure 48, percolator guided mobile robot explore more connected voids in the second experiment that has obstacle width parameter $\alpha = 0.6$ while the mobile robot exploration of the first experiment that has obstacle width parameter $\alpha = 0.8$ has been restricted at the bottom corner of the simulation environment where the region has been interpreted as narrow due to the large α parameter.

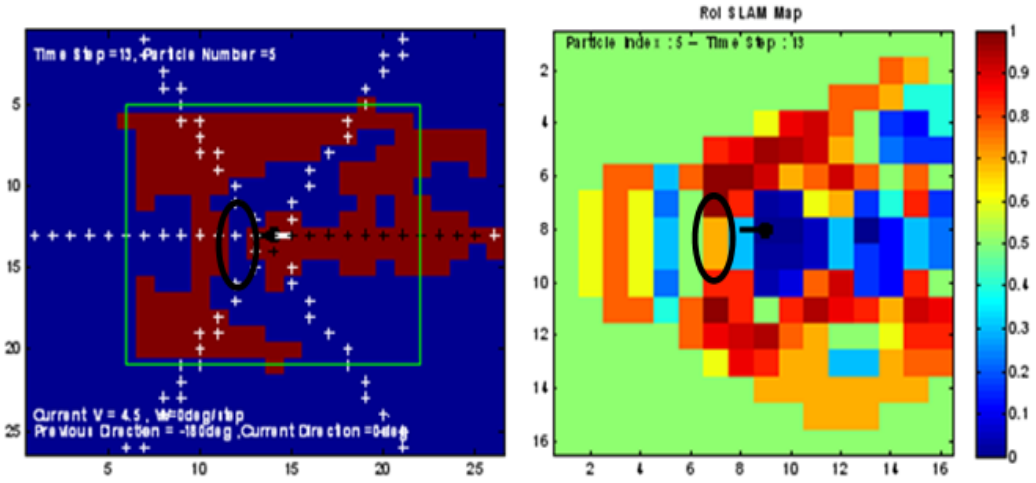

Figure 49 Percolator Map (left) and Corresponding SLAM Map (right) for Time Step 13 of Figure 47

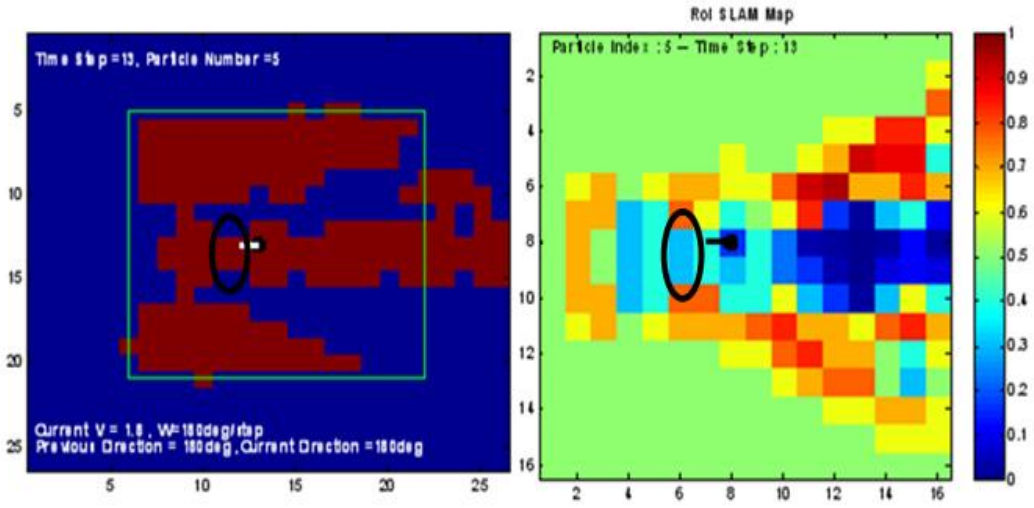

Figure 50 Percolator Map (left) and Corresponding SLAM Map (right) for Time Step 13 of Figure 48

We can also explain that situation on their percolator maps at the left of Figure 49 and Figure 50. Before interpreting the percolator map and SLAM map figures we recall that SLAM maps at the right side of Figure 49 and Figure 50 correspond to the obstacle occupation probabilities of obtained maps by sensor readings. Occupancy grid cells with probabilities lower than 0.5 and represented by blue color tones in SLAM maps correspond to the void regions, while the grid cells that has occupation probability greater than 0.5 and represented by red color tones correspond to the obstacle regions. On the other hand, green color toned grid cells correspond to uncertain regions in the SLAM map. On the other hand, we generate percolator maps from SLAM maps (on the left of the figures) and red colored grid cells represent spanning voids or available regions while blue colored grid cells represent obstacle regions . As is seen from Figure 49 (left) and Figure 50 (left), percolator maps generated from SLAM maps, their isolated regions are blue colored cells and are not available for robot motion while invaded regions that are red colored regions are available for robot motion After those reminders, we can investigate the percolator and SLAM maps of the experiments to understand how obstacle width affects percolation exploration performance. Figure 49 corresponds to the experiment that is displayed in terms of mobile robot navigation in Figure 47 with $\alpha = 0.8$ for obstacle width parameter. We have seen from those figures that percolator guides the mobile robot through the connected voids until time step 13. Because there exist red colored regions in front of the robot motion within the percolator maps given at time step 13 represented with black circle, its percolator map given by Figure 49 cannot

lead the mobile robot through the continuum of voids since its SLAM map causes an isolated region on Figure 49 (left) represented by the black circle. Hence the percolator considers that tunnel as a dead end as seen in Figure 49 and starts to seek new guidance. We have shown those guidance directions with following black and white marker sequences that is seen on the percolator map given with Figure 49 (left). Black markers correspond to available paths while white ones correspond to unavailable paths. So the percolator guided mobile robot selects a reverse direction as shown with white arrow meaning the next action bearing while black arrow corresponds to current robot bearing. On the other hand, SLAM map of the experiment that is given in Figure 48 with $\alpha = 0.6$, has void cells at the bottleneck which is represented by the black circle on Figure 50(right) and those void cells are the ones on percolator map guidance given by black circle on the map at the left in Figure 50. Percolator map in Figure 50(left) has invaded regions at the black circled bottleneck represented with red colored grid cells. Since red colored grid cells in percolator map correspond to available regions for mobile robot motion, percolator map in Figure 50 (left) leads mobile robot through the tunnel as seen in Figure 48 at time step 13. Therefore percolator map region in Figure 50 (left) with black circle is red colored invaded part as a consequence of void regions in SLAM map circled in Figure 50 (right)and correspond to regions that are available for robot motions. The mobile robot continues its current direction toward that direction of Figure 50. Therefore we have compared decision making phase of two percolator based exploration methodology experiments by their percolator maps given by Figure 49 and Figure 50. It is a noticeable point that their SLAM maps affect percolator maps that determine mobile robot guidance. Since larger obstacle width parameter α has been used in Figure 49, exaggerated mapped obstacle cells prevent to percolate through the tunnel in Figure 49 while percolation continues in Figure 50 by appropriate obstacle width α value selection. So we have represented how percolator depends upon SLAM map and its mapping parameter. Let us take the experiment that is given in Figure 48 and apply angular and linear odometry errors with percentage of 0.1 at each step. So we will analyze how our percolator exploration guidance is affected from injected odometry noise as represented in Figure 51. It is noticed that injected odometry noise can drop coverage performance according to the noise free case. Due to those intensive odometry errors injected as 0.1 percentages at each time step, our mobile robot localization faults mislead obtained SLAM maps. Since such a faulty SLAM map, our percolator guidance cannot be reliable as seen in the situation given in Figure 49 and as a consequence of that fact; our mobile

robot has a stuck condition in the dead end of simulation environment as seen from Figure 51 bottom right.

