2013

A Novel Approach To Intelligent Navigation Of A Mobile Robot In A Dynamic And Cluttered Indoor Environment

Daniel Opoku
North Carolina Agricultural and Technical State University

A Novel Approach to Intelligent Navigation of a Mobile Robot in a Dynamic and Cluttered Indoor Environment

Daniel Opoku

North Carolina A&T State University

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Department: Electrical and Computer Engineering

Major: Electrical Engineering

Major Professor: Dr. Abdollah Homaifar

Greensboro, North Carolina

2013

School of Graduate Studies
North Carolina Agricultural and Technical State University
This is to certify that the Doctoral Dissertation of

Daniel Opoku

has met the dissertation requirements of
North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2013

Approved by:

Dr. Abdollah Homaifar Major Professor	Dr. Edward W. Tunstel, Jr Co-Advisor
Dr. Albert Esterline Committee Member	Dr. Marwan Bikdash Committee Member
Dr. Numan Dogan Committee Member	Dr. John C. Kelly Department Chairperson

Dr. Sanjiv Sarin
Dean, The Graduate School

Biographical Sketch

Daniel Opoku was born on September 24, 1982 in Tafo-Kwahu, Ghana. He attended St. Peters' Senior High School, Nkwatia-Kwahu, Ghana. He then attended Kwame Nkrumah University of Science and Technology (KNUST) where he received a Bachelor of Science degree in Electrical and Electronic Engineering with First Class Honors in 2007. Following his graduation, he served as a Teaching Assistant to Mr. E. K. Anto in the Electrical and Computer Engineering department at KNUST. In 2008, he enrolled as a direct PhD student at North Carolina Agricultural and Technical State University (NC A&T SU), USA, where he pursued a doctorate in Electrical Engineering under the supervision of Dr. Abdollah Homaifar.

His achievements at NC A&T SU include publication of two research papers (one journal paper, one refereed conference paper) and being a four-time recipient of Wadaran L. Kennedy 4.0 Scholars Award.

Dedication

I would like to dedicate this to my dear mother Ms. Gladys Oforiwaa. Mum, I would like you to know that I have successfully completed this journey only because you succeeded in your duty as a God-fearing mother. I will always love and cherish you.

Acknowledgements

My utmost gratitude goes to our Omnipotent God for His strength that saw me through. I am grateful to my advisor, Dr. Abdollah Homaifar and co-advisor, Dr. E. W. Tunstel for their enormous support and guidance. I am so thankful to the Ph.D. committee members, Dr. Albert Esterline, Dr. M. Bikdash and Dr. N. Dogan; and the Chairperson Dr, John Kelly for their presence in times of need and direction. Without them, this would not have been possible. I would like to thank Dr. Owusu-Ofori and his family for providing help from the get-go of this journey in 2008 and for their hospitality, love and support. I am grateful to Mr. E. K. Anto and Dr. Francis Momade for initiating and encouraging me to take on this journey. I am thankful to my dearest wife, Anna Opoku and my mother, Gladys Oforiwaa, for all their love and care till this day.

My thanks go to Pastor and Mrs. Joseph Gyamfi for their prayers, advice and support throughoutthis research. My thanks also go to my friends Dr. Ruben Buaba, Mr. Ruben Kotoka and Dr. Kwadjo Mensah-Darkwa for their supports. I also acknowledge and appreciate all the love and help from all my colleagues in the Autonomous Control and Information (ACIT) center, Dr. Abrham Workineh, Chaunte Lacewell, Dr. Mohammed Gebril and Dr. Gholamreza Fetanat, and to all the other names I could not include. May all your efforts get rewarded a thousand fold!

Table of Contents

List of Figures

List of Tables

List of Symbols

S	Finite set of all nodes to be searched $\Rightarrow S_b \cup S_f = S$
S_f	Set of all unoccupied/unblocked nodes
S_b	Set of all blocked nodes
S_{Open}	Set of all the nodes that ever make it to the $OPEN$ list, thus $S_{Open} \subseteq S_f$
S_{Closed}	Set of all the nodes on the $CLOSED$ list, thus $S_b \subseteq S_{Closed}$
$n(S)$	Cardinality of set S
s	An individual node, $s \in S$
$c(s', s)$	Cost of traversing from node s to its neighbor s'
s_{Goal}	Goal node
s_{Start}	Starting node
$g(s)$	Cost of moving from s_{Start} to s
$h(s)$	User defined heuristic cost of moving from node s to s_{Goal}
$f(s)$	The estimated cost of a path from s_{Start} to s_{goal} passing through s
$succ(s)$	Set of all the successors of s
$pred(s)$	Set of all the predecessors of s
$parent(s)$	Parent of s, $\Longrightarrow parent(s) \in pred(s)$
C_{free}	Free Region on the configuration space
v	Linear Velocity
ω	Angular velocity
ICR	Instantaneous Center of Rotation

R	Distance from the ICR to the midpoint between two differential wheels
l	Base distance between the two differential drive wheels on a robot
x	The abscissa of a world frame
y	The ordinate a of a world frame
d	Distance
T	Period of a cycle
k_1	The fixed, known distances separating the door-marker sensors on the robot
k_2	The fixed, known distances separating the door-markers on the doorpost
t_1	The time interval between the instances when the door-marker sensor 1 and door-marker sensor 2 respectively detects the same door-marker
t_2	The time interval between the instances when the same door-marker sensor detects door-marker 1 and door-marker 2 respectively
δ	The angle in radian that the moving robot makes with a door-marker
$\boldsymbol{x}$	Represents a state vector
H	An observation model
$\mathcal{N}$	Gaussian noise vector
z	An observation
Q	Noise Covariance
EIR	Estimation Improvement Ratio
rms	Root Mean Square

Abstract

The need and rationale for improved solutions to indoor robot navigation is increasingly driven by the influx of domestic and industrial mobile robots into the market. This research has developed and implemented a novel navigation technique for a mobile robot operating in a cluttered and dynamic indoor environment. It divides the indoor navigation problem into three distinct but interrelated parts, namely, localization, mapping and path planning. The localization part has been addressed using dead-reckoning (odometry). A least squares numerical approach has been used to calibrate the odometer parameters to minimize the effect of systematic errors on the performance, and an intermittent resetting technique, which employs RFID tags placed at known locations in the indoor environment in conjunction with door-markers, has been developed and implemented to mitigate the errors remaining after the calibration.

A mapping technique that employs a laser measurement sensor as the main exteroceptive sensor has been developed and implemented for building a binary occupancy grid map of the environment. A-r-Star pathfinder, a new path planning algorithm that is capable of high performance both in cluttered and sparse environments, has been developed and implemented. Its properties, challenges, and solutions to those challenges have also been highlighted in this research. An incremental version of the A-r-Star has been developed to handle dynamic environments. Simulation experiments highlighting properties and performance of the individual components have been developed and executed using MATLAB. A prototype world has been built using the Webots™ robotic prototyping and 3-D simulation software. An integrated version of the system comprising the localization, mapping and path planning techniques has been executed in this prototype workspace to produce validation results.

CHAPTER 1

Introduction

1.1 Research Scope

Robotic navigation is the act of moving a robot from one place in an environment to another on a collision-free path which may be either a predefined path (e.g., using offline path planning methods such as A^*) or a path defined by maximizing a cost function (such as in artificial potential fields (Guldner et al., 1995; Sfeir et al., 2011)). The process of examining the available information about the environment and computing a path that satisfies one or more conditions and/or constraints (e.g., shortest, collision free, safest path, etc.) is referred to as path planning. At least three main questions need to be answered in a navigation task, namely, *localization* (Where am I?), *mapping* (How does my environment look?) and *path planning* (How do I get to my destination from here without crashing into an obstacle?) (Leonard and Durrant-Whyte, 1991).

Researchers have investigated many ways of solving the navigation problem. Satellite-based and satellite related navigation is well-researched, and the payoff has invariably resulted in different navigation solutions for both commercial and domestic consumption. The most common examples include the Global Positioning System (GPS), which has become a common commodity found in modern communities. This has greatly improved navigation in areas where sufficient satellite coverage is available (commonly called outdoor navigation). However, there exist some environments devoid of sufficient satellite signals and hence satellite applications are either unreliable or inapplicable. Such satellite-signal deficient areas are commonly referred to as indoor environments, and navigating in such environments is called indoor navigation (Lukianto

et al., 2010). Examples include large public office buildings, shopping malls, hospital corridors and wards, residential homes, etc.

Indoor navigation is functionally similar to outdoor navigation in the sense that both involve positioning/localization, mapping and some sort of obstacle avoidance. However, they differ in the technologies they use, the requirement for routing and directions and the physical space within which they operate (Karimi, 2011). This implies that some implementation modifications are needed, and in some cases a total redesign is necessary, when transferring an outdoor navigation system into an indoor environment to ensure effective and efficient performance of the system. However, it is worth mentioning that some technologies which are applicable to both indoor and outdoor navigation have been designed even though these often have limited applications.

As pointed out in Wang (2012), justifying the need for outdoor navigation was much easier as situations necessitating their use were more glaring than situations requiring indoor navigation. However, the proliferation of domestic robots such as vacuum robots, robots for assistive care, intelligent wheelchairs (Matsumoto, Ino andOgasawara, 2001), etc. for various applications has whetted researchers' appetite for addressing indoor navigation problems. The sections below introduce the three aspects of indoor navigation highlighted above, outline the main problem that this research seeks to tackle and summarize the contributions of this research.

1.2 Motivation

The past few decades have seen a great interest in research to enhance mobile robot navigation in an indoor environment. However, the vast majority of indoor navigation algorithms presented in the literature focuses more on guiding the mobile robot relative to the surrounding

environment, in the absence of coordinate information. Such systems are handicapped when it comes to certain applications that require point-to-point navigation. The popularity of autonomous indoor navigation rests on the increasing demand for industrial and domestic mobile robots (e.g., robot vacuum cleaners, etc.). Wang (2012), has emphasized that defending development of indoor navigation systems has been made much more reasonable and convenient by the influx of indoor and industrial robots.

For example, in the health care industry, it is a well-known fact that the worldwide population has seen a tremendous growth within the past decade and research has confirmed the possibility of continuous growth of population in the next decade (see Figure 1.1). The population of elderly in this demographic is gaining much attention since the rate of growth of the elderly population significantly exceeds that of the general population (Division, 2009) . The population of older persons is growing at a rate of 2.6 percent per year faster than the whole population (see Figure 1.2). The projection is that by 2025-2030, the population growth rate for individuals of age 60 and over will be 4 times as rapid as the total population growth rate (i.e., 2.8 : 0.7). These considerations point to the need for robotic assistive care for several reasons including that: (a) the existing supply of domestic and health-care services does not seem to be commensurate with the demand; (b) the high cost of elderly care; and (c) the increasing desire for aging-in-place (making changes in the home to allow seniors to live at home for as long as possible).

Many indoor operating robots are confined to predefined trajectories or fixed locations due to the lack of more prestigious navigating techniques to handle autonomous navigation of these robots. Our motivation therefore is to develop an easy to implement, inexpensive and

adaptable navigating system for a mobile robot operating in an indoor environment. This will facilitate the use of domestic and industrial mobile robots.

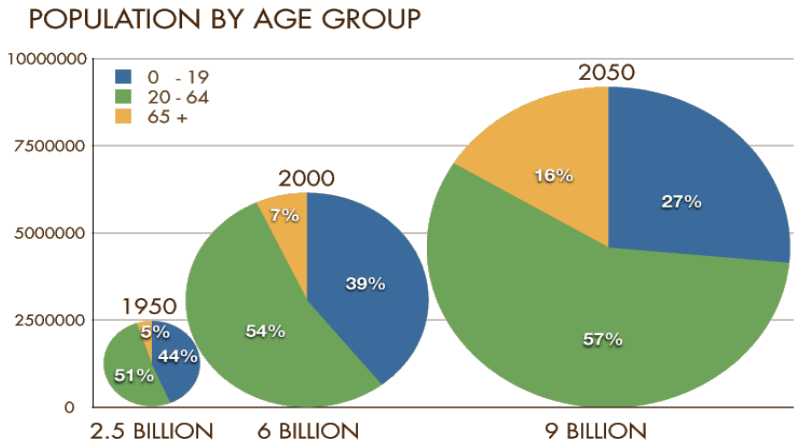


Figure 1.1. Projected population growth from 1950 to 2050 by Department of Economic and Social Affairs report (Division, 2009).

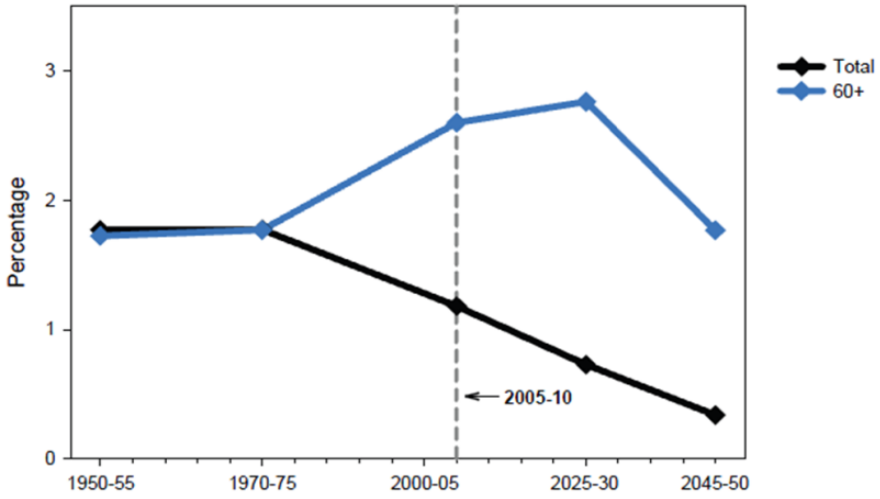


Figure 1.2. Average annual growth rate of the world's population for the total population and population aged 60 or over, 1950-2050 (from United Nations Report: World Population Aging 2009) (Division, 2009).

1.3 Indoor Localization

One of the fundamental requirements for achieving autonomy in mobile robot navigation is localization. Indoor localization of a mobile robot involves estimating the current position and orientation of the robotic system in a given indoor environment, usually by using measured data and *a priori* knowledge of the environment in the form of a map (Sgouros et al., 1996). Without accurate knowledge of the mobile robot's position relative to a given environment, autonomous task execution becomes a very difficult problem. Indoor mapping involves the gathering and representation of the layout of the indoor environment so that the robot can interpret the environment and exploit the layout for context awareness and subsequent maneuvering within the environment (Sgouros et al., 1996).

The past decade has seen tremendous progress in research into localization and mapping in an indoor environment (Nieuwenhuisen et al., 2010). Localization and mapping have been handled separately as well as simultaneously in what is commonly referred to as Simultaneous Localization And Mapping (SLAM) (Tuna et al., 2012). Localization and Mapping in a well-structured environment is a solved problem. However, issues such as getting a system to work in real time, error handling, loop closure, reducing the computation and storage requirement, etc. make this field still very viable for research. Another challenge is how to handle dynamisms in the environment, which are common in a typical indoor environment (e.g. moving objects, opening and closing of doors, displacement of objects). Most of the current work presents solutions requiring robots to be equipped with a comprehensive sensor suite, such as ultrasonic rangers, LIDAR, and cameras as well as computers with substantial capabilities, which can be

excessively cumbersome for practical robotic applications. For example, Ying et al. (2010) presented one such system for navigating in a cluttered environment.

There are also several techniques highlighted in literature for indoor localization. Wireless Local Area Network (WLAN)-based localization is very popular (Bahl and Padmanabhan, 2000; Castro and Munz, 2000). Most of the WLAN localization systems use Received Signal Strength Indicator (RSSI) for localization. These systems use two main phases, namely, an offline training phase (Nguyen et al., 2005; Youssef and Agrawala, 2005) and an online localization phase (Addesso et al., 2010; Kushki et al., 2010). Research into these phases forms the two main branches of the WLAN-based Localization. Junjun et al. (2011) presented some methodologies for WLAN-based indoor localization using RSSI. The authors used grids, in conjunction with a transfer matrix, which are sampled prior to the training phase to estimate probabilities of neighboring nodes to calculate Access Points (APs) for each grid. Some APs from the neighbors of the last estimated location and these new APs from current iteration are selected and used to compute the likelihood of the nodes for the localization. A statistical threshold is calculated and used to avoid the condition of the system being trapped in a local minimum.

Landmark-based localization is another solution to the indoor localization problem. Two broad categories of landmarks can be highlighted, namely, artificial landmarks and natural landmarks. Artificial landmarks are prevalent in structured environments, and they usually involve less environmental engineering to achieve a flexible, robust and highly accurate localization system compared to systems that use natural landmarks (Sooyong and Jae-Bok, 2007). The artificial landmark systems can be grouped into active landmarks, which usually

require an electrical power source for their activation (e.g., Active Radio Frequency Identification (RFID)), and passive landmarks, which require no power source for their operation (e.g., barcodes, QR coded tags, passive RFID, etc.). Passive landmark-based localization is relatively low-cost and consumes less power. This makes passive systems easily scalable (where the technological requirement is not a limitation) and easily maintained. However, most passive landmark-based navigation systems need higher 'landmark density' to match the accuracy of active landmark-based navigation, a condition that makes unique coding and subsequent identification of individual landmarks a burden for large environments (Lee, 2009; Zhou and Shi, 2009). In (Ul-Haque and Prassler, 2010), the authors evaluated passive artificial landmark-based localization system using landmarks made of a retro-reflective coated film able to strongly reflect the infrared light coming from the CMOS infrared camera. The authors argue that the system is easy to scale and maintain after installation and that the ease of assigning unique landmark IDs to thousands of landmarks enables large area coverage.

1.4 Mapping

Mobile robot mapping is another important and challenging aspect of robotic navigation that has been investigated more increasingly in the past few decades. Mapping involves the building of a consistent representation of the robot's unknown environment based on the information obtained through the robot's sensors, such as laser, sonar, visual and infrared sensors. The Sound-based (sonar) sensors such as ultrasonic sensors are among the most prolific because they are relatively easy to use and inexpensive to acquire. However, their major drawback lies in their being highly prone to interference from environmental noise.

Despite the many efforts of researchers, mapping remains a difficult challenge because the robot has to deal with unexpected circumstances, such as moving objects, and also must handle complex and rough environments with little or no prior knowledge. In the case of mobile robots, these challenges are compounded by the non-existence of a perfect strategy for estimating the exact instantaneous location of the robot at a given time since this is at the core of the mapping process.

Mapping can be categorized under two broad headings, namely, (a) grid/graph-based environment maps and; (b) object-specification based maps. Research into the grid/graph-based environmental maps such as the occupancy grids (Noykov and Roumenin, 2007) is well advance due to the intuitive nature of these maps and their ease of implementation. Yenilmez and Temeltas (2007) presented a new map building method for a mobile robot operating in an obstacle populated environment by fusing sensor data using the sequential principal component (SPC) method. An approximate but highly efficient approach to mobile robot mapping with Rao–Blackwellized particle filters has been presented in (Grisetti et al., 2007). A novel method for the integration of fuzzy logic and genetic algorithms for solving the mobile robot mapping problem has been presented in Begum et al. (2008).

The inception of Simultaneous Localization And Mapping (SLAM) (Smith et al., 1990) has led many researchers to consider object-specification based maps, especially feature-based maps, and landmark-based maps. The author, Milton Roberto (2010), presented a new algorithm for feature-based environment mapping where the environment is represented using multivariate Gaussian mixture models using data from either sonar sensors or laser range finders. Similar

work includes that reported in literature (de la Puente et al., 2009; Vázquez-Martín et al., 2009; Zen et al., 2011).

1.5 Path-planning

Given a map of an environment, one common task is for the mobile robot to determine a collision-free path from its current position (state) to a specified position (state) called the goal. The process of determining this path is called path planning. The solution to the path planning problem usually falls under one of three categories: search-based, sampling-based and combinatorial-based solutions. The combinatorial methods are the oldest and possibly most studied branch of planning, with application in many areas ranging from computer graphics to very large scale integrated circuit design. The search-based techniques dominate the other two categories of techniques because they are relatively easy to implement and also due to the fact that they received an early establishment of dynamic search-based algorithms. Research, however, seems to be shifting towards the sampling-based techniques due to their powerful potential for planning in high dimensional space such as that of the serial robot manipulators (Wooden, 2006).

1.5.1 Search-based planning. Search-based planning techniques usually operate on occupancy grids. The configuration space is represented as a tessellation of regularly sized grid cells with the start location of the robot and the goal location within the grid. A search is then performed on the grid to solve the point-to-point problem by finding a chain of free cells (grid cells that are free of obstacles) linking them. Usually these cells form the shortest possible path.

1.5.1.1 Limitations. The grid on which search-based techniques are performed is usually a graph with a fixed topology. One requirement of such graphs is that their resolution needs to be

specified prior to the search. This gives rise to what is usually referred to as resolution completeness (i.e., the path found by the search is only optimal at the specified resolution of the grid). The ramification is that the shortest possible path found by the search might not be the shortest possible continuous path in the workspace. The higher the resolution of the grid, the closer the searched path is to the shortest continuous path possible on the workspace. However, using high resolution is not without a price tag. High resolution means high computational cost in terms of memory for storage and processor time for computation. Using coarse resolution can result in the computation of non-intuitive paths (see Figure 1.3), or render some regions inaccessible from other regions (see Figure 1.4). Note that for both Figure 1.3 and Figure 1.4, subfigure (a) is the continuous world, subfigure (b) is the course gridded world and subfigure (c) is the fine gridded world. There is therefore always a tradeoff between grid resolution and computational cost. A search-based method is complete when it always returns a path through the configuration space of free cells, C_{free}, whenever such a path exists.

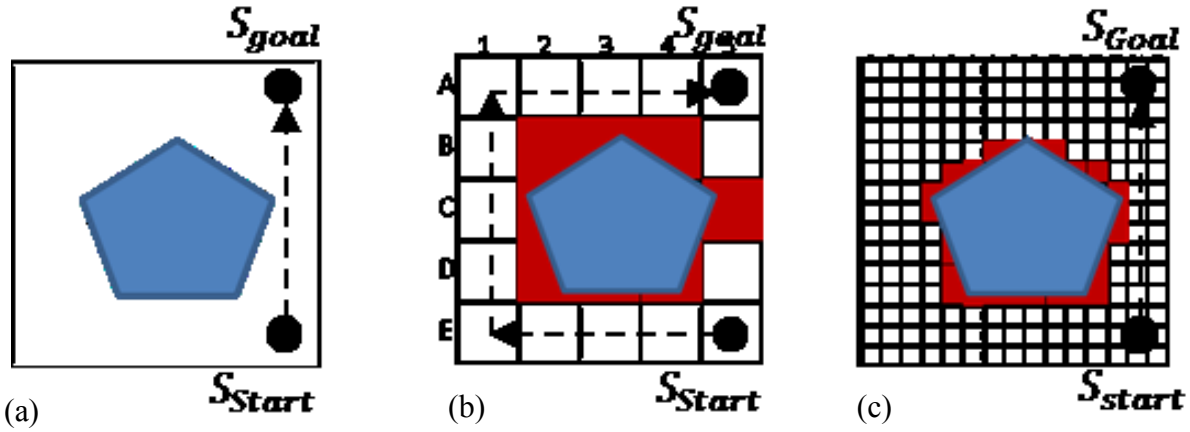
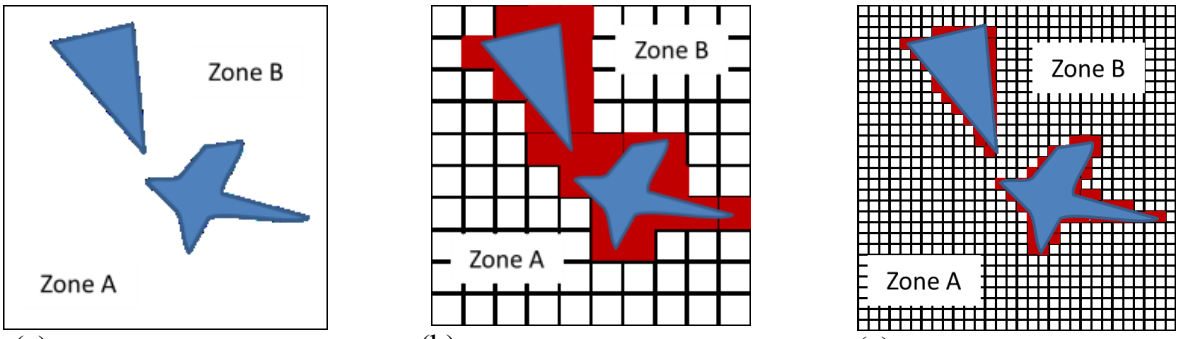


Figure 1.3. How increasing resolution can help graph search algorithms



(a) (b) (c)

Figure 1.4. How increasing resolution can help graph search algorithms.

On the other hand, a regular resolution grid can be decimated to a multi-resolution grid or hierarchical tree structures (e.g., a quad-tree (Noborio et al., 1990) and framed quad-tree (Yahja et al., 1998)). This is usually accomplished as preprocess before the search technique is applied. It is worth mentioning that, this can be a non-trivial process.

1.5.2 Sampling-based path planning. The idea of the sampling-based path planning is to run a search that probes the configuration space with a sampling scheme. The system employs a collision detecting module for the probing of the configuration space (LaValle, 2006). That is, the system generates a vertex and checks whether the generated vertex is interior to an obstacle using the collision detecting module. If so, the vertex is discarded, and otherwise the vertex is considered good and edges are added to it and its mutually visible immediate neighbors (Wooden, 2006). The neighbors are usually nearest neighbors or k-nearest neighbors. The vertex generation can either be stochastic, such as in a probabilistic roadmap (PRM) method (Kavraki et al., 1996; Zhong and Su, 2011), or deterministic. The sampling based path-planning approach has proven powerful in solving robotics, biological and manufacturing applications, especially those involving thousands and even millions of geometric primitives. Such problems are difficult if not impossible to be handled with techniques that rely on explicit representation of the

environment (Choset et al., 2005). The idea here is to eliminate the need for explicit representation of the environment and only generate information about the environment based on sampling. After identifying sufficient vertices and edges, a graph search technique (such as A^*) can be employed to find the path linking the start and goal positions. A recent trend in this branch of path planning has been to investigate methods of making the sampling-based approach dynamic (Yershova et al., 2005; Ferguson et al., 2006).

1.5.2.1 Limitation. The main drawback of the sampling-based path planning approach is its weak guarantee of completeness (usually termed probabilistic completeness). The completeness property guarantees that the problem will be solved if a solution exists. Thus, for any given start and goal locations, the approach correctly reports a solution if at least one exists and no solution if none exists. Since only sample vertices are generated, a false conclusion can result when the samples are not a true representation of the environment. The solution to this problem lies in the quality of the sampling technique used.

1.5.3 Combinatorial-based approach. As the name suggests, the combinatorial-based approach to path planning takes a polyhedral representation of the environment and connects the vertices with edges and/or connects the edges with vertices to form what is usually referred to as roadmap. A search method can then be employed to determine the shortest path between the start and the goal positions. The three main categories in literature are shortest-path (visibility graphs), maximum clearance, and cell decomposition. Combinatorial approaches find paths through the continuous configuration space without employing any approximation technique such as discretization or sampling and hence are usually referred to as exact techniques.

Combinatorial approaches are complete (i.e., for a given input, they are able to return whether a solution exists or does not exist).

1.5.3.1 Limitations. The main challenge with combinatorial algorithms is that they can become impractical for workspaces filled with numerous geometric primitives. This is because the number of vertices and edges will grow exponentially; making excessive demand on storage and making search too time consuming. The other limitations are approach dependent; for example, visibility graphs suffer from obstacle edge hugging which is undesirable in practical implementations.

1.6 Statement of Research Problem

1.6.1 Localization. The challenges affecting the implementation of absolute positioning systems such as the use of navigating beacons and GPS have led to the proliferation of relative positioning systems. Relative positioning systems usually use dead-reckoning for position estimation. Dead-reckoning integrates all pose (position and orientation) changes made by the robot from its initial to current pose. Dead-reckoning is very simple and inexpensive in its modeling and implementation but suffers from substantial sources of error (Cox, 1991). The greatest weakness of dead-reckoning is the accumulation of errors and thus, the uncertainty in the robot's position estimation increases without bound over time. The origin of errors in dead-reckoning can be classified under two broad headings, systematic and non-systematic errors. Systematic errors originate from incorrect parameterization/ measurements inherent in the system due to imprecise manufacturing and modeling.

Many researchers have considered some of the causes of the observable systematic errors and have developed calibration and tuning methods for mitigating them (Borenstein and Liqiang,

1996). However, to the best of our knowledge, none of these researchers claim to perfectly eliminate all of the causes. This means there will still be some unaccounted for causes introducing errors into the estimation in the long run. Besides, non-systematic errors are unpredictable and mostly considered unobservable and so their effects cannot be adequately modeled. Many dead-reckoning systems employed in mobile robots use odometers, leading to the conventional name odometry. An intuitive way to improve the position estimation is to fuse odometry information with additional data/information from sensor sources such as global positioning systems (Thrapp et al., 2001), inertial navigation systems (Borenstein and Feng, 1996) or cameras (Abuhadrous et al., 2003).

1.6.2 Path planning. Map building using fixed node decomposition (i.e., the continuous world is tessellated into a discrete approximation of the continuous map) is inexact resulting in the loss of narrow passages in this transformation. The higher the resolution of the grid, the closer the approximation is to the continuous world. But increasing the resolution introduces more free nodes and increases the search space leading to a sparse grid (see Figure 1.3 and Figure 1.4). The map of most indoor environments can be considered sparse if decomposed into nodes of high resolution. Most of the path planning algorithms in literature are great in solving the path planning problem on cluttered grids but not so much on sparse grids whiles others are good on sparse grids but not on cluttered grids. However, many real world environments comprise of both cluttered and sparse sections. A common way to solve this is the use of multi-resolution gridding (Cowlagi and Tsiotras, 2010; Shih-Ying et al., 2012), that means regions close to obstacles are represented with high resolution gridding and regions far from obstacles are represented by much courser gridding. But building multi-resolution/adaptive grids requires

more work in the decomposition than building uniform/fixed node grids. Besides, determining which node a given position belongs to and finding the neighbors of a given node in a multi-resolution grid is a non-trivial task (Aizawa and Tanaka, 2009). The motivation is therefore to develop a complete and correct search algorithm that can plan paths in high-dimension, high-resolution grid maps faster across the spectrum from highly sparse grids to heavily cluttered grids. This will enable us to handle large grid sizes and thus encourage increasing the resolution of the grid without the need for multi-resolution.

