



Using Architectural Blueprints for Intelligent Robot Cognitive Mapping as a Knowledge-Based System

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Master Degree in
Computer Information Systems

By
Suhail Ibrahim al-Ghazi

Supervisor
Dr. Hussein H. Owaid

Faculty of Information Technology

Middle East University
Amman, Jordan

July, 2010

AUTHORIZATION FORM

إقرار تفويض

أنا صهيب إبراهيم الغازى أفوض جامعة الشرق الأوسط بتزويد نسخ من رسالتي
للمكتبات أو المؤسسات أو الهيئات أو الأفراد عند طلبها.

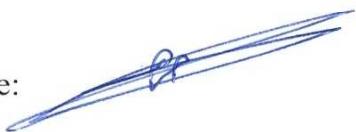
التوقيع:

التاريخ: ٢٠١٥ / ٩ / ١٩

Authorization statement

I, Suhaib Ibrahim al-Ghazi, authorize the Middle East University to supply a copy of my Thesis to libraries, establishments or individuals upon their request.

Signature:

A handwritten signature in blue ink, appearing to read "Suhaib Ibrahim al-Ghazi".

Date: ١٩ - ٩ - ٢٠١٥

Middle East University

Examination Committee Decision

This is to certify that the thesis entitled "Using Architectural Blueprints for Intelligent Robot Cognitive Mapping as a Knowledge-Based System" was successfully defended and approved on August 18th 2010.

Examination Committee Members

Dr. Hussein H. Owaid

Associate Professor, Department of Computer Science
(Middle East University)

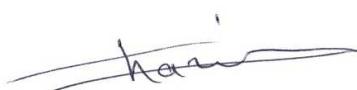
Dr. Hazim A. Farhan

Assistant Professor, Department of Computer Information Systems
(Middle East University)

Prof. Saleh Abu Soud

Professor, Department of Engineering & Computing Sciences
(New York Institute of Technology)

Signature



DECLARATION

I do hereby declare the present research work has been carried out by me under the supervision of Dr. Hussein H. Owaid, and this work has not been submitted elsewhere for any other degree, fellowship or any other similar title.

Signature:



Date: ١٩١٢٠١٠

Suhail Ibrahim al-Ghazi

Department of Computer Information System

Faculty of Information Technology

Middle East University

DEDICATION

Almighty Allah says “And remembers! Your Lord caused to be declared (publicly): "If you are grateful, I will add more (favours) unto you".

So all praise is for Allah, the exalted, for his favours that cannot be counted.

I dedicate this work to my Parents, my brother, my sisters, and my relatives, my friends, and for all those who helped, support and taught me.

ACKNOWLEDGMENTS

I would like to thank my supervisor Dr. Hussein H. Owaid for his support, encouragement, for his proofreading of thesis drafts, and helping me throughout my thesis, putting me on the right track of Artificial Intelligence and intelligent robots' field. I thank the Information Technology Faculty members at the Middle East University for Graduate Studies. I thank my father for his continuous support during my study. I thank the team of the National Robot Center (Amman, Jordan) and the support center of RobotShop Distribution Inc (Boisbriand, Canada) for their guidance on robotics field. I would also like to thank my brother Bilal, my brothers in law Ammar al-Ali and Mohammad al-Shuraida and my friend Jamal Haikal for their support during writing my thesis.

Table of Contents

Chapter One	2
Introduction.....	2
1.1. Overview	2
1.2. Problem Statement	4
1.3. The Objectives of the proposed project.....	5
1.4. Research Importance	6
1.5. Study Boundaries	6
1.6. Study limitations	6
1.7. Thesis Structure.....	7
Chapter Two.....	9
Literature Survey and Related Work	9
2.1 Overview	9
2.2 Literature Survey Related to Intelligent Robots.....	9
2.3 Literature Survey Related to Intelligent Robot Mapping.....	9
2.4 Literature Survey Related to Cognitive Mapping	10
2.5 Literature Survey Related to Analyzing Architectural Maps.....	12
2.5.1 Recognizing Maps' Objects.....	13
Chapter Three.....	16
Terminology.....	16
3.1 Overview	16
3.2 Architectural Blueprints	16
3.2.1 Architectural Blueprint Maps Drafting Standards.....	16

3.3	Pixel & Pixel Neighbors.....	18
3.4	Masks & Scanned Pixels.....	19
3.5	Pattern Recognition & Matching.....	20
3.6	Path Definition	21
3.7	Main Path	22
3.8	Rooms.....	22
3.9	Turnings	23
3.10	Vertical and Horizontal Corridors.....	23
3.11	Start and end points	24
3.12	Labeling Maps.....	24
3.13	Medial Axis (skeleton).....	25
3.14	Cognitive Mapping.....	26
	Chapter Four	31
	The Implementation of Proposed Methodology	31
4.1	Overview	31
4.2	The Algorithms	31
1-	Threshold algorithm.....	36
2-	Flood Fill Algorithm	39
3-	Start Point Finding	40
4-	End Point Finding	41
5-	Main Path Finding.....	42
6-	Targeted Roo@m Finding	44
7-	Similarity Algorithm.....	46
a.	Pattern Matching.....	47