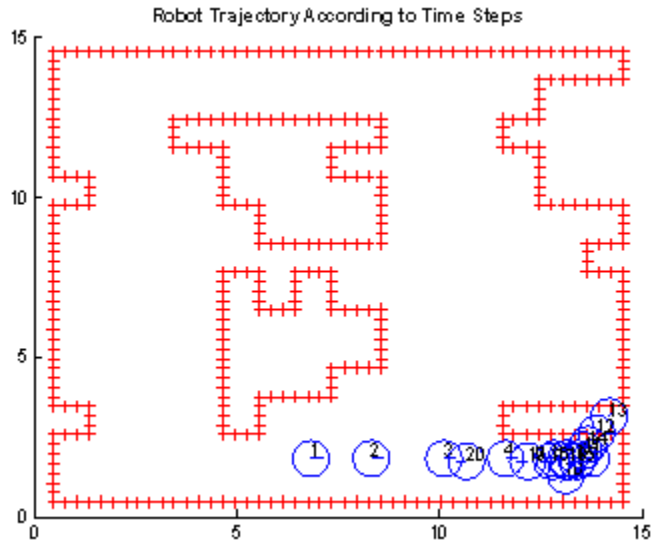

Figure 51 Percolator Based Exploration Guided Mobile Robot Trajectory with Injected Odometry Noise

But in order to handle such unwanted failures to noise that our purely applied percolator guided exploration can encounter and not solve, we have developed our proposed percolator enhanced methodology that can handle those abnormal disruptive situations. We will run comparisons of both methodologies in section 5.7.

5.6 Percolator Enhanced Entropy Based Exploration

We have discussed sub optimality of purely applied percolator based exploration methodology versus the classical entropy based exploration methodology in the previous sections. In order to integrate the advantages in both methodologies we chose to use entropy based and percolator based explorations in a switching mode. We have explained the details of the switching mode hybrid methodology in section 3.3. In this hybrid methodology, we can say that exploration activities are driven by percolator based guidance while exploitation actions are driven by entropy based exploration methodology. Hence we expect the switching mode to be more reliable and robust against odometry measurement noises when compared to purely applied percolator based exploration methodology and to have more enhanced coverage capability when compared to the entropy based exploration methodology. Hence we use entropy based exploration methodology to maintain map accuracy and position awareness but in the meanwhile

switching mode utilizes percolator based exploration guidance to speed up map coverage through the simulation environment. We have realized sensitivity analyses in Table 21and Table 22. As is seen from Table 21, we have carried out the localization performance analyses of percolator enhanced entropy based exploration technique while coverage performances are compared in Table 22. Those experiments are realized with 20 particles, 9 proximity sensor and maximum sensor range as 2br through 10 time steps. On the other hand, 0.1 percentages rated angular error has been injected to odometry measurements within the implementation of those experiments in Table 21 and Table 22. As is seen from Table 21, PEE_Exp_1 has reached approximately 0 particle set variance with particle weight values of 0.98. While we decreased the obstacle width value as in PEE_Exp_2, our mobile robot reaches particle variance of 1.96 with most reliable particle weight 0.52 valued. Increasing particle set variance, namely position entropy level; correspond to the increment of the localization uncertainty. That decrement of the obstacle widths in PEE_Exp_2 causes lower existence possibilities of obstacles in SLAM map that are used for ray tracing compared to PEE_Exp_1. So the absence of obstacle cells in SLAM map leads to worsen localization certainty in PEE_Exp_2 compared to PEE_Exp_1. On the other hand, increasing the grid cell sizes of SLAM maps causes to worsen localization in a similar way as for the percolator based results of section 5.5. Therefore coarse grid cells involve less information with respect to finer ones as expected. As a consequence of that, better localization can be realized with finer grid cell sizes. This is attested based on results by PEE_Exp_3 with 0.7 grid cell size. Its most reliable particle weight is 0.05 while PEE_Exp_1 and PEE_Exp_2 have larger particle weight values and smaller particle set variance. That means the betterment of robot localization with narrower particle set variance. On the other hand, we have also applied motion model with larger variances as seen from PEE_Exp_5 motion model parameter columns. Motion model parameters $\alpha_3$and α_4 in Table 22 correspond to the linear error sample generation variance as explained in section 2.2.1.2.We have set $\alpha_3$and α_4 motion model parameters with larger variance as 0.5 for PEE_Exp_4 while using $\alpha_3$and α_4 motion model parameters set as 0.1.Increase of sampling variance also decreases mobile robot localization. PEE_Exp_4 has most reliable particle weight values of 0.82 with decreased particle set variance compared to PEE_Exp_2. Hence increasing the variance of motion model sampling leads to better localization performance (PEE_Exp_4) than other trial (PEE_Exp_2) that has finer grid cell size. On the other hand, coverage

performances differ with respect to the tuning between occupancy grid cell sizes and α width parameter values.

Table 21 Percolator Enhanced Entropy Based Exploration (PEEBE) Sensitivity Analyses for Map Accuracy and Robot Position Awareness

Initial Simulation Parameters												
Total Time Step										**10**		
Particle Number										**20**		
Sensor Number										**9**		
Maximum Proximity Sensor Range										**2br**		
Sensitivity Analyses of Map Accuracy and Position Awareness Performance												
Experiment Name	**α Possible Obstacle Width**	**Occupancy Grid Map Cell Size**	**Sensor Cone Width**	**Motion Model Parameters**				**Injected Noise Parameter**		**Exploration Methodology**	**Most Reliable Particle Weight Value**	**Particle Set Variance(Position Entropy Level)**
				α_1	α_2	α_3	α_4	e_L	e_A			
PEE_Exp_1	1.2	0.5	40	0	0	0.1	0.1	0	0.1	PEEBE	0.98	0
PEE_Exp_2	1	0.5	40	0	0	0.1	0.1	0	0.1	PEEBE	0.52	1.946
PEE_Exp_3	1.5	0.7	40	0	0	0.1	0.1	0	0.1	PEEBE	0.05	2.96
PEE_Exp_4	2	1	40	0	0	0.5	0.5	0	0.1	PEEBE	0.82	1.386

Table 22 Percolator Enhanced Entropy Based Exploration (PEEBE) Sensitivity Analyses for Coverage Performance

Initial Simulation Parameters									
Total Time Step								**10**	
Particle Number								**20**	
Sensor Number								**9**	
Maximum Proximity Sensor Range								**2br**	
Sensitivity Analyses of Coverage Performance									
Experiment Name	**α Possible Obstacle Width**	**Occupancy Grid Map Cell Size**	**Sensor Cone Width**	**Motion Model Parameters**				**Exploration Methodology**	**Coverage Area(br^2)**
				α_1	α_2	α_3	α_4		
PEE_Exp_1	1.2	0.5	40	0	0	0.1	0.1	PEEBE	22.5 br^2
PEE_Exp_2	1	0.5	40	0	0	0.1	0.1	PEEBE	23 br^2
PEE_Exp_3	1.5	0.7	40	0	0	0.1	0.1	PEEBE	40 br^2
PEE_Exp_4	2	1	40	0	0	0.5	0.5	PEEBE	21br^2

Larger occupancy grid cells size leads the percolator to explore more connected voids from within the percolator enhanced entropy based exploration methodology as seen from the comparisons of trials of PEE_Exp_3 with PEE_Exp_2 and PEE_Exp_1. Similar

to the other comparisons in sections 5.5. and 5.4, bad tuning between possible obstacle width and occupancy grid map cell size causes to assign smaller number of void cells with respect to PPE_Exp_3 so that its exploration performance through the connected voids decreased in PEE_Exp_4 versus PEE_Exp_3.

5.7 Comparative Performance Analyses of Novel and Classical Exploration Approaches

In this section we compare the performance results of purely percolator based exploration, entropy based exploration and percolator enhanced entropy based exploration methodologies under noisy odometry measurements. We have realized our comparison over 20 time steps keeping motion model and mapping parameters, constant for all three approaches. Firstly, we start from purely applied percolator based exploration results. As is seen from Figure 52, our mobile robot has navigated through the found connected voids in a straightforward direction. Numbers written within the circles represent the iteration number, where circles represent the location of the robot at that instant.