1.6.3 Problem definition. The indoor navigation problem has been formulated thus: Given an unknown and unstructured indoor environment with a navigating mobile robot and a model of how the robot interacts with the environment (a) design a system for lifelong estimation of the position of the robot in such an environment; (b) using this information about the robot's instantaneous locations in the environment, build a consistent map for the environment; and (c) given a current state of the robot and expected destination state, search for the minimum cost path between these two states maneuverable by the robot. Chapter 3 presents a solution to parts (a) and (b) while Chapter 4 presents a solution to part (c).

1.6.4 General assumptions. This section presents some of the general terminologies for describing robots and their environment vis-à-vis some generally accepted assumption for robotic navigation. The robot is a rigid body equipped with mobility capability, in this case a wheeled robot with differential drive motion system. The world in which the robot operates is termed the workspace which is assumed to comprise two distinct regions: Obstacle region – denotes places in the workspace where the robot certainly should not go or cannot go either because it is preoccupied with another object or because it is too close to another object; Free

Space – denotes places in the workspace where the robot can safely go or is free to move. To simplify especially the planning problem, the concept called configuration space introduced in the influential work of Lozano-P and Wesley (1979) has been adopted. Thus the robot is modeled as a point mass capable of omnidirectional translation in R^2 Euclidean space. This alleviates the difficulty, such as geometrical and mobility constraints, associated with directly planning over the workspace with the true physical model of the robot. The workspace is then transformed into the configuration space by "bloating" the obstacles by the maximum radius from the center of the robot. This is achieved by applying the Minkowski sum or mathematical morphology dilation operator to each obstacle in the workspace forming a convolution of the robot's geometry and the obstacle's geometry in the configuration space. In common notations, regions in configuration space occupied by obstacles are termed obstacle regions denoted by C_{obs} and remaining regions available for the robot to maneuver are termed free space denoted by C_{free} (Latombe, 1991). The third dimension of a planner robot is then factored into the low level controls of the robot or by using heuristics stemming from the knowledge of the robot's kinematics constraints.

1.7 Summary of Contributions

The contributions that this research has made to the field of indoor navigation can be classified under the two major areas namely localization and path planning. Under localization, a system for mitigating the long term error accumulation of odometry errors has been designed and implemented. Under path planning, a new pathfinder called $\boldsymbol{A_r^*}$ (pronounced "A-r-Star") has been developed for path planning on occupancy grid maps. Finally, the components are integrated in a simulation using an indoor world prototype developed using the Webots™ software – a

development environment used to model, program, and simulate mobile robots and their behavior in physically realistic virtual environments (Michel, 2004).

1.7.1 Localization.

- Odometry error mitigation using Radio Frequency Identification (RFID) and door-markers.
- Header angle estimation improvement for a navigation robot by fusing the data from three fiber-optic gyroscopes using fuzzy integrals.

1.7.2 Path planning.

- Development of Basic A-r-Star and A-r-Star – A modified, more general and fast version of $\boldsymbol{A}^*$
    - Simulation experimental study of the properties of A-r-Star and presentation of some informal proofs of its theorems
    - Study of the challenges of A-r-Star and development of some solutions to handle these challenges and improve optimality
        - A-r-Star with post smoothing algorithm
        - A-r-Star with interleave smoothing algorithm
        - A-r-Star with interleave smoothing and post dissociative smoothing
        - A-r-Star with interleave smoothing and iterative post dissociative smoothing
- Study of the DAG formed by various graph search techniques and the subsequent development of the incremental version of A-r-Star algorithm that allows for previous

path information reusability to speed up re-planning when a change in the environment invalidates a previously planned path.

1.7.3 Integration.

- Development of a Webots prototype for an indoor simulation world and subsequent application of an integrated version of the navigation system developed in this research to operate in this prototype indoor environment.

1.8 Summary of the Introduction

This chapter has introduced the indoor navigation problem. Also, a brief introduction to the aspects of the indoor navigation algorithm has been presented with their drawbacks. The motivation for this research stemming from the quest to develop a system robust for navigating diverse indoor environments has also been introduced. The chapter ends with the research contributions to solving the indoor navigation problem. The next chapter will give a more detailed state-of-the-art exposition on some of these topics.

CHAPTER 2

Literature Review

This chapter first presents a general overview of some indoor navigation systems and their implementations. This is followed by a brief description of some of the more common indoor navigation algorithms. Also presented is a literature review of related research areas such as indoor localization using RFID and odometry, occupancy grid based mapping and search-based path planning techniques.

2.1 Indoor Navigation Systems

The proliferation of domestic robots for various applications, such as vacuum cleaning, assistive care, intelligent wheelchairs (Matsumoto, Ino andOgsawara, 2001), has whetted researchers' appetite for addressing indoor navigation problems. Researchers have proposed various solutions to the indoor navigation problem. Among the most important solutions presented are inertial navigation systems, Radio Frequency Identification (RFID) based systems and Wifi-based/Bluetooth-based systems. Some highlights of current major trends of indoor navigation techniques are given below with their pros and cons where relevant. For more information, the reader is advised to read (Retscher, 2006; Mautz, 2009; Wendel, 2011).

2.1.1 Satellite-based techniques. Satellite-based navigation systems are still widely used in indoor navigation despite the fact that conditions present during indoor navigation almost imply the absence of viable satellite signals (Dovis et al., 2008). The most common satellite-based navigation systems make use of the American Global Positioning System (GPS), which uses a constellation of 24 satellites orbiting the Earth at altitudes of approximately 11,000 miles (Manapure et al., 2004). Also, there are alternative counterparts, such as Russia's Global

Orbiting Navigation Satellite System (GLONASS) and the Global Navigation Satellite System (GNSS) under development by the European Union (EU) and European Space Agency (ESA) called GALILEO (Manapure et al., 2004).

2.1.1.1 Working Principles. In Global Navigation Satellite Systems (GNSS), the satellites act as a signal transmitters and the device used acts as the signal receiver. The precise instantaneous positions of these satellites are known, and this makes them suitable as reference points for GPS navigating devices on Earth to estimate their positions. The satellites (24 satellites for the current US GPS and a projected 30 satellites for the GALILEO system) continuously transmit their position and a highly accurate time-signal from their corresponding orbits to the surface of the Earth. The GPS navigating devices measure signals from the satellites and make use of the time-of-flight to estimate their distances from these satellites. The satellite signals (microwaves) travel at the speed of light, and thus using the velocity of light and time allows the receiver to calculate its distance to the satellite. Using the principle of trilateration, the intersection of the spheres centered at three satellites is used to estimate the position of the GPS device on the Earth's surface. Because the GPS devices are not usually equipped with the ability to measure the precise time of flight of the signal, a fourth satellite is used to enhance the precision and measurement of elevation or altitude.

2.1.1.2 Limitations/Challenges (Wendel, 2011). The following are some limitations or challenges with satellite-based techniques.

- The initial setup requirement for the GNSS system is both expensive and cumbersome, and this limits its installation to only a few well-to-do countries or multi-national organizations (e.g., US, Russia, and joint EU and ESA).

- The operation and maintenance of the satellites needed for the GNSS is also expensive and demands much effort.
- Satellite-based navigation systems require a direct line-of-sight (DLOS) between the satellite acting as transmitter and the GPS device acting as the receiver for proper operation. Where this DLOS does not exist, the signal may be greatly attenuated, and this reduces the reliability for navigation.
- The accuracy of satellite navigation is still in the meters range.

2.1.2 Inertial navigation systems (INS). Inertial navigation systems exploit the principles of inertia to measure the acceleration and angular velocity of a body and estimate its position. They rely on inertia sensors such as accelerometers and gyroscopes. At its inception in the early 1990's, INS was mainly used for guiding missiles. Researchers later lost interest in this approach since satellite-based navigation proved superior to INS. However, the inherent limitations of the satellite-based navigation systems such as those mentioned in section 2.1.1 have prompted researchers to consider INS as a viable alternative or supplement to satellite based navigation, especially for indoor applications and small areas of navigation.

2.1.2.1 Working principle. Inertial navigation systems are usually implemented using an Inertial Measurement Unit (IMU), which consists of several inertia sensors (both linear accelerometers and angular velocity sensors) assembled together. This IMU forms the core of the INS and is used to observe the acceleration and the angular velocity of the navigation system (Savage, 2007; Liu and Li, 2010). The IMU integrates the acceleration forces acting on it to generate the velocity vector and another integration produces the position vector of the navigation system. To ensure accurate operation, the IMU should be free from external

perturbations and the update rate should be high. These requirements generally dictates the use of large laser-gyroscopes and pendulum accelerometers in high accuracy INS to make the gravimetric measurements. The birth of semiconductor-based micro-electro-mechanical system (MEMS) sensors has made the integration of IMUs into mobile devices possible (Smailagic and Kogan, 2002).

2.1.2.2 Limitations/Challenges. The following are some limitations or challenges with INS.

- INS needs an initial reference position and heading, which must be fixed and accurate vis-a-vis altitude stabilization. These requirements make INS implementation cumbersome.
- IMUs are very sensitive to external perturbations forces, which can greatly degrade their performance.
- Drifting of numerical consistency in IMU measurements occurs if the system is not augmented with external, independent position updates. This makes its long term application impractical given certain levels of sensor quality.

2.1.3 Sound-based navigation. Sound-based navigation systems use the principles of sound wave propagation to estimate the position of a navigating system. Such a system usually consists of a transmitter and receiver placed at the ends of the distance to be measured or collocated together and used to measure the distance of an object (e.g., a wall or cliff) in the direct line of sight from the transmitter/receiver source (Castro et al., 2001).

2.1.3.1 Working principle. There are different implementations of sound-based navigation systems. The most popular is the sonar system. The sonar device houses both the ultrasound transmitter and receiver. It emits either unidirectional or omnidirectional sound (depending on application). When this sound reflects off a surface, the receiver measures the reflected signal. The device then uses its knowledge of the sound velocity and the time-of-flight to estimate its distance from the reflecting surface (Langer and Thorpe, 1992). This implementation is usually employed for mapping applications.

A second implementation is the Active Bat system. Active Bat is based on the principle of trilateration. It consists of a single ultrasonic source/transmitter (often whose position is to be estimated) and multiple receivers embedded in the environment. These sensors measure the time-of-flight, and the system uses trilateration techniques to estimate the position of the transmitter (Ward et al., 1997).

A third implementation uses a transmitter installed on the navigator and a receiver on the point of reference. The properties of sound are then used to estimate the distance between the transmitter and the receiver, and this is used to estimate the position of the navigation system. These are but a few of the existing implementations.

2.1.3.2 Limitations/disadvantages. The following are some limitations or disadvantages of sound-based navigation systems.

- Sound-based navigation systems are very sensitive to environmental noise (e.g., similar sounds from other objects or multiple reflections from the same surface or multiple surfaces) and this can drastically degrade their performance.

- Sound-based navigation systems, especially the Active Bat system, require pre-installation of the multiple receivers in the environment, which can be cumbersome and relatively expensive.
- Sound-based navigation systems require direct line of sight for their proper operation and so their usage in cluttered environments is limited.
- Sound has low frequency and so attenuates fast, limiting sound-based navigation systems to short distance applications.

2.1.4 Electromagnetic wave-based navigation. Electromagnetic waves (EMW) are high frequency waves such as visible or invisible light and radio waves. The properties of electromagnetic waves can be exploited for the design of a navigation system in ways similar to some of those mentioned above (sound-based and satellite-based). In recent applications, the common examples of EMW based navigation include: laser range finders/laser measurement systems, Radio Frequency Identification (RFID) systems, wireless LAN systems (Castro et al., 2001; Ladd et al., 2002; Smailagic et al., 2002), bluetooth positioning and ultra-wide band systems.

2.1.4.1 Working principles. The principle of operation is usually dependent on the type of electromagnetic wave being used in the application. The light based systems, such as infrared-based (e.g., Active Badge system), laser-based systems (e.g., laser range finders/laser measuring instruments) employ the measurement of time-of-flight to estimate distance. The operating principle of the Active Badge system is similar to that of the Active Bat system described above (Section 2.1.3). The radio wave-based methods, commonly implemented in the form of RFID, work by proximity detection or reading of transmitted codes, such as is the case of RFID tags.

Other implementations include measuring and calculating of distances based on the received signal strength (RSSI) of a signal transmitted by installed infrastructure nodes such as WLAN, ultra-wide band (UWB) or Bluetooth access points (Grimson and Lozano-Perez, 1987; Latombe, 1991).

2.1.4.2 Disadvantages/Limitations. The following are some limitations or disadvantages of electromagnetic-wave-based navigation.

- Some of the EMW based techniques (e.g., the Active Badge system) require the pre-installation of uniquely encoded transmitters, and the setup requires a sufficient number of receivers to achieve room-level precision. This makes their implementation expensive (Boontrai et al., 2009).
- The necessary fingerprinting process (the recording of RSSI at each grid point of the environment) can be very time consuming.

2.1.5 Image-based or optical techniques. Image-based or optical techniques use image analysis and image processing techniques for estimation of the position of a navigating system. They usually involve the processing of visual information in the form of still or continuous images provided by a camera. The image source can be a single camera usually for 2D mapping or stereo camera for 3D mapping.

2.1.5.1 Working Principles. These approaches employ some kind of camera device for taking the image of the environment and image processing techniques for localization and path planning. Many researchers have employed different imaging devices, such as monocular cameras, or the stereo cameras which add the depth dimension. Some of the popular optical

navigation techniques are the optical flow navigation methods (McCarthy and Bames, 2004), image annotation (Kawaji et al., 2010) and feature based mapping (Se et al., 2001).

2.1.5.2 Disadvantages/Limitations. The following are some limitations or disadvantages of image-based and optical techniques.

- The performance of these techniques highly depends on the state of the environment (lighting, occlusions, etc.);
- Lack of depth information and reliance on complex image processing algorithms create high computational burden and other difficulties.

2.2 Indoor Navigation Algorithms

The previous section gives a general overview of the sensor hardware solutions to the indoor navigation problem. These sensor hardware techniques are usually implemented with various algorithms that operate on the data acquisition or collection and processing and subsequent estimation of position (localization), building of a representation of the environment (mapping), and direction and routing from one position to another (path planning and obstacle avoidance). Some of these pairings of sensor hardware and algorithms have been highlighted below.

2.2.1 Kalman filter/extended Kalman filter. The Kalman filter uses a series of measurements observed over time, an observation model, and a system model containing noise (random variations) and other inaccuracies to estimate unknown parameters, which usually have higher accuracy than estimates that rely on a single measurement. A Kalman filter models the system using a random variable with a Gaussian distribution and uses the first and second moments (mean and covariance) to represent this probability distribution. It uses the system

model and observation models to predict and uses the observations made by its sensors to correct/reduce the estimation error. The Kalman filter works with linear system models but it can be extended to work with non-linear system models (extended Kalman filter). The most common navigation application of Kalman filters is in the implementation of Simultaneous Localization And Mapping (SLAM). Common hardware for implementation dead reckoning includes image-based or optical techniques, INS and electromagnetic-based navigation (e.g., RFID beacons).

2.2.2 Dead reckoning. Dead reckoning is a position/pose estimation technique that uses the integration of all the displacements made since the navigating object first left/passed a reference point. These displacement estimates can be in the form of changes in the heading and distance or Cartesian coordinates. Example applications include pedestrian dead reckoning (Retscher, 2006), odometry (SungHwan et al., 2012), Strapdown (Savage, 2007), etc. Common hardware for implementation dead reckoning includes INS, odometers and optical navigation system.

2.2.3 Particle filter. Particle filters are used when the system state cannot be modeled as a (linear) Gaussian distribution random variable. Here a sum of weighted samples called particles is used to approximate the density function of the random variable. The number of the particles can be chosen to meet the desired performance requirement. A higher number of particles achieves a good approximation, but their processing is computationally expensive. Too few particles, however, can cause system divergence. Like the Kalman filter, the particle filter is mainly used in the implementation of SLAM (the so called FastSLAM) (Montemerlo and Thrun, 2007). Common hardware implementation using particle filters includes image-based/Optical techniques.

2.3 Robot Localization

Determination of the exact position of a navigating mobile robot remains fundamental to mobile robot automation. Although researchers have proposed and investigated many systems, sensor combinations, techniques and algorithms, it remains a viable challenge since no elegant and broadly robust solution has been developed yet. Localization can be classified under two broad categories, namely, absolute position estimation (e.g., magnetic compasses, active beacons, Global Positioning System (GPS), landmark navigation, and model matching) and relative position estimation (e.g., dead reckoning (odometry) and inertial navigation) (Casals, 1989).

2.3.1 RFID-based indoor localization. Radio frequency identification (RFID) is a form of automatic identification technology that utilizes remote storing and retrieving of data using readers or scanners and tags. The principles of RFID localization are similar to those of WLAN. Most existing RFID localization methods employ the RF signal strength, instead of time-of-arrival of signal, as an indicator of distance. Researchers have explored many different methods for localization based on RFID tagging and identification, meeting different application demands and available hardware. The application requirement and the available hardware usually dictate which parameters can be chosen to achieve good localization estimates. Also, existing localization techniques have been enhanced by employing the object identification potential of RFID, artificial landmarks or global reference points as an accuracy enhancing tool. Several reviews in literature (Bouet and dos Santos, 2008; Sanpechuda and Kovavisaruch, 2008; Zhou et

al., 2009; Nikitin et al., 2010) give good overviews of many of these systems and approaches and highlight some of their pros and cons.

An RFID based localization technique is presented in Nick et al. (2012) that applies the Constrained Unscented Kalman Filter (CUKF) to the RSSI measured from an unknown tag to localize an RFID. A camera-based localization technique is implemented to supplement the noisy RSSI estimation and to increase the accuracy of the localization. The accuracy performance of this camera-assisted localization technique proves superior to that of the CUKF without camera assistance by a factor of two and to Unscented Kalman Filter (UKF) by a factor of about four. RFID localization techniques find their application in healthcare (Mautz, 2009), construction material management (Song et al., 2007), local positioning for road safety (Hui and Zekavat, 2007), production process control (Thiesse et al., 2006), automated guided vehicle routing (Langer et al., 1992), mining safety (Ruff and Hession-Kunz, 1998) and other areas.

2.3.2 Dead reckoning (odometry). Dead reckoning (odometry) denotes the integration of incremental motion information of a given navigation system over a given run time. It is usually implemented by using odometers (e.g., optical encoders) to measure the wheels' angular velocity and using this data to compute the navigating system's offset from a known reference point. Odometry is most widely used in mobile robot navigation due to its ability to provide good short-term accuracy, its ability to allow very high sampling rates, and its lower implementation cost and complexity. The integration of this information inevitably leads to the accumulation of errors, however. Table 2.1 outlines some of the known causes of systematic and non-systematic errors. Accumulation of errors in the position estimates is an issue but of greater concern is the accumulation of the orientation errors, which translates into large position errors (Borenstein,

Everett, et al., 1996), and increase proportionally with the distance traveled by the robot thereby causing the estimation error to diverge. However, because odometry forms an integral part of many navigating systems, research into odometry accuracy improvement and error mitigation has advanced. Some of the earliest includes the test method for detecting and correction of mainly systematic errors called "UMBmark" and "extended UMBmark" (Borenstein, 1998). This method, also called internal position error correction (IPEC), was implemented on the OmniMate robot, which was specifically designed for the implementation of the IPEC method.

Table 2.1

Sources of Systematic and Non-Systematic Errors in Mobile Robot Odometry

Systematic Errors	Non Systematic Errors
a) Limited encoder sampling rate b) Limited encoder resolution c) Unequal wheel diameters d) Average of both wheel diameters differing from nominal diameter e) Wheel misalignment f) Uncertainty about the effective wheelbase (due to nonpoint wheel contact with the floor)	a) Unexpected external forces (interaction with external bodies) b) Unexpected internal forces (e.g., from castor wheels) c) Non-point wheel contact with the floor d) Uneven operating floors e) Unexpected objects on the operating floor f) Wheel-slippage resulting from slippery floors, skidding in fast stops or turns, etc.

Results show that OmniMate can improve odometry accuracy by one order of magnitude over conventional mobile robots. However, the UMBmark test (Borenstein and Evans, 1997) is

relatively difficult to perform and very sensitive to non-systematic errors. Besides, its accuracy depends on a large number of tests with precise measurements of the final position and orientation of the robot, which is difficult if not impossible in real-life application (Bostani et al., 2008).

A low cost navigation system is developed by fusing inertial sensor information provided by gyroscopes and odometry using Kalman Filters and rule set-based fusion strategies (Tarin Sauer et al., 2001). The gyroscope information helps to improve orientation estimation. The autonomous mobile robot B21 was used as the testing platform. This system reduces both the systematic and non-systematic errors in the navigation system and improves the accuracy of their system (Bostani et al., 2008).

Abbas et al. (2006) identifies the main limitation of UMBmark as its inability to incorporate the non-systematic stochastic wheelbase error that arises from less turning in the test path execution and wheel slippage at corners due to stops and wheel direction reversals. The authors then present a technique for measurement and correction of systematic odometry error caused by kinematics imperfections in the differential drive mobile robots; the technique uses occasional systematic calibration. The paper proposes a Bi-Directional Circular Path Test (BCPT) for modeling the systematic odometry errors and claims that this approach significantly reduces the amount of effort required to model parameters involved in the systematic. This approach is also claimed to be very simple to perform, robust, and relatively free from random errors, especially from measurement noise. It is also said to yield good results at the end of a single test, thereby eliminating the need to repeat the test over and over again.

Other researchers have focused on visual odometry; see, for example, (Corke et al., 2004; Johnson et al., 2005; Milella and Siegwart, 2006; Zhiwei et al., 2007). However, the closest work in literature to the research in this dissertation is presented in Kubitz et al. (1997) where Radio Frequency Identification (RFID) tags were used as artificial landmarks in the robot's environment. However, as the authors rightly pointed out, the accuracy of the measurement of the relative position between the tag and the robot depends on the abilities of their RFID system. In most cases, it is relatively inaccurate and is strongly influenced by the reading range and the ability to detect whether the robot is approaching or moving away from a tag. This challenge is resolved in the current research by employing door-markers (as detailed in section 3.2.3).

2.4 Mapping

Map building is considered an important component of mobile robotic applications in partially-known or unknown environments since knowledge of the environment plays an important role in developing robots and robotic software capable of exhibiting fully autonomous behavior.

2.4.1 Occupancy grids mapping. An occupancy grid is an approach for representing an environment using regular tessellation of the space into a number of cells (usually rectangular cells). It is the most common low-level environment modeling approach for fusion of noisy data in robotics. Each cell stores the probability that a given area in the environment corresponding to that cell is occupied by an obstacle. This approach assumes that neighboring cells of the grids are independent from each other and thus avoids a combinatorial explosion of possible grid configurations. Occupancy grids find their direct application in robotic navigation, path planning, and collision avoidance.

There are two main types of occupancy grid approaches in the literature, namely, the static and the dynamic occupancy grid. As the names suggest, static occupancy grids rest on the assumption that the environment is static while dynamic occupancy grids incorporate the dynamism of the environment (Fulgenzi et al., 2007). Most researchers assume a static environment and therefore static occupancy grids are common in the literature.

Moravec and Elfes (1985) proposed occupancy grids for constructing an internal model of static environments based on ultrasonic range data (Moravec, 1988; Schiele and Crowley, 1994). Their approach utilizes probability or certainty values to handle the uncertainty in the sensory data. The authors in (Stepan et al., 2005) presented a novel approach for building an occupancy grid from a monocular color camera. They have also developed a method of fusing the monocular camera data with data from laser range finders. This leads to a more accurate result.

The two drawbacks to the application of occupancy grids in robotics have been the modeling of dynamic obstacles and localization of the robot building the map. Multi-resolution matching has been proposed in Schiele et al. (1994) and Moravec et al. (1985) as a solution to these challenges by matching global and local occupancy grids.

For the past few decades, dynamic occupancy grids have been attracting more interest. Coué et al presented a 4D occupancy grid in (Coué et al., 2006) in which each cell is defined in terms of its position and two speed components along each axis. This 4D model permits the computation of the speed of the classical cells in the 2D grid using the estimation of the occupancy of each cell in the 4D. Chen et al. (2006) presented another solution that does not use 4D but uses a distribution of speed for each cell in the form of a histogram.

Recently, Vatavu et al. (2011) have developed a real-time flexible method for occupancy grid modeling and representation using particles that move from cell to cell. A border scanner algorithm extracts polylines from occupied cells by performing radial scanning using the position of the ego vehicle as the scan rotation center. An average speed is then computed for each resulting polyline as an average speed of the grid occupied cells neighboring the polyline. This method uses the measurements derived from stereovision to determine when to create or destroy particles. The occupancy grid approach presented in (Vatavu et al., 2011) relies on a hidden Markov model (HMM) to explicitly represent both the belief about the occupancy state and state transition probabilities of each grid cell, and thus enables the modeling of how the occupancy changes over time.

Hao and Nashashibi (2012) presented a new approach for merging occupancy grid maps built by different observers (robots). They achieve this by measuring the consistency degree of map alignment using occupancy likelihood. The optimization of the objective function is achieved through genetic algorithms implemented in a dynamic scheme. Their approach has been implemented on a scheme of multi-vehicle cooperative local mapping and moving object detection as well as demonstration of the effectiveness of the algorithm using real-data tests.

2.5 Search-Based Path Planning

Search-based path planning received its popularity from the early development of graph search algorithms with direct application in discovering paths on grids. One such algorithm is Dijkstra's algorithm, a breadth-first search technique (Dijkstra, 1959). It was first conceived by Dutch computer scientist Edsger Dijkstra. It finds the single source shortest path from a single vertex to all the vertices of a given graph with positive edge costs. The $\boldsymbol{A}^*$ (pronounced "A-

Star") algorithm, which is similar to Dijkstra's algorithm was independently introduced by Nils Nilsson (1968) (Hart et al., 1968), and uses admissible heuristics to speed up the search algorithm by narrowing the search space. The $\boldsymbol{A}^*$ algorithm has been proven to be optimal, which means that it will search at least as fast (in terms of computational speed) as all other solution methods provided the other method is not using a better-heuristic (Hart et al., 1968). Also, if the heuristic function used for the cost modeling never overestimates the actual minimal cost of reaching the goal (i.e., it is admissible), then the solution returned by the $\boldsymbol{A}^*$ algorithm is optimal (i.e. the best possible path available given the graph/tree constraints) (Dechter and Pearl, 1985). Use of the $\boldsymbol{A}^*$ algorithm for path planning in both robotics and video games has increasingly gained popularity.

However, the $\boldsymbol{A}^*$ algorithm uses a fixed heuristic which can negatively affect the performance where the system changes frequently. The $\boldsymbol{LRTA}^*$ (Korf, 1990) solves this problem by learning a perfect heuristic function. Near optimal hierarchical path-finding ($\boldsymbol{HPA}^*$**)** introduced in (Botea et al., 2004), is essentially a hierarchical approach that abstracts the map into linked local clusters before applying the $\boldsymbol{A}^*$ algorithm thereby reducing the complexity of the search process.

One drawback of the $\boldsymbol{A}^*$ algorithm is its offline nature, which makes its application suitable for only static environments. This implies that, whenever there is an environmental change that invalidates the searched path, the $\boldsymbol{A}^*$ algorithm has to do search-from-scratch (commonly referred to as re-planning). It has no built-in capability to reuse the information accumulated from the previous search (Hart et al., 1968). This limits the $\boldsymbol{A}^*$ algorithm's

applicability to static environments and the algorithm is less useful in dynamic, partially known or unknown environments.

The need for algorithms that will solve this problem gave rise in the mid-1990 to the $\boldsymbol{D}^*$ algorithm developed by Stentz (1994). $\boldsymbol{D}^*$ (Stentz, 1994, 1995b), which is possibly the first of the category of algorithms called incremental search algorithms. These are incremental in the sense that the information from the previous search is used whenever a change in the environment invalidates the previously searched path. The reusability of information from the previous search speeds up the re-planning process, and makes their application to real world scenarios involving dynamic environment or partially-known or unknown environments more attractive.

However, the $\boldsymbol{D}^*$ algorithm at its initial stage operates as a breadth-first search and thus its initial search is time consuming (Stentz, 1995a). To handle this limitation, the $\boldsymbol{focussed - D}^*$ algorithm (Stentz, 1995a) was developed. It utilizes heuristics to focus the search to significantly reduce the total time required for both the initial path calculation and subsequent re-planning operations thereby making the $\boldsymbol{D}^*$algorithm a full generalization of $\boldsymbol{A}^*$ for dynamic environments. The $\boldsymbol{Field\ D}^*$ algorithm is a variant of $\boldsymbol{D}^*$ that uses interpolation to improve the smoothness of the path returned both in the planning and the re-planning phase (Ferguson and Stentz, 2007) (Carsten et al., 2006). Another variant of the $\boldsymbol{D}^*$ algorithm is the Anytime $\boldsymbol{D}^*$ (a.k.a. Anytime Dynamic $\boldsymbol{A}^*$) algorithm, which combines the benefits of anytime and incremental planners to provide efficient solutions to complex, dynamic search problems (Likhachev et al., 2005). $\boldsymbol{D}^*$ and its variants are algorithmically complex and so their implementation is not very attractive to many pathfinder developers.

Another incremental search, developed by Ramalingam and Reps (1996) and called the DynamicSWSF-FP shifts the re-planning computational burden to the initial stage by computing and storing all the distances from the free nodes to the goal. When the robot detects a change in the environment while navigating, the system only needs to update the nodes whose distances have changed to obtain consistent paths from multiple nodes to the goal. This makes it a powerful tool in problems where there is the need to determine the distances from multiple nodes to the goal after a change in the environment.

Koenig et al. (2004) developed the incremental $\boldsymbol{A}^*$ which combines the powers of $\boldsymbol{A}^*$ and DynamicSWSF-FP. The incremental $\boldsymbol{A}^*$ is also referred to as Lifelong Planning $\boldsymbol{A}^*$ ($\boldsymbol{LPA}^*$), in analogy to "lifelong learning" (Eaton and Ruvolo, 2013), due to its ability to reuse the information from previous searches. The initial search of $\boldsymbol{LPA}^*$ is the same as that of $\boldsymbol{A}^*$**,** but the subsequent re-plannings are much faster than that of $\boldsymbol{A}^*$. $\boldsymbol{LPA}^*$ forms the foundation for the development of the $\boldsymbol{D}^* - \boldsymbol{Lite}$ algorithm (Koenig and Likhachev, 2002), which differs algorithmically from and is easier to understand and analyze than $\boldsymbol{D}^*$, but implements the same behavior as $\boldsymbol{Focused\ D}^*$. Koenig and Sun (2009) presented a comparison of $\boldsymbol{D}^* - \boldsymbol{Lite}$ and $\boldsymbol{LRTA}^*$.