b.	Target room door	48
8-	Medial Axis.....	49
a.	Finding Central Pixels for Corridors.....	51
b.	Vertical Corridors	51
c.	Horizontal Corridors	53
d.	Finding Turnings Centers	56
9-	Labeling the Medial-Axis	58
10-	Finding the Shortest Path	59
a.	Creating Graph (Connecting Nodes)	60
b.	Floyd's Algorithm.....	62
11-	Translating to Directions.....	63
a.	Finding the Angle	65
b.	Allocate Doors in Directions	65
12-	The Cognitive Mapping Algorithm.....	68
a.	Represent knowledge in the knowledge base	69
b.	Cognitive Mapping Structure in the knowledge base	70
c.	Cognitive Mapping Storing Algorithm.....	72
	Chapter Five.....	74
	Conclusion, Discussion & Future work	74
5.1.	Overview	74
5.2.	Conclusion.....	74
5.3.	Discussion	75
5.4.	Future Work	75

List of Figures

Figure 3-1 : Architectural Blueprint Maps Symbols.....	17
Figure 3-2 : Pixel's Neighbors	18
Figure 3-3 : Masking.....	19
Figure 3-4 : Pixels Matching.....	21
Figure 3-5 : Main Path in a Building	22
Figure 3-6 : Horizontal (right) & Vertical (left) Corridors	24
Figure 3-7 : Different Labeling Techniques	25
Figure 3-8 : 2d Skeleton Example (Sundar et al. 2003).....	25
Figure 3-9 : 3d Skeleton Example (Sundar et al. 2003).....	26
Figure 3-10: Functional Model of Human System Owaied (2009)	27
Figure 3-11 : Intelligent Robot Knowledge Based System Owaied (2009)	27
Figure 3-12 : The Architecture of Intelligent Robot as Knowledge-Based System Owaied (2009).....	28
Figure 3-13 : Main Algorithms as Processes as Knowledge-Based System	29
Figure 4-1: The Data Flow Diagram for the Algorithms inside the Program.....	33
Figure 4-2 : Offices Building at Regus Whitefields Centre, Bangalore,	34
Figure 4-3 : Labeled Architectural Blueprint for the Center Floor at Regus Whitefields Centre, Bangalore, India.	35
Figure 4-4 : Scanned Blueprints have a lot of Noise.	36
Figure 4-5 : Thresholding Pseudo Code	37
Figure 4-6 : Blueprint after Threshholding, Using MATLAB R2009b.	38
Figure 4-7 : Flood Fill Algorithm Pseudo code	40
Figure 4-8 : StartPoint Finding Algorithm Pseudo code	41

Figure 4-9 : endpoint Algorithm Pseudo Code	42
Figure 4-10 : Start (right (shape with x symbol)) & End (left (shape with o symbol)) Point Labels.....	42
Figure 4-11 : Main Path Finding Algorithm Pseudo code.....	43
Figure 4-12 : Main Path Extracted from the Blueprint after Applying the Algorithm.	44
Figure 4-13 : Targeted Room Finding Algorithm Pseudo code	45
Figure 4-14 : Targeted Door Extracted from the Blueprint after Applying the Algorithm.	45
Figure 4-15 : Similarity Algorithm Pseudo code	46
Figure 4-16 : Different Patterns Examples.	47
Figure 4-17 : Different Door Patterns.....	47
Figure 4-18 : Target Door Round Pixels Set Within Target Room and Main Path Pixels.	48
Figure 4-19 : Finding Doors Pseudo Code.	49
Figure 4-20 : Finding Boarder Pixels Pseudo Code.....	50
Figure 4-21 : Center Pixels for Vertical Corridors Pseudo Code.	52
Figure 4-22 : the Path after Finding Center Pixels for Vertical Corridors Only.....	53
Figure 4-23 : Center Pixels for Horizontal Corridors Pseudo Code.	54
Figure 4-24 : Connecting Doors with Medial Axis Pseudo Code.	55
Figure 4-25 : Path After Finding Center Pixels For Vertical & Horizontal Corridors.	56
Figure 4-26 : Finding Turns Pseudo Code.....	57
Figure 4-27: Original Medial Axis after Applying The Algorithm on MATLAB R2009b.	58
Figure4-28 : Start Point Node in The Map.	59
Figure 4-29 : Labeled Medial Axis with Nodes on Red for Simulation Purpose.....	59
Figure 4-30 : Two Cases Appears on Connecting Nodes.....	60

Figure 4-31 : Connecting Nodes Pseudo Code.....	61
Figure 4-32 : Shortest Path by Floyd Algorithm Pseudo Code.....	62
Figure 4-33 : Angels Translated into Directions.....	63
Figure 4-34 : Converting Nodes to Directions Algorithm Pseudo Code.....	64
Figure 4-35 : Translating Doors Algorithm Pseudo Code.....	66
Figure 4-36 : Door Position Translation.....	67
Figure 4-37 : Directions Cognitive Map.....	69
Figure 4-38 : Frame Structure.....	70
Figure 4-39 : The Cognitive Map, Network of frames.....	71
Figure 4-40 : Cognitive Mapping Storing Algorithm Pseudocode.....	72
Figure 5-1 : Multi Floored Building Architectural Map (Alex et al. 2009).....	76
Figure 5-2 : Different Kinds of Doors (Sam, 2009).....	77

ABSTRACT

This thesis presents an algorithmic approach to constructing cognitive maps for the intelligent robots from labeled architectural blueprint maps for sophisticated buildings.