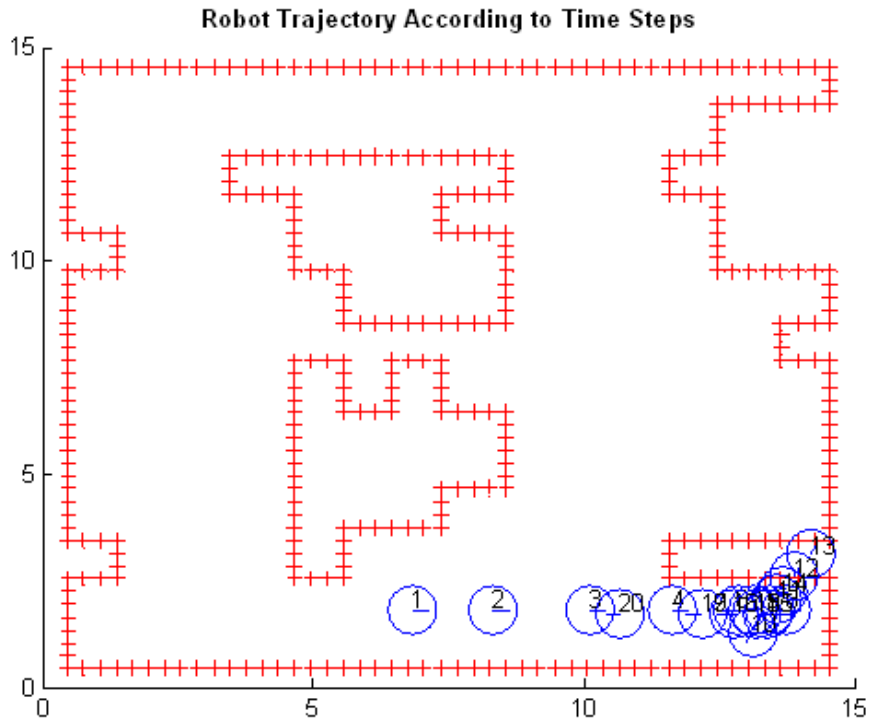

Figure 52 Purely Applied Percolator Based Exploration

It is seen that mobile robot has navigated towards the "cul de sac" to the bottom right of Figure 52 from time step 4 to 20 and got trapped. We notice here that purely applied percolator exploration guidance has quickly converged the mobile robot to the dead end through connected voids very speedily in 4 steps. But after the 4 th time step, purely applied percolator algorithm cannot recover its guidance to enable escape from that dead end consequently the robot has demonstrated that it cannot navigate through the more available and wide regions of the environment , when guided by the percolator. On the other hand, entropy based exploration methodology as classical approach is also applied to our mobile robot within the same initial conditions. As is seen from the Figure 53, entropy based exploration methodology leads our mobile robot to navigate through an exploitation random walk around the initial point rather than directing itself towards a new exploration direction at the first step.

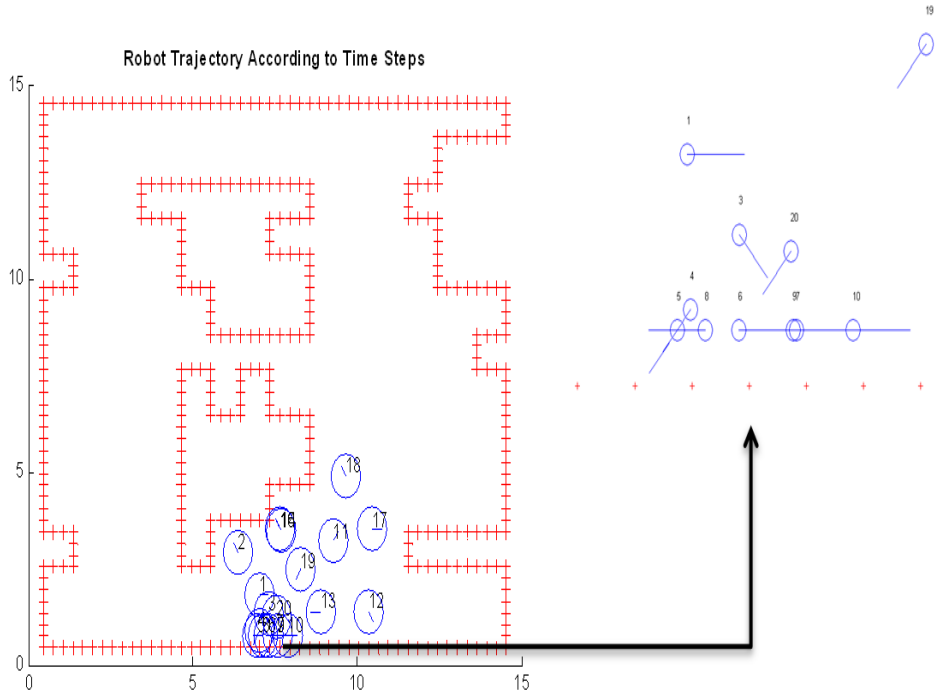

Figure 53 Entropy Based Exploration

At the end of the black bold arrow on the right of the figure, one can find the enlarged behavior of the robot at the crowded spot of the Figure 53 itself. Entropy based exploration guided mobile robot is found to exploit unknown regions by roaming its surrounding rather than exploring new distant sites. Therefore its localization information is improved within the roamed regions due to data acquired from similar regions. For this

reason, entropy based exploration guided mobile robot undergoes stuck situation shortly with respect to purely applied percolator based exploration guided mobile robot in Figure 52.

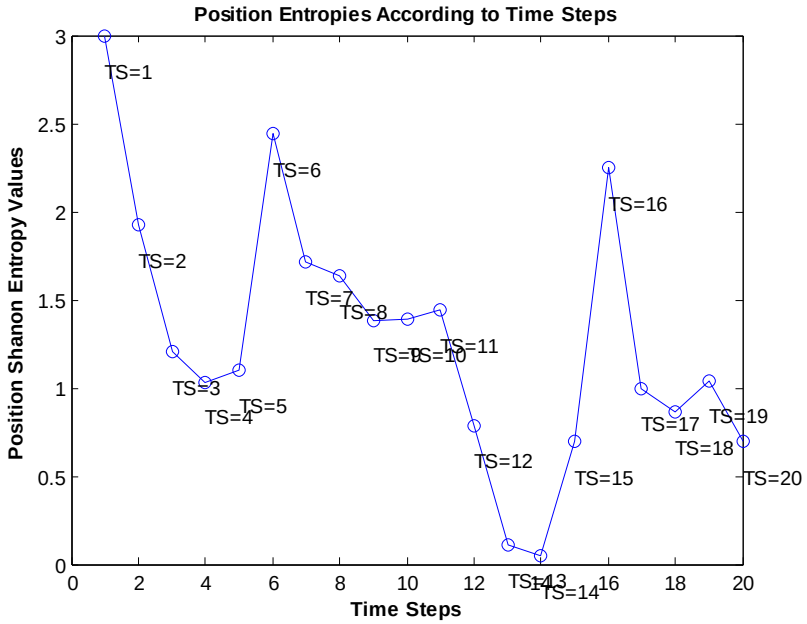

Figure 54 Entropy Based Exploration Guided Mobile Robot Position Entropy Values through Time Steps