Most of the grid-based path planning algorithms that operate on a 2D world use discrete state transitions that are artificially constrained to a small set of possible headings angles (e.g., $0, \frac{\pi}{4}, \frac{\pi}{2}$, etc.). The ramification is that the path returned by even the optimal grid planner will not be the shortest possible continuous path. The $\boldsymbol{Theta}^*$ algorithm (Daniel et al., 2010) reduces this discrete angular constraint by exploring connectivity between a node and its parents as well as its

grandparent. Where the grandparent has a line-of-sight between it and the child, the grandparent is made the parent of the current node and the parent is discarded.

The new trend in this branch of path planning includes the works of Sven Koenig et al in the development of the Adaptive $\boldsymbol{A}^*$ algorithm (Koenig and Likhachev, 2006) and their variants such as the Generalized Adaptive $\boldsymbol{A}^*$ (GAA*) (Sun et al., 2008a) and Real-Time Adaptive $\boldsymbol{A}^*$ ($\boldsymbol{RTAA}^*$) (Koenig et al., 2007; Sun et al., 2008b). These algorithms have been shown to be effective in moving target tracking and multiple goal search problems.

Most of the graph search algorithms highlighted so far perform well in cluttered environments but suffer substantial performance degradation when operating in sparse grids. The natural solution to this is to preprocess the world into some kind of multi-resolution grid or tree structures such as demonstrated in (Noborio et al., 1990; Yahja et al., 1998; Hern et al., 2011). However, the decomposition problem is not trivial. It can result in time consuming pre-processing as well as huge memory requirements, especially where the given image map has some 'salt and pepper noise'(random spots present in the image which are not present in the original object that was imaged). The $\boldsymbol{A}_r^*$ algorithm presented Opoku et al. (2013) and reviewed in Chapter 4 of this dissertation has the power of working across the spectrum from cluttered to sparse grids without great changes in its performance. Therefore, for the environments that are mixtures of both sparse and cluttered sectors, $\boldsymbol{A}_r^*$ becomes the planner's choice.

2.6 Summary of Literature Review

This chapter has looked into both the general indoor navigation problem and some hardware and software approaches to solving the problems presented in literature. An overview of research closely related to the indoor navigation problem (state-of-the-art) in the areas of

localization, mapping and path planning has been presented, highlighting the weaknesses that necessitate the research presented in this dissertation. Chapter 3 presents the localization and mapping approaches developed and used in this work.

CHAPTER 3

Localization and Mapping

3.1 Introduction

This chapter presents the first part of the methodology for solving the indoor navigation problem developed by this research. This is outlined under the two main broad headings localization and mapping. The localization employs an odometry technique using a least square error calibration approach for alleviating the effect caused by systematic errors. Besides, an RFID and door-markers based resetting system has been developed for intermittent resetting of the positions relative to the global reference point to mitigate both systematic errors that escape the calibration and the non-systematic errors. The mapping used in this system is the occupancy grid mapping using a laser range finder. The efficiency of this mapping is enhanced by the improved accuracy of the localization system. The modeling used in this chapter assumes a differential wheel driven mobile robot even though an extension can be made to other drivers such as the car drive (Ackerman steering). The wheels are equipped with encoders (odometers) that can measure the angular velocity of the wheels.

3.2 Localization

3.2.1 Differentially driven mobile robot odometry equations. Differential drive is considered to be one of the simplest drive mechanisms employed on many mobile robots platforms (especially robots designed for indoor navigation) such as the iRobot Roomba, Pioneer 2, Khepera, Labmate, Robuter, etc. platforms. In its simplest design, a differential wheel drive robot is driven by two wheels on a common base/axis with each wheel controlled by a reversible motor. The robot is able to execute various trajectories by independently varying the speed and

direction of the two wheels. The two wheels then exhibit rolling motion with the center of rotation of the robot lying on the axis linking the two wheels, and the position on this axis depends on the speed of the two wheels (Dudek and Jenkin, 2000).

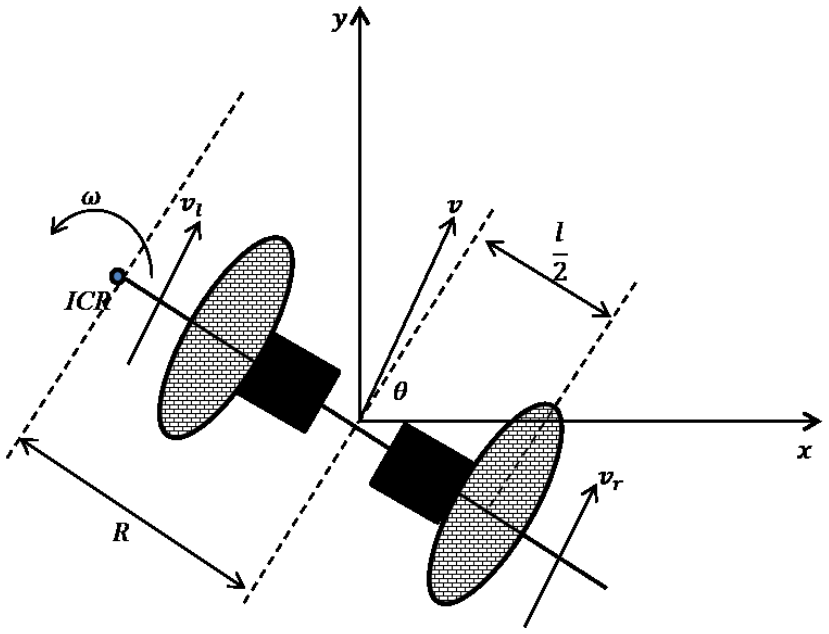


Figure 3.1. Derivation of the kinematics for a differential drive mobile robot.

Using the idea that the instantaneous tangential linear velocity at a given point is the product of the angular velocity and the radius of that point from the *ICR* (see Figure 3.1), Equation (3.1) can be derived.

$$v_r = \omega(t)(R + \frac{l}{2})$$
$$v_l = \omega(R - \frac{l}{2}) \tag{3.1}$$

The two equations can be solved simultaneously for R and ω to produce Equation (3.2).

$$R = \frac{l}{2}\left(\frac{v_l + v_r}{v_r - v_l}\right) \tag{3.2}$$

$$\omega = \frac{v_r - v_l}{l}$$

It can be inferred from these equations that if $v_l = v_r$, then the radius R if infinite which implies that the robot will move on a straight line. This can also be deduced from the fact that ω is zero. If $v_l = -v_r$ then the radius is zero and the robot rotates about the midpoint of the two wheels. This can also be seen from the fact that $\omega = \frac{v_l}{0.5l} = \frac{v_r}{0.5l}$. When $v_l = v_r = 0$, then the radius $R = 0$ and $\omega = 0$ implies the robot remains stationary. Aside from these special cases, any other values of v_l and v_r will cause the robot to execute a curved trajectory with the center of rotation R away from the midpoint between the two wheels. It is also apparent that the differential drive is highly sensitive to any error introduced to either or both wheel velocities, since that can result in a totally different trajectory.

Let d be the 1D distance travelled by the robot and v the velocity from time t_0 to t_f, then the distance can be determined from Equation (3.3).

$$d = \int_{t_0}^{t_f} v dt \tag{3.3}$$

This distance can be resolved into its corresponding distances travelled in the 2D coordinate plane as shown in Equation (3.4).

$$\begin{aligned} x &= \int_{t_0}^{t_f} v dt \cos\theta \\ y &= \int_{t_0}^{t_f} v dt \sin\theta \end{aligned} \tag{3.4}$$

This defines the principle of dead reckoning which is simply the integration of all the distances moved by the robot and can be generalized for higher dimensions. In practice, the effective

rectilinear and angular velocities ($v = v(t)$ and $\omega = \omega(t)$) of the robot are much difficult if not impossible to observe. Thus, angular velocity measurement instruments (e.g., tachometer (continuous), encoders/counters (discrete)) are employed to measure the angular velocities of the individual wheels; the effective rectilinear and angular velocities are deduced from it as shown below and also in (Kelly, 2004; Corke, 2011)): Assuming that the two driver wheels are equipped with odometers to measure the angular velocities of the wheels gives signal Equation (3.5).

$$\boldsymbol{u}(t) = [\omega_r(t)\ \ \omega_l(t)]^T \tag{3.5}$$

Where X^T denotes X transpose. Define the state vector $\boldsymbol{x}(t)$ as the instantaneous position and orientation/heading angle of the robot, then (Papadopoulos and Misailidis, 2007)

$$\boldsymbol{x}(t) = [x(t)\ y(t)\ \ \theta(t)]^T \tag{3.6}$$

Thus the dead reckoning equation can be derived as shown in Equation (3.7)

$$\begin{pmatrix} x(t+1) \\ y(t+1) \\ \theta(t+1) \end{pmatrix} = \begin{bmatrix} x(t) + v(t)\cos\theta\ \delta t \\ x(t) + v(t)\sin\theta\ \delta t \\ \theta(t) + \omega(t)\delta t \end{bmatrix} \tag{3.7}$$

Where δt is assumed to be very small.

The right and the left wheel angular velocities can be represented by Equations (3.8) and (3.9).

$$\omega_r(t) = \frac{1}{r_r}\left[v(t) + \frac{l}{2}\omega(t)\right] \tag{3.8}$$

$$\omega_l(t) = \frac{1}{r_l}\left[v(t) - \frac{l}{2}\omega(t)\right] \tag{3.9}$$

Solving the two equations simultaneously for $v(t)$ and $\omega(t)$ yields Equations (3.10) and (3.11)

$$v(t) = \frac{(r_r\omega_r(t) + r_l\omega_l(t))}{2} \tag{3.10}$$

$$\omega(t) = \frac{(r_r\omega_r(t) - r_l\omega_l(t))}{l} \tag{3.11}$$

These can be rewritten in the form shown in Equation (3.12).

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \tag{3.12}$$

Where $\boldsymbol{M} \in R^{2\times 2}$ is the matrix of the parameters to be determined by a calibration technique (described below) and is defined as Equation (3.13)

$$\boldsymbol{M} = \begin{bmatrix} \frac{r_r}{2} & \frac{r_l}{2} \\ \frac{r_r}{l} & -\frac{r_l}{l} \end{bmatrix} \tag{3.13}$$

If $\boldsymbol{M}$ could be accurately obtained, then the systematic errors in the system due to imprecision in the wheel axis length, l and wheel radii r_r and r_l can be totally eliminated. For a 2D Cartesian coordinate system, the dynamic equation for the robot's motion can be written as Equation (3.14).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \tag{3.14}$$

Discretizing Equation (3.7) and assuming data acquisition frequency of $\frac{1}{T}$ Hz for obtaining the velocities, Equation (3.7) can be written as Equation (3.15).

$$\begin{aligned} x_{k+1} &= x_k + Tv_k\cos(\theta_k + \frac{T\omega_k}{2}) \\ y_{k+1} &= y_k + Tv_k\sin\left(\theta_k + \frac{T\omega_k}{2}\right) \\ \theta_{k+1} &= \theta_k + T\omega_k \end{aligned} \tag{3.15}$$

3.2.2 Systematic error calibration using least square error approach. The least squares odometry calibration technique described in (Antonelli et al., 2005) for the calibration of mobile robot for its simplicity and implementation flexibility has been adopted for this research. From Equation (3.15), an iterative model can be formulated as Equation (3.16).

$$\theta_N - \theta_0 = Tm_{2,1}\sum_{i=1}^{N-1}\omega_{\mathrm{r,i}} + Tm_{2,2}\sum_{i=1}^{N-1}\omega_{\mathrm{l,i}} \tag{3.16}$$

where $m_{i,j}$ is the $(i,j)th$ entry in the $\boldsymbol{M}$ matrix in Equation (3.13). Equation (3.16)can be rewritten in the form of Equation (3.17).

$$\theta_N - \theta_0 = \boldsymbol{\Phi}\begin{bmatrix} m_{2,1} \\ m_{2,2} \end{bmatrix} \tag{3.17}$$

Where $\boldsymbol{\Phi}$ is the (1×2) regressor given by Equation *(3.18)*.

$$\boldsymbol{\Phi} = T\begin{bmatrix} \sum_{i=0}^{N-1}\omega_{\mathrm{r,i}} & \sum_{i=0}^{N-1}\omega_{\mathrm{l,i}} \end{bmatrix} \tag{3.18}$$

Here, N is the number of paths executed by the robot for the calibration procedure. By executing p suitable paths, the least squares model can be written as Equation (3.19).

$$\begin{bmatrix} m_{2,1} \\ m_{2,2} \end{bmatrix} = \left(\bar{\boldsymbol{\Phi}}_{\theta}^{\mathrm{T}}\bar{\boldsymbol{\Phi}}_{\theta}\right)^{-1}\bar{\boldsymbol{\Phi}}_{\theta}^{T}\begin{bmatrix} \theta_{N,1} - \theta_{0,1} \\ \vdots \\ \theta_{N,p} - \theta_{0,p} \end{bmatrix} \tag{3.19}$$

Where $\bar{\boldsymbol{\Phi}}_{\theta}$ is the vector of regressors shown in Equation (3.20).

$$\bar{\boldsymbol{\Phi}}_{\theta} = \begin{bmatrix} \boldsymbol{\Phi_1} \\ \boldsymbol{\Phi_2} \\ \vdots \\ \boldsymbol{\Phi_p} \end{bmatrix} \tag{3.20}$$

A similar expression can be derived using the position variables in Equation (3.15) as shown in Equation (3.21).

$$\begin{bmatrix} m_{1,1} \\ m_{1,2} \end{bmatrix} = \left(\bar{\boldsymbol{\Phi}}_{\mathrm{xy}}^{\mathrm{T}} \bar{\boldsymbol{\Phi}}_{\mathrm{xy}}\right)^{-1} \bar{\boldsymbol{\Phi}}_{xy}^{T} \begin{bmatrix} x_{N,1} - x_{0,1} \\ y_{N,1} - y_{0,1} \\ \vdots \\ x_{N,p} - x_{0,p} \\ y_{N,1} - y_{0,1} \end{bmatrix} \tag{3.21}$$

Where, $\overline{\boldsymbol{\Phi}}_{\mathrm{xy}}$, the vector of the position regressors, is given by Equation (3.22).

$$\bar{\boldsymbol{\Phi}}_{xy} = \begin{bmatrix} \boldsymbol{\Phi}_{xy} \\ \boldsymbol{\Phi}_{xy} \\ \vdots \\ \boldsymbol{\Phi}_{xy} \end{bmatrix} \tag{3.22}$$

Also, the position regressor, $\boldsymbol{\Phi}_{xy}$ is given by Equation *(3.23)*.

$$\boldsymbol{\Phi}_{xy} = \mathrm{T} \begin{bmatrix} \sum_{i=0}^{N-1} \omega_{\mathrm{r,i}} \cos\left(\theta_i + \frac{T\omega_i}{2}\right) & \sum_{i=0}^{N-1} \omega_{\mathrm{l,i}} \cos\left(\theta_i + \frac{T\omega_i}{2}\right) \\ \sum_{i=0}^{N-1} \omega_{\mathrm{r,i}} \sin\left(\theta_i + \frac{T\omega_i}{2}\right) & \sum_{i=0}^{N-1} \omega_{\mathrm{l,i}} \sin\left(\theta_i + \frac{T\omega_i}{2}\right) \end{bmatrix} \tag{3.23}$$

Note, however, that if we know the absolute value of l, and obtain $m_{2,1}$ and $m_{2,2}$ from (3.21), then we can obtain the values of $m_{1,1}$ and $m_{1,2}$ from Equation (3.13) and (3.18) by simple algebra.

$$\begin{bmatrix} m_{1,1} \\ m_{1,2} \end{bmatrix} = \begin{bmatrix} m_{2,1} \\ -m_{2,2} \end{bmatrix} * \frac{l}{2} \tag{3.24}$$

3.2.3 Intermittent resetting. The indoor environment has some features that can easily be exploited in the system design to enhance the accuracy during navigation. Such features include the orthogonal arrangement of walls and presence of doorposts. This research has exploited the latter in the design of the intermittent pose resetting system. The heading angle contributes immensely to the odometry drift. The information from the RFID has therefore been used in conjunction with the fixed locations of door-markers to reset the position and heading

angle of the robot intermittently. Figure 3.2 is the flowchart illustrating how the system operates. This serves to eliminate non-systematic errors that have accumulated during its navigation in a particular room so that this error will not propagate into the pose estimation within other rooms.

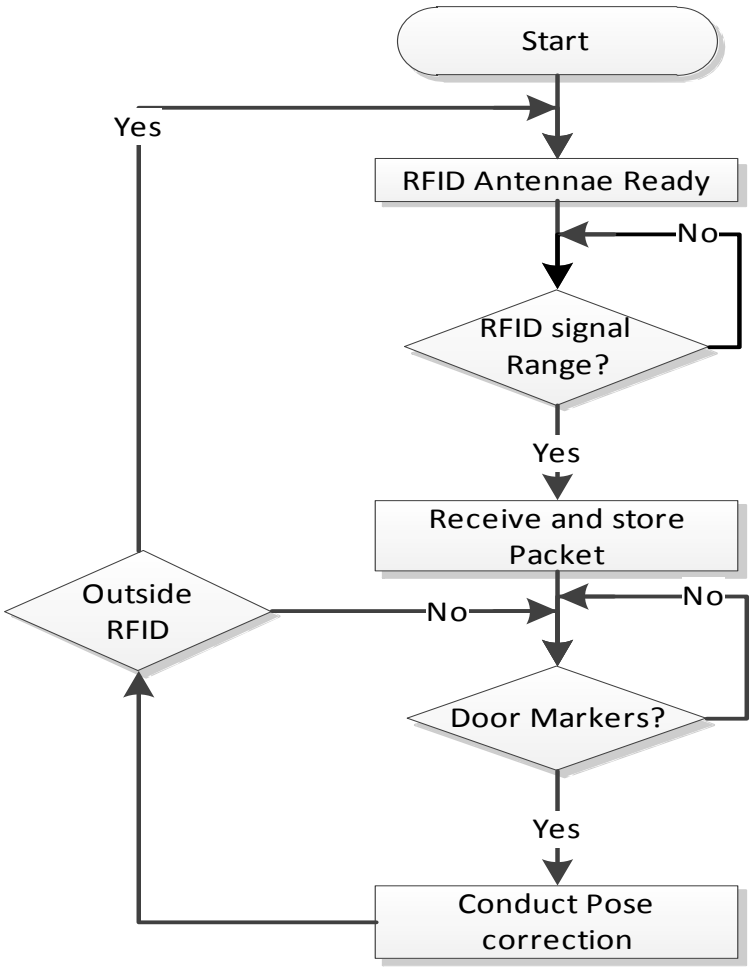


Figure 3.2. This is the flow chart of the intermittent resetting technique.

Such exploitation of indoor environment structure is a reasonable and effective approach to facilitate the navigation task for mobile robots. And once installed, the use of RFID door-markers impose minimal burden (in terms of service or maintenance) on elderly users of the

mobile robot system. Each doorpost is equipped with an RFID-tag and door-markers and the robot carries with it a tag identification antenna. The tag is coded with information including its associated door-marker global position in the indoor environment, the doorpost orientation (along the global x axis or y axis), name of the door, etc. Whenever the robot comes into the communication range of the doorpost tag, the antenna reads and stores the tag's data. These data are used for the resetting of the true position and the heading angle of the robot when it crosses the door-markers.

The door-markers are mounted on the doorposts and arranged in a pattern as shown in Figure 3.3. In Figure 3.3, δ is defined as the angle between the robot direction of motion and the doorpost global orientation. Using trigonometric equations and taking into account the geometric arrangements of the door-markers, we arrive at Equation (3.25) and Equation (3.26)

$$tan\,\delta = \frac{v_1 t_1}{k_1} = \frac{sin\,\delta}{cos\,\delta} \tag{3.25}$$

$$cos\,\delta = \frac{k_2}{v_2 t_2} \tag{3.26}$$

Combining the two equations gives Equation *(3.27)*.

$$sin\,\delta = \frac{v_1 t_1}{k_1} \cdot \frac{k_2}{v_2 t_2} \tag{3.27}$$

Here, k_1 and k_2 are the fixed, known distances separating the door-marker sensors on the robot and the door-markers on the doorpost, respectively. t_1 is the time interval between the instances when the door-marker sensor 1 and door-marker sensor 2 respectively detects the same door-marker (e.g., door-marker 1).

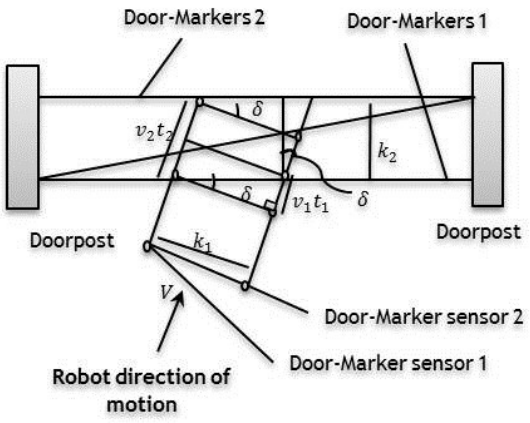


Figure 3.3. The arrangement of the door-markers showing the geometric relationship between the door-markers and the door-marker sensors on the robot.

Also, t_2 is the time interval between the instances when the same door-marker sensor (e.g., door-marker sensor 1) detects door-marker 1 and door-marker 2 respectively. Since we do not have a system for measuring the accurate instantaneous velocity in the system, we assume the velocity of the vehicle remains constant from the time it crosses the first door-marker till it crosses the second door-marker, i.e., acceleration is negligible and $v_1 = v_2$. This is a reasonable assumption since the distance between the two markers is reasonably small. Factoring this assumption into the derivation results in Equation (3.28)

$$\delta = sin^{-1}\left(\frac{t_1 k_2}{t_2 k_1}\right) \tag{3.28}$$

Let φ be the global orientation of the door; then the new true heading θ_{new} of the robot is defined as Equation (3.29).(3.29)

$$\theta_{new} = \varphi + \delta \tag{3.29}$$

Here, a performance measure for comparing the quality of the estimation per the introduction of the intermittent resetting technique called *Estimation Improvement Ratio (EIR)* has been defined as expressed in Equation (3.30).

$$EIR = \frac{Estimation\ Error\ without\ Resetting}{Estimation\ Error\ with\ Resetting} \tag{3.30}$$

This is a "unitless" quantity that expresses the improvement in estimation error as the proportional number of times the estimation error is reduced due to application of the error mitigation technique.

3.3 Mapping

Having achieved an improved accuracy for the localization, we continue to use the information from the robot's position to project the position of obstacles in the environment observed by the robot whiles navigating the environment. From above, the state vector is defined as the position and orientation of the robot given by Equation (3.6). This research used the SICK Laser Measurement Sensor (LMS) (see Figure 3.4 and Figure 3.5) for measuring distance. Such laser-based sensors are available commercially from a number of manufacturers with eye-safe characteristics making them suitable for use on a navigating robot in the home. The observation model can be represented as Equation (3.31).

$$z(t) = H\boldsymbol{x}(t) + \mathcal{N}(0, Q) \tag{3.31}$$

Where the H is the observation matrix and the $z(t)$ is the observation at time t and the $\mathcal{N}(0, Q)$ is a Gaussian noise vector with zero mean and covariance Q.

Less emphasis will be placed on the noise when assuming a small indoor environment which means short measurement distances and the error is therefore negligible. The map building is done by first referencing the points into the local coordinate frame of the robot and then using

coordinate transformation to determine the global position of the observed obstacle. Referencing the point (x_{obs}, y_{obs}) to the coordinate frame of the robot can be done by resolving the point directly into x and y components using the measuring angle of the LMS's laser ray.

Figure 3.4. A picture of the SICK Laser Measurement Sensor (LMS).

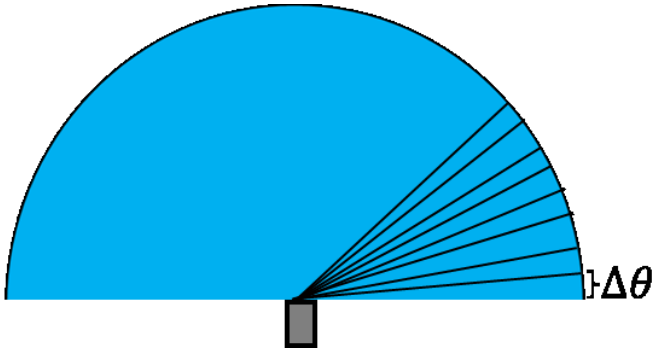


Figure 3.5. The field of view of the LMS.

$$\begin{pmatrix} x_{obs}^R \\ y_{obs}^R \end{pmatrix} = \begin{pmatrix} x_k^R \\ y_k^R \end{pmatrix} + r_n \begin{pmatrix} \cos(\theta + n\Delta\theta) \\ \sin(\theta + n\Delta\theta) \end{pmatrix} \tag{3.32}$$

Here, $\Delta\theta$ is the angular resolution of the LMS and n denotes the nth ray counting counterclockwise as shown in Figure 3.5. The (x_k^R, y_k^R) is the current position of the robot in the robot's coordinate frame and (x_{obs}^R, y_{obs}^R) denotes the true position of a point on the observed obstacle with respect to the robot's coordinate frame. The next step will be to transform the point

from the robot's coordinate frame into the global coordinate frame to form a point in the global map. This can be done using the compounding equation in Equation (3.33); assuming the current position of the origin of the robot's coordinate frame $\boldsymbol{R}$ is located at (x_R^G, y_R^G), that is, position (x_R, y_R) relative to the global coordinate frame.

$$\begin{pmatrix} x_{obs}^G \\ y_{obs}^G \end{pmatrix} = \begin{pmatrix} x_R^G \\ y_R^G \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_{obs}^R \\ y_{obs}^R \end{pmatrix} \tag{3.33}$$

Equation (3.32) can be used to build the local map of the current environment as seen by the navigating robot and then the entire local map can be transformed to the global map intermittently using Equation (3.33). This research employs this technique to build the occupancy grid by discretizing the entire environment into a regular grid. The observed points are projected onto this grid world, and where an obstacle is observed is denoted by binary ones and the free spaces are left as binary zeros.

3.4 Summary of Localization and Mapping Methodology

This chapter has outlined the methodologies employed in the estimation of the location and orientation (localization) of a navigating wheeled robot. An odometry error mitigation approach using RFID and door-markers has been designed to improve the position estimation. This is followed by how the system's observations and location are combined to build a consistent global map of the environment. The modeling assumes a laser measuring instrument such as the one shown in Figure 3.4. Chapter 4 presents a pathfinder that works on the occupancy grid map built using the methods covered in this chapter.

CHAPTER 4

Path Planning (The A-r-Star)

4.1 Introduction

This chapter presents the graph search algorithm developed by this research for the path finding in a given binary grid. The algorithm will enable a mobile robot to navigate intelligently in both dynamic and cluttered indoor environment. This novel algorithm is called $\boldsymbol{A}_{\boldsymbol{r}}^{*}$ (pronounce "A-r-Star"), and is a modified version of the well known $\boldsymbol{A}^{*}$ pathfinder for path planning in a uniform grid world. This new algorithm outperforms $\boldsymbol{A}^{*}$ pathfinder in a sparse uniformly gridded world and matches $\boldsymbol{A}^{*}$ in a cluttered world. It does this by interweaving the building of a non-uniform grid out of a uniform gridded world with path-finding. When given a uniform grid world, $\boldsymbol{A}_{\boldsymbol{r}}^{*}$ decimates the nodes within a given radius/range (r) and forms bigger nodes out of them. Because the $\boldsymbol{A}_{\boldsymbol{r}}^{*}$ builds upon the $\boldsymbol{A}^{*}$ algorithm, it is expedient to introduce briefly the $\boldsymbol{A}^{*}$ algorithm, however for detailed understanding of $\boldsymbol{A}^{*}$ algorithm; its properties and limitations, the reader is directed to (Hart et al., 1968; Dechter et al., 1985). This chapter first introduces the $\boldsymbol{A}_{\boldsymbol{r}}^{*}$ pathfinder with its properties, challenges and some developed solutions to these challenges. Additionally, an incremental version of $\boldsymbol{A}_{\boldsymbol{r}}^{*}$ has been developed along with a means to exploit the information stored in a previously explored/searched graph for subsequent planning (given that the world remains static).

4.2 A-star Algorithm Description

$\boldsymbol{A}^{*}$ is a best-first search algorithm that finds the least costly path from an initial configuration to a final configuration in a given finite and static grid world. It uses an estimate of the start distance $g(s_{Start}, s)$ and heuristic estimation of the goal distance $h(s, s_{Goal})$ to define a

cost/sorting function, $f(s)$ as shown in Equation (4.1). Where $f(s)$ is a heuristic estimate of the cost of going from $s_{Start}$to s_{Goal} passing through node s. For the purpose of simplicity of notation, the goal node and start nodes will be dropped when they appear as part of the argument of a function giving $g(s_{Start}, s) = g(s)$ and $h(s, s_{Goal}) = h(s)$. The heuristic cost function is therefore defined by (4.1)

$$f(s) = g(s) + h(s) \qquad (4.1)$$

Generally, the $\boldsymbol{A}^*$ algorithm maintains two lists, namely the $OPEN$ list and $CLOSED$ list. The $OPEN$ list is a priority queue of all the states (nodes), which have at least one of their predecessors already explored, and as such are potential candidates for next exploration. The $CLOSED$ list holds the candidates that have been explored at least once (and often the blocked nodes). The algorithm starts with an empty $OPEN$ list and populates it with the starting node. At the beginning of every iteration, the node with the minimum $cost(s)$ is popped from the $OPEN$ list. If that is the goal node, the algorithm terminates and follows back-pointers to extract the shortest path (i.e., the path with minimum cost). Else, that node is placed on the $CLOSED$ list and then expanded (i.e., the neighbors are generated and conditionally placed on the OPEN list). It proceeds with the iteration until the goal node is expanded or the $OPEN$ list becomes empty in which case it returns *'no path found'*. The pseudo code is shown in Algorithm 1**.**

{1} $\boldsymbol{Main}()$

{2} $g(s_{Start}) := 0;$

{3} $parent(s_{Start}) := s_{Start};$

{4} $OPEN := \emptyset$

```
{5}   OPEN.Insert(s_Start, g(s_Start) + h(s_Start));
{6}   CLOSED := S_b
{7}   while OPEN ≠ ∅ do
{8}       s := OPEN.Pop();
{9}       if s = s_goal then
{10}          return "Unblocked path found";
{11}      CLOSED := CLOSED ∪ s;
{12}      neighbors := Expand(s, CLOSED)
{13}      foreach s' ∈ neighbors do
{14}          if s' ∉ CLOSED
{15}              if s' ∉ OPEN
{16}                  g(s') := ∞;
{17}                  parent(s') := NULL;
{18}          UpdateNode(s, s');
{19}  return "no unblocked path found'
{20}  end
{21}  Expand(s, CLOSED)
{22}      r := 1;
{23}      neigbor(s) := NeighborsGen(s)
{24}      foreach s' ∈ neighbor(s)
{25}          if s' ∉ CLOSED;
```

```
{26}            neighbors := neighbors ∪ s'
{27}        return neighbors
{28}    end
{29}    UpdateNode(s, s')
{30}        if g(s) + c(s, s') < g(s') then
{31}            g(s') := g(s) + c(s, s');
{32}            parent(s') := s;
{33}            f(s') := g(s') + c(s', s_goal)
{34}            if s' ∈ OPEN then
{35}                OPEN.Remove(s');
{36}            OPEN.Insert(s', g(s'), f(s'));
{37}    end
```

Algorithm 1. This is the pseudo code for the A^* algorithm.