Since most of the previous methods were studied in different areas, such as intelligent robot, artificial intelligence and image processing, none of those methods are considered this approach for development of cognitive maps.

The purpose of this thesis is to give intelligent robots the ability to be mobilized in critical and unsafe city buildings without the need of human guidance or interference, using architectural blueprint maps as the only knowledge source to accomplish their jobs.

New algorithms have been developed and built to be the core of the procedures and steps, which will be used by the intelligent robot for development of cognitive maps from the architectural blueprint maps. These algorithms were defined and written using the standard programming code (pseudo code). Some of these algorithms were tested and proved using MATLAB R2009b.

These algorithms will be facilitating the intelligent robot to be more capable of dealing with external resources (maps) and having the ability to read and benefit from these kinds of maps.

الخلاصة

تمثل هذه الدراسة نهج خوارزمي لبناء خرائط إدراكية للروبوتات الذكية من خلال خرائط معمارية مؤسرة للأبنية المعقّدة.

الكثير من الدراسات السابقة غطت منهجيات وطرق تم استخدامها في مختلف المجالات سواء تلك المتعلقة بالذكاء الإصطناعي أو الروبوتات الذكية أو تحليل الصور، أو مجالات أخرى غير متعلقة بذلك، ولكن يجدر الإشارة بأن أي من هذه المنهجيات لم تستخدم لبناء الخرائط الإدراكية كما في هذه الدراسة.

الهدف من هذه الدراسة هو إعطاء الروبوت الذكي القدرة كي يكون مؤهلاً على استكشاف وإتمام الأعمال المنوط به في الأماكن غير المؤهلة والخطيرة بدون الحاجة إلى أي توجيه أو تدخل من جانب الإنسان. وذلك عن طريق استخدام الخرائط المعمارية للأبنية كمصدر وحيد للمعرفة لإتمام واجباته.

تم بناء وتطوير خوارزميات جديدة كي تكون لب الإجراءات والخطوات التي يتم اتخاذها من قبل الروبوت الذكي لبناء قاعدة معرفة للخرائط الإدراكية من الخرائط المعمارية.

تم كتابة تعريف هذه الخوارزميات وكتابتها باستخدام المعايير الثابتة لكتابة الكود البرمجي (pseudo code). والبعض من هذه الخوارزميات تم تجربته وإثباته خلال الدارس باستخدام برنامج MATLAB R2009b.

هذه الخوارزميات بنيت لتساعد الروبوت الذكي من التعامل مع المصادر الخارجية (الخرائط)، وذلك من خلال القدرة على قراءة والإنتفاع من هذه الخرائط.

Chapter One

Chapter One

Introduction

1.1. Overview

Artificial Intelligence has been highlighted and extensively researched, trying to build machines that have some sort of intelligence. that Owaied, Abu-A'ra & Farhan (2010) discussed that the built machines should have certain capabilities, such as the capability of behaving “like a human being, smart, problem solver of unstructured and complex problems as human does, understands languages, learner, and able to reason and analyze data and information, and so on”. For these capabilities to be reached, as Avron, Paul & Edward (1990) state, the machine “should have the facilities of recognition, analysis, deduction, and induction”.

In order to understand the Artificial Intelligence concept, a clear and recent definition was mentioned by Owaied, Abu-A'ra (2007) as the following: “Artificial Intelligence is a concept of study and research for finding a relationship between cognitive science and computation theories in order to represent these relationships as either data structures, search techniques, problem solving methods or representation forms for knowledge and the final goal of AI is to build an intelligent machine, with another benefit which is better understanding of human thinking”. The previous definition was built on the fact that in order for a machine to be intelligent, it should be smart and problem solver. Smart can be defined as “everything gives pleasure and happiness to humans, through the facilities available in all sort of multimedia equipments”. On the other hand, for the intelligent machine to be a problem solver of an unstructured and complex problem, it should be like the human where “humans usually solve algorithmic and non algorithmic problems and most problems are non algorithmic”, this capability is the most important and most of the pioneers of A.I. are concentrating on them like (Winston, 1992) and (Luger, 1999).

As a matter of fact, not only human beings have intelligence, but also all living creatures do. All the living beings have the quality of intelligence even if their abilities are considered by humans as simple. This is due to the fact that these abilities can run sophisticated systems, such as ant colonies, bird flocking, animal herding, bacterial growth, bees forging and human neuron system. These systems either work under supervision or without Supervision (Owaid, Sa'ab, 2008).

In the present time intelligent robots need to be more dynamic and reliable. They should also have the ability to be mobilized even within indoor environments or buildings that are abandoned without the needs of guidance. This ability will facilitate the intelligent robot's reaching its target in the simplest and shortest way.

The previously mentioned ability is more demanded for the intelligent robots with special jobs such as police or fire fighters, when human guidance is not possible.