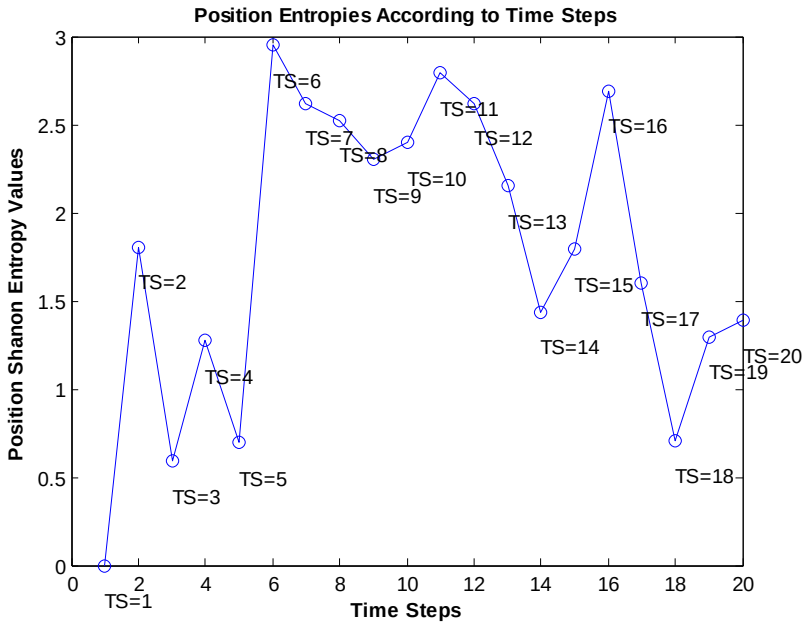

Figure 55 Purely Applied Percolator Based Exploration Guided Mobile Robot Position Entropy Values through Time Steps

Hence entropy based exploration guided mobile robot shown in Figure 53 exploiting its surrounding while purely applied percolator exploration guided one shown in Figure 52 explored other sites and even dead ends , but unfortunately getting stuck there. Let us compare now position entropy levels and coverage speeds of purely applied percolator based exploration guided mobile robot navigation in Figure 52 and the ones of entropy based exploration guided method of Figure 53. As is seen from the differences on Figure 54 and Figure 55, localization behavior of entropy based exploration guided mobile robot in Figure 54 has dramatic decay in the position uncertainty (i.e. 2-4 and 6-14 time step intervals) versus that of percolator based exploration guided mobile robot uncertainty levels in Figure 55 leading to better localization accuracy in the first classical method in the time steps 6 and 14. In summary, entropy based exploration guided mobile robot has lower position uncertainty levels compared to those of the purely applied percolator based method. When we compare the coverage speeds of both exploration methods by plotting obtained SLAM map coverage of void cells with respect to time steps. We notice from the Figure 56 and Figure 57 that entropy based exploration guided mobile robot undergoes short stuck situation in the 18-19, 16-17 time step intervals, but purely applied percolator based exploration guided mobile robot is more adventurous and tends to suddenly go further around from its current position to increase sharply and suddenly coverage of the area. Therefore this type of navigation not being cautious as for the entropy based method can run into longer and dramatic stuck conditions such as in the 3-10, 13-16 time step intervals. Furthermore, coverage level decreasing after time step 16 in Figure 57 represents the deterioration of SLAM map accuracy thus degrading coverage. On the other hand, the horizontal axis in Figure 56 and Figure 57 corresponds to the elapsed time during the computation of the exploration that in fact informs us about the computational burden in the corresponding exploration algorithm. As is seen from Figure 56, coverage levels of entropy based exploration guided mobile robots are increased through time steps (TS) more quickly than percolator based exploration guided one in Figure 57. But it is seen that percolator based exploration guided mobile robot computations are performed in shorter time intervals than entropy based one in Figure 56 rise time of the coverage in purely percolation guided SLAM is much shorter than classical entropy based one, therefore the percolation guidance provide the robot swiftness in response and more curiosity in exploring unentered regions, which a critical assest for search and rescue. For instance, entropy based exploration guided mobile robot reaches coverage level of 32 br^2 at the 12 th time step while percolator based exploration

guided one reaches 17 br^2at that time step. On the other hand, entropy based exploration guided mobile robot completed its 20 time step in 270 seconds approximately while purely applied percolator based exploration guided mobile robot completed its 20 time step in 147 seconds approximately.

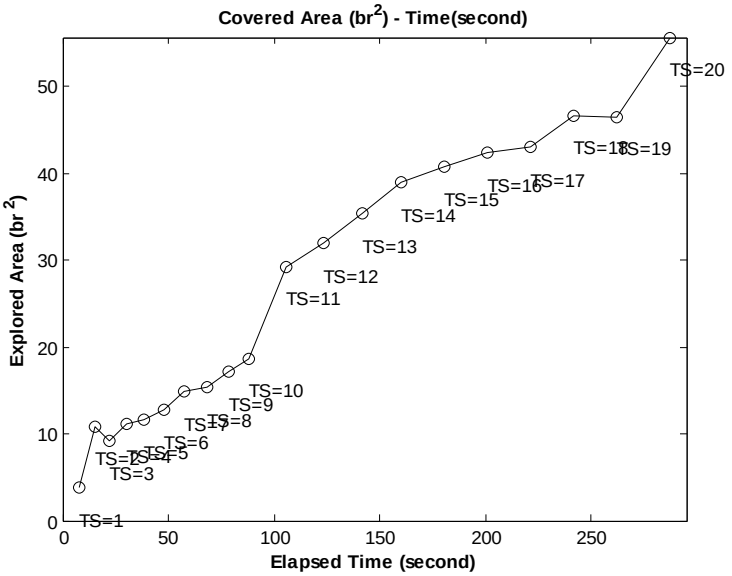

Figure 56 Entropy Based Exploration Guided Mobile Robot Coverage Levels

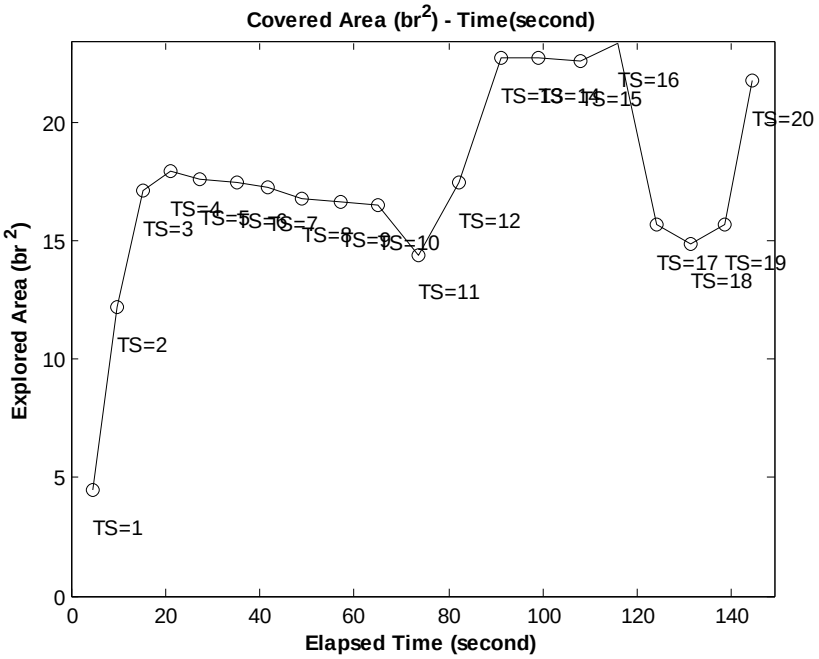

Figure 57 Purely Applied Percolator Based Exploration Guided Mobile Robot Coverage Levels

That interpretation shows us that percolator based exploration methodology lower the computational burden on the mobile robot decision making processes. On the other hand, purely applied percolator based exploration methodology has more impressive coverage speed at the initial time steps before the discussed stuck condition due to the localization inaccuracies. We observe that impressive coverage speed at 1-3 time steps interval in Figure 57. We notice that purely applied percolator based exploration guided mobile robot reaches 17 br^2 coverage within 3 time step and 20 seconds approximately while entropy based exploration guided one reaches that coverage level in 8 time step and 70 seconds approximately. That shows clearly the potential of the percolator based methodology to reach and explore connected voids quickly with less computational burden. But its performance cannot be observed until 20 time step due to the deterioration in mobile robot localization.