4.3 A-r-Star Algorithm

4.3.1 Definitions. *Node Distance* of s'' from s is defined as the fewest number of nodes that will be touched by a straight line connecting s and s'' as shown in Figure 4.1. This is equivalent to the distance from s to s'' using the *'chessboard'* distance metric. This implies that two nodes with different Euclidian distances can have the same node distances. *Level-R-Neighbors* of a node s comprise of all nodes with node distances equal to R from s. In Figure 4.1, all the nodes at a given level bear their node distance, e is the Euclidean distance and n is the node distance (R). This implies that, the *Level-1-Neighbors* of s are its 8 contiguous

neighbors (see Figure 4.1). Here *R* refers to the radius of the 'ball' (a box in the case of square grid) formed by connecting the centers of all *Level-R-Neighbors*.

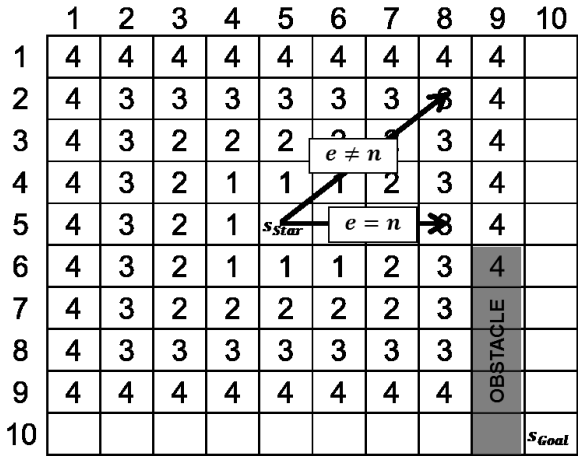


Figure 4.1. Showing the Level-R-Neighbors for a given node start node.

4.3.2 Algorithm description and implementation. The A_r^* algorithm is a modified version of the A^* algorithm that interweaves node decimation with path-finding in a uniform grid/mesh. For a given node, A^* counts only immediate nodes (4- or 8- connected nodes) as its neighbors. This implies that even when all the nodes in a particular area are similar, it will still do some computation for all of them. The effect is that, if an obstacle blocks the direct line of sight close to the goal, the number of nodes needed to be explored more than doubles which translates into increasing the search time. Figure 4.2 illustrates this phenomenon and how this contributes to 'kinks' in the path returned by A^*. The circles indicate the nodes that were explored before the path was found.

Using the idea from non-uniform mesh building (Schroeder et al., 1992), a cluster of nodes with similar characteristics can be represented with a bigger node with minimal loss of information (see Figure 4.3). The A_r^* algorithm therefore allows collapsing of such nodes into a

single node with properties that commensurate with the union of those nodes. For instance, in Figure 4.3, the nodes at Level-1 to Level-3 have been decimated to form one big node with the original Level-4-Neighbors of s_{Start} now forming the neighbors of this new node. For multi-level terrain, one will use a distance transform (Huang and Mitchell, 1994; Matsumoto, Ino and Ogasawara, 2001) to identify changes in the nodes; but for the binary occupancy grid such as the one used in this paper, the task reduces to identifying the nearest obstacle to the current node.

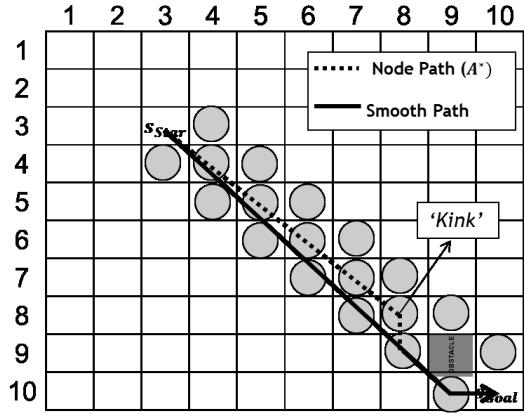


Figure 4.2. This figure shows how A-Star responds to an obstacle.

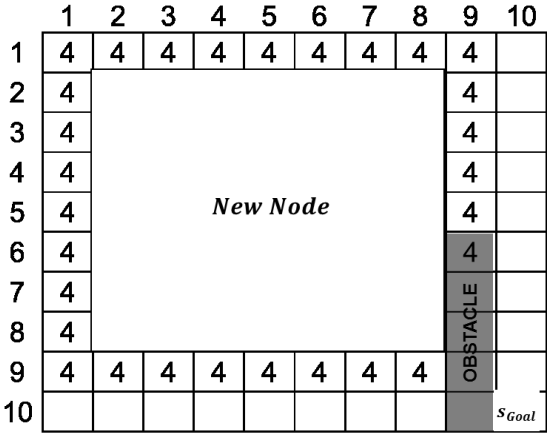


Figure 4.3. This figure shows decimated Level-1 through Level-3 nodes to form a single node.

The overall effect is a reduction in the number of nodes needed to be explored, computation cost and increased search speed in a sparse uniform gridded world. The *'star'* in the name does not suggest that it always finds an optimal path, it is just intended to retain its resemblance to its namesake, $\boldsymbol{A}^*$. The *r* stands for radius (range) defined as the maximum allowable radius (in node distance) of a 'ball' of nodes that can be counted as the neighbors of a given node (see Figure 4.1). Thus, only *Level-R-Neighbors* are considered during the search where $1 \leq R \leq r$.

Let the node distance from the closest obstacle node to a given node s be $r_0(s)$, then at the end of the search: $r_0(s) \leq r \; \forall \; s \in S_{Open}$. Implementation-wise, this is achieved by searching for the minimum *R,* such that, at least one of the *Level-R-Neighbors* of a node being expanded belongs to the set of blocked nodes. Then all nodes in the neighborhood of s such that $R < r_0(s)$ are tagged as skip nodes ($s'.tag = Skip$) and nodes such that $R = r_0(s)$ are returned as the *Level-R-neighbors*. The pseudo code for the algorithm is similar to that of $\boldsymbol{A}^*$ with two modifications. The first modification is done to the *Expand subroutine* and the resulting algorithm is referred to as $\boldsymbol{Basic\ A_r^*}$. This is achieved by replacing the lines {21} to {28} in Algorithm 1. by the lines in Algorithm 2.

Note that nodes that are tagged skip will never make it to the $OPEN$ list (Algorithm 2 line {30}) but a node might make it to the $OPEN$ list before being tagged as skip. Thus, in the $\boldsymbol{Basic\ A_r^*}$ algorithm, tag *open* dominates/overwrites tag *skip*. The second modification is to switch the tag dominance to *skip* dominating *open*. Thus, nodes that made it to the $OPEN$ list from one node before being tagged as skip from another node will not be expanded (i.e., they will stay on the $OPEN$ list till the algorithm terminates).

{21} $\boldsymbol{Expand}(s, S_{Closed}, S_f, s_{goal}, r)$

{22} $R := 1;$

{23} $skipflag := true$

{24} $while\ R \leq r$

{25} $levelrneighbor(s) := \boldsymbol{LRNG}(R, s)$

{26} $\boldsymbol{foreach}\ s' \in levelrneighbor(s)$

{27} $\boldsymbol{if}\ s' \notin S_f$

{28} $skipflag := false$

{29} $\boldsymbol{continue}$

{30} $\boldsymbol{Elseif}\ \ s'.tag \neq skip\ and\ s' \notin S_{Closed}$

{31} $lrneighbors := lrneighbors \cup s'$

{32} $\boldsymbol{if\ not}(skipflag)$

{33} $\boldsymbol{return}\ lrneighbors$

{34} $\boldsymbol{Foreach}\ s' \in lrneighbors$

{35} $s'.\boldsymbol{tag} := skip;$

{36} $r++$

{37} $\boldsymbol{return}\ lrneighbors$

{38} $\boldsymbol{end}$

Algorithm 2. First modification resulting in the Basic A-r-Star

The resulting algorithm after the last modification is called the $\boldsymbol{A_r^*}$ algorithm. The $\boldsymbol{A_r^*}$ pseudo code can be derived from the $\boldsymbol{Basic\ A_r^*}$ by inserting the lines in Algorithm 3 after line {10} in Algorithm 1..

{11}	$\boldsymbol{if}\ s.\boldsymbol{tag} = skip\ \boldsymbol{then}$
{12}	$\boldsymbol{continue}$

Algorithm 3. Second modification resulting in the A-r-Star

4.3.3 Finite radius (r) verses infinite radius (∞). The choice of r can be fixed from the start of the algorithm to a finite value. For example, choosing $r = 1$ reduces the algorithm to $\boldsymbol{A_1^*}$ which is essentially $\boldsymbol{A}^*$. But choosing an appropriate finite value for r requires absolute knowledge of the environment since the choice of r affects the performance of the algorithm. The simplest solution is to allow the algorithm to evolve and discover the value of *r* during the search. This is achieved by pegging the value of *r* at infinity (i.e., choosing $r = \infty$). This leads to what is referred to as the $\boldsymbol{A_\infty^*}$ (A-Infinity-Star). Here, infinity (∞) is defined as $r \geq r'$ where r' is the radius of the biggest 'ball' of continuous free nodes available in the search space. It is trivial to derive that for an $M\ x\ M$ grid world, $r' < M/2$ always holds (strictly less because the node under consideration cannot be counted as part of the radius and r is undefined for $s \notin S$).

4.3.4 Choice of Level-R-Neighbors Generator (LRNG). The ***LRNG*** function is responsible for generating *Level-R-Neighbors* of a given node s. A good choice of the ***LRNG*** function should return at every *Level-R*, all and only the nodes at radius *R* from s as the *Level-R-Neighbors* of s. This is a necessary condition for $\boldsymbol{A_r^*}$ to be complete and correct. On square grids, choosing the ***LRNG*** is a trivial task but this is not trivial when other geometric shapes are used for the gridding.

Theorem 1: *If the Level-R-Neighborhood generation function of* $\mathrm{A_r^*}$ *at every Level-R returns all and only the nodes at radius R from s for* $R = 1, \ldots, r$ *then the* $\boldsymbol{A_r^*}$ *algorithm is complete and correct.*

Proof: Let us assume the contrary that the path, $s_1, \ldots, s_i, \ldots, s_n$ returned by the algorithm is incorrect, thus there exists at least one s_i between s_1 and s_n, $1 < i < n$, such that $s_i \in S_b$. This will imply that a blocked node s_i got expanded by the algorithm which contradicts the line {30} of Algorithm 2 and therefore cannot be true. Similarly, let us assume that a path actually exists but $\boldsymbol{A_r^*}$ did not find a path. This will imply at a certain *Level-R*, the algorithm failed to return a node $s \in S_f$ and hence assumed there was not a path available. This is the necessary condition for a function to qualify as an ***LRNG*** and hence poses a contradiction that cannot surface.

4.4 Properties of the A-r-Star Algorithm

4.4.1 Completeness. Like $\boldsymbol{A^*}$, the $\boldsymbol{A_r^*}$ algorithm is complete meaning it will find a path if one exists between the start and the goal node. The condition for completeness solely depends on the ***LRNG*** as stated in **Theorem 1**.

4.4.2 Correctness. The correctness property holds for the $\boldsymbol{A_r^*}$ algorithm. This implies that if $\boldsymbol{A_r^*}$ does return a path for a given starting and ending node, then that path is a truly unblocked path (that is, the path exists and is correct).

4.4.3 Termination in finite time. The use of the *CLOSED* list and the tagging ensure that $\boldsymbol{A_r^*}$ expands (or tags) every node once. Since the world is an $M \times M$ grid, where M is finite, it is implicit that the algorithm will terminate in finite time.

4.4.4 Convergence to A-Star. The performance of the $\boldsymbol{A_r^*}$ (and for that matter $\boldsymbol{Basic\ A_r^*}$) approaches that of $\boldsymbol{A^*}$ for worlds with increasing clutter. Let us define a perfectly cluttered 2D world as a grid configuration such that every *Level-1-Neighbor* of

$s\ \forall s: s \in S_f$ contains at least one $s : s \in S_b$. Given a perfectly cluttered world, the $\boldsymbol{A_r^*}$ will be forced not to tag any node as skip. Thus, $\boldsymbol{A_r^*}$ will implicitly operate as $\boldsymbol{A_1^*}$ which is essentially $\boldsymbol{A}^*$.

Theorem 2: In a perfectly cluttered world, $\boldsymbol{A_r^*}$ and $\boldsymbol{Basic\ A_r^*}$ converge to $\boldsymbol{A}^*$ for all positive integer values of r.

***Proof*:** Assume the set S is a perfectly cluttered 2D environment. Then every *Level-R-Neighborhood* of a free cell $s \in S_f$ will contain at least one $s': s' \in S_b$ and thus $r_0(s) = 1\ \forall\ s \in S_f$; from subsection 4.3.2 , $1 \leq R \leq r_0(s) \Longrightarrow R = 1$ for all nodes. But the $\boldsymbol{A_r^*}$ runs at *Level-R* = 1 throughout the search and so it is intuitive that after the search, $r_0(s) = 1\ \forall\ s \in S_{Open}$ and hence the proof.

4.4.5 Any angle path planning. Most of the grid-based path planning algorithms that operate on a 2D world use discrete state transitions that are artificially constrained to a small set of possible headings angles (e.g., $0, \frac{\pi}{4}, \frac{\pi}{2}$, etc.). The ramification is that the path returned by even the optimal grid planner will not be the shortest possible continuous path (see Figure 4.2). The $\boldsymbol{A_r^*}$ algorithm is not constrained to a finite set of angles. This means that it sometimes returns a more natural and smooth path than $\boldsymbol{A}^*$. The $\boldsymbol{A_r^*}$ algorithm under sparse conditions can therefore be considered as an *'any angle path planner'*.

4.4.6 Reaction to obstacle. The $\boldsymbol{A_r^*}$ algorithm reacts to an obstacle by planning in small steps till it avoids the obstacle. This mimics intuitive navigating behavior. Much caution is taken when navigating close to an obstacle than when navigating far from an obstacle.

4.4.7 Definitions. Given two nodes, s_{Start} and s_{Goal}, a *node path* is defined as any chain of nodes $s_1, s_2, \ldots, s_n\ \in S_p$ such that every s_{i+1} belongs to the *Level-1-Neighbors* of s_i and

$s_i = s_j$ *iff* $i = j$ and $s_i \notin S_b \; \forall \; i, j \in [1, n]$. A continuous path is an unbroken curve drawn from the center of s to the center of s'' without passing through a blocked node. Let $\Gamma(s, s'')$ be the set of all continuous paths linking the centers of s and s'', and let $\Gamma_{\mathrm{p}}(s, s'')$ be a specific continuous path linking the centers of s and s''. Define $\Gamma_{\mathrm{m}}(s, s'') = \min_{cost(s,s'')} \Gamma(s, s'')$. Thus, for a given grid map, if $Line\ Of\ Sight, \boldsymbol{LOS}\ (s, s'') = true$ (i.e., there is an unobstructed straight line path from s to s'') then $\Gamma_{\mathrm{m}}(s, s'')$ = a straight line.

4.5 Challenges of the A-r-Star Algorithm

4.5.1 Bulges. $\boldsymbol{A_r^*}$ usually returns a path with $bulges$ in it (see Figure 4.4). This is a major challenge to the performance of $\boldsymbol{A_r^*}$. Given two nodes, s and s'', if $\boldsymbol{LOS}\ (s, s'') = true$ and $\Gamma_p(s, s'') \neq \Gamma_{\mathrm{m}}(s, s'')$ = straight line, then $\Gamma_p(s, s'')$ is called a $bulged$ path and, in general, any $\Gamma_{\mathrm{p}}(s, s'') \in \Gamma(s, s'')$: $\Gamma_{\mathrm{p}}(s, s'') \neq \Gamma_{\mathrm{m}}(s, s'')$ is said to be a *bulged* path. This definition implies that a *'kink'* is a type of *bulge* (see Figure 4.2). There are two main causes of *bulges* in $\boldsymbol{A_r^*}$; namely, *angular constraint (kinks)* and *premature tagging*. *Angular constraint* occurs around obstacles where the algorithm navigates in small steps; the navigation angles are thus artificially constrained to a small set of angles. *Premature tagging* occurs due to local minima. The greedy heuristic of $\boldsymbol{A_r^*}$ is initially drawn into a local minimum. This is accompanied by the tagging of nodes as *skip*. When the algorithm bounces back from the local minimum, these nodes which were prematurely tagged as *skip* are not considered for expansion and this creates a *bulge*.

4.5.2 Non-optimality. Due to *bulges*, the path returned by $\boldsymbol{A_r^*}$ is not always optimal. *Bulges* introduce extra cost into the sorting function by increasing the estimated goal distance thereby placing some nodes at a disadvantage. The goal distance becomes dependent on the configuration of the obstacles in the environment.

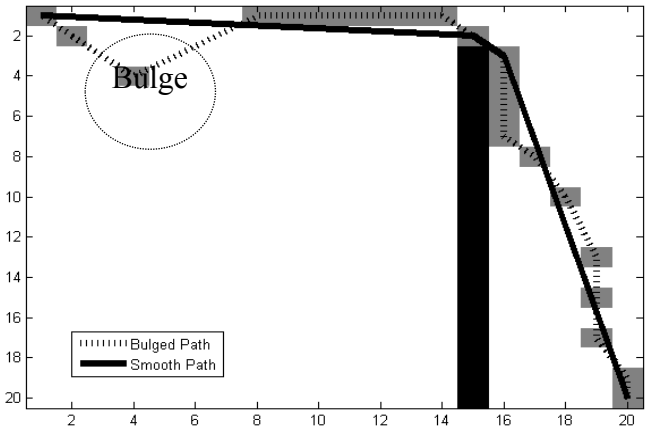


Figure 4.4. An example of a path planned by A-r-Star highlighting the challenges posed by bulges and bulge elimination using post smoothing.

Consequently, $g(s_1) < g(s_2)$ does not always imply $g^*(s_1) < g^*(s_2)$ during the search (where $g^*(s_n)$ is the actual optimal path between s_{start} and s_n). Thus, unlike the $\boldsymbol{A}^*$ algorithm, $\boldsymbol{A}_{\boldsymbol{r}}^*$ does not always guarantee an optimal path.

4.6 Proposed Solutions to the Challenges

4.6.1 Bulge removal using smoothing. A Post Dissociation Smoothing (PDS) algorithm similar to that outlined in (Daniel et al., 2010) has been developed and implemented to eliminate the *bulges* in the path returned by $\boldsymbol{A}_{\boldsymbol{r}}^*$. This shortens the path and gets it closer to the shortest possible path. Algorithm 4 shows the pseudo code for the PDS. Both $\boldsymbol{LOS}$ and $\boldsymbol{DissociatePath}$ are derived from the Bresenham line drawing algorithm.

4.6.2 Non-optimality: Interleave Smoothing with Post Dissociative Smoothing (IS-PDS). Some path configurations cannot be smoothed into the shortest possible/optimal path. To increase the chances of returning a path that can be smoothed to optimal, the search has been interleaved with the smoothing algorithm. This is similar to the idea in (Daniel et al., 2010). The post dissociative smoothing is then applied to the path as a post process. Note that *PDS* can be

applied iteratively from goal to start and vice versa until subsequent application does not shorten the path by a distance greater than ε (where ε is a user defined threshold).

```
{1} Smooth(Path)
{2}     DisPath = DissociatePath(Path)
{3}     s_p = DisPath.pop(); s_cur = DisPath.pop();
{4}     smoothPath.push(s_p)
{5}     while Path ≠ ∅
{6}         if LOS(s_p, s_cur) then
{7}             s_old = s_cur
{8}             s_cur = DisPath.pop
{9}         continue
{10}        smoothPath.push(s_p)
{11}        s_p = s_old
{12}        smoothPath.push(s_cur)
{13}    return smoothPath
{14}    end
```

Algorithm 4. Post Dissociative Smoothing Algorithm.

This results in *Interleave Smoothing with Iterative Post Dissociative Smoothing (IS-IPDS)*. To implement the interleave smoothing; replace the ***UpdateNode*** function of the $\boldsymbol{A}_{\boldsymbol{r}}^{*}$ algorithm with Algorithm 5.

{1} $\boldsymbol{UpdateNode}(s, s')$

{2} $\quad \boldsymbol{if}\ LOS(parent(s), s')$

{3} $\quad \boldsymbol{if}\ g(parent(s)) + c(parent(s), s') < g(s')\ \boldsymbol{then}$

{4} $\quad\quad g(s') := g(parent(s)) + c(parent(s), s');$

{5} $\quad\quad parent(s') := parent(s);$

{6} $\quad\quad f(s') := g(s') + c(s', s_{goal})$

{7} $\quad\quad \boldsymbol{if}\ s' \in OPEN\ \boldsymbol{then}$

{8} $\quad\quad\quad OPEN.\boldsymbol{Remove}(s');$

{9} $\quad\quad OPEN.\boldsymbol{Insert}(s', g(s'), f(s'));$

{10} $\quad \boldsymbol{else}$

{11} $\quad\quad \boldsymbol{if}\ g(s) + c(s, s') < g(s')\ \boldsymbol{then}$

{12} $\quad\quad\quad g(s') := g(s) + c(s, s');$

{13} $\quad\quad\quad parent(s') := s;$

{14} $\quad\quad\quad f(s') := g(s') + c(s', s_{goal})$

{15} $\quad\quad\quad \boldsymbol{if}\ s' \in OPEN\ \boldsymbol{then}$

{16} $\quad\quad\quad\quad OPEN.\boldsymbol{Remove}(s');$

{17} $\quad\quad\quad OPEN.\boldsymbol{Insert}(s', g(s'), f(s'));$

Algorithm 5. Post Dissociative Smoothing Algorithm

4.7 The Incremental A-r-Star Pathfinder

4.7.1 The direct acyclic graph. Incremental search algorithms are more powerful in handling path planning in dynamic environments. This section highlights the incremental version of $\boldsymbol{A_r^*}$ pathfinder. The development is similar to that of $\boldsymbol{D^* - Lite}$. The planning algorithms

produce a plan that essentially takes the regular grid and builds a directed acyclic graph (DAG) rooted at the node from which the search starts (NB: this node may be different from the start node). Table 4.1 lists some of the planning algorithms and their root nodes. This means after the search pauses, all the nodes on the CLOSED list (explored nodes) are roots of sub-graphs in the DAG and the nodes on the OPEN list are the leaves of DAG. The blocking of a single node on the CLOSED list means that node has been discontinued from the main DAG and so the system has to find a systematic way of reconnecting all the sub-graphs (and leaves) rooted at that blocked node to the graph where possible.

Table 4.1

Root Nodes for the DAG Built by the Various Pathfinders

Path Finder	Root Node
$\boldsymbol{A^*}$	Start Node
$\boldsymbol{D^*/Focussed-D^*}$,	Goal Node
Incremental $\boldsymbol{A^*}$ ($\boldsymbol{LPA^*}$)	Start Node
$\boldsymbol{D^*-Lite}$	Goal Node

This principle has been harnessed in the various incremental planning algorithms in different ways. In some circumstance, the blocked node will put all the leaf nodes beyond a lower bound of the path cost to the goal, and that means that all the leaves on a sub-graph cannot become part of the main graph anymore. Under such circumstances, the system will take the goal and place it as part of another sub-graph. This idea has been exploited in the developing of incremental search algorithms such as $\boldsymbol{LPA^*, D^*, Focussed-D^*}$and $\boldsymbol{D^*-Lite}$, and etc. D* uses the

concept of *RAISED* (Nodes whose cost has increased due to the change in edge cost) and *LOWERED* (nodes whose cost has decreased by the change in edge cost) to propagate the cost changes to all the sequences that contain the edge whose cost has changed. $\boldsymbol{LPA}^*$ exploits this knowledge by using consistency evaluation (*over-consistent* and *under-consistent nodes*) to propagate the edge cost changes to the affected sequences of back-pointers (sub-graphs). $\boldsymbol{D}^* - \boldsymbol{Lite}$ operates in a way similar to $\boldsymbol{LPA}^*$ except that it start searching from the goal and so implements a routine to remove prevent cycles in the graph when propagating the cost changes.

4.7.2 Incremental A-r-Star. The incremental $\boldsymbol{A}_{\boldsymbol{r}}^*$ algorithm consists of essentially two main procedures namely ***PROCESS-STATE*** and ***PRUNE-BRANCH***. The ***PROCESS-STATE*** procedure handles the computation of the DAG and the ***PRUNE-BRANCH*** procedure effects edge cost changes and prunes the sub-graphs and/or leaves from the main DAG by dissolving all the sub-graphs centered at the parent of the blocked node. Note: the dissolution starts from the parent and not the current node because the current node is blocked and need to be removed from the tree along with all its siblings. The ***PROCESS-STATE*** procedure is the same as the $\boldsymbol{A}_{\boldsymbol{r}}^*$ pathfinder presented earlier except that it starts its search from the goal node. This implies the DAG that will be built will be rooted at the goal node. Secondly, we introduce an array that keeps track of the parent-child relationships. The pseudo code for the algorithm for the ***PRUNE-BRANCH*** procedure is as shown in Algorithm 6. The algorithm first looks forward for the parent of the node affected, and saves that in s_p. This node is then placed on the $PNodes$ array. The system then enters into the looping mode where the system continues until $PNodes$ becomes empty. At every loop, the system pops a node from $PNodes$ and stores it in s_c which is the current node under consideration.

{18} $\boldsymbol{PRUNE-BRANCH}(s)$

{19} $s_p = parent(s)$

{20} $PNodes = \emptyset$

{21} $PNodes.\boldsymbol{Insert}(s_c)$

{22} $while\ PNodes \neq \emptyset$

{23} $s_c = PNodes.\boldsymbol{pop}()$

{24} $\boldsymbol{if}\ TAG(s_c) = closed$

{25} $Branches = \boldsymbol{RootBranches}(s_c)$

{26} $PNodes.\boldsymbol{Insert}(Branches)$

{27} $CLOSED.\boldsymbol{Remove}(s_c)$

{28} $\boldsymbol{esleif}\ TAG(s_c) = open$

{29} $OPEN.\boldsymbol{Remove}(s_c)$

{30} $TAG(s_c) = NEW$

{31} $g(s_c) = \infty$

{32} $OPEN.Insert(s_p)$

{33} $\boldsymbol{end}$

Algorithm 6 The pseudo code for ***PRUNE-BRANCH***

The system then accrues all its children using $\boldsymbol{RootBranches}$ and puts them on $PNodes$. Set $TAG(s_c) = NEW$ to free the node for subsequent re-planning. NB: If s_c belongs to the CLOSED or OPEN list, it must be removed from it. For example in Figure 4.5, if the robot start navigating from the starts node (3,10) towards the goal node (1,1) and it discovers that node (1,5) is blocked, iterations 1 to 10 are summarized in Table 4.2.

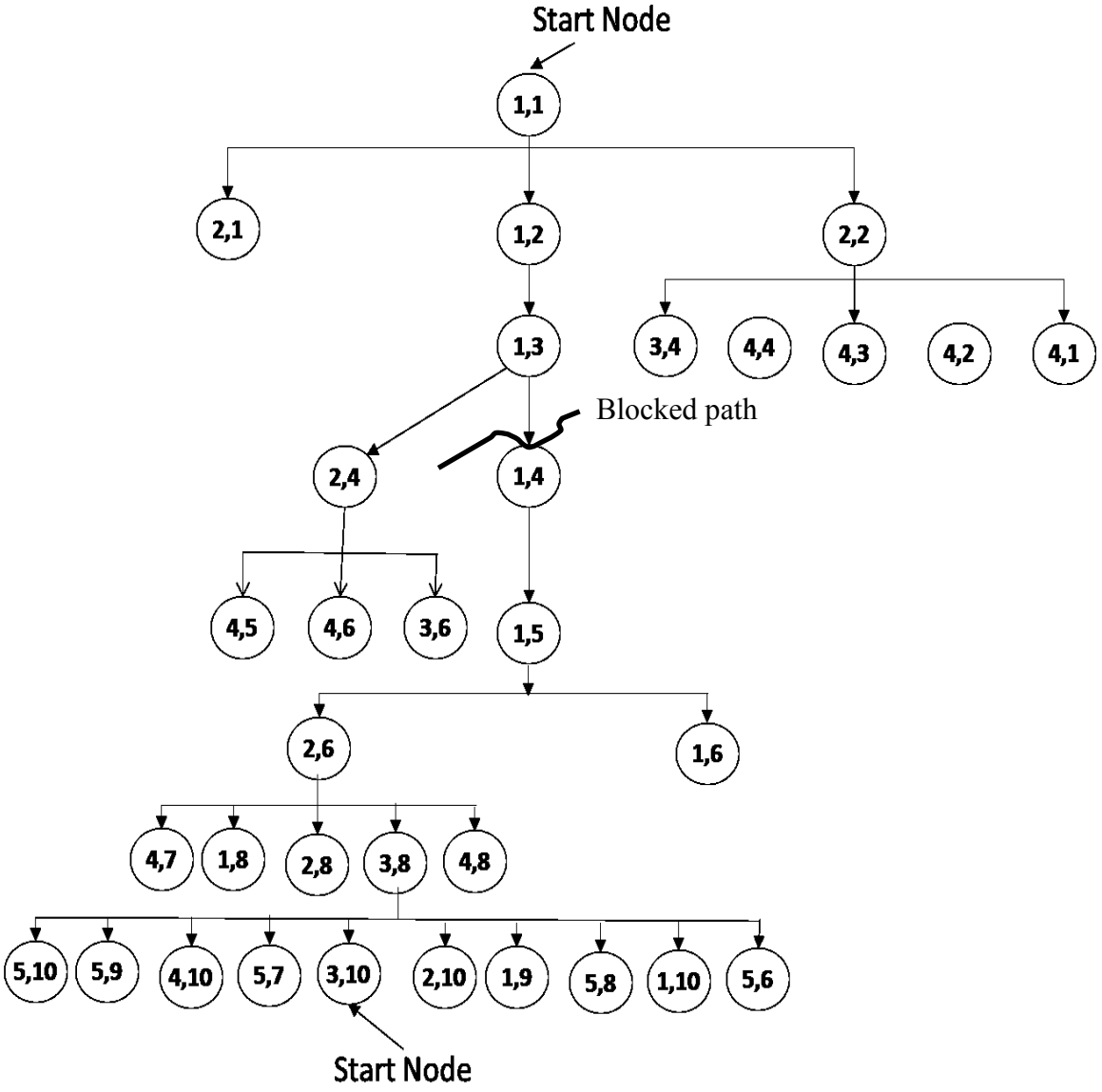


Figure 4.5. A Sample DAG built by the A-r-Star for a 10 x10 grid world without obstacle, when searching from node (1,1) to (3,10).