The talent which is previously described can be hardly reached without external knowledge recourses, such as architectural maps (architectural blueprints maps) or geographical maps.

It is well-known that, there are no satellite images for indoor environments, which makes geographical maps unuseful inside such environments. As a result, the architectural blueprints maps where used in order to completely cover the indoor environments.

There are many differences between indoor and outdoor environments, such as the fact that there are more objects, elements and obstacles in the indoor environments.

Indoor environments vary from simple to very complicated environments, but even simple indoor environment are very hard to be swept by the intelligent robots or even an adult human in time manner without having a simple knowledge about the indoor environment he is facing.

Since knowledge is the human brain's soul, solving problems and communicating with others, require applying the human brain model on the intelligent robot increased by

the demand of building an intelligent robot. The knowledge based system must be applied by covering one of its fields, i.e. receiving, thinking and storing knowledge.

As defined by Stuart-Hamilton (1995), Cognition is “The understanding, acquisition, and processing of knowledge, or, more loosely, thought processes”. However, cognitive science refers to “the interdisciplinary study of the acquisition and use of knowledge. It includes as contributing disciplines: artificial intelligence, psychology, linguistics, philosophy, anthropology, neuroscience, and education” (Eysenck, 1990).

In order to understand the concept of cognitive mapping, which is used during the research, it would be useful to refer to Downs & Stea (1973) who define cognitive mapping is “a process composed of a series of psychological transformations by which an individual acquires, codes, stores, recalls, and decodes information about the relative locations and attributes of phenomena in their everyday spatial environment”.

The project’s aim is to give the ability to the intelligent robot read and gain knowledge from the architectural blueprints maps of indoor environments to build its own cognitive maps as a knowledge-based system.

1.2. Problem Statement

Using architectural blueprints for intelligent robot cognitive mapping faces problems such as:

1. Studying the possibility of using the architectural blueprints maps to build the cognitive maps of the intelligent robot to facilitate intelligent robot navigation.
2. Identifying the relationships between the architectural blueprints maps and the cognitive maps. These relationships will be represented either by data, procedures, searching techniques, or solving problems.
3. Identifying the required information in the architectural blueprint maps which are needed for creating cognitive maps.

4. Solving the difficulties that face simplifying the architectural blueprints maps to gain knowledge that constructs the intelligent robot cognitive maps.
5. Dealing with false information from architectural blueprints maps that could lead to probabilities in building incorrect cognitive maps for the building.
6. Creating the dynamic knowledge base based on the required information for creating cognitive maps, through the interaction with the environment, to be added to the intelligent robot's knowledge base.

1.3. The Objectives of the Proposed Project

Building Cognitive Maps can make it easier for the intelligent robot to work without the need of direct human interaction.

It could also be used to reduce the time to reach a target room in a new building and this could be reduced more by using architectural blueprints maps where what could take hours to search and map a new building (Benjamin, Yung, 1993) will take minutes.

However, we will need less information from the environment to build knowledge which means less space of knowledge stored and then retrieved, which also means less time retrieving that knowledge from the knowledge base and so less energy wasted on processing.

Cognitive Maps supported by knowledge from architectural blueprints maps will also increase the ability of the intelligent robot to sweep more sophisticated buildings where you can notice that all researches have been made for indoor intelligent robot mapping assuming few rooms for the building to be swept, where the proposed method could gain the ability to the intelligent robot to sweep building without taking in consideration its size, since there is an architectural blueprint for the building.

The proposed method does not build the cognitive maps based on the architectural blueprint, but it makes the cognitive map for the intelligent robot dynamic where the

architectural blueprint is not the only source for the cognitive mapping, and the knowledge could be updated based on the reality.

This will also give the ability to the intelligent robot to sweep unlivied (abandoned or uninhabited) buildings, with less human guidance.

1.4. Research Importance

The new enhancement in cognitive mapping helps the security and the emergency forces to overcome any obstacle in the human jobs (jobs that are dangers to human).

A human being in many situations needs to have basic information about the place or building he will surface so that he can navigate inside it, start building knowledge about the spatial that he will face, either acquiring that from a person who knows the place or using the information gained from reading the architectural blueprints of a building, and giving the ability to the intelligent robot to read the architectural blueprints of a building will be a large step in intelligent robot mapping and navigation.

1.5. Study Boundaries

The study covers buildings with one floor or one floor apartment, parts of its field of appliance is non-secured buildings during the emergency situations.

1.6. Study limitations

The study is within the indoor environments without discussing outdoor environments; taking into consideration that there must be only one door that separates between the path that the robot is deployed onto, and the room that the robot must reach as a result of its sweeping job. The study discusses working inside one floor without mentioning multi story buildings and using the stairs, elevators and inside non-secure and emergency situations that have required speed in reaching rooms where the crowded, urban, or dynamic areas will not be taken into consideration.

1.7. Thesis Structure

The thesis includes five chapters, as proceeding in this chapter the introduction was covered. Chapter two is the literature survey for the thesis, showing the related work regarding intelligent robot, intelligent robot mapping, analyzing architectural maps and recognizing shapes in these maps.