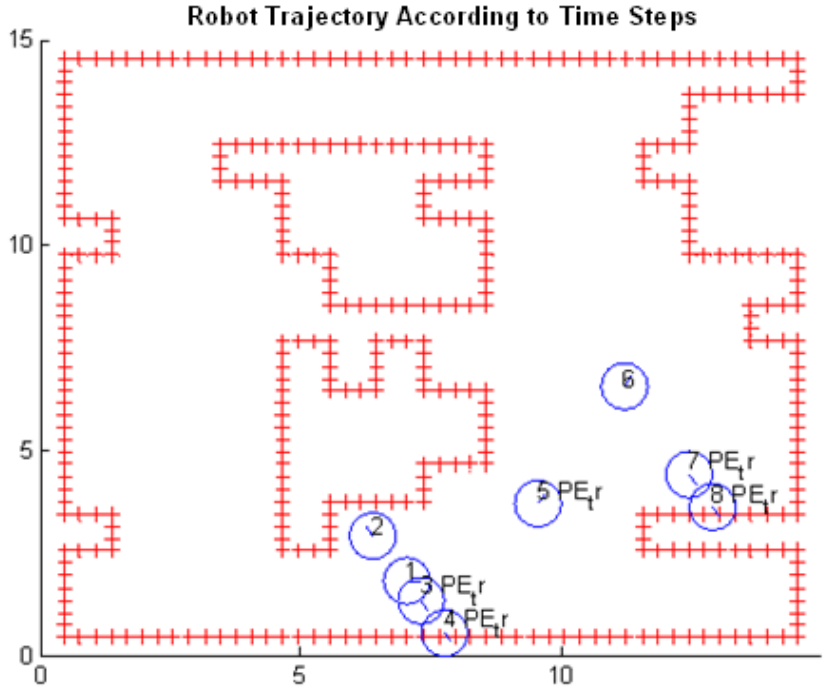

Figure 58 Percolator Enhanced Entropy Based Exploration

Since percolator only aims to predict connected voids through the upcoming unknown environment, it is not equipped with any instrument to recover position uncertainty during navigating mobile robot into unknown regions. In order to equip this methodology with a localization recovery tool, we have developed a switching methodology that

utilizes the localization part of classical entropy based exploration methodology and the exploration part of our percolator based exploration methodology within a hybrid system called percolator enhanced entropy based exploration. As is seen from Figure 58, we applied the identical previous initial conditions to the percolator enhanced entropy based exploration methodology as well.

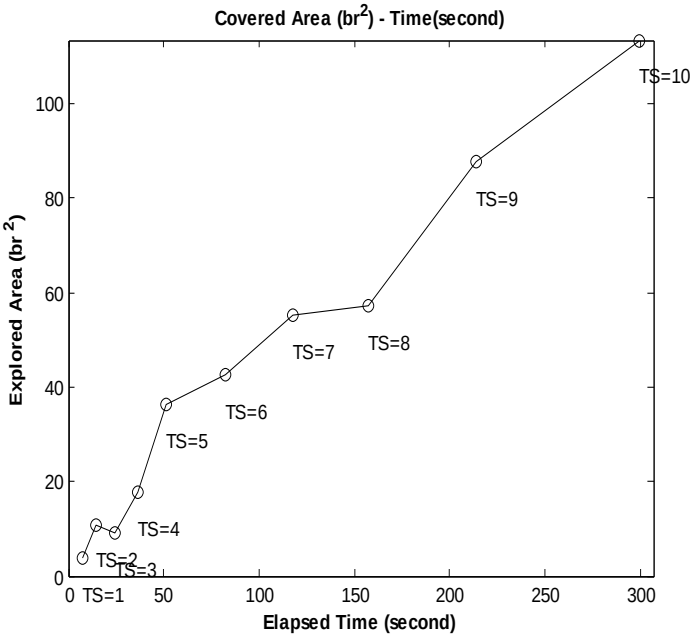

Figure 59 Percolator Enhanced Entropy Based Exploration Guided Mobile Robot Coverage Levels

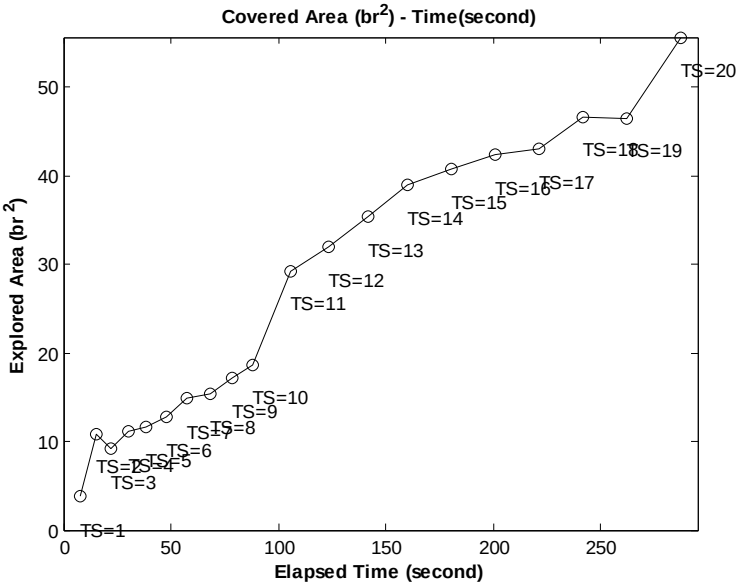

Figure 60 Entropy Based Exploration Guided Mobile Robot Coverage Levels

We have represented the percolator enhanced entropy based exploration guided mobile robot navigation in Figure 58 until time step 8 since a collision with a wall is detected at that time step. In this experimental run, collision avoidance is disabled to be able to compare the 2 novel approaches and the classical one in equivalent terms in terms of computational burden. We will then enable the collision avoidance mechanism of our mobile robot navigation algorithms at the later comparisons at the end of this subsection 5.7. We are now comparing our percolator enhanced entropy based exploration guided mobile robot navigation of Figure 58 with the classical entropy based exploration guided mobile robot navigation performance of Figure 53. We base our comparison on position uncertainty and coverage performances. As is seen from the comparison of Figure 59 and Figure 60, the percolator enhanced entropy based exploration guided mobile robot is showing better coverage performance with respect to the classical entropy based exploration guided mobile robot. The percolator enhanced entropy exploration reaches approximately 60 br^2 coverage area at time step 8 while entropy based exploration based mobile robot reaches that coverage levels at 20 the time step. On the other hand, the percolator enhanced entropy based exploration methodology reaches time step 10 in 300 seconds while entropy based exploration implementation reaches 20 time step within the same time interval. This means that computational burden of percolator enhanced entropy based exploration is larger than classical entropy based one. Moreover the percolator enhanced exploration methodology reaches 60 br^2 in 160 seconds while classical entropy based one reaches approximately that value in 300 seconds. This represents the impressive performance of percolator enhanced entropy based exploration methodology over the classical approach. On the other hand, we neglect to interpret other time steps after time step 8 in Figure 59 since a collision has occurred during the simulation of percolator enhanced entropy based methodology. Since mobile robot cannot sense the wall in Figure 59, it continues to its path through walls. Therefore we consider the simulation of percolator enhanced entropy based exploration until time step 8. Although we have a collision avoidance mechanism for the mobile robot keeping it from hitting walls or passing through them, we have realized our experiments of Figure 52, Figure 53, Figure 58 without considering any collision avoidances. We have only compared methods for their coverage and localization performances and investigated the stuck points during navigation. As is seen from the comparison of Figure 61 and Figure 62, the percolator enhanced entropy based exploration guided mobile robot demonstrates better localization performance compared to the classical entropy based approach. There,

mobile robot position uncertainty decrement levels are more dramatically realized within the percolator enhanced entropy based exploration guided mobile robot navigation as seen in Figure 61.