Iterations 11 to the end only involve the freeing of the leaf nodes. In effect, the other sub-graphs rooted at (1,3) and those which are not rooted at (1,3) are left. The information stored in these sub-graphs can therefore be used for the planning.

Table 4.2

*First Ten Iterations of **PRUNE-BRANCH** Acting on the Sample DAG in Figure 4.5 When Blocked at the Node (1,5)*

Iteration	$PNodes$	S_c	$\boldsymbol{RootBranches}$
1	(1,4)	(1,4)	(1,5)
2	(1,5)	(1,5)	(1,6),(2,6)
3	(1,6), (2,6)	(1,6)	
4	(2,6)	(2,6)	(4,7), (1,8), (2,8), (3,8), (4,8)
5	(4,7), (1,8), (2,8), (3,8), (4,8)	(4,7)	
6	(1,8), (2,8), (3,8), (4,8)	(1,8)	
7	(2,8)	(2,8)	
8	(3,8), (4,8)	(3,8)	(1,9), (1,10), (2,10),(3,10), (4,10), (5,6), (5,7), (5,8), (5,9), (5,10)
9	(1,9), (1,10), (2,10),(3,10), (4,10), (5,6), (5,7), (5,8), (5,9), (5,10)	(1,9)	
10	(1,10), (2,10),(3,10), (4,10), (5,6), (5,7), (5,8), (5,9), (5,10)	(1,10)	

4.7.2.1 Challenge for the incremental A-r-Star. As can be inferred from the ***PRUNE-BRANCH*** algorithm, when the blocked node is very close to the goal; it means there will be a lot of branches to dissolve and this will take a longer time to complete. To handle this for a real

navigating robot, the nodes will be dissolved as soon as they are navigated through with all their branches and thus the dissolution time will spread over the run time.

4.8 Multiple Goal Path Planning

The structure of searched nodes developed by $\boldsymbol{A_r^*}$ can be harnessed in subsequent path searches; this is the case if all subsequent path search tasks starting from the same node but have different destination or they all have the same destination but different starting nodes. The node they share in common is called the root node. Multiple-destination problems can apply in multiple agent based applications wherein a single planner plans paths to send agents from a single point (base station) to multiple destinations. The reverse can apply to scenarios wherein a planner dispatches agents from multiple destinations to a single point to accomplish a single goal that may be beyond the capability of a single agent. Here, the assumption is that the environment remains unchanged. Consider the cost function

$$f(s) = g(s) + h(s) \tag{4.2}$$

If the environment remains unchanged, it is expected that after the search, the cost function will become (4.3) for all the nodes which are on the CLOSED list.

$$f(s) = g^*(s) + h(s) \tag{4.3}$$

This implies that, the same structure can be used to plan the path to any point in the environment from the same root node.

Expanding Equation (4.2) for the fixed environment gives

$$f(s) = g(parent(s)) + c(s, parent(s)) + h(s) \tag{4.4}$$

Since $c(s, parent(s))$ remains constant for a static environment; for every node on the OPEN list, there exists a parent on the CLOSED list and thus $g(parent(s)) = g^*(parent(s))$. This

implies that, the task of planning from the root node to another node with parent on the CLOSED list is trivial because $g(parent(s)) = g^*(parent(s))$. Thus, it reduces to path planning from the parent to the child (NB: there is a direct line of sight between every child and the parent) and the path planning between the root node and the parent of that node which exists already in the previous structure. This approach is similar to the ROADMAP (SungHwan et al., 2012) approach to path planning. The task of planning from the root node to another node beyond the reach of the previous search can be done by changing the heuristic for all the nodes in the *OPEN* list to conform to the current goal node. That is, $h(s, s_{Goal})$ becomes $h(s, s_{NewGoal})$, where $s_{NewGoal}$ is the new destination. This will naturally extend the previous graph towards that new goal.

4.9 Conclusion

This chapter has presented the methodology for path planning developed by through this research. The $\boldsymbol{A_r^*}$ algorithm is detailed with its properties, limitations and the extension to incremental search as well as for handling the multiple destination problems. The next chapter presents some simulation experimental results of the application of both the methodologies developed in Chapter 3 and Chapter 4.

CHAPTER 5

Simulation and Results

5.1 Localization

5.1.1 The prototyping and simulation testbed. The setup for simulation experimental verification of the methodologies outlined in this section was developed on the WebotsTM robotic prototyping and simulation platform from Cyberbotics (see Figure 5.1). The environment is a 3-D interior representation of a five compartment home comprised of a kitchen, living room, bedroom, bathroom and a fitness room.

Figure 5.1. A screenshot of the prototype indoor environment, from the Webots graphics window

This environment is intended to represent a model domestic operating domain for an assistive robot. A model of the Pioneer 2 robot from ActivMedia robotics equipped with SICK LMS 200 has been adopted as the robotic platform in this scenario (see Figure 5.2). A feature of the Webots supervisor node (a software object that can access the state of all objects in a Webots

simulation) was programmed to read at each time instance the actual position of the robot. This serves as our ground truth data.

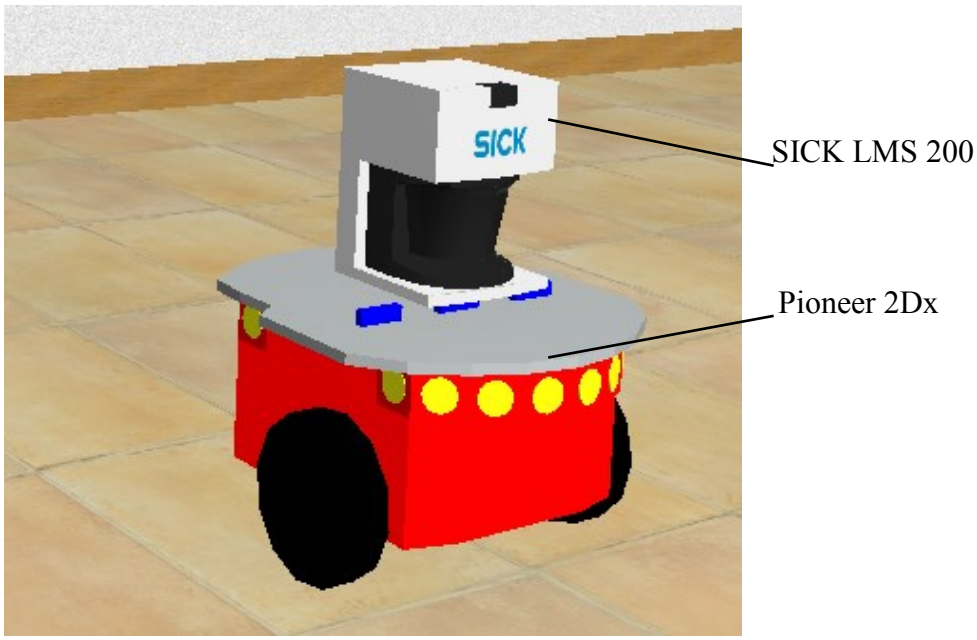


Figure 5.2. This is a screenshot for the pioneer 2DX prototype in Webots.

5.1.2 Calibration results. This simulation assumes direct access to the odometer readings. A modification was made to Equation (3.18) to enable the incorporation of this information for the calibration as follows.

$$\boldsymbol{\Phi} = \left[\sum_{i=0}^{N-1} \frac{\Delta s_{r,i}}{s} \qquad \sum_{i=0}^{N-1} \frac{\Delta s_{l,i}}{s} \right] \tag{5.1}$$

Here, s is the number of encoder readings per one complete rotation of the wheel and the wheels are assumed to be identical, therefore s is the same for both wheels. Four types of trajectories were executed in order to identify the calibration parameter matrix, $\boldsymbol{M}$ in Equation (3.14). From (Borenstein and Liqiang, 1996), the set of trajectories must include at least a straight line

movement, a clockwise rotation and a counterclockwise rotation. Therefore, three types of each of these trajectories have been captured in our calibration: forward-straight-line, forward-left-turn, forward-right-turn, left-revolution, right-revolution, counterclockwise-rotation and clockwise-rotation. These trajectories/movements are defined as

$$Traj. = \begin{cases} No\ Movement & if\ \omega_r = \omega_l = 0 \\ Straight\ Line & if\ \omega_r = \omega_l \\ Rotation & if\ \omega_r = -\omega_l \\ Revolution & otherwise \end{cases} \tag{5.2}$$

After executing these trajectories/movements on the simulated Pioneer 2 robot in Webots, the calibration method identifies the following matrix

$$\boldsymbol{M} = \begin{bmatrix} 0.0412 & 0.0413 \\ 0.2578 & -0.2578 \end{bmatrix} \tag{5.3}$$

This is consistent with this simulated robot's wheel diameter of $d_r = d_l = 0.165\ m$ and axis length of $l = 0.32\ m$, thus validating the accuracy of the least squares calibration technique. The same approach is applicable to a physical robot based on trajectories/movements executed in the real world and the associated odometer readings from real sensors.

5.1.3 Intermittent resetting results. The graphs in Figure 5.3 and Figure 5.4 illustrate the run-time errors in the estimated heading angle and position of the robot after the calibration is applied to mitigate systematic errors. The vertical lines in Figure 5.3 and Figure 5.4 indicate the time instances at which pose resets were made by the robot to update its true position and orientation and thus correct for non-systematic errors that accumulate during navigation. These graphs show how the pose errors vary with and without the intermittent resetting.

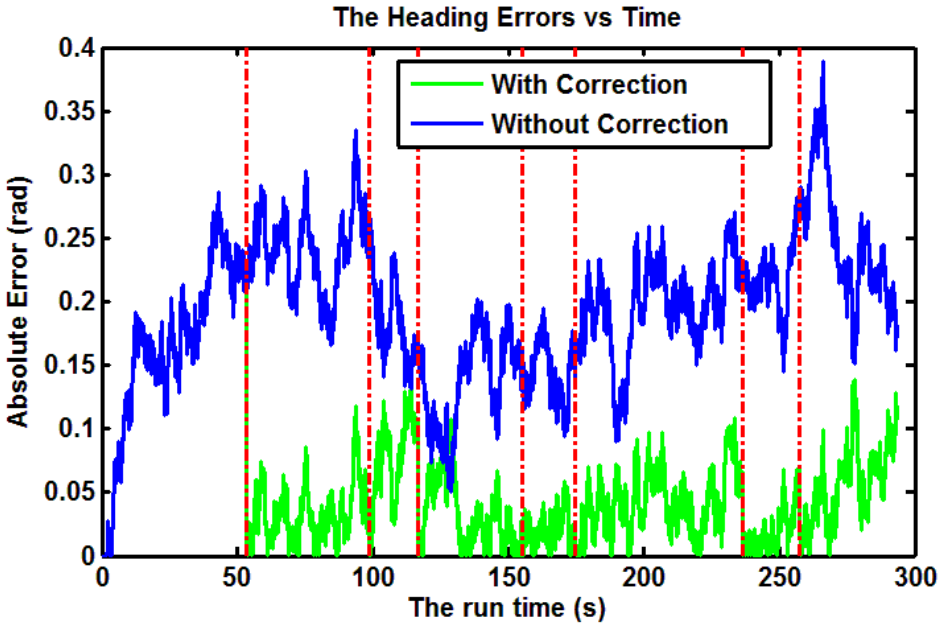


Figure 5.3. The absolute errors in the heading angle estimate by the robot.

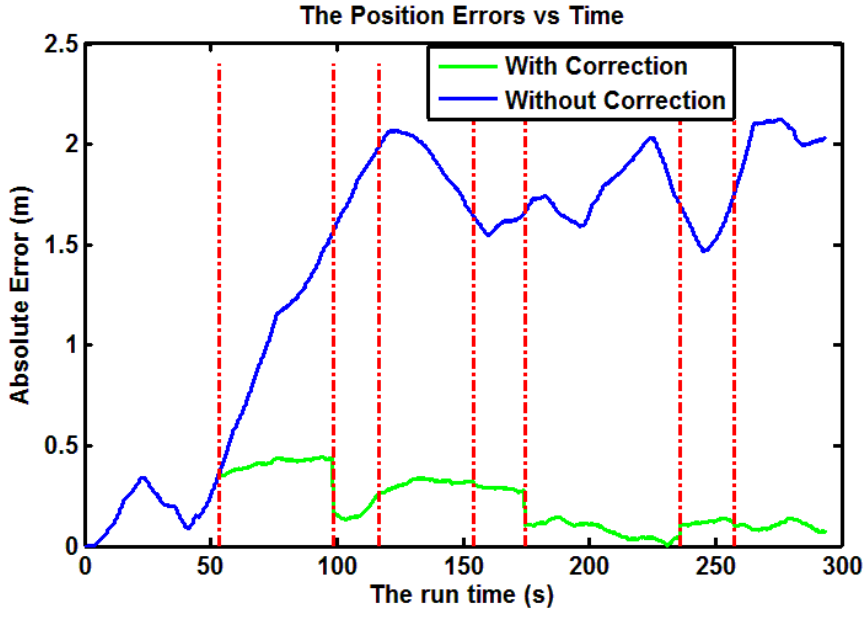


Figure 5.4. The absolute errors in the position estimate by the robot.

It is seen that the maximum absolute error for the heading angle and the position without the intermittent resetting during the 300-second run is about $0.38\ rad$ and $2.2\ meters$ while that with the resetting is about $0.28\ rad$ and $0.4\ meters$ respectively, see Table 5.1.

Table 5.1

The Maximum Absolute Errors and the Root Mean Square Errors for $300 - second$ *Runs with Corresponding* $EIR's$

Estimation	**Position Estimation**		**Heading Estimation**	
	Max. Error	RMS Error	Max. Error	RMS Error
Without Resetting	$2.20m$	$1.55m$	$0.38rad$	$1.60rad$
With Resetting	$0.40m$	$0.24m$	$0.28rad$	$0.54rad$
EIR	5.50	6.46	1.36	2.96

Using the resetting achieves an EIR of about 6 for the position and about 3 for the heading angle, when using the rms error, thus, the long term quality of the pose estimation improves and, therefore, the pose accuracy increases with the intermittent resetting.

5.2 Mapping

A binary occupancy grid of the prototyped world in Figure 5.1 has been built using the SICK LMS 200 as described in the methodology of Chapter 4. Note that the map is flipped vertically so that the living room is up instead of down. The red dots indicate the preset exploration path followed by the robot in building the map, the blue cells indicate obstacle regions and the white cells indicate free spaces.

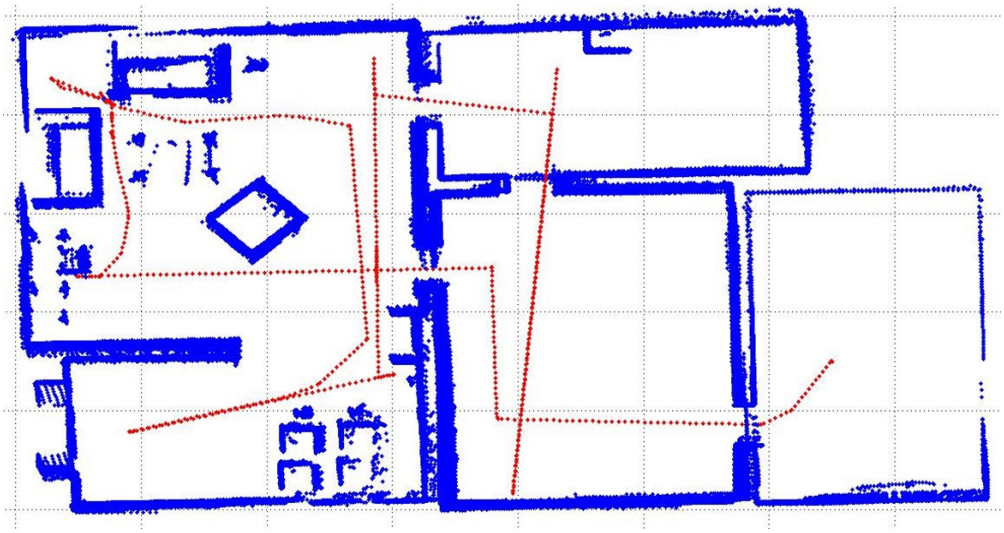


Figure 5.5. A binary occupancy grid map of the model environment prototype in Figure 5.1.

5.3 Path Planning

5.3.1 Path planning simulation experimental setup. In this section, simulation experiments have been used to highlight the properties of the ***Basic*** $\boldsymbol{A_r^*}$/$\boldsymbol{A_r^*}$ pathfinder and show its performance as compared to $\boldsymbol{A}^*$ on different world scenarios using both uniform and non-uniform gridding. The simulations were developed using MATLAB (*2011b, The MathWorks*) running on PC with the Windows 8 OS. The simulation world comprises a grid of size $256x256$ (which amounts to 65536 nodes). The performance parameters include: (a) ***Search Time***: the time it takes to plan a path; (b) ***Number of cells on OPEN list***: the total number of cells that ever made it to the OPEN list throughout the search; (c) ***Number of cells explored***: the number of cells that were actually explored before the goal was reached. In addition, example pathfinder applications to maze solving and indoor navigation are presented.

5.3.2 Effect of congestion/clutter on performance of A-Star, Basic A-r-Star and A-r-Star. In the experiment shown in this subsection, the simulation environment was populated with obstacle nodes having congestion/clutter probability varied from 0 to 0.75, and with $s_{Start} =$

[1,20] and $s_{Goal} = [256,256]$. Results are shown in Figure 5.7. It was observed that no path existed beyond congestion probability of 0.6. Since over half of the nodes are occupied, it makes sense that searching from one extreme corner of the world to another will not have an unblocked path. Secondly, as the clutter increases, the number of free nodes decreases and this explains the sudden reduction in the graphs of performance parameter values after congestion probability of 0.55.

The simulation results in Figure 5.7 demonstrate that ***Basic*** $\boldsymbol{A_r^*}$ and $\boldsymbol{A_r^*}$ converge to $\boldsymbol{A^*}$ beyond some degree of congestion, an assertion of Theorem 2. Figure 5.6 is an instance of the environment at clutter probability of 0.5 showing the path returned by the three algorithms. Figure 5.7 compares the performance for the three algorithms at different congestion probabilities in terms of their (a) search time; (b) size of OPEN list and (c) number of nodes explored.

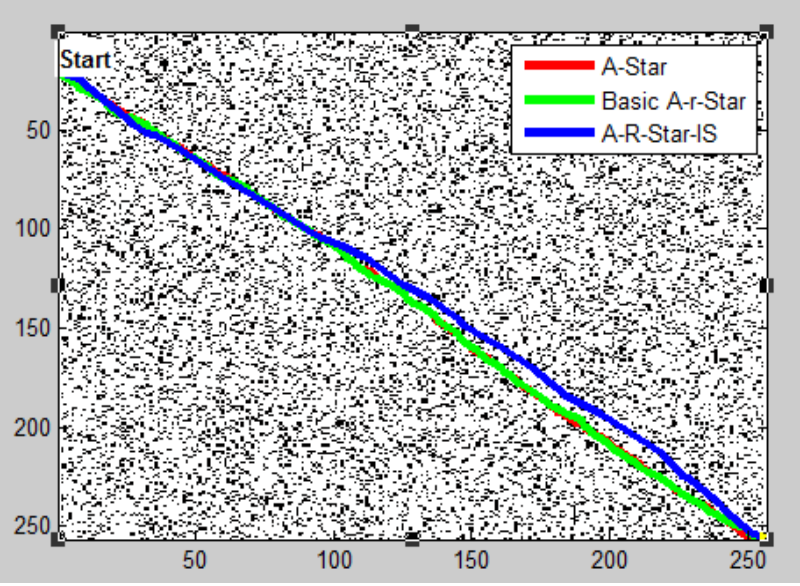


Figure 5.6. An instance of the environment at clutter/congestion probability of 0.5.

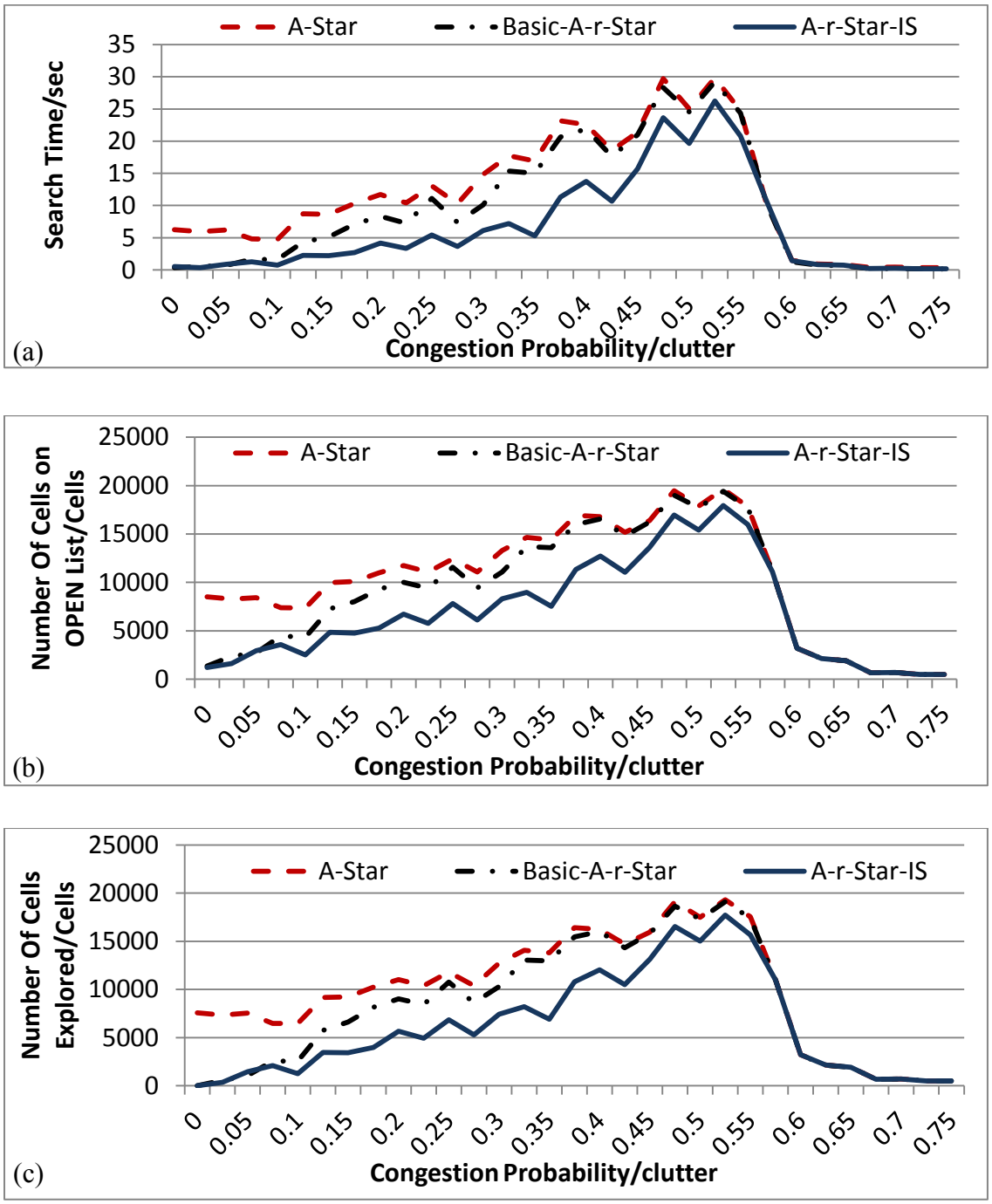


Figure 5.7. The effect of congestion/clutter on A-Star, Basic A-Star and A-r-Star operating on a uniform grid world.

5.3.3 Effect of changing obstacle configuration on performance of A-Star, Basic A-r-Star and A-r-Star (sliding obstacle). This simulation experiment shown in this subsection demonstrates that changing the obstacle configuration has little effect on $\boldsymbol{A_r^*}$ performance whereas it can drastically degrade the performance of $\boldsymbol{A}^*$ operating in a sparse world such as the one shown in Figure 5.8. The obstacle is assumed to be a long rigid wall in the environment separating the $s_{Start} = [1,20]$ and $s_{Goal} = [256,256]$. The horizontal position of this obstacle was varied from 11 to 231 and the performances of the pathfinders were recorded after each run. Figure 5.8 is an instance of the environment at obstacle position 131on the horizontal axis showing the path returned by the three algorithms. NB: these paths can all be post smoothened to the same path (overlap). Figure 5.9 compares the performance for the three algorithms at different obstacle positions in terms of their (a) search time; (b) size of OPEN list and (c) number of nodes explored.

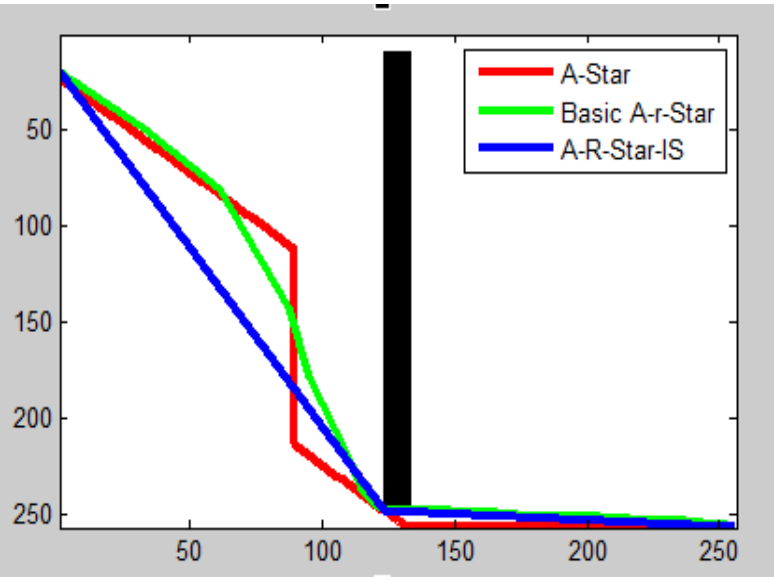


Figure 5.8. An instance of the environment at obstacle position 131on the horizontal axis.

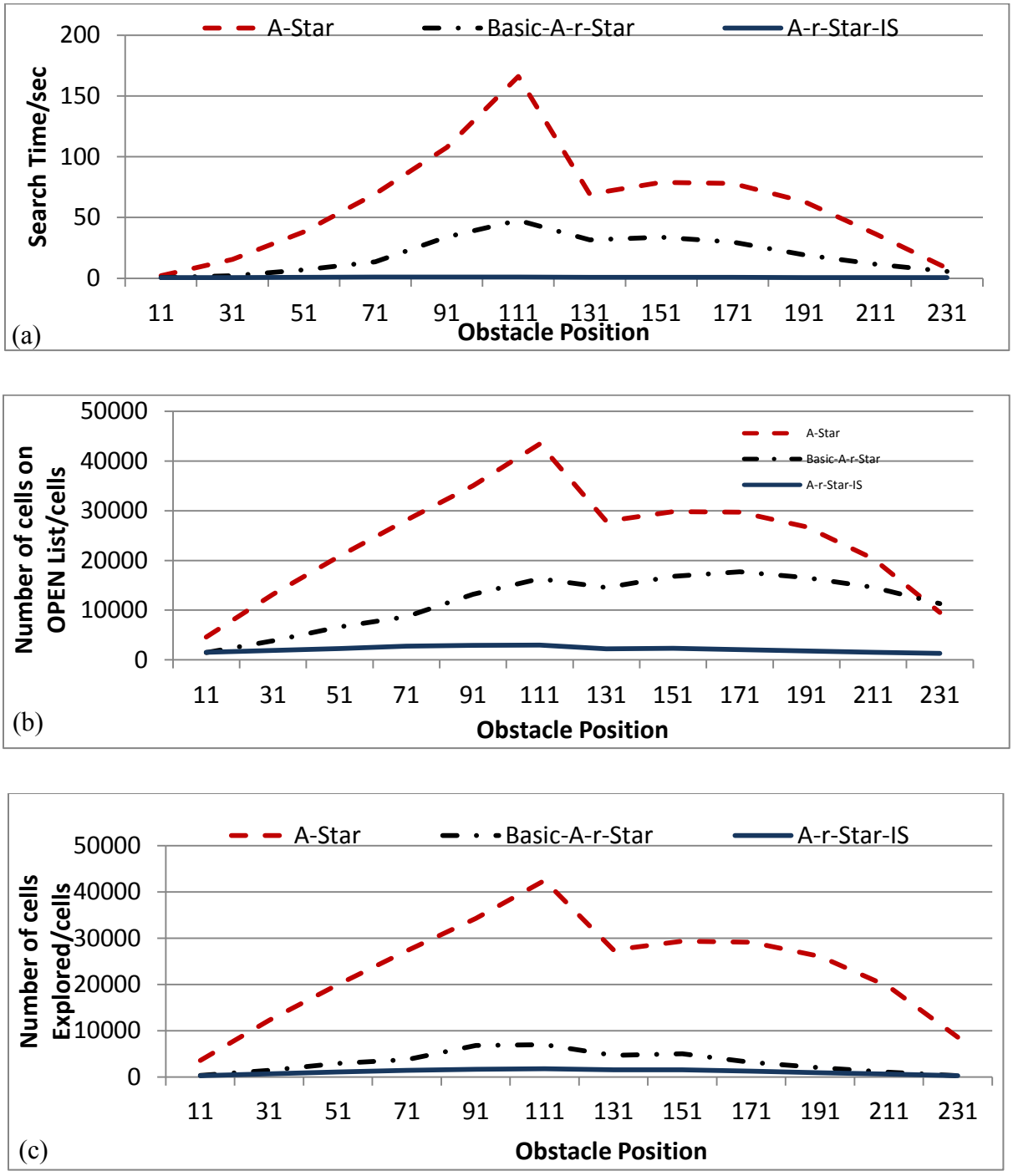


Figure 5.9. The effect of changing obstacle configuration on A-Star, Basic A-Star and A-r-Star operating on a uniform grid world.

5.3.4 Effect of changing start and goal node configuration on performance of A-Star, Basic A-r-Star and A-r-Star in the presence of a concave obstacle. The simulation experiment shown in this subsection indicates that, $\boldsymbol{A_r^*}$ can better handle a large concave obstacle such as the one shown in Figure 5.10 than $\boldsymbol{A}^*$. Here, the obstacle is assumed to be a large rigid concave wall in the environment separating s_{Start} and s_{Goal}. *Table 5.2* shows the nine different combinations of s_{Start} and s_{Goal} used to generate the performance results shown in Figure 5.11.