Chapter three views the terminology used in the study in different research areas. These methods are used within the work in order to satisfy different objectives.

However, chapter four goes within the implementation of the proposed methodology of the work.

Finally, chapter five contains the conclusion and the future work for the thesis. The thesis also includes references and an appendix.

Chapter Two

Chapter Two

Literature Survey and Related Work

2.1 Overview

In recent years, intelligent robot mapping attracted many researchers in the field of intelligent robotics and artificial intelligence; so the literature survey and related work will be divided into four sections which are; literature survey related to intelligent robot, literature survey related to intelligent robot mapping, literature survey related to cognitive map, and literature survey related to analyzing architectural maps.

2.2 Literature Survey Related to Intelligent Robots

For more than 50 years, the concepts and applications of intelligent robots have been explored by the researchers in the field of Robots and AI. These intelligent robots are often modeled by what we see when we look into a mirror or what we can perform as humans. Over the years, a variety of intelligent robots have been described from game playing robots to novel industrial robots to mobile sensor guided robots. These intelligent robots are remarkable combinations of mechanisms, sensors, computer controls and power sources. Each component, as well as the proper interfaces among and between the components is essential to a successful robust intelligent robot.

2.3 Literature Survey Related to Intelligent Robot Mapping

Sebastian (2002) made a good survey discussing the techniques that have been used on intelligent robot mapping. He mentioned that many concepts have raised, some of it researched concepts that psychologists think how human thinks and builds maps and addressed it with the machines architecture like Shrihari, Stefan, Viet & Roland (2007), while others tried to find a new mapping concepts to perform human jobs far away from human thinking concepts like Benjamin (2000), Emilio & Benjamin, (2003) or Puneet, Stergios & Gaurav (1999). Resent study about that was to build a network of camera systems linked with a centralized unit that give direction for a “Town Robot” by (Hiroshi,

Hiroshi & Katsumi, 1997). However, recent studies assure that making intelligent robots perform human jobs requires using the concept of human intelligence like Chandler (2009), and the way psychologists think that human represent space is Cognitive Mapping, which we are going to build the study based on it.

EL & BC (1985) defines an intelligent robot as “one that responds to changes to its environment through sensors connected to a controller”. Much of the research in robotics has been concerned with vision and tactile sensing. For example, one of the most important considerations in using a robot in a workplace is human safety. A robot equipped with sensory devices that detects the presence of an obstacle or a human worker within its workspace and automatically stops its motion or shuts itself down in order to prevent any harm to itself for the human worker is an important current implementation in most robotics work cells.

To reach the best performance for a robot to be intelligent, human intelligence concepts are now being studied to be implemented on intelligent robots. Most recent research on this field was made by (Owaid, 2009), trying to study human system in term of functions, since the framework model is for a Robot doing a job as human in a specified and specific domain.

2.4 Literature Survey Related to Cognitive Mapping

Many definitions where written for cognitive mapping like those written by Stephanie (1998), and (Yoichiro & Ronald, 2003). However, Tolman (1984) defined it as “inferred the existence of cognitive maps by recording the spatial behavior of a maze-running rat who took a -short cut- to the final destination by running across the top of a maze instead of following a route through It”. But a very recent and accurate definition by Yeap & Jefferies (2001) stated that: “Cognitive mapping, the process by which one’s mental representation of the environment is acquired and manipulated”. Also, it has been enhanced by many researchers gaining advantages from other fields like using fingerprints for places like Adriana & Roland. And some studies have been made to enhance a certain field on cognitive mapping like Shrihari, Stefan, Viet & Roland,

(2007). Others studied cognitive mapping on dynamic environments like (Dirk, Rudolph, Wolfram & Sebastian, 2003).

Using architectural blueprint maps in building cognitive maps can facilitate it for humans to do their jobs better, and to improve building their cognitive maps, a research on this field has discovered that even two years of work in the same building did not build a correct cognitive map like persons who used blueprints for the building from the first time, while the research by Shannon (1988) discusses that: “The series of studies reported in this article examined the cognitive mapping systems of student nurses who had worked in the hospital for various periods of time. After inspecting several different measures, it was concluded that the student nurses had failed to form ‘survey’- type cognitive maps of the building even after traversing it for two years. A control experiment was tested, using naive subjects who were first asked to memorize floor plans of the building. These naive subjects performed significantly better on objective measures of cognitive mapping than did the nurses with two years' experience working at the hospital.”

A survey was made for school students by Tolman (1984) for outdoor environments also showed problems in directions of areas without maps. The study showed that “When students at U.C. San Diego were asked to draw the direction between San Diego and Reno, they incorrectly indicated that San Diego was west of Reno. Indeed, it was surprising to learn that Reno was in fact west of San Diego. After all, California is on the western coast of the United States, and Reno is far inland, in Nevada. A glance at a map reveals that the coast of California, far from running north-south, in fact, cuts eastward as it cuts southward. Stevens and Coupe attributed their findings to hierarchical representations of space. People do not remember the absolute locations of cities. Instead, they remember the states cities are part of, and the relative locations of the states. Then they infer the relative locations of cities from the locations of their superset states”.