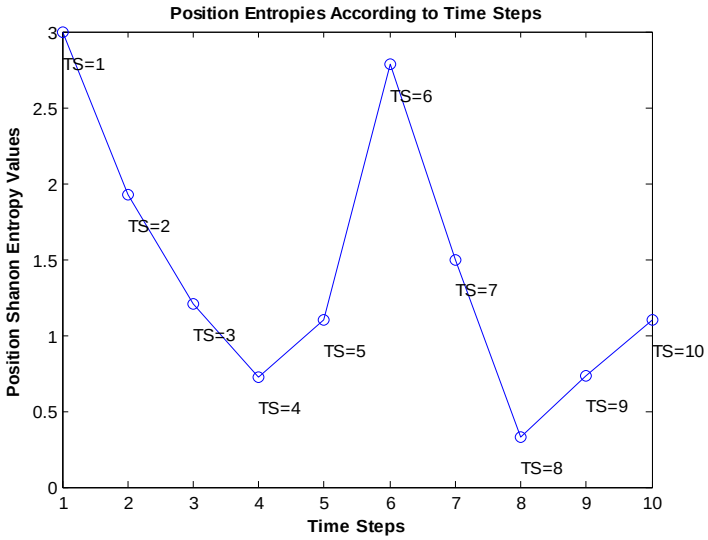

Figure 61 Percolator Enhanced Entropy Based Exploration Guided Mobile Robot Localization Levels

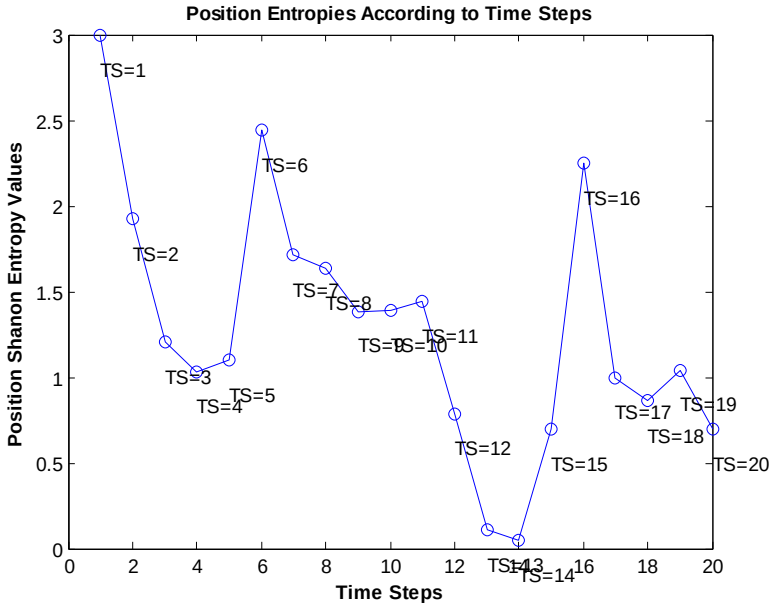

Figure 62 Entropy Based Exploration Guided Mobile Robot Localization Levels

A considerable increment of the localization recovery can be seen at time step intervals between 1st and 4th time steps and 6th and 8th time steps in Figure 61. On the other side, there is no any tremendous change within the localization level decrements of Figure 62. As we recall from the coverage comparison, we neglect the time steps beyond the time step 8 in Figure 61 due to collision. As a comparison of Figure 61 and Figure 62, we have seen that percolator enhanced entropy based exploration methodology has a much better localization performance than the classical one.

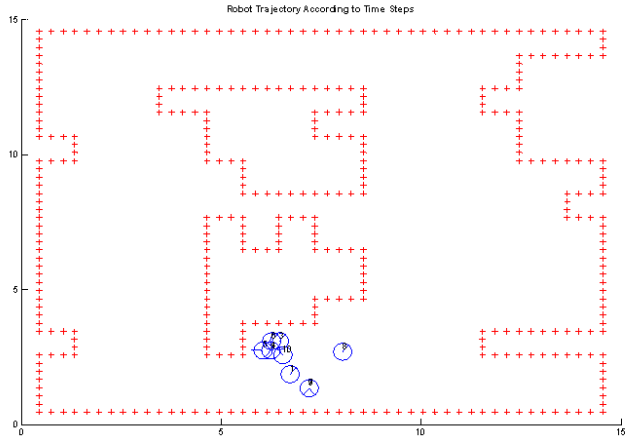

Figure 63 Entropy Based Exploration Guided Mobile Robot with Collision Detector

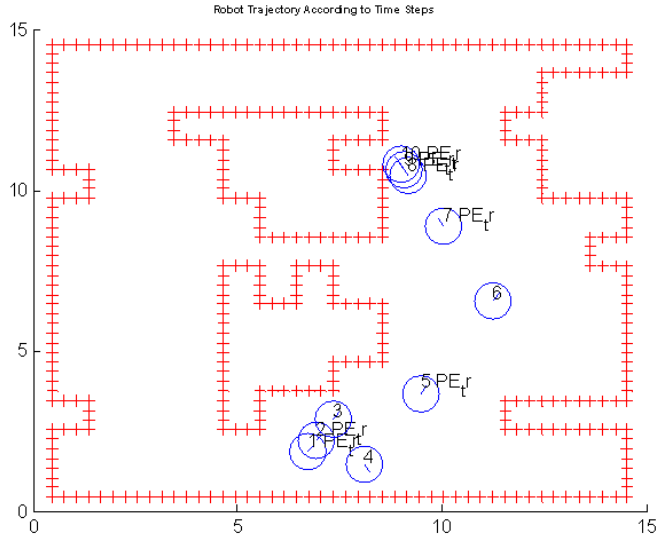

Figure 64 Percolator Enhanced Entropy Based Exploration Guided Mobile Robot with Collision Detector

Since we have applied our exploration methodologies without collision avoidance in the experiments of Figure 52,Figure 53 and Figure 58.We now repeat those enabling the collision avoidance mechanism. Collision avoidance in our simulator is based on the detection of any obstacle or wall part by proximity sensor in all directions during navigation. When a wall or corner pieces have been detected then collision detector stops the mobile robot in order to seek to a new guidance. Using collision avoidance indirectly restricts the guided robot actions for passing through walls without hitting any one in our simulation environment. The obtained coverage performance comparisons for the percolator enhanced entropy based exploration and the entropy based exploration implementations both equipped with collision avoidance mechanism are given in Figure 65 and Figure 66. As is seen from the comparison of Figure 65 and Figure 66, the percolator enhanced entropy based exploration guided mobile robot reaches coverage level of 64 br^2 while classical entropy based exploration guided one navigates an area of 27 br^2 at the same time step interval. On the other hand, the horizontal axis in Figure 65 and Figure 66 corresponding to the computational times for those exploration algorithms, we see that the computational time of our percolator based methodology is larger than the classical based one. But in real applications total time step does not only depend upon computational processes.

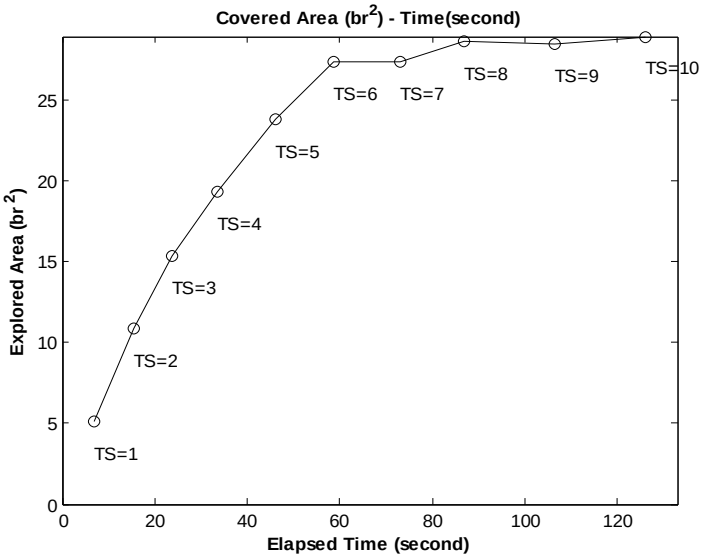

Figure 65 Entropy Based Exploration Guided Mobile Robot Coverage Levels with Collision Detector