Table 5.2

The Nine Different Start and Goal Combinations for the Simulation in this Subsection

Simulation	Start		Goal	
	X	**Y**	**X**	**Y**
1	5	5	251	5
2	5	5	251	128
3	5	5	251	251
4	5	128	251	5
5	5	128	251	128
6	5	128	251	251
7	5	251	251	5
8	5	251	251	128
9	5	251	251	195

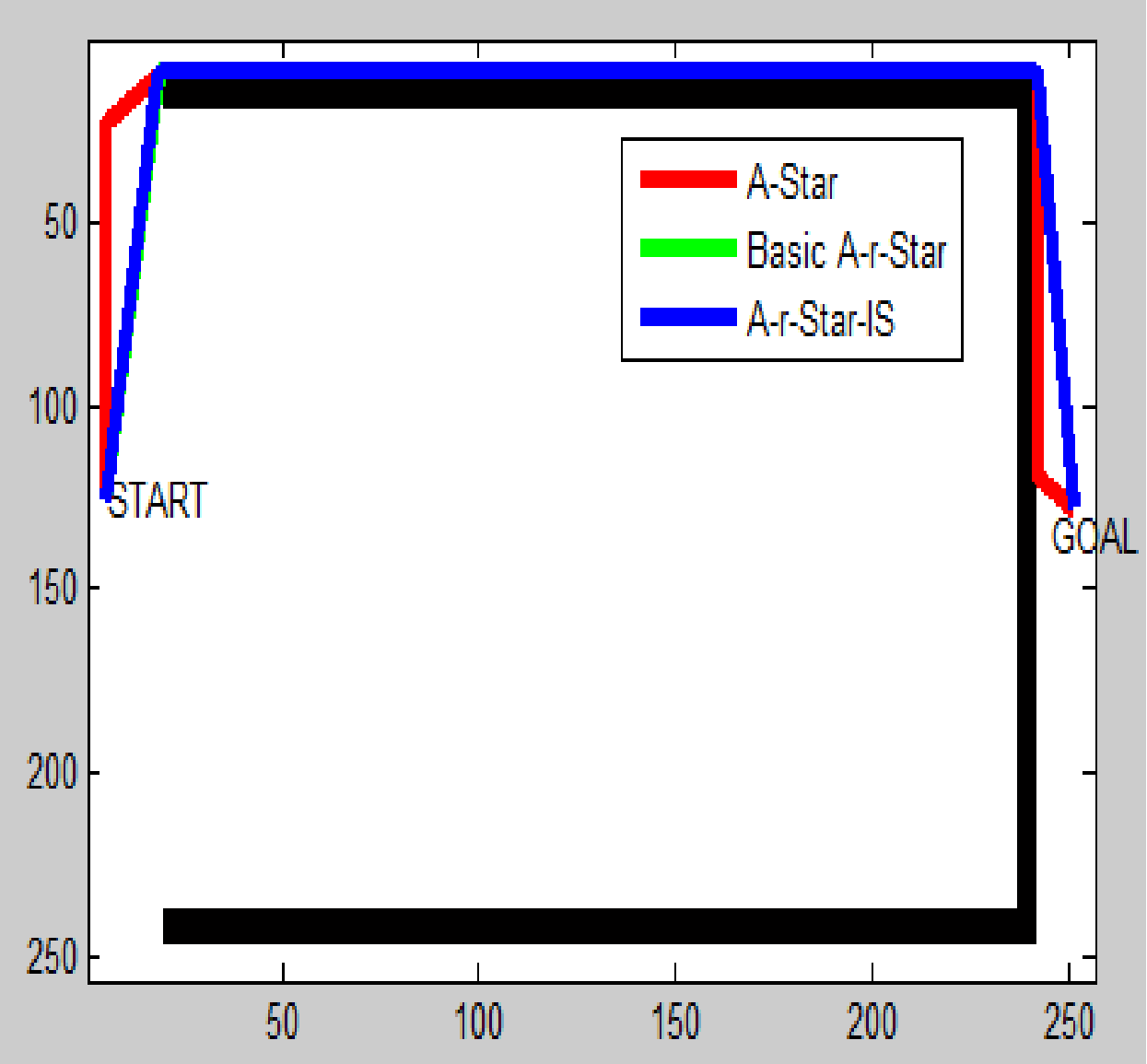


Figure 5.10. An instance of the environment with concave obstacle.

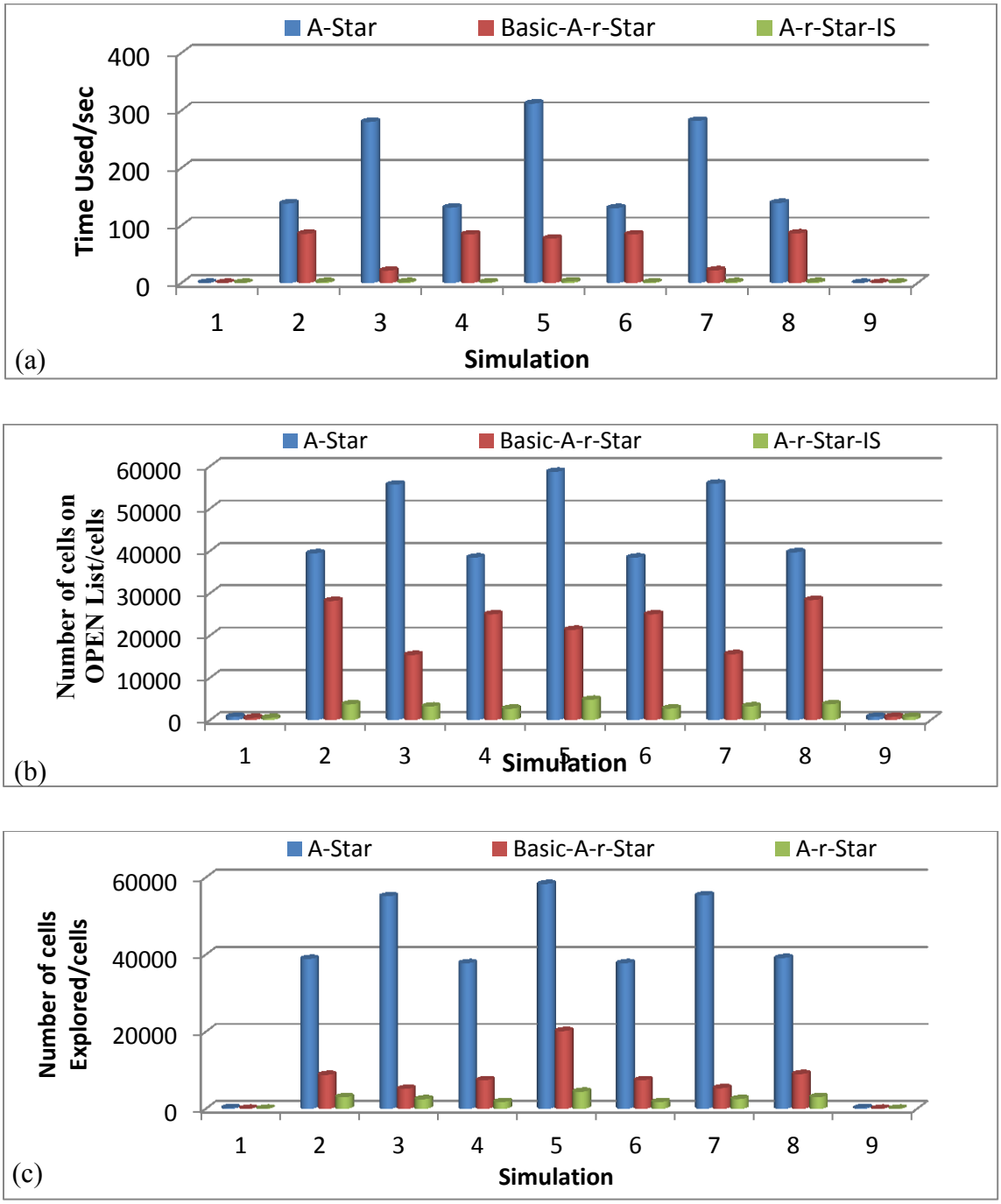


Figure 5.11. The effect of changing start and goal node position with respect to a large concave obstacle on A-Star, Basic A-Star and A-r-Star operating on a uniform grid.

5.3.5 Effect of increasing the resolution of the same environment on performance of A-Star, Basic A-r-Star and A-r-Star. This simulation shows that increasing the resolution of the same environment degrades the performance of $\boldsymbol{A}^*$ exponentially but that of $\boldsymbol{A}_r^*$ only degrades linearly. The obstacle is assumed to be a long rigid wall in the environment separating s_{Start} and s_{Goal}. The resolution of the grid was varied from $256x256$ to $10x10$. At each resolution, the performances of the pathfinders were recorded. This confirms the earlier assertion that $\boldsymbol{A}^*$ search time increases exponentially with increasing the grid size. Figure 5.12 is an instance of the environment at resolution of $123x123$ (i.e. at scale 0.5) showing the path returned by the three algorithms. NB: these paths can all be post smoothened to the same path (overlap). Figure 5.13 compares the performance for the three algorithms for different grid resolutions in terms of their (a) search time; (b) size of OPEN list and (c) number of nodes explored.

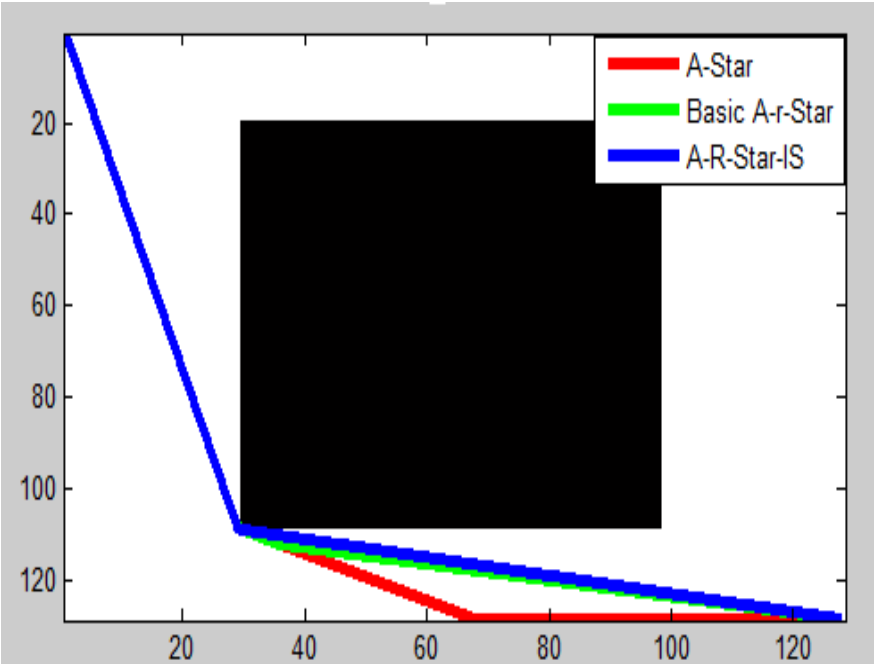


Figure 5.12. This is an instance of the environment at resolution of 123x123 (i.e. at scale 0.5).

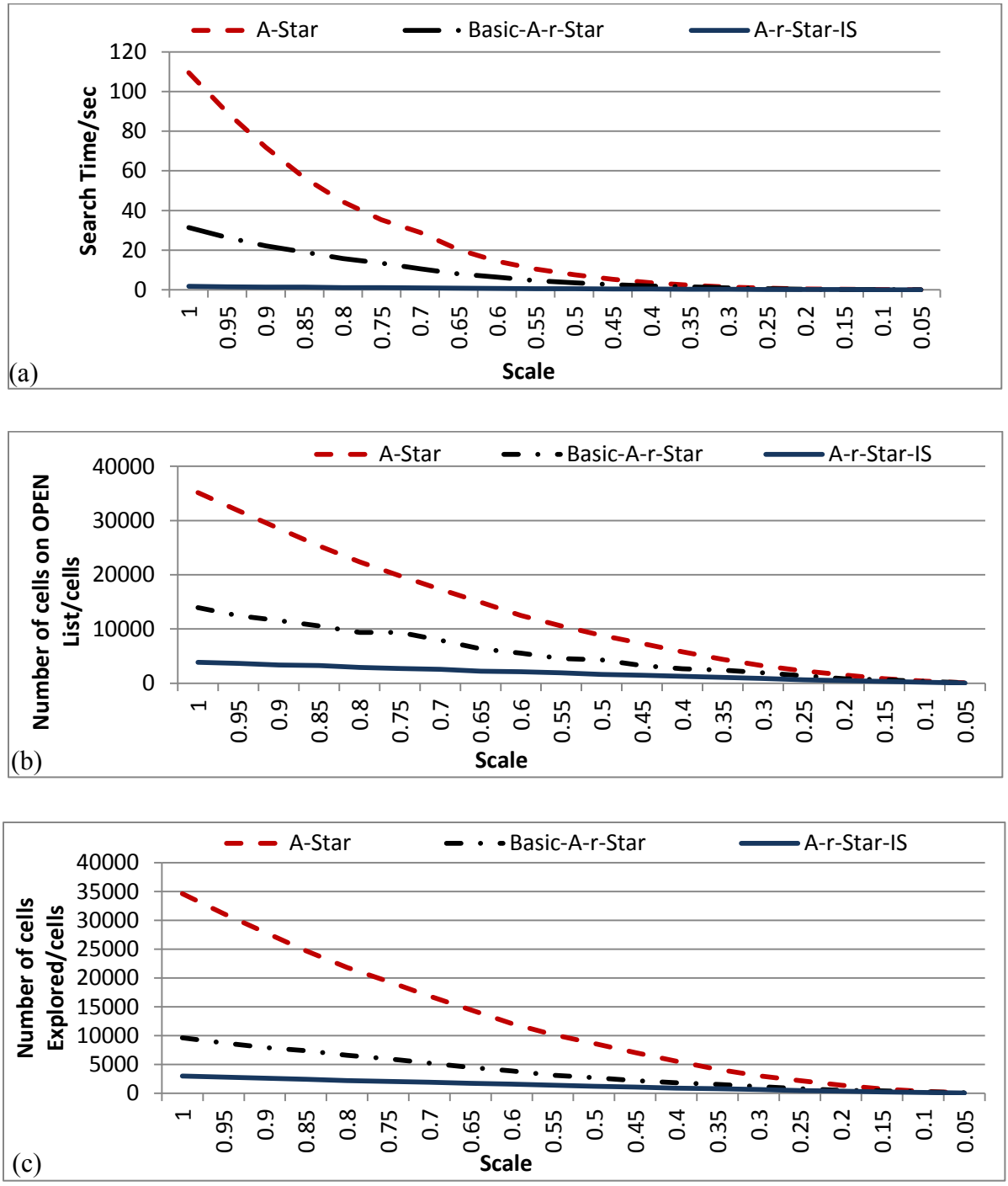


Figure 5.13. The effect of changing the gridding resolution of a given continuous world on A-Star, Basic A-Star and A-r-Star operating on a uniform grid world.

5.3.6 Comparing the performance of the A-r-Star with that of the A-Star running on quadtree. For this simulation experiment, an obstacle of size $(236\ x\ 20)\ nodes$ is placed between the start position $s_{Start} = [3, 3]$ and the goal position $s_{Goal} = [254, 254]$. An instance of the world after quadtree decomposition is shown in Figure 5.14 and the world after $\boldsymbol{A}_{\boldsymbol{r}}^{*}$ search in Figure 5.16. The comparison in Figure 5.15 (a) and Figure 5.15 (c) shows that at certain obstacle configurations $\boldsymbol{A}^{*}$ running on an environment preprocessed into a quadtree almost always outperforms $\boldsymbol{A}_{\boldsymbol{r}}^{*}$; however, it must be noted that the preprocessing takes a longer time in the quadtree case. Besides, as highlighted above in Section 2 and in (Kambhampati and Davis, 1986), the performance of the quadtree approach degrades drastically with increasing congestion.

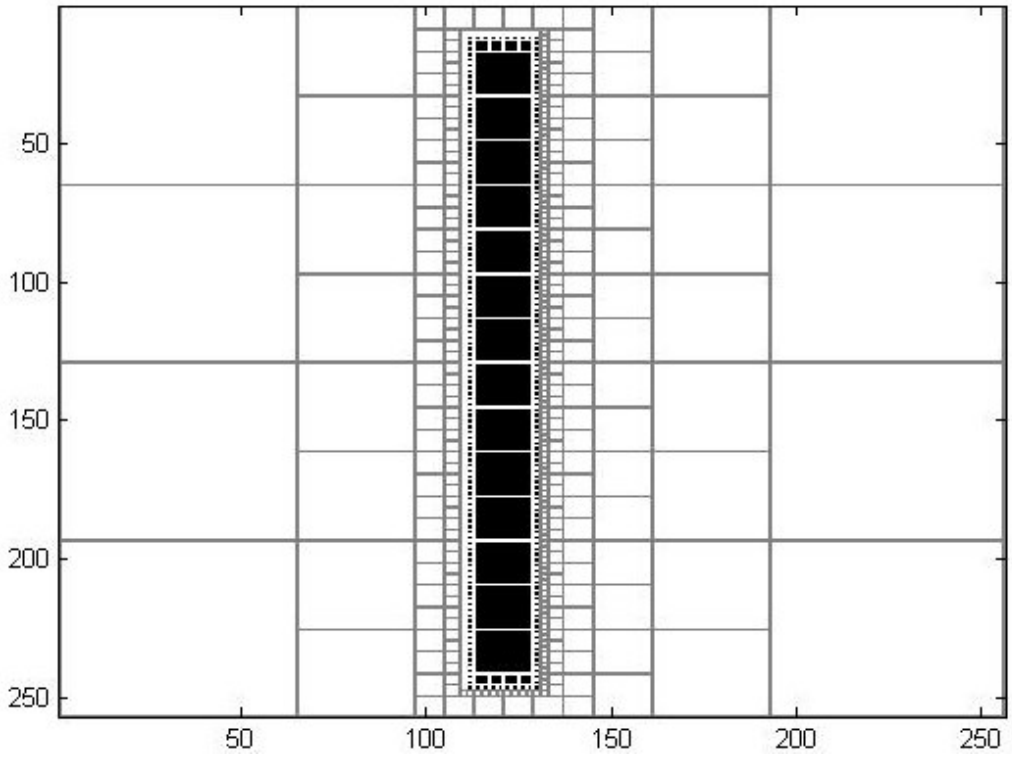
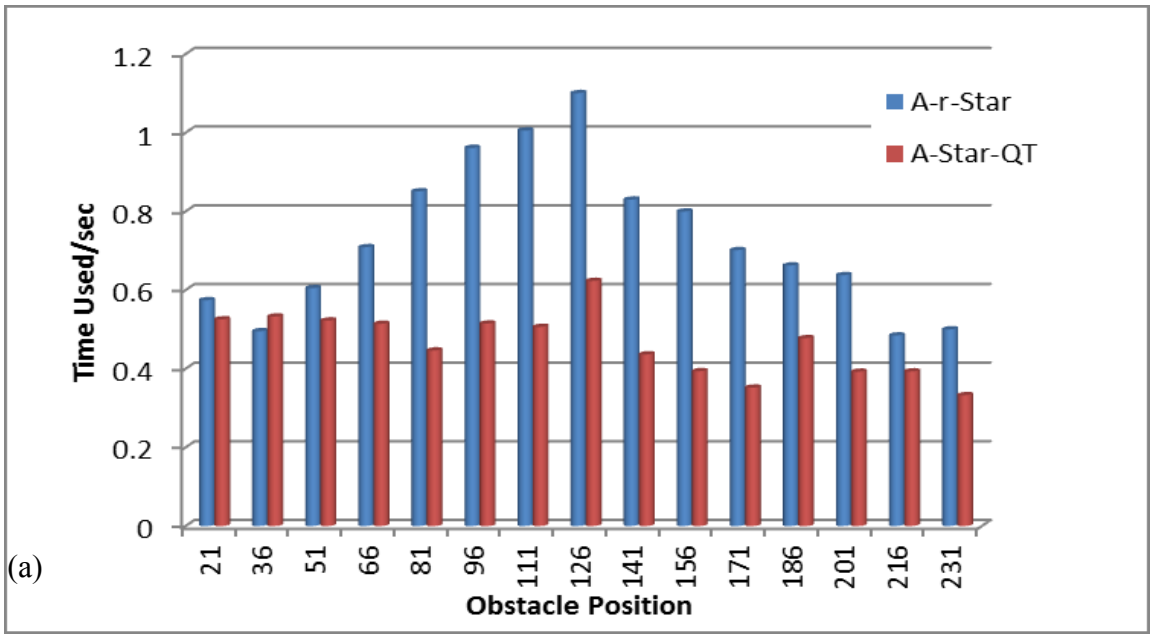


Figure 5.14. The world after quadtree decomposition (preprocessing). White represents free nodes, gray represents node borders and black represents obstacle.



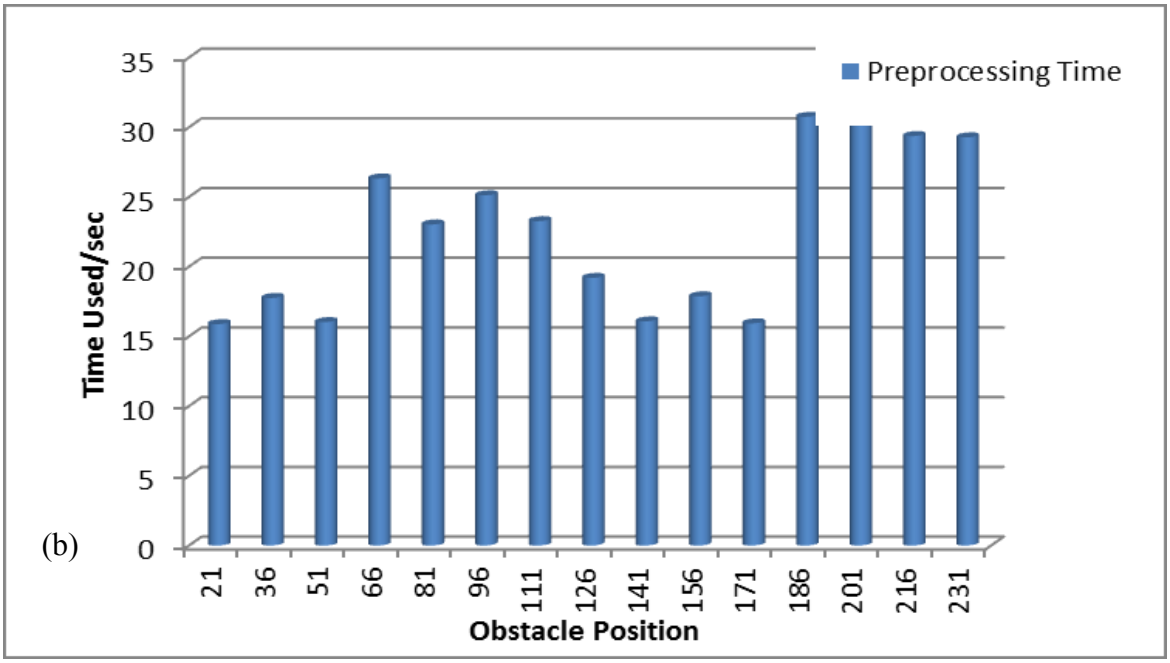


Figure 5.15. The performance comparison for A-Star operating on a quadtree and A-r-Star operating on a uniform grid of the same continuous environment.

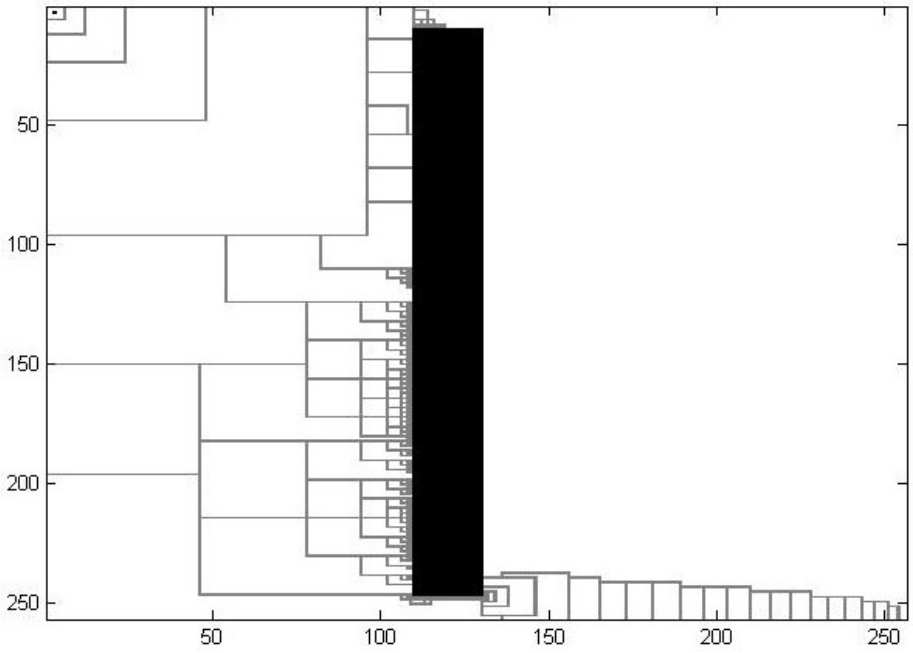


Figure 5.16. The multi-resolution grid built by A-r-Star during the search. White represents free nodes, gray represents node borders and black represents obstacle.

5.3.7 Solving a maze problem with the A-Star, Basic A-r-Star and A-r-Star algorithms. Artificial intelligence search algorithms are often used to solve maze problems that are common in tortuous games such as the *Pacman Maze Game*. Such maze problems are similar, and equivalent in some cases, to path finding problems faced by robots operating in maze-like environments such as building floor plans and underground mines, for example. The first application is to use the three pathfinder algorithms to solve a simple 256 x 256 maze problem. Figure 5.17 shows the maze and respective paths found between the indicated start and goal nodes. The performances of the algorithms are summarized in Table 5.3. Here, the path length is measured using the Euclidean distance of all path segments. Note that the paths returned by $\boldsymbol{Basic\ A_r^*}$ and $\boldsymbol{A_r^*}$ have *bulges* that make the path suboptimal. These bulges can be eliminated using the path smoothing techniques in Algorithm 1., as shown in Figure 4.2.

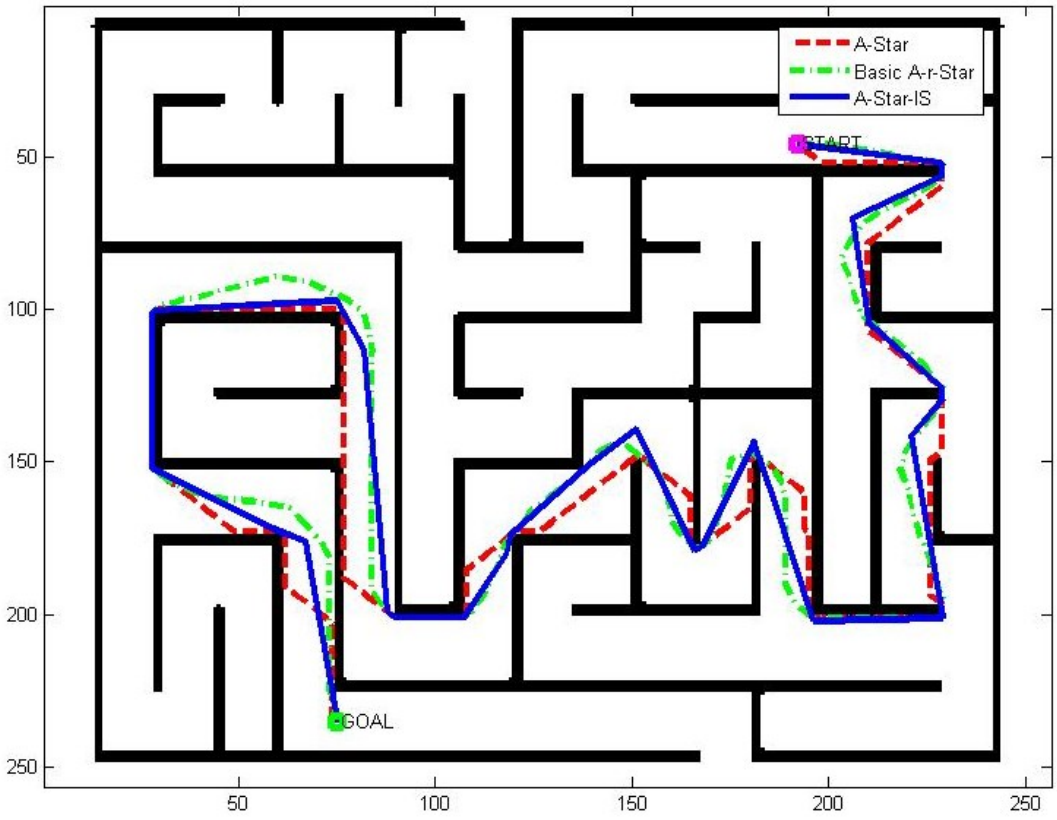


Figure 5.17. How A-Star, Basic A-Star and A-r-Star operating on a uniform grid world solves a maze problem.

Table 5.3

The Performance Comparison of the Three Algorithms for the Maze Problem Solving

Algorithm	Time Used (sec)	Path Length (units)	Number of cells on Open List (cells)	Number of cells Explored (cells)
A^*	122.45	779.39	37378	37142
$Basic\ A_r^*$	102.62	795.77	33011	31581
$A_r^* - IS$	17.47	789.56	14980	14675

5.3.8 Integration of the methodologies in a simulated home using A-r-Star Pathfinder.

The next simulation involves running the integrated system in a simulated 3D home environment which was developed using Webots as shown in Figure 5.1. Webots (Leonard et al., 1991) is commercial software for robotic systems prototyping and simulation. A prototype of the Pioneer 2DX robot was run in this environment to build a binary occupancy grid map using a simulated SICK Laser Measurement Sensor (LMS) 200. Note that, the simulation prototypes for the robot and sensor come with Webots. A 2D map representing the floor plan of the home environment is then fed to $\boldsymbol{A}_r^*$ to plan a path from a point in the fitness room to a destination in the living room.