Another related research for Benjamin & Yung (1993) shows that the huge amount of time needed by an intelligent robot in to explore a new building without human guide or supporting maps and move from room to room inside the building.

However Jochen, Chee, Wai (2007) discussed that: “When animals (including humans) first explore a new environment, what they remember is fragmentary knowledge about the places visited. Yet, they have to use such fragmentary knowledge to find their way home. Humans naturally use more powerful heuristics while lower animals have shown to develop a variety of methods that tend to utilize two key pieces of information, namely distance and orientation information.”

Depending on maps during robot mapping field, some studies have done using geographical maps during mapping, like (Motilal & Kurt, 2006), and others tried to put a theory to topological maps like (Emilio & Benjamin, 2003) and (Schönberg, Ojala, Suomela, Torpo & Halme, 1995).

The knowledge based-system witnessed a huge research area and a good paper ‘Frame Model for Intelligent Robot as Knowledge-Based System’ Owaid (2009), describes its architecture. This paper was as an inspiration for using the knowledge-based system architecture in the proposed project.

2.5 Literature Survey Related to Analyzing Architectural Maps

An analysis of architectural drawings has been discussed by few researchers. The architect’s point of view has been first discussed by Koutamanis (1989, 1990, 1992, 1995), who takes into account the dual nature of the components of an architectural drawing: the building elements themselves and the spaces which they delimitate. According to him, only six building elements, with all possible reflections and 90° rotations, are necessary in a design drawing. He therefore recognizes these configurations by simple template matching. When all these elements are found, the spaces are delineated in the building by following the recognized elements and localizing successive corners at 90° and 270°. Although he did not, as far as we know, tackle real-world, scanned drawings, his ideas are a good foundation for spatial analysis methods.

Recently, Llad’os (1996) has designed a system for analysis of hand-drawn sketches. Structural knowledge about the drawing is described using attributed graphs; thus, models are available for usual patterns such as beds, tables, bathroom furniture,

etc... These model graphs are stored in a library. The analysis itself is then performed by looking for a subgraph isomorphism between the segmented drawing and the library of models. In addition, the Hough transform is used to recognize elements which are not easily described by a symbol model, such as walls or hatching patterns. The results are interesting for the data used by the authors, but the variability of representations in architecture is not taken into account yet.

Aoki et al. have a similar system Aoki, Shio, Arai & Odaka (1996). To reduce vectorization errors, they work with very strict conditions (line thickness, grid location, etc.) for the input data. Construction elements are recognized by a combination of two main primitives: line elements (walls and windows) and closed region elements (doors and closets).

Ryall & Shieber (1995) have proposed a low-level technique working directly on the bitmap, by using proximity metric and delineating partially or fully bounded regions. This can be quite interesting to localize rooms, but a high-level analysis leading to 3D reconstruction is lacking, and the method is only semi-automatic.

In the field of finding best routes after extracting information from maps, a good thesis by (Alex et al. 2009) discussed many methods to help students move from one building to another inside a university with the help of xml based map for the university.

2.5.1 Recognizing Maps' Objects

As the architectural blueprint maps contain many objects and shapes, a plenty of algorithms were raised to recognize these shapes.

Sundar (2003) and Ashbrook & Thacker (1998) discussed that shape matching algorithms have been used to detect certain shapes using skeletonization algorithms. A good research in this field was made by (Pavel & Carlos, 2000), who have declared different methods used for skeletonization. Some also used arc segmentation like Liu & Dov (1998) who have defined arc segmentation as “an open problem in the area of line drawing interpretation”. Coeurjolly, Gérard, Reveillés & Tougne (2004) and João,

Marco, Daniel & António (2005) also used the same method. In the same area, some have used edge based features to detect shapes like (Mikolajczyk, Zisserman & Schmid, 2003). Parameter shape recognition has been used to detect shapes in images as (Jay & Jonathan, 2002).

Chapter Three

Chapter Three

Terminology

3.1 Overview

This chapter includes the declaration of the methods and terms that are used in the algorithms and how these algorithms are used.

3.2 Architectural Blueprints

The term “Architectural Blueprints” is part of the general term in the construction industry which is “blueprint”, where the blueprint is defined by Sam (2009) as follows: “blueprint generally refers to a composite of several plans, such as the foundation plan, the floor plan, elevations, sections, mechanical plans and details, etc., that are assembled into an organized set of drawings to transmit as much information about a project as can be placed on paper in two-dimensional views.”

Architectural Blueprint map or what sometimes is called floor plans is the foundation plan of a building, or the map that semi-presents the vertical top cross-view of buildings in the real world.

This thesis suggests the use of a map for building cognitive maps of knowledge-based intelligent robot. And since the purpose of this robot is currently indoor environments, then the best, reliable and accurate maps in this field are the Architectural Blueprint maps.

3.2.1 Architectural Blueprint Maps Drafting Standards

Architectural Blueprint Maps have standards to be followed, so that the Institute of Architects worldwide does not approve the drawings that have a leak in following these standards.

Sam (2009) has discussed these standards in his book: “The American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) have adopted drafting standards that are voluntarily accepted and widely used throughout the world. These standards incorporate and complement other architectural/engineering standards developed and accepted by professional organizations such as the American Institute of Architects (AIA), the American Society of Mechanical Engineers (ASME) and others”.