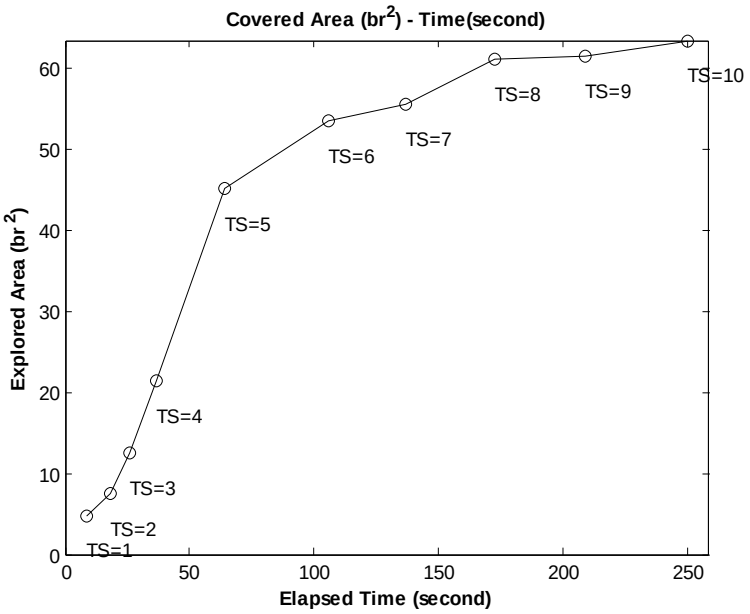

Figure 66 Percolator Enhanced Entropy Based Exploration Guided Mobile Robot Coverage Levels with Collision Detector

Displacement through the mobile robot navigation avoiding obstacles consumes mobile robot's time in reaching targets. Based on this number of time steps is more important and determinative than interval between time steps that reflect the computational burden at those steps. Hence our algorithm requires less time step than the classical approach so we can infer from that we can reach survivors trapped under the disaster ruins much more quickly. On the other hand, we also compare localization performances of percolator enhanced and classical based entropy explorations. As is seen from Figure 67 and Figure 68, percolator enhanced entropy based methodology reaches lower uncertainty levels than entropy based methodology. While the entropy based exploration methodology localizes robot position at the initial time steps of its simulation, the percolator enhanced one localizes its position after time step 6. Until time step 6 percolator enhanced entropy based exploration guided robot tends to weight its actions selections to explorer new regions. Meanwhile classical based approach in Figure 68 tends to decrease its position uncertainty with roaming around its initialized point as is seen in Figure 63. This behavior of "curiosity" is a characteristic passed through by the percolation guidance. For the entropy based exploration, as a consequence of that action selection, the robot position uncertainty level is dramatically, decreased at those initial steps. But percolator enhanced entropy based methodology putting preference on exploring new regions, and

only localizing its position sparsely at the initial time steps. We can observe those initial time steps until 6th time step in Figure 67 and Figure 68.

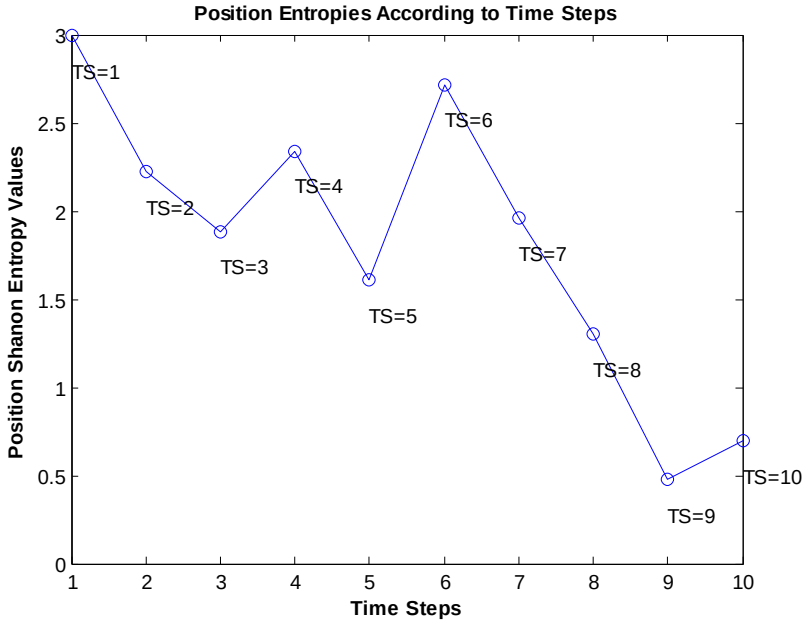

Figure 67 Percolator Enhanced Entropy Based Exploration Guided Mobile Robot Position Entropy Levels

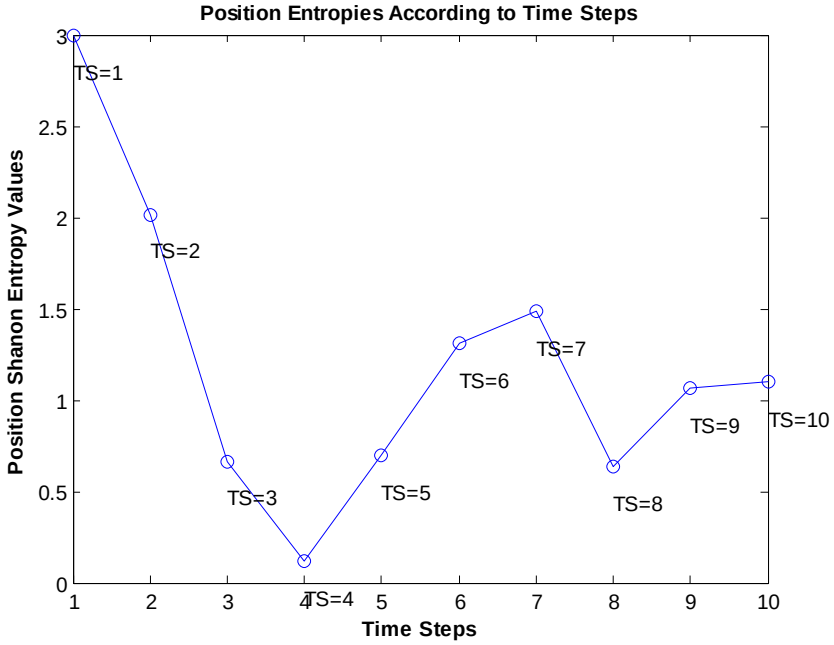

Figure 68 Entropy Based Exploration Guided Mobile Robot Position Entropy Levels

But after 6^{th} time step the percolator enhanced methodology decreases the robot position uncertainty levels more than the classical entropy based approach. Since obtained SLAM map of percolator enhanced entropy based methodology have more details for localization with respect to the classical entropy based exploration approach, percolator enhanced localization in Figure 67 is more successful than classical entropy based approach in Figure 68. Therefore our percolator enhanced modification on the classical entropy based approach leads the hybrid system to have better exploration performance while maintaining high accuracy in its localization performance.

CHAPTER 6

CONCLUSION

6. Conclusion

According to the simulation analyses of both our proposed methodologies, we have drawn a general conclusion in section 6.1 from the interpretations of the sensitivity analyses obtained from experiment results in section 5.4, section 5.5, section 5.6 and section 57. As an expansion to those drawn conclusions, we propose alternative future works related to problems handled within the balance of our thesis work.