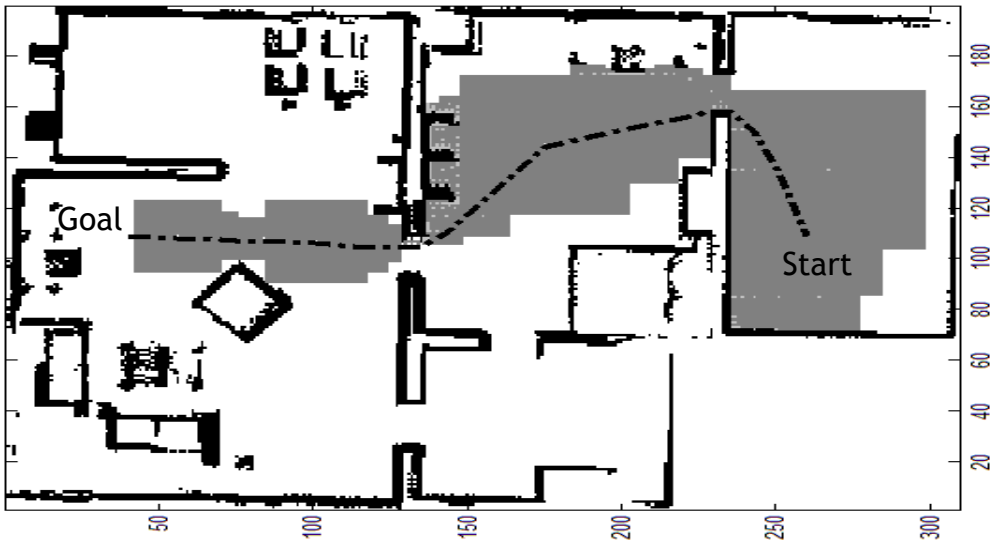


Figure 5.18. Path planning in a prototype home environment (see Figure 5.1) using the A-r-Star pathfinder.

The result is shown in Figure 5.18 where the obstacle regions are represente by black nodes; the obstacle-free zone are represented by white nodes and the gray nodes represent those cells that the $\boldsymbol{A}_r^*$ skipped while searching for the path. Furthermore, the dark center line indicates

the path that was returned by the $\boldsymbol{A_r^*}$ search planning from a point in the fitness room to a point in the living room.

5.3.9 Results for the incremental A-r-Star search compared to D*-lite. Figure 5.20 illustrates the re-plan time comparison between the incremental $\boldsymbol{A_r^*}$ and $\boldsymbol{D^* - Lite}$. The cells that were blocked have been indicated on the horizontal axis. Also note that the node blocking was accumulative, meaning when a node is blocked, it remains blocked in the next iteration. The Incremental $\boldsymbol{A_r^*}$ Algorithm outperforms $\boldsymbol{D^* - Lite}$ because this is a sparse world and $\boldsymbol{D^* - Lite}$'s initial search is similar to $\boldsymbol{A_r^*}$. Also take note that, in instances where the change does not demand searching of many new cells, $\boldsymbol{D^* - Lite}$ does perform very well and even in some instances better than $\boldsymbol{A_r^*}$.

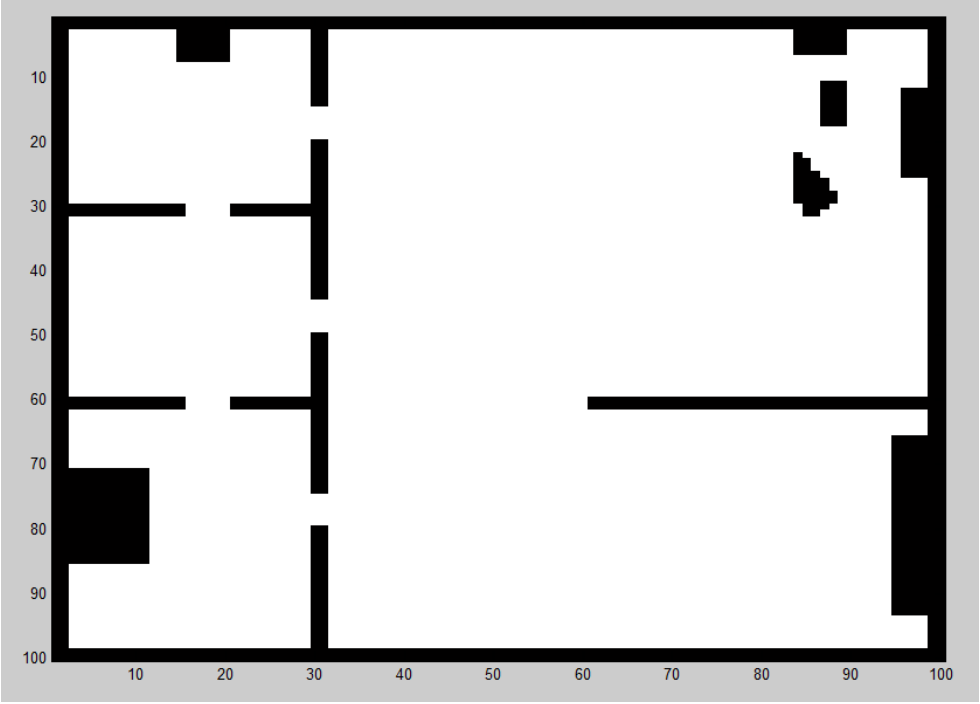


Figure 5.19. The grid map used for the simulation.

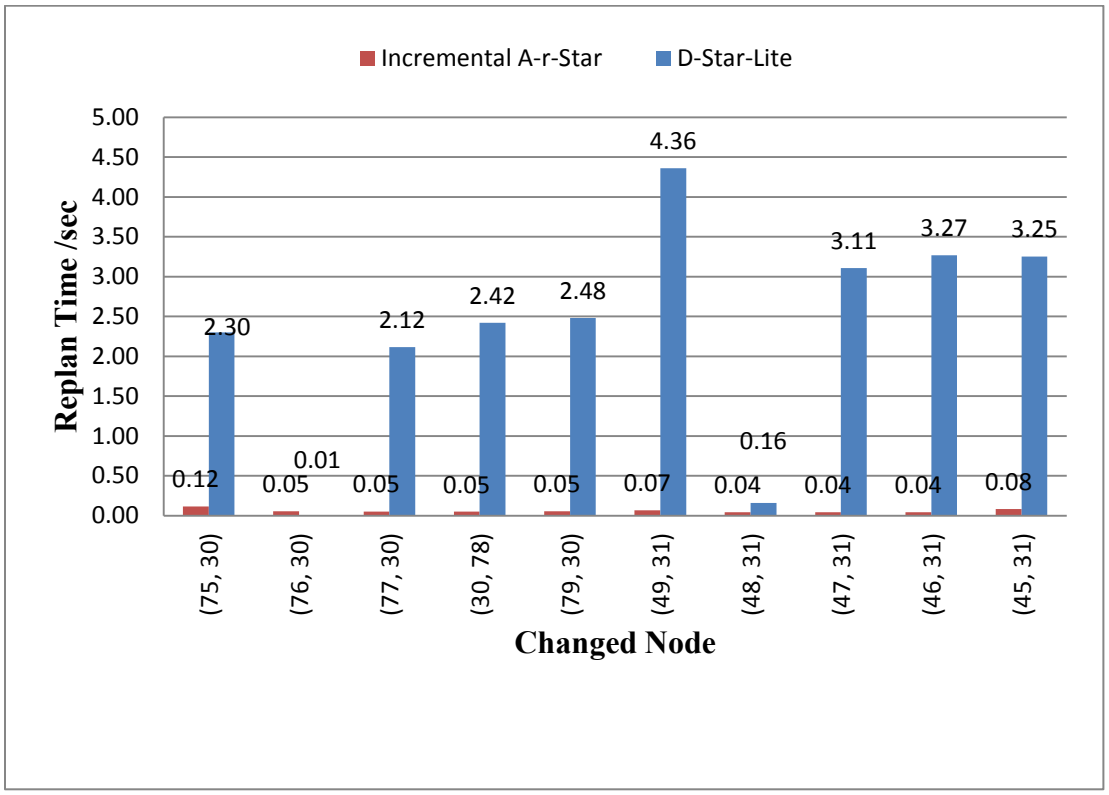


Figure 5.20. The comparison between the re-planning time of Incremental-A-r-Star and D*-Lite.

5.3.10 Results of the multiple destination planning. Figure 5.21 shows a time comparison for the multiple-goal search on the maze shown in Figure 5.17. Different destinations were searched in turn as shown in the horizontal axis from left to right. The environment is kept constant throughout the search. Also, s_{Start} was kept constant at $(280,250)$. The first scenario plans from scratch anytime it is queried with a new goal but the second one reuses the information from the DAG from the previous searches. It can be seen that reusing the information from previous search amounts to a substantial increase in speed depending on how close the new goal is to a leaf in the previous graph.

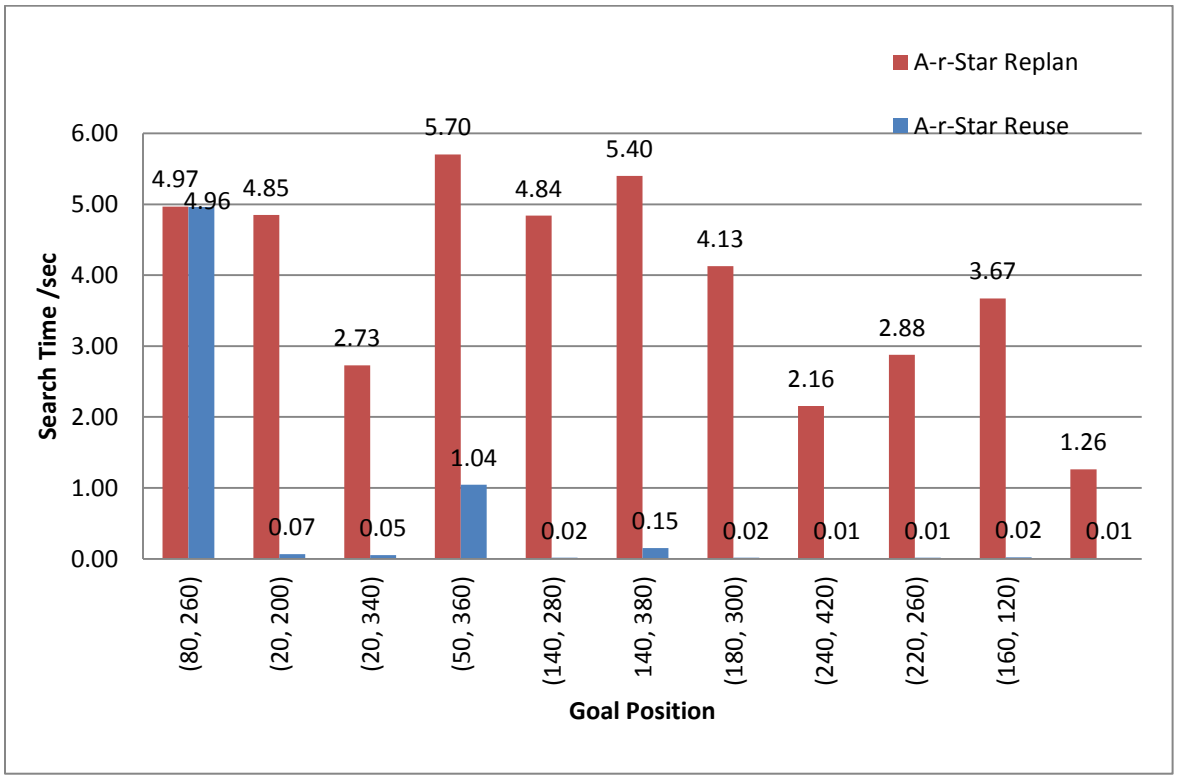


Figure 5.21. The run time comparison for the A-r-Star algorithm searching multiple number of times in a static environment when it reuses the previous information and when it plans from scratch.

CHAPTER 6

Conclusion and Possible Research Extensions

6.1 Research Overview

This research has addressed the indoor navigation problem using a novel approach. This approach divides the problem of indoor navigation into three distinctive parts, namely localization, mapping and path planning. These parts can be solved independently and yet are interrelated. In other words, the performance of the localization technique directly or indirectly affects efficiency of the mapping, and the accuracy of the mapping influences the effectiveness of the path planning. These three components have therefore been solved individually and then integrated to form a single navigation system. To solve the localization problem, dead-reckoning (odometry) was adopted. The greatest problem with dead reckoning – the accumulation of errors from both systematic and non-systematic error sources– has been effectively handled both theoretically and experimentally. A least squares numerical approach to odometry error calibration was utilized to reduce the effect of the systematic odometry errors on the navigation system. An intermittent resetting technique that employs RFID tags placed at known fixed locations in the environment in conjunction with door-markers has been developed and implemented to mitigate the errors remaining after the calibration (mainly non-systematic errors).

This research has developed and implemented a technique for building a binary occupancy grid map of the environment using a laser range finder, SICK LMS 200, as the main exteroceptive sensor. Path planning using various graph search techniques such as $\boldsymbol{A}^*$, $\boldsymbol{D}^*$, $\boldsymbol{D}^* - \boldsymbol{Lite}$, $\boldsymbol{LPA}^*$ and Adaptive $\boldsymbol{A}^*$ have been investigated and implemented. $\boldsymbol{A}_{\boldsymbol{r}}^*$ Pathfinder, a new path

planning algorithm that is capable of high performance both in cluttered and sparse environment has been developed and implemented. Its properties, challenges and solutions to the challenges have also been highlighted in the research. Simulation experiments highlighting properties and performance of the individual components have been developed and executed using MATLAB. A prototype world (five compartments home) has been built using the Webots robotic prototyping and simulation software, incorporating use of the Webots models for the Pioneer 2 robot and LMS. These respectively served as a representation of an indoor domestic operating environment and an assistive robot model. An integrated version of the developed navigation system comprising the localization, mapping and path planning techniques has been executed on the simulated assistive robot system in this prototype workspace.

6.2 Theoretical and Experimental claims

6.2.1 Intermittent resetting technique. The intermittent resetting technique for odometry error mitigation is novel. There are systems developed in literature which are similar but not exactly the same in principle. Its application to the localization of the robot results in improvement in the position and orientation estimation over the long run. Also, since the system doesn't rely on the RSSI of the RFID tags, as many applications do, the issue with signal strength degradation has less influence on the system. This system is effective as long as the coded ID can be read. The door-marker system can be implemented with inexpensive proximity sensors and timers. Thus, the system is less expensive to setup because it requires low cost off-the-shelf sensors.

6.2.2 A-r-Star pathfinder. The development of the $\boldsymbol{A_r^*}$ algorithm is a major contribution of this research to the field of robotic path planning. Both formal and informal proofs have

asserted its superiority over existing techniques in terms of its simplicity, flexibility and search speed. The linear scalability with increasing grid resolution makes its application to large sparse grid worlds more attractive than most existing algorithms. Furthermore, the performance degradation in cluttered environment is less and so it possesses the ability to plan across the congestion spectrum from much cluttered environments to very sparse environments. To the best of the author's knowledge, this research is the first to study the DAG built by the graph search algorithms and the subsequent understanding of their extension to incremental algorithms. Compared with other multi-resolution gridding path planning techniques such as the quadtree approach, this approach is desirable for various reasons some of which are highlighted below:

- $\boldsymbol{A_r^*}$ combines the multi-resolution gridding with the search and so requires no pre-processing time
- $\boldsymbol{A_r^*}$ builds the grid for only the part of map which is of interest without spending time building the entire map
- Unlike other multi-resolution gridding path planning techniques, the effect of pepper noise on the $\boldsymbol{A_r^*}$ algorithm is negligible
- $\boldsymbol{A_r^*}$ is simple algorithmically and implementation-wise and yet has powerful application advantages
- The exploitation of information from previous searches enhances subsequent searches in the same environment for a given root node. This holds potential for solving the moving target problem.

6.3 Industrial Application

The three different components of this research have separate as well as combined industrial applications. While a focal application for this research is assistive robotics, indoor navigation also has applications to healthcare institution robots; robotic vacuum cleaners; robotic nursing; museum tour guide robots; warehouse and factory robotics; etc. The $\boldsymbol{A}_{\boldsymbol{r}}^{*}$ pathfinder in particular also has applications in path planning in underground bunkers, complex buildings such as shopping malls, video games; and more.

6.4 Possible Research Extensions

The intermittent resetting technique developed in this research can be extended to include more general scenarios, where the robot is assumed to perform motions such as turning and acceleration or deceleration from the time it crosses the first door-marker to the second door-marker. Besides, the resetting of just one coordinate dimension at each door can be generalized to include the resetting of all the state variables involved at each door. Also, one can look into the inclusion of probabilistic modeling of the non-systematic errors to reduce the error accumulation between intermittent resetting stations.

The mapping technique assumes perfect LMS readings since the system has high accuracy for short distance measurements. However, incorporation of error modeling of the observation to cater for uncertainty in the sensor measurement can be a great extension of this research. Also, various feature extraction techniques such as Split and Merge, Random Sample Consensus (RANSAC), Hough Transform; etc. can be used in conjunction with data association techniques such as Individual Compatibility or Joint Compatibility Branch and Bound to enhance

the accuracy of the mapping. Ultimately, SLAM methodologies can be employed to automate the localization and mapping process.

Possible improvements to the $\boldsymbol{A}_{\boldsymbol{r}}^{*}$ algorithm include research into better path smooth algorithms. If the optimality of $\boldsymbol{A}_{\boldsymbol{r}}^{*}$ algorithm can be guaranteed, that will be a ground breaking accomplishment in the field of artificial intelligence and video gaming. Also, the incremental $\boldsymbol{A}_{\boldsymbol{r}}^{*}$ algorithm can be generalized to include the scenarios where the cost can decrease. Finally, there can be an extension of this system to operate on higher dimensional configuration space.

References

Abbas, T., Arif, M., and Ahmed, W. (2006, 18-21 Oct. 2006). *Measurement and Correction of Systematic Odometry Errors Caused by Kinematics Imperfections in Mobile Robots.* Paper presented at the SICE-ICASE, 2006. International Joint Conference.

Abuhadrous, I., Nashashibi, F., and Laurgeau, C. (2003). 3-D land vehicle localization: a real-time multi-sensor data fusion approach using (RT)MAPS. *Proceedings of the 11th International Conference on Advanced Robotics 2003, Vol 1-3*, 71-76.

Addesso, P., Bruno, L., and Restaino, R. (2010, 5-7 May 2010). *Adaptive localization techniques in WiFi environments.* Paper presented at the 5th IEEE International Symposium on Wireless Pervasive Computing (ISWPC), 2010

Aizawa, K., and Tanaka, S. (2009). A Constant-Time Algorithm for Finding Neighbors in Quadtrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(7), 1178-1183.

Antonelli, G., Chiaverini, S., and Fusco, G. (2005). A calibration method for odometry of mobile robots based on the least-squares technique: theory and experimental validation. *IEEE Transactions on Robotics, 21*(5), 994-1004.

Bahl, P., and Padmanabhan, V. N. (2000). *RADAR: an in-building RF-based user location and tracking system.* Paper presented at the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE INFOCOM 2000.

Begum, M., Mann, G. K. I., and Gosine, R. G. (2008). Integrated fuzzy logic and genetic algorithmic approach for simultaneous localization and mapping of mobile robots. *Appl. Soft Comput., 8*(1), 150-165.

Boontrai, D., Jingwangsa, T., and Cherntanomwong, P. (2009). *Indoor localization technique using passive RFID tags*. Paper presented at the Proceedings of the 9th international conference on Communications and information technologies, Incheon, Korea.

Borenstein, J. (1998). Experimental results from internal odometry error correction with the OmniMate mobile robot. *IEEE Transactions on Robotics and Automation, 14*(6), 963-969.

Borenstein, J., and Evans, J. (1997, 20-25 Apr 1997). *The OmniMate mobile robot-design, implementation, and experimental results.* Paper presented at the Proceedings., 1997 IEEE International Conference on Robotics and Automation, 1997. .

Borenstein, J., Everett, H. R., and Feng, L. (1996). *Navigating mobile robots : systems and techniques*. Wellesley, Mass.: A K Peters.

Borenstein, J., and Feng, L. (1996, 22-28 Apr 1996). *Gyrodometry: a new method for combining data from gyros and odometry in mobile robots.* Paper presented at the Proceedings., 1996 IEEE International Conference on Robotics and Automation, 1996. .

Borenstein, J., and Liqiang, F. (1996). Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation, 12*(6), 869-880.

Bostani, A., Vakili, A., and Denidni, T. A. (2008, 4-7 May 2008). *A novel method to measure and correct the odometry errors in mobile robots.* Paper presented at the CCECE 2008. Canadian Conference on Electrical and Computer Engineering, 2008. .

Botea, A., Müller, M., and Schaeffer, J. (2004). Near optimal hierarchical path-finding. *Journal of Game Development, 1*, 7--28.

Bouet, M., and dos Santos, A. L. (2008, 24-27 Nov. 2008). *RFID tags: Positioning principles and localization techniques.* Paper presented at the WD '08. 1st IFIP Wireless Days, 2008. .

Carsten, J., Ferguson, D., and Stentz, A. (2006, 9-15 Oct. 2006). *3D Field D: Improved Path Planning and Replanning in Three Dimensions.* Paper presented at the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems.

Casals, A. (1989). *Sensor devices and systems for robotics*: Springer-Verlag New York, Inc.

Castro, P., Chiu, P., Kremenek, T., and Muntz, R. R. (2001). *A Probabilistic Room Location Service for Wireless Networked Environments*. Paper presented at the Proceedings of the 3rd international conference on Ubiquitous Computing, Atlanta, Georgia, USA.

Castro, P., and Munz, R. (2000). Managing context data for smart spaces. *Personal Communications, IEEE, 7*(5), 44-46.

Chen, C., Tay, C., Laugier, C., and Mekhnacha, K. (2006, 5-8 Dec. 2006). *Dynamic Environment Modeling with Gridmap: A Multiple-Object Tracking Application.* Paper presented at the 9th International Conference on Control, Automation, Robotics and Vision, 2006. ICARCV '06. .

Choset, H., Burgard, W., Hutchinson, S., Kantor, G., Kavraki, L. E., Lynch, K., and Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementation*: MIT Press.

Corke, P. (2011). *Robotics, Vision and Control : Fundamental algorithms in MATLAB* (Vol. 73). Germany: Springer.

Corke, P., Strelow, D., and Singh, S. (2004, 28 Sept.-2 Oct. 2004). *Omnidirectional visual odometry for a planetary rover.* Paper presented at the Proceedings. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004). .

Coué, C., Pradalier, C., Laugier, C., Fraichard, T., and Bessiere, P. (2006). Bayesian Occupancy Filtering for Multitarget Tracking: an Automotive Application. *International Journal of Robotics Research, 25*(1), 19--30.

Cowlagi, R. V., and Tsiotras, P. (2010, 15-17 Dec. 2010). *Multi-resolution path planning: Theoretical analysis, efficient implementation, and extensions to dynamic environments.* Paper presented at the 49th IEEE Conference on Decision and Control (CDC), 2010

Cox, I. J. (1991). Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation, 7*(2), 193-204.

Daniel, K., Nash, A., Koenig, S., and Felner, A. (2010). Theta*: Any-Angle Path Planning on Grids. *Journal of Artificial Intelligence Research, 39*, 533-579.

de la Puente, P., Rodriguez-Losada, D., Valero, A., and Matia, F. (2009, 10-15 Oct. 2009). *3D feature based mapping towards mobile robots' enhanced performance in rescue missions.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009.

Dechter, R., and Pearl, J. (1985). Generalized best-first search strategies and the optimality of A*. *J. ACM, 32*(3), 505-536.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematic, 1*(1), 269-271.

Division, P. (2009). *World Population Ageing 2009* (ESA/P/WP/212). Department of Economic and Social Affairs, 19 December 2009.

Dovis, F., Lesca, R., Margaria, D., Boiero, G., and Ghinamo, G. (2008, 5-8 May 2008). *An assisted high-sensitivity acquisition technique for GPS indoor positioning.* Paper presented at the IEEE/ION Position, Location and Navigation Symposium, 2008

Dudek, G., and Jenkin, M. (2000). *Computational principles of mobile robotics*: Cambridge University Press.

Eaton, E., and Ruvolo, P. L. (2013). *An Efficient Lifelong Learning Algorithm*.

Ferguson, D., Kalra, N., and Stentz, A. (2006, 15-19 May 2006). *Replanning with RRTs.* Paper presented at the Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006. .

Ferguson, D., and Stentz, A. (2007). Field D*: An Interpolation-Based Path Planner and Replanner. In S. Thrun, R. Brooks & H. Durrant-Whyte (Eds.), *Robotics Research* (Vol. 28, pp. 239-253): Springer Berlin Heidelberg.

Fulgenzi, C., Spalanzani, A., and Laugier, C. (2007, 10-14 April 2007). *Dynamic Obstacle Avoidance in uncertain environment combining PVOs and Occupancy Grid.* Paper presented at the IEEE International Conference on Robotics and Automation, 2007.

Grimson, W. E., and Lozano-Perez, T. (1987). Localizing overlapping parts by searching the interpretation tree. *IEEE Trans Pattern Anal Mach Intell, 9*(4), 469-482.

Grisetti, G., Tipaldi, G. D., Stachniss, C., Burgard, W., and Nardi, D. (2007). Fast and accurate SLAM with Rao-Blackwellized particle filters. *Robotics and Autonomous Systems, 55*(1), 30-38.

Guldner, J., Utkin, V. I., Hashimoto, H., and Harashima, F. (1995, 21-23 Jun 1995). *Tracking gradients of artificial potential fields with non-holonomic mobile robots.* Paper presented at the Proceedings of the 1995 American Control Conference.

Hao, L., and Nashashibi, F. (2012, 5-7 Dec. 2012). *A new method for occupancy grid maps merging: Application to multi-vehicle cooperative local mapping and moving object detection in outdoor environment.* Paper presented at the 12th International Conference on Control Automation Robotics & Vision (ICARCV), 2012

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics, 4*(2), 100-107.

Hern, C., Sun, X., Koenig, S., and Meseguer, P. (2011). *Tree Adaptive A**. Paper presented at the The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1, Taipei, Taiwan.

Huang, C. T., and Mitchell, O. R. (1994). A Euclidean distance transform using grayscale morphology decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 16*(4), 443-448.

Hui, T., and Zekavat, S. A. (2007). A Novel Wireless Local Positioning System via a Merger of DS-CDMA and Beamforming: Probability-of-Detection Performance Analysis Under Array Perturbations. *IEEE Transactions on Vehicular Technology, 56*(3), 1307-1320.

Johnson, A., Montgomery, J., and Matthies, L. (2005, 18-22 April 2005). *Vision Guided Landing of an Autonomous Helicopter in Hazardous Terrain.* Paper presented at the Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. .

Junjun, X., Haiyong, L., Fang, Z., Rui, T., and Yiming, L. (2011, 26-28 Oct. 2011). *Dynamic indoor localization techniques based on Rssi in WLAN environment.* Paper presented at the 6th International Conference on Pervasive Computing and Applications (ICPCA), 2011.

Kambhampati, S., and Davis, L. S. (1986). Multiresolution Path Planning for Mobile Robots. *Ieee Journal of Robotics and Automation, 2*(3), 135-145.

Karimi, H. A. (2011). Indoor Navigation. *Universal Navigation on Smartphones*, 59-73.

Kavraki, L. E., Svestka, P., Latombe, J. C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Journal of Robotics and Automation, 12*(4), 566-580.

Kawaji, H., Hatada, K., Yamasaki, T., and Aizawa, K. (2010). *Image-based indoor positioning system: fast image matching using omnidirectional panoramic images*. Paper presented at the Proceedings of the 1st ACM international workshop on Multimodal pervasive video analysis, Firenze, Italy.

Kelly, A. (2004). Linearized Error Propagation in Odometry. *The International Journal of Robotics Research, 23*(2), 179-218.

Koenig, S., and Likhachev, M. (2002). *D*lite*. Paper presented at the Eighteenth national conference on Artificial intelligence, Edmonton, Alberta, Canada.

Koenig, S., and Likhachev, M. (2006). A new principle for incremental heuristic search: Theoretical results. *Proceedings of the International Conference on Automated Planning and Scheduling*, 410-413.

Koenig, S., Likhachev, M., and Furcy, D. (2004). Lifelong planning A*. *Artif. Intell., 155*(1-2), 93-146.

Koenig, S., Likhachev, M., and Sun, X. (2007). *Speeding up moving-target search*. Paper presented at the Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, Honolulu, Hawaii.

Koenig, S., and Sun, X. (2009). Comparing real-time and incremental heuristic search for real-time situated agents. *Autonomous Agents and Multi-Agent Systems, 18*(3), 313-341.

Korf, R. E. (1990). Real-time heuristic search. *Artif. Intell., 42*(2-3), 189-211.

Kubitz, O., Berger, M. O., Perlick, M., and Dumoulin, R. (1997, 4-7 May 1997). *Application of radio frequency identification devices to support navigation of autonomous mobile robots.* Paper presented at the IEEE 47th Vehicular Technology Conference, 1997, .

Kushki, A., Plataniotis, K. N., and Venetsanopoulos, A. N. (2010). Intelligent Dynamic Radio Tracking in Indoor Wireless Local Area Networks. *IEEE Transactions on Mobile Computing, , 9*(3), 405-419.

Ladd, A. M., Bekris, K. E., Marceau, G., Rudys, A., Wallach, D. S., and Kavraki, E. E. (2002, 2002). *Using wireless Ethernet for localization.* Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002.

Langer, D., and Thorpe, C. (1992, 7-10 Jul 1992). *Sonar Based Outdoor Vehicle Navigation And Collision Avoidance.* Paper presented at the Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on.

Latombe, J.-C. (1991). *Robot Motion Planning*: Kluwer Academic Publishers.

LaValle, S. M. (2006). *Planning Algorithms*: Cambridge University Press.

Lee, S. (2009). Use of infrared light reflecting landmarks for localization. *Industrial Robot-an International Journal, 36*(2), 138-145.

Leonard, J. J., and Durrant-Whyte, H. F. (1991). Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on, 7*(3), 376-382.

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S. (2005, June, 2005.). *Anytime Dynamic A*: An Anytime, Replanning Algorithm.* Paper presented at the International Conference on Automated Planning and Scheduling (ICAPS).

Liu, Y., and Li, X. R. (2010). Aided strapdown inertial navigation for autonomous underwater vehicles. 76981H-76981H.

Lozano-P, T., and Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM, 22*(10), 560-570.

Lukianto, C., Ho, x, nniger, C., and Sternberg, H. (2010, 15-17 Sept. 2010). *Pedestrian smartphone-based indoor navigation using ultra portable sensory equipment.* Paper presented at the International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2010

Manapure, S. S., Darabi, H., Patel, V., and Banerjee, P. (2004, 2004). *A comparative study of radio frequency-based indoor location sensing systems.* Paper presented at the IEEE International Conference on Networking, Sensing and Control, 2004.