Within these standards are symbols’ standards, which are the symbols that the architects should follow while presenting objects (doors, windows ...) in the Architectural Blueprint Maps. Figure 3-1 shows some of these symbols.

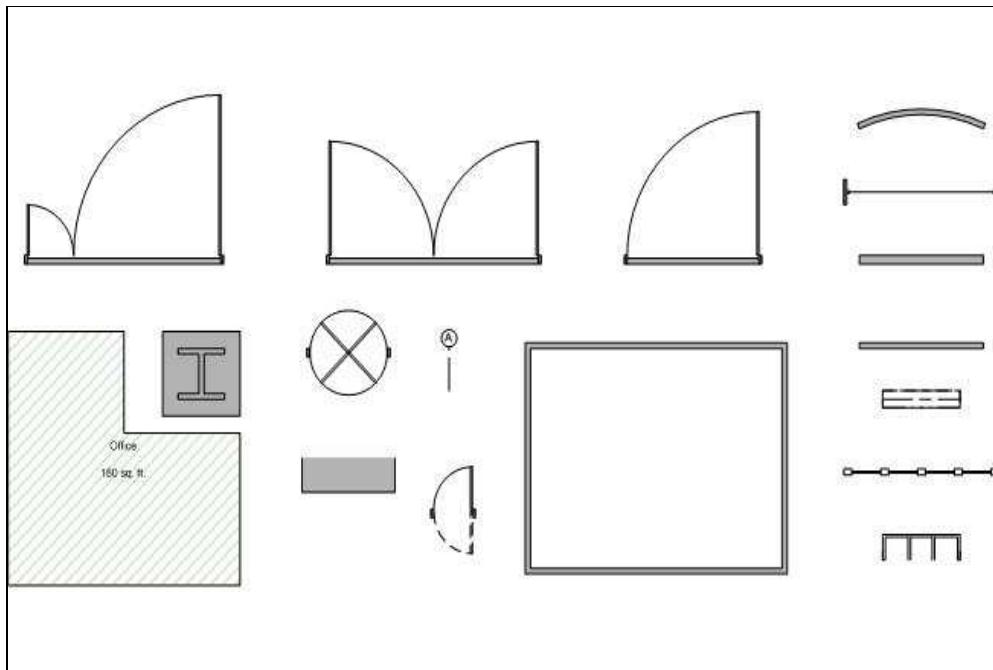


Figure 3-1 : Architectural Blueprint Maps’ Symbols

These standards were used in this thesis for reading the architectural blueprints maps. Hence, any non-approved drawings by the Institute of Architects worldwide are not supported by this work of algorithm.

3.3 Pixel & Pixel Neighbors

Rudolf (1999) defines Pixel as “a spatial resolution element. It is the smallest distinguishable and resolvable area in an image”.

In this work, Architectural blueprints analyzing algorithms used the pixel for visioning, simplifying and analyzing the Architectural blueprints maps. Where the maps images, after being received by one of the communication interfaces, should be converted to RGB image, where each pixel in the image has a set of color ranges starting from perfect black that its hexadecimal value is (000000), to perfect white that its hexadecimal value is (FFFFFF), where the first two hexadecimals present the red value of the pixel, the second two hexadecimals present the green value of the pixel, and the third two hexadecimals present the green value of the pixel.

The neighbors of the pixel are the eight pixels surrounding the pixel from its eight directions, as the Figure 3-2 shows.

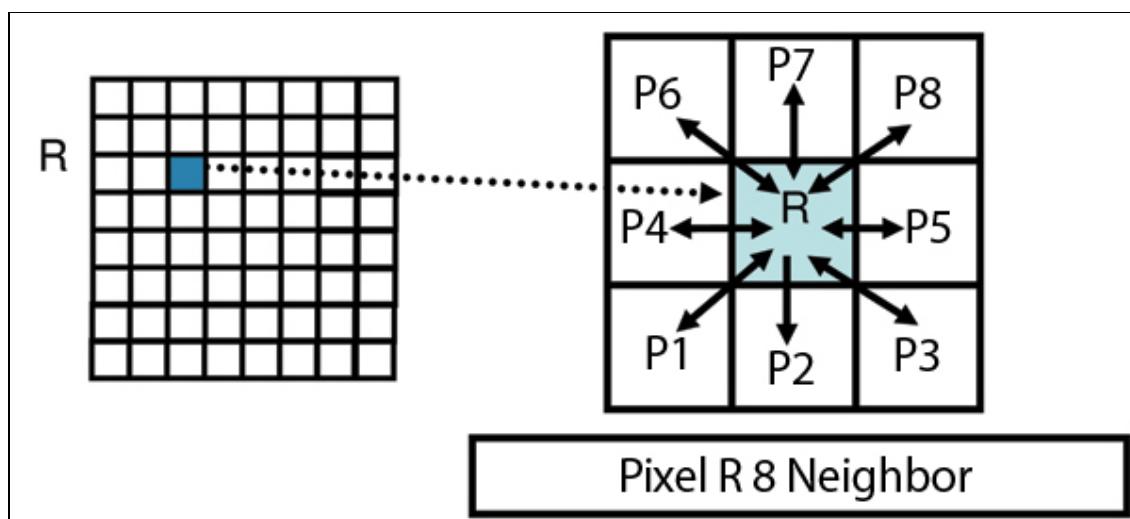


Figure 3-2 : Pixel's Neighbors

In this thesis, when the term “P left pixel” is mentioned then it is the fifth neighbor pixel of pixel P , “P right pixel” is the fourth neighbor pixel of pixel P , “P top pixel” is the seventh neighbor pixel of pixel P , “P bottom pixel” is the second neighbor

pixel of pixel P. However, the four other pixels (first, third, sixth and eighth neighbor pixels) are mentioned directly.