6.1 Experiment Analyses Conclusions

Map and localization accuracies have been concerns that led to slowing down the SLAM process and to being more conservative in the coverage actions. Speed and poor coverage have been the major drawbacks of classical entropy based SLAM methodologies when tried to be used for search and rescue. Coverage and speed are critical in rescue operations without getting stuck at dead ends of unstructured environments. Thus connected voids are of the guides to search and rescue robot navigation, where in this disaster environment odometry measurements are also poor. Mapping the disaster area with suitable accuracy is also critical feedback that rescuers expect from robots sent for search in the disaster area. Hence we have investigated map accuracy and position awareness performances of exploration algorithms according to their odometry command guidance instead and their mapping parameters. We have witnessed to the localization deficiencies during sensitivity analyses of purely applied percolator based methodology since their actions tend to explore new regions that weaken the robot position awareness, while the pure percolator based SLAM shows superiority. In order to provide better

localized odometry commands to the SLAM mechanisms we chose to pair the percolator with entropy based method, collaborating in a switching mode Then we witnessed in our study that percolator enhanced and classical entropy based methodologies proposed better localized odometry commands regarding purely applied percolator based exploration methodology. Mobile robot position awareness is necessary for our exploration activity due to its effect on the map accuracy. Since inaccurate map of explored environment is useless to reach possible survivors, we have to sacrifice from noticeable coverage performance of the percolator based exploration methodology on behalf of obtaining accurate maps that refer correct routing through the unknown environment for subsequent followers. Coverage performance is another our primary issue in our thesis study since we aim to reach voids as soon as possible while maintaining position awareness through the unknown environment. We have seen that percolator guided exploration methodologies are greedy to reach connected voids. But accuracy of their maps cannot be maintained in the long run of SLAM activities during mobile robot exploration. But their exploration performances are noticeable high when coverage is analyzed sensitivity analyses when compared to entropy based methodologies. This coverage superiority has been coupled to high map accuracy and position awareness performances when percolator guidance has been coupled in a switching mode to the entropy based SLAM. Moreover, this percolator enhanced entropy based SLAM has been found to significantly when compared to entropy based methodology alone. So the switching mode in the percolator enhanced entropy based methodology has lead to more effective robot position recovery then the purely applied percolator based exploration methodology since switching mode methodology utilizes exploitation actions arisen from entropy based exploration methodology so as to minimize position uncertainty. On the other hand, this switching composition can be considered as an enhancement of exploration action proposals, selected by entropy based trade off mechanism to minimize either position or map uncertainty. Hence our percolator guidance is a suboptimal speedy coverage method related with the continuity of explored voids that does not guarantee to be more effective localization at each trial like in entropy based methodology. Appropriate environments that consisting of connected voids such as in search and rescue are useful platforms to show the efficiency of percolator guidance contributions versus the slow and insufficiency in coverage of the entropy based methodology. On the other hand, for more scattered environments deprived from connected voids, there will be hardly noticeable differences in coverage

performances between percolator enhanced and classical entropy based exploration methodologies. As a consequence that percolator enhanced entropy based exploration can be thought as ad hoc solutions for the environments consisting of connected voids or tunnels. Since such structures naturally existed within disaster environments, our percolator enhanced switching mode methodology is highly promising with effective performance to reach possible survivors at the end of the connected tunnels.

6.2 Future Work

As a consequence of our thesis work, we develop a novel mobile robot exploration methodology that enhance search and rescue reconnaissance missions in the objective of reaching trapped survivors under the disaster ruins. We proposed our exploration methodologies on the metric based map representations with obtained sonar proximity sensor models. Since metric based representations are not realistic ways for particle filter based SLAM computations, feature based representations of our proposed exploration methods need to be devised and a better computational performance need to be obtained. More complicated feature extraction methodologies can also be devised in accordance with irregular disaster environments. Therefore different sensing types such as camera, laser, infrared and RF based sensors can be utilized other than sonar based ones to extract distinguishable features from irregular disaster environments. Our approach in the scope of this thesis work consists of theoretical proposals and their performance analyses implementations within a simulator environment. For a future work, our theoretical proposals need to be implemented in real environments with equipped mobile robots. The proposed methodologies should also be tried on SAR swarms mapping the disaster area in 3D using sensors at multiple levels of valuation. Those experimental activities can be realized with the feedback of search and rescue professionals that are come up with their practical advices and critics on the mobile robot navigation. More progressive systems consisting of hardware improvements such as using FPGA or Power PC processors and software enhanced by such parallel computing techniques can be utilized so as to improve by far the computational facility of our simulator. Embedded guided SLAM systems should also be developed that guides real mobile robot systems for practical onboard data acquisition, mapping, area monitoring experiments using our proposed active guided SLAM methodologies.

REFERENCES

1. **Murphy, R. R,** *Search and Rescue Robotics,* Chapter 5, Handbook of Robotics, Springer Verlag, 2008.

2. **Murphy, K. P.,** "*Bayesian Map Learning in Dynamic Environments*", Proceedings of Neural Information Processing Systems Conference , USA, 1999.

3. **Doucet, A., Godsill, S., Andrieu, C.** "*On Sequential Monte Carlo Sampling Methods for Bayesian Filtering*", Journal of Statistics and Computing, Volume 10, Issue 3, Pages:197-208, Springer Verlag, 1999.

4. **Liu, J. S.,Chen , R.,** "*Sequential Monte Carlo Methods for Dynamic Systems",* Journal of the American Statistical Association, Volume 93, Pages:1032-1044, ASA, 1998.

5. **Koenig, S. and Tovey, C.,** "*Improved analysis of greedy mapping",* Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, USA, 2003.

6. **WeiB, G., Wetzler, C. and von Puttkamer, E.,** "*Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans*". Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Germany, 1994

7.**Mataric, M.,** "*Integration of representation into goal-driven behavior-based robots",* IEEE Transactions on Robotics and Automation, Volume 8, Issue 3, Pages:304-312, IEEE Press, 1992

8. **Thrun, S., Bücken, A.,** "*Integrating grid-based and topological maps for mobile robot navigation",* Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, 1996.

9.**Moravec, H. and Elfes, A.,** "*High Resolution maps from wide angle sonar",* Proceedings of the IEEE International Conference on Robotics and Automation, UK, 1985.

10.**Yamauchi, B.,** "*Frontier-based exploration using multiple robots*", Proceedings of the International Conference on Autonomous Agents, USA, 1998.

11. **Makarenko, A.A.,**"*An experiment in integrated exploration*", Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Switzerland, 2002.

12. **Stachniss, C., Hahnel, D. and Burgard, W.,**"*Exploration with active loop closing for fast SLAM* "**,** Proceedings of IEEE/RSJ International Conferences on Intelligent Robots and Systems, Japan, 2004.

13. **Montemerlo, M.,**"*Fast SLAM : A factored solution to the simultaneous localization and mapping problem*"**,** Proceedings of the AAAI National Conference on Artificial Intelligemce, UK, 2002.

14. **Thrun, S., Wolfram, B. and Fox, D.,** *Probabilistic Robotics*, MIT Press, 2005.

15.**Stanley,H.E., Andrade,J.S., Havlin,S., Makse,H.A., Suki B.,.** "*Percolation phenomena: a broad-brush introduction with some recent applications to porous media,liquid water and city growth-test of a percolation model*", Journal of Physics A: Mathematical and General, Volume 266, Issue 1, Pages: 5-16, Elsevier, 1999

16. **Stauffer, D and Aharony, A.,** *An Introduction to Percolation Theory*. Taylor and Francis, 1994.

17. **Makse, H.A,** "*Modelling urban growth patterns with correlated percolation"*. Physics Reviews, Volume 58, Issue 8, Pages:7054-7062, Elsevier, 1998.

18. **Gouyet, J.F and Rosso, M.,** "*Diffusion fronts and gradient percolation".,* Journal of Physics A: Mathematical and General, Volume 357, Issue 1, Pages: 86-96, Elsevier,2005.

19. **Wilkinson, D. and Willemsem, J.F.** "*Invasion percolation: a new form of percolation theory*".Journal of Physics A: Mathematical and General, Volume 16, Issue 14, Pages: 3365-3376, Elsevier, 1983.

20 **Stachniss, C., Grisetti, G. and Burgard, W.** "*Information gain-based exploration using Rao-Blackwellized particle filter*", Proceedings of Robotics: Science and Systems, USA, 2005.

21 **Soares, R. F., Corso, G., Lucena, L. S., Freitas J. E, Da Silva,L. R., Paul, G., and Stanley, H. E,** "*Distribution of shortest paths at percolation threshold: Application to oil recovery with multiple wells*", Journal of Physics A: Mathematical and General, Volume 343, Issue 1, Pages: 739-747 , Elsevier, 2004