Matsumoto, Y., Ino, T., and Ogasawara, T. (2001). Development of intelligent wheelchair system with face and gaze based interface. *Robot and Human Communication, Proceedings* 262-267.

Matsumoto, Y., Ino, T., and Ogsawara, T. (2001, 2001). *Development of intelligent wheelchair system with face and gaze based interface.* Paper presented at the Proceedings. 10th IEEE International Workshop on Robot and Human Interactive Communication, 2001. .

Mautz, R. (2009, 19-19 March 2009). *The challenges of indoor environments and specification on some alternative positioning systems.* Paper presented at the Positioning, Navigation and Communication, 2009. WPNC 2009. 6th Workshop on.

McCarthy, C., and Bames, N. (2004, 26 April-1 May 2004). *Performance of optical flow techniques for indoor navigation with a mobile robot.* Paper presented at the Proceedings 2004 IEEE International Conference on Robotics and Automation, 2004. .

Michel, O. (2004). Webots: Professional Mobile Robot Simulation. *Journal of Advanced Robotics Systems, 1*(1), 39-42.

Milella, A., and Siegwart, R. (2006, 04-07 Jan. 2006). *Stereo-Based Ego-Motion Estimation Using Pixel Tracking and Iterative Closest Point.* Paper presented at the Computer Vision Systems, 2006 ICVS '06. IEEE International Conference on.

Milton Roberto, H. (2010). *Feature-Based Mapping Using Incremental Gaussian Mixture Models*.

Moravec, H. (1988). Sensor fusion in certainty grids for mobile robots. *AI Mag., 9*(2), 61-74.

Moravec, H. P., and Elfes, A. (1985, Mar 1985). *High resolution maps from wide angle sonar.* Paper presented at the 1985 IEEE International Conference on Robotics and Automation..

Nguyen, X., Jordan, M. I., and Sinopoli, B. (2005). A kernel-based learning approach to ad hoc sensor network localization. *ACM Trans. Sen. Netw., 1*(1), 134-152.

Nick, T., Cordes, S., Gotze, J., and John, W. (2012, 13-15 Nov. 2012). *Camera-assisted localization of passive RFID labels.* Paper presented at the 2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN),.

Nieuwenhuisen, M., Stuckler, J., and Behnke, S. (2010, 3-7 May 2010). *Improving indoor navigation of autonomous robots by an explicit representation of doors.* Paper presented at the IEEE International Conference on Robotics and Automation (ICRA), 2010.

Nikitin, P. V., Martinez, R., Ramamurthy, S., Leland, H., Spiess, G., and Rao, K. V. S. (2010, 14-16 April 2010). *Phase based spatial identification of UHF RFID tags.* Paper presented at the RFID, 2010 IEEE International Conference on.

Noborio, H., Naniwa, T., and Arimoto, S. (1990). A quadtree-based path-planning algorithm for a mobile robot. *Journal of Robotic Systems, 7*(4), 555-574.

Noykov, S., and Roumenin, C. (2007). Occupancy grids building by sonar and mobile robot. *Robot. Auton. Syst., 55*(2), 162-175.

Opoku, D., Homaifar, A., and Tunstel, E. (2013). *The A-r-Star (Ar*) Pathfinder*. International Journal of Computer Applications.

Papadopoulos, E., and Misailidis, a. M. (2007, July 2-5, 2007). *On Differential Drive Robot Odometry with Application to Path Planning* Paper presented at the European Control Conference, Kos, Greece.

Ramalingam, G., and Reps, T. (1996). An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms, 21*(2), 267-305.

Retscher, G. (2006, April 25-27, 2006). *Location Determination in Indoor Environments for Pedestrian Navigation.* Paper presented at the Position, Location, And Navigation Symposium, 2006 IEEE/ION.

Ruff, T., and Hession-Kunz, D. (1998, 12-15 Oct. 1998). *Application of radio frequency identification systems to collision avoidance in metal/nonmetal mines.* Paper presented at the Industry Applications Conference, 1998. Thirty-Third IAS Annual Meeting. The 1998 IEEE.

Sanpechuda, T., and Kovavisaruch, L. (2008, 14-17 May 2008). *A review of RFID localization: Applications and techniques.* Paper presented at the ECTI-CON 2008. 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008..

Savage, P. G. (2007). *Strapdown Analytics*: Strapdown Associates.

Schiele, B., and Crowley, J. L. (1994, 8-13 May 1994). *A comparison of position estimation techniques using occupancy grids.* Paper presented at the 1994 IEEE International Conference on Robotics and Automation, 1994..

Schroeder, W. J., Zarge, J. A., and Lorensen, W. E. (1992). Decimation of triangle meshes. *SIGGRAPH Comput. Graph., 26*(2), 65-70.

Se, S., Lowe, D., and Little, J. (2001, 2001). *Vision-based mobile robot localization and mapping using scale-invariant features.* Paper presented at the Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation, 2001. .

Sfeir, J., Saad, M., and Saliah-Hassane, H. (2011, 17-18 Sept. 2011). *An improved Artificial Potential Field approach to real-time mobile robot path planning in an unknown*

environment. Paper presented at the IEEE International Symposium on Robotic and Sensors Environments (ROSE), 2011

Sgouros, N. M., PapaKonstantinou, G., and Tsanakas, P. (1996, 22-28 Apr 1996). *Localized qualitative navigation for indoor environments.* Paper presented at the Proceedings., 1996 IEEE International Conference on Robotics and Automation, 1996. .

Shih-Ying, C., Tsui-Ping, C., and Zhi-Hong, C. (2012, 4-6 June 2012). *An Efficient Theta-Join Query Processing Algorithm on MapReduce Framework.* Paper presented at the Computer, Consumer and Control (IS3C), 2012 International Symposium on.

Smailagic, A., and Kogan, D. (2002). Location sensing and privacy in a context-aware computing environment. *Wireless Communications, IEEE, 9*(5), 10-17.

Smith, R., Self, M., and Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics. In J. C. Ingemar & T. W. Gordon (Eds.), *Autonomous robot vehicles* (pp. 167-193): Springer-Verlag New York, Inc.

Song, J., Haas, C. T., and Caldas, C. H. (2007). A proximity-based method for locating RFID tagged objects. *Adv. Eng. Inform., 21*(4), 367-376.

Sooyong, L., and Jae-Bok, S. (2007, 17-20 Oct. 2007). *Mobile robot localization using infrared light reflecting landmarks.* Paper presented at the ICCAS '07. International Conference on Control, Automation and Systems, 2007. .

Stentz, A. (1994, 8-13 May 1994). *Optimal and efficient path planning for partially-known environments.* Paper presented at 1994 IEEE International Conference on.the Robotics and Automation, 1994.

Stentz, A. (1995a). *The Focussed D* Algorithm for Real-Time Replanning*. Paper presented at the International Joint Conference on Artificial Intelligence.

Stentz, A. (1995b). Optimal and Efficient Path Planning for Unknown and Dynamic Environments. *International Journal of Robotics & Automation, 10*(3), 89-100.

Stepan, P., Kulich, M., and Preucil, L. (2005). Robust data fusion with occupancy grid. *Systems, Man, and Cybernetics, Part C: IEEE Transactions on Applications and Reviews, 35*(1), 106-115.

Sun, X., Koenig, S., and Yeoh, W. (2008a). *Generalized Adaptive A*.* Paper presented at the International Joint Conference On Autonomous Agents And Multiagent Systems (Aamas).

Sun, X., Koenig, S., and Yeoh, W. (2008b). *Real-Time Adaptive A*.* Paper presented at the International Joint Conference On Autonomous Agents And Multiagent Systems.

SungHwan, A., Sukjune, Y., Seungyong, H., Nosan, K., and Kyung Shik, R. (2012, 7-12 Oct. 2012). *On-board odometry estimation for 3D vision-based SLAM of humanoid robot.* Paper presented at the Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on.

Tarin Sauer, C., Brugger, H., Hofer, E. P., and Tibken, B. (2001, 2001). *Odometry error correction by sensor fusion for autonomous mobile robot navigation.* Paper presented at the Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE.

Thiesse, F., Fleisch, E., and Dierkes, M. (2006). LotTrack: RFID-based process control in the semiconductor industry. *Pervasive Computing, IEEE, 5*(1), 47-53.

Thrapp, R., Westbrook, C., and Subramanian, D. (2001, 2001). *Robust localization algorithms for an autonomous campus tour guide.* Paper presented at the Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation, 2001. .

Tuna, G., Gulez, K., Gungor, V. C., and Veli Mumcu, T. (2012, 25-28 Oct. 2012). *Evaluations of different Simultaneous Localization and Mapping (SLAM) algorithms.* Paper presented at the IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society.

Ul-Haque, I., and Prassler, E. (2010, 7-9 June 2010). *Experimental Evaluation of a Low-cost Mobile Robot Localization Technique for Large Indoor Public Environments.* Paper presented at the Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK).

Vatavu, A., Danescu, R., and Nedevschi, S. (2011, 25-27 Aug. 2011). *Environment perception using dynamic polylines and particle based occupancy grids.* Paper presented at the Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on.

Vázquez-Martín, R., Núñez, P., Bandera, A., and Sandoval, F. (2009). *Spectral clustering for feature-based metric maps partitioning in a hybrid mapping framework*. Paper presented at the Proceedings of the 2009 IEEE international conference on Robotics and Automation, Kobe, Japan.

Wang, L. E. (2012). *iNavigation: An Image Based Indoor Navigation System.* Master of Computer and Information Sciences, Auckland University of Technology. Retrieved from http://hdl.handle.net/10292/4743

Ward, A., Jones, A., and Hopper, A. (1997). A new location technique for the active office. *Personal Communications, IEEE, 4*(5), 42-47.

Wendel, J. (2011). *Integrierte Navigationssysteme: Sensordatenfusion, GPS und Inertiale Navigation*: Oldenbourg Wissensch.Vlg.

Wooden, D. T. (2006). *Graph-based Path Planning for Mobile Robots.* Doctor of Philosophy, Georgia Institute of Technology. Retrieved from https://smartech.gatech.edu/bitstream/handle/1853/14055/wooden_david_t_200611_phd.pdf?sequence=1

Yahja, A., Stentz, A., Singh, S., and Brumitt, B. L. (1998, 16-20 May 1998). *Framed-quadtree path planning for mobile robots operating in sparse environments.* Paper presented at the IEEE International Conference on Robotics and Automation, 1998. Proceedings. 1998

Yenilmez, L., and Temeltas, H. (2007). A new approach to map building by sensor data fusion: sequential principal component-SPC method. *The International Journal of Advanced Manufacturing Technology, 34*(1-2), 168-178.

Yershova, A., Jaillet, L., Simeon, T., and LaValle, S. M. (2005, 18-22 April 2005). *Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain.* Paper presented at the Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005. .

Ying, Z., Juan, L., Hoffmann, G., Quilling, M., Payne, K., Bose, P., and Zimdars, A. (2010, 8-12 Nov. 2010). *Real-time indoor mapping for mobile robots with limited sensing.* Paper presented at the IEEE 7th International Conference on Mobile Adhoc and Sensor Systems (MASS), 2010

Youssef, M., and Agrawala, A. (2005). *The Horus WLAN location determination system*. Paper presented at the Proceedings of the 3rd international conference on Mobile systems, applications, and services, Seattle, Washington.

Zen, H., Nankaku, Y., and Tokuda, K. (2011). Continuous Stochastic Feature Mapping Based on Trajectory HMMs. *IEEE Transactions on Audio, Speech, and Language Processing, 19*(2), 417-430.

Zhiwei, Z., Oskiper, T., Samarasekera, S., Kumar, R., and Sawhney, H. S. (2007, 14-21 Oct. 2007). *Ten-fold Improvement in Visual Odometry Using Landmark Matching.* Paper presented at the IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007.

Zhong, J., and Su, J. (2011, 22-24 July 2011). *Narrow passages identification for Probabilistic Roadmap Method.* Paper presented at the 30th Chinese Control Conference (CCC), 2011

Zhou, J., and Shi, J. (2009). RFID localization algorithms and applications—a review. *Journal of Intelligent Manufacturing, 20*(6), 695-707.

Appendix A

This is a sample unoptimized matlab code for the A-Star algorithm

```
%The A* algorithm programmed in the algorithmic way
function Res=AStarPathfinder (MAP,inStart,inGoal)
%This function finds the shortest path of a given grid system using the
%AStar algorithm.
tic
MAP(MAP==0)=inf;
global OPEN g goal start cmap open closed new xMax yMax TAG
goal=inGoal; start=inStart; cmap=MAP; new=0; open=1; closed=2;
[xMax,yMax]=size(cmap);g=inf*ones(size(cmap));
g(start(1),start(2))=0; TAG=zeros(size(cmap));TAG(start(1),start(2))=open;
OPEN=[];
Insert(start,start,h(start));
while (1)
   if isempty(OPEN)
      path=-1;
      pathflag=0;
      break;
   else
      s=pop();
```

```
20          if isequal(s,goal)
21              pathflag=1;
22              break;
23          end
24          TAG(s(1),s(2))=closed;
25          neighbors=Expand(s);
26          for n=1:size(neighbors,1)
27              UpdaNode(s,neighbors(n,:));
28          end
29      end
30  end
31  if pathflag
32  Res.OPEN=OPEN; Res.g=g; Res.path=ExtrPath(); %Output Variables
33  plotPath(cmap,Res.path);
34  time=toc
35  end
36  end
37  function path=ExtrPath()
38  %This is a greedy approach to extracting the path after it has been built
39  %by get shortest pat
40  global start goal g
41  s=goal; %Start from the goal
```

```
42  path=goal;
43  while ~isequal(start,s) %Check whether we are at the goal
44     ming=inf; %Set to the highest
45     neighbors=Expand(s); %Get all the neighbors
46     %Check for the minimum g among the candidates
47     for n=1:size(neighbors,1)
48        ss=neighbors(n,:);
49        if g(ss(1),ss(2))<ming
50           ming=g(ss(1),ss(2)); %If less, winner
51           minss=ss;
52        end
53     end
54     s=minss;
55     path=[path;s]; %Append the current winner to the path
56  end
57  end
58  function UpdaNode(s,u)
59  %This is for updating the current vertex/node
60  % disp('In Update Vertex');
61  global g TAG open %new
62     if (g(s(1),s(2))+CalcCost(s,u))<g(u(1),u(2)) %Calculate the rhs
63        g(u(1),u(2))=(g(s(1),s(2))+CalcCost(s,u));
```

```
64          parent=s;
65          f=g(u(1),u(2))+h(u);
66          if TAG(u(1),u(2))==open  %if u is on the U Queue
67              Remove(u); %Remove it
68          end
69          Insert(u,parent,f);
70      end
71  end
72  function neighbors=Expand(s)
73  %use to generate the neighbors of a node s in an 2D 8-connected grid
74  r=s(1); %Extract the row
75  c=s(2); %Extract the column
76  global xMax yMax cmap
77  neighbors=[];
78  temp=[r-1, r-1, r-1, r,r,r+1,r+1,r+1;
79            c-1, c, c+1, c-1,c+1, c-1,c,c+1]'; %Find the neigbors
80          for i=1:size(temp,1)
81              if (temp(i,1)>0 && temp(i,1)<=xMax)&&(temp(i,2)>0 && temp(i,2)<=yMax)
                    if cmap(temp(i,1),temp(i,2))==inf %||TAG(temp(i,1),temp(i,2))==closed %If it
82  is a wall,
                        %drop it. NB: This is for excluding the blocked nodes and so if any node is
83  not blocked
```

```
84                  % then it becomes ineffective
85                  continue
86              end
87              neighbors=[neighbors;temp(i,:)];
88          end
89      end
90  end
91  function neighbors=GeneNeighbors(s)
92  %use to generate the neighbors of a node s in an 2D 8-connected grid
93  r=s(1); %Extract the row
94  c=s(2); %Extract the column
95  global xMax yMax  TAG closed
96  neighbors=[];
97  temp=[r-1, r-1, r-1, r,r,r+1,r+1,r+1;
98          c-1, c, c+1, c-1,c+1, c-1,c,c+1]'; %Find the neigbors
99      for i=1:size(temp,1)
100         if (temp(i,1)>0 && temp(i,1)<=xMax)&&(temp(i,2)>0 && temp(i,2)<=yMax)
                if TAG(temp(i,1),temp(i,2))~=closed %If it is a wall, drop it. NB: This is for
101 excluding
                    % the blocked nodes and so if any node is not blocked then it becomes
102 ineffective
                continue
103
```

```
104                 end
105                 neighbors=[neighbors;temp(i,:)];
106               end
107           end
108   end
109   function [u,parent,f]=pop()
110   %pop the top node from the U queue
111   global OPEN closed  TAG
112   u=OPEN(1,1:2); %pop the top node
113   parent=OPEN(1,3:4); %pop the top node parents
114   f=OPEN(1,5); %pop the top node
115   OPEN(1,:)=[]; %Remove it
116   TAG(u(1),u(2))=closed;
117   end
118   function Insert(s,parent,f)
119   %For inserting a given node onto the UQUEUE node
120   global OPEN TAG open
121   if isempty(OPEN) %If the U queue is empty then just insert onto it
122      OPEN=[s,parent,f];
123       TAG(s(1),s(2))=open;
124   else
125         downNodes=(OPEN(:,5)<f);
```

```
126       OPEN=[OPEN(downNodes,:);[s,parent,f];OPEN(~downNodes,:)];
127        TAG(s(1),s(2))=open;
128   end
129   end
130   function Remove(s)
131   %For removing a node from the
132   global OPEN TAG closed
133   % ind=find();
134   OPEN(OPEN(:,1)==s(1)& OPEN(:,2)==s(2),:)=[];
135   TAG(s(1),s(2))=closed;
136   end
137   function h=h(s)
138   %Calculates the heuristic cost estimate from s to the goal
139   global goal   %NB goal is a global variable
140   h=norm(s-goal,2); %The norm in this case
141   end
142   function cost=CalcCost(s,ss)
143   %This calculates the cost of a given node s and its predecessor ss given
144   %the cost map cmap
145   global cmap
146   eucdist=norm(s-ss,2); %First calculate the euclidean distance
147   cost=0.5*eucdist*(cmap(s(1),s(2))+cmap(ss(1),ss(2))); %then multiply
```

```
148    % the average cost with the euclidean distance
149    end
```

Appendix B

This is a sample non-optimized code of the A-r-Star Pathfinder

```
1  %The A* algorithm programmed in the algorithmic way
2  function Res=ArStarPathfinder (MAP,inStart,inGoal)
3  %This function finds the shortest path of a given grid system using the
4  %AStar algorithm.
5  tic
6  MAP(MAP==0)=inf;
7  global OPEN CLOSED g goal start cmap open closed new xMax yMax TAG skip r
8  goal=[inGoal(1) inGoal(2)]; start=[inStart(1) inStart(2)]; cmap=MAP; new=0;
   g(start(1),start(2))=0; TAG=zeros(size(cmap));TAG(start(1),start(2))=open; r=inf;
9  CLOSED=[];
10 OPEN=[];
11 Insert(start,start,h(start));
12 while (1)
13    %Check whether the OPEN is empty=>there is no path linking them
14    if isempty(OPEN)
15       path=-1;
16       pathflag=0;
17       break;
18       %The Next three lines change the algorith from the Basic A-r-Star to the A-r-Star
19    elseif TAG(OPEN(1,1),OPEN(1,2))==skip;
```

```
20          OPEN(1,:)=[];
21           continue;
22       else %Get the next best node and expand
23          s=pop();
24          if isequal(s,goal) %Check whether we are at goal
25              pathflag=1;
26              break;
27          end
28  %          TAG(s(1),s(2))=closed;
29          neighbors=RExpand(s);
30          for n=1:size(neighbors,1)
31              UpdaNode(s,neighbors(n,:));
32          end
33  %          g
34       end
35  end
36  if pathflag
37  % path=ExtrPath();
38  imagesc(TAG);
39  % CLOSED
40  path=ExtrPathR();
41  % plotPath(cmap,path);
```

```
42  time=toc
43  end
    Res.OPEN=OPEN; Res.g=g; Res.path=path; Res.CLOSED=CLOSED;
44  Res.TAG=TAG;%Output Variables
45  figure, imagesc(Res.g);
46  figure, imagesc(Res.TAG);
47  end
48
49  function path=ExtrPath()
50  %This is a greedy approach to extracting the path after it has been built
51  %by get shortest pat
52  global start goal g
53  s=goal; %Start from the goal
54  path=goal;
55  while ~isequal(start,s) %Check whether we are at the goal
56     ming=inf; %Set to the highest
57     neighbors=RExpand(s); %Get all the neighbors
58     %Check for the minimum g among the candidates
59     for n=1:size(neighbors,1)
60        ss=neighbors(n,:);
61        if g(ss(1),ss(2))<ming
62           ming=g(ss(1),ss(2)); %If less, winner
```

```
63              minss=ss;
64          end
65      end
66      s=minss;
67      path=[path;s]; %Append the current winner to the path
68  end
69  end
70
71  function UpdaNode(s,u)
72  %This is for updating the current vertex/node
73  % disp('In Update Vertex');
74  global g TAG open %new
75      if (g(s(1),s(2))+CalcCost(s,u))<g(u(1),u(2)) %Calculate the rhs
76          g(u(1),u(2))=(g(s(1),s(2))+CalcCost(s,u));
77          parent=s;
78          f=g(u(1),u(2))+h(u);
79          if TAG(u(1),u(2))==open  %if u is on the U Queue
80              Remove(u); %Remove it
81          end
82          Insert(u,parent,f);
83      end
84  end
```

```
85
86  function leveRNeighbors=RExpand(s)
87  %use to generate the neighbors of a node s in an 2D 8-connected grid
88  x=s(1); %Extract the row
89  y=s(2); %Extract the column
90  R=1;
91  skipflag=true;
92  global xMax yMax cmap  TAG closed skip r goal
93
94  while(1) %Iterate forever unless interrupted
95  leveRNeighbors=[];
96  temp= GeneLRNeighbors(x,y,R)'; %Get the level R neighbors
97  for i=1:size(temp,1)
            if (temp(i,1)<=0 || temp(i,1)>xMax)||(temp(i,2)<=0 || temp(i,2)>yMax) %IF any is
98  out of bounds
99              skipflag=false; %we cannot skip
100             continue
101         elseif cmap(temp(i,1),temp(i,2))==inf %It is a wall, we cannot skip
102             skipflag=false;
103             continue
104         elseif isequal(temp(i,:),goal)
105             leveRNeighbors=[leveRNeighbors;temp(i,:)];
```

```
106                 skipflag=false;
107                 continue
              elseif  TAG(temp(i,1),temp(i,2))==closed || TAG(temp(i,1),temp(i,2))==skip%If
108   we have already mark this as a skip or closed, leave it
109                 continue
110             end
111                 leveRNeighbors=[leveRNeighbors;temp(i,:)];
112   end
113           if ~skipflag ||R>=r %If permissible radius or don't skip
114             leveRNeighbors;
115             break;
116           else %Else tag all of them as skip
117             leveRNeighbors;
118             for n=1:size(leveRNeighbors,1)
119                 TAG(leveRNeighbors(n,1),leveRNeighbors(n,2))=skip;
120             end
121             R=R+1; %increase the R and continue
122           end
123   end
124   end
125
126   function path=ExtrPathR()
```

```
% This function extracts the path using the back pointers after the ArStart search.
global goal start CLOSED %Declare the global variables needed in this function
child=goal; %Start from the goal and work your way backwards.
% parent=[]; %Parent is null
path=child;
while ~isequal(child,start) %loop until you are at the start
   parent=CLOSED(CLOSED(:,1)==child(1)&CLOSED(:,2)==child(2),3:4); %Get the
parent
   child=parent; %Make the parent the next child
   path=[path;child]; %Place the child on the path array
end
end

function [u,parent,f]=pop()
%pop the top node from the U queue
global OPEN closed  TAG CLOSED
u=OPEN(1,1:2); %pop the top node
parent=OPEN(1,3:4); %pop the top node parents
f=OPEN(1,5); %pop the top node
CLOSED=[CLOSED;OPEN(1,:)];
OPEN(1,:)=[]; %Remove it
TAG(u(1),u(2))=closed;
```

```
148 end
149
150
151 function Insert(s,parent,f)
152 %For inserting a given node onto the UQUEUE node
153 % disp('In Insert')
154 global OPEN TAG open
155 % s
156 if isempty(OPEN) %If the U queue is empty then just insert onto it
157    OPEN=[s,parent,f];
158     TAG(s(1),s(2))=open;
159 else
160       downNodes=(OPEN(:,5)<f);
161      OPEN=[OPEN(downNodes,:);[s,parent,f];OPEN(~downNodes,:)];
162       TAG(s(1),s(2))=open;
163 end
164 end
165
166 function Remove(s)
167 %For removing a node from the
168 global OPEN TAG closed
169
```

```
% UQUEUE(UQUEUE(:,1)==s(1)&&UQUEUE(:,1)==s(2))=[];
% ind=find();
OPEN(OPEN(:,1)==s(1)& OPEN(:,2)==s(2),:)=[];
TAG(s(1),s(2))=closed;
end

function h=h(s)
%Calculates the heuristic cost estimate from s to the goal

global goal   %NB goal is a global variable
h=norm(s-goal,2); %The norm in this case
end

function cost=CalcCost(s,ss)
%This calculates the cost of a given node s and its predecessor ss given
%the cost map cmap
global cmap
eucdist=norm(s-ss,2); %First calculate the euclidean distance
cost=0.5*eucdist*(cmap(s(1),s(2))+cmap(ss(1),ss(2))); %then multiply the average cost
with the euclidean distance
end

```

```
191
192 function Neighbors = GeneLRNeighbors( x,y,R)
    %Neighbors FXNRNEIGHBORS( R, x,y ) Finds the R box neighborhood of (x,y) in a 2D
193 space.
194
195 % (x,y) is the center and R is the radius from the center in terms of tiles
196
197 % tic
198 dx=1;
199 dy=1;
200 fx=x-R;
201 fy=y-R;
202 % coord=[fx,y-1;fx,y+1;x-1,fy;x+1,fy];
203 coord=[];
204     coord=[coord;[[fx:x+R]',repmat(y-R,[2*R+1,1])]];
205     coord=[coord;[[fx:x+R]',repmat(y+R,[2*R+1,1])]];
206     coord=[coord;[repmat(x-R,[2*R-1,1]),[fy+dy:y+R-1]']];
207     coord=[coord;[repmat(x+R,[2*R-1,1]),[fy+dy:y+R-1]']];
208 Neighbors=coord';
209 end
```

Appendix C

The Post Dissociative Smoothing Algorithm's MATLAB implementation using the Bresenham's Line Plotting Algorithm. The inputs are the binary image of the map and the Path to be smoothened. It returns New Path (NP) which is a smooth version of the given path.

Path Smoothing

```
1  function NP=fxnSmoothPath(Path,MAP)
2  %% To find the points that have line of sight on a path on a map
3  ps=size(Path,1)
4  NP=Path(ps,:);
5  j=ps;
6  for i=ps-2:-1:1
7      if ~BresenhamLOS(Path(j,:),Path(i,:),MAP)
8          NP=[NP;Path(i+1,:)];
9          j=i+1;
10     end
11 end
12 NP=[NP;Path(i,:)];
13
14 end
15
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
17  function [bol,map]=BresenhamLOS(A,B,map)
18  %This function is for check the validity ofa line of sight from tile in a
19  %grid to then other.
20
21  %The Bresenham for 2nd 3rd 6th and 7th octants
22  x0=A(1);
23  y0=A(2);
24  x1=B(1);
25  y1=B(2);
26  if x0<1 || y0<1 || x1<1 || y1<1
27      disp('sorry, Im yet to learn how to handle these type of problem');
28      return
29  end
30  % tic
31  dy=y1-y0;
32  dx=x1-x0;
33  f=0;
34  bol=true;
35  if dx<0 sx=-1; else sx=1; end
36  if dy<0 sy=-1; else sy=1; end
37  gx=abs(dx);
38  gy=abs(dy);
```

```
39  f=gx/2;
40  if gx>=gy
41  %    display('loop 1');
42      for i=0:gx
43          coord=[x0,y0];
44          if ~map(x0,y0)
45              bol=false;
46  %              timeused=toc;
47              return
48  %          else
49  %              map(x0,y0)=3;
50          end
51          f=f+gy;
52          if (f>gx)
53              f=f-gx;
54              y0=y0+sy;
55          end
56          x0=x0+sx;
57      end
58  else
59      f=gy/2;
60  %    display('loop 2');
```

```
61      for i=0:gy
62          coord=[x0,y0];
63          if ~map(x0,y0)
64              bol=false;
65  %               timeused=toc;
66              return
67  %           else
68  %               map(x0,y0)=3;
69          end
70          f=f+gx;
71          if (f>gy)
72              f=f-gy;
73              x0=x0+sx;
74          end
75          y0=y0+sy;
76      end
77  end
78  % timeused=toc
79  % imshow(map,[0,10]);
80  % truesize([128,128]);
81  end
```

Appendix D

Below are the publications resulting from this research work.

Journal

Opoku, D., Homaifar, A., and Tunstel, E. (2013). *The A-r-Star (Ar*) Pathfinder*. International Journal of Computer Applications; vol. (67), pp. 0975-8887 .

Refereed Conference

Opoku, D. and Abdollah Homaifar, *Non-Classical Multi-Sensor Data Fusion Techniques*, **Conference proceedings**, IEEE Aerospace Conference, 2010, ISBN 978-1-4244-3888-4.

Submitted/In preparation

Daniel Opoku, Abdollah Homaifar, and Edward W. Tunstel, *RFID-Augmentation for Improving Long-term Pose Accuracy of an Indoor Navigating Robot*, **Conference paper**, ***(About to be submitted***).

Poster

1. **Opoku, D.** and A. Homaifar, "Intelligent Navigation of a Robot in a Dynamic Home Environment using Laser Range Finder," **Poster,** *1st Annual COE Graduate Student Research Poster Competition*, NC A&T SU, April 2012.
2. **Opoku, D.,** A Workineh and A. Homaifar, "Design and Implementation of Assistive Robotic Residence Home (DIARRH)," **Poster**, *COE Healthcare Day*, NC A&T SU, February 2012.

3. **Opoku, D.** and A. Homaifar, "Intelligent Navigation of a Robot in a Dynamic Home Environment using Laser Range Finder," **Poster,** *13th Annual Science & Engineering Technology Conference / Defense Tech Exposition*, Charleston SC, April 2012.
4. Workineh, A., **D. Opoku,** A Homaifar, "Evolutionary Learning, Navigation and Target Identification for Assistive Robotic Application", **Poster,** *3rd BEACON Annual Congress*, MSU, MI, August 2011.
5. **Opoku, D.** and A. Homaifar, "The A-r-Star and its applications, **Poster**, *2st Annual COE Graduate Student Research Poster Competition*, NC A&T SU, April 2012.