In the purpose of declaring the colors in the figures, symbols will be used instead, it will be declared when the usage of symbols is necessary.

3.4 Masks & Scanned Pixels

The general definition of a mask as defined by Rudolf (1999) is: “A frame mounted in front of a television picture tube to limit the viewing area of the screen”.

In vision, mask is used to store specific pixels for an image in the short memory, where it helps during calculations and algorithms of vision.

Before being able to use the mask, it should first have a preserved area in the memory, at the beginning, the size of the preserved area is specified by the program that initialized the mask, considering the type of colors (RGB, Bitmap ...) used by the original image, then it can be extended while trying to add more pixels to the mask if the current preserved area is fully loaded. Then the mask can receive pixels with the pixels parameters, like the original position of the pixel and its color value. For example, in Figure 3-3, if blue (boxes with o symbols) pixels are to be masked, then mask is initiated in the short memory with a specific name, the type of colors used by the left image, after that blue pixels are sent to the initiated mask, sending each pixel along with its x and y coordinates and its color.

	0	1	2	3	4	5	6	7	8	9	
0	o	o	o	o	x	x	x	x	x	x	0
1				o	o	o	x	x	x	x	1
2				o	x	x	x	x	x	x	2
3	o	o	o	x	x	x	x	x	x	x	3

Original Image Pixels

	0	1	2	3	4	5	6	7	8	9	
0	o	o	o	o							0
1				o	o	o					1
2					o						2
3											3

Specific pixels in mask

Figure 3-3 : Masking

Tagging pixels as scanning pixels can be done in many ways; putting a copy of the pixel in an area with its coordinates or putting the scanned pixels in a mask.

Algorithms of this thesis use multiple masks. Whenever terms like “scanned pixels” or “sent to mask x” are used, then a mask in the memory is initiated. Moreover, when terms like “check mask pixels” or “if pixel is within the mask x ...” are mentioned, then a mask is retrieved from the short memory, then it is checked if a specific pixel does exists within it.

3.5 Pattern Recognition & Matching

According to Sergios & Konstantinos (2006) pattern recognition is: “the scientific discipline whose goal is the classification of these objects into a number of categories or classes” they also add: “Machine vision is an area in which pattern recognition is of importance. A machine vision system captures images via a camera and analyzes them to produce descriptions of what is imaged.”

Pattern recognition could be assuring that an image contains specific shapes or objects. For example, if a car to be recognized using patterns, at least one door, one window and one wheel should be found on the image. The problem occurs if the previous items are in the image but it is not a car, because the image might be for a scrap items. So to increase the accuracy, the position of those patterns (shapes) must be more accurate.

If we are searching for more than one set of patterns, pattern matching is used. In the previous example, if the car was not found, then a plane should be found, with specific patterns (wings, tail ...), then it is pattern matching.

Patterns for both pattern recognition and matching could be pixels, searching for a set of pixels in the image. If this set does exist in the image, then recognition (or matching) is done. Matching in the set of pixels we are searching for may not be 100% accurate, sometimes the image is rotated, or little mismatching because of noise problems, here the demand of decreasing the accuracy appears. Pixel patterns are used in thesis to give more accuracy.

An example of pattern recognition is as follows: in Figure 3-4, assuming that the colored boxes are pixels, then the image is similar the one that is on our mind if it contain red pixel (boxes with x symbols) in position (1,1), black pixels in (2,3), (5,3), (6,3), blue pixels (boxes with o symbols) in (2,1), (2,2), (4,4), and does not contain a yellow pixel (boxes with y symbols) in (5,1). Most of the patterns match the ones in the image except one; the image is recognized in terms of accuracy.

	1	2	3	4	5	6
1	x			y		
2	o	o	o	o	o	o
3						
4	o	o	o	o	o	o
5	y				y	
6	o	o	o	x	o	o

Figure 3-4 : Pixels Matching

3.6 Path Definition

As Rudolf (1999) said “In navigation, an imaginary line connecting a series of points in space and constituting a proposed or traveled route” is called a path.

In building the path is any space that a navigator, either a human or an intelligent robot, could navigate inside (move from point to point). The space could be a corridor, room ... etc.

3.7 Main Path

In this thesis, the main path is a set of corridors connected with each others, where one of these corridors is the intelligent robot start point exists, translating this definition to the architectural blueprint maps, the main path is the shape the red label (start point) is positioned. Figure 3-5 shows the main path (lines with x symbols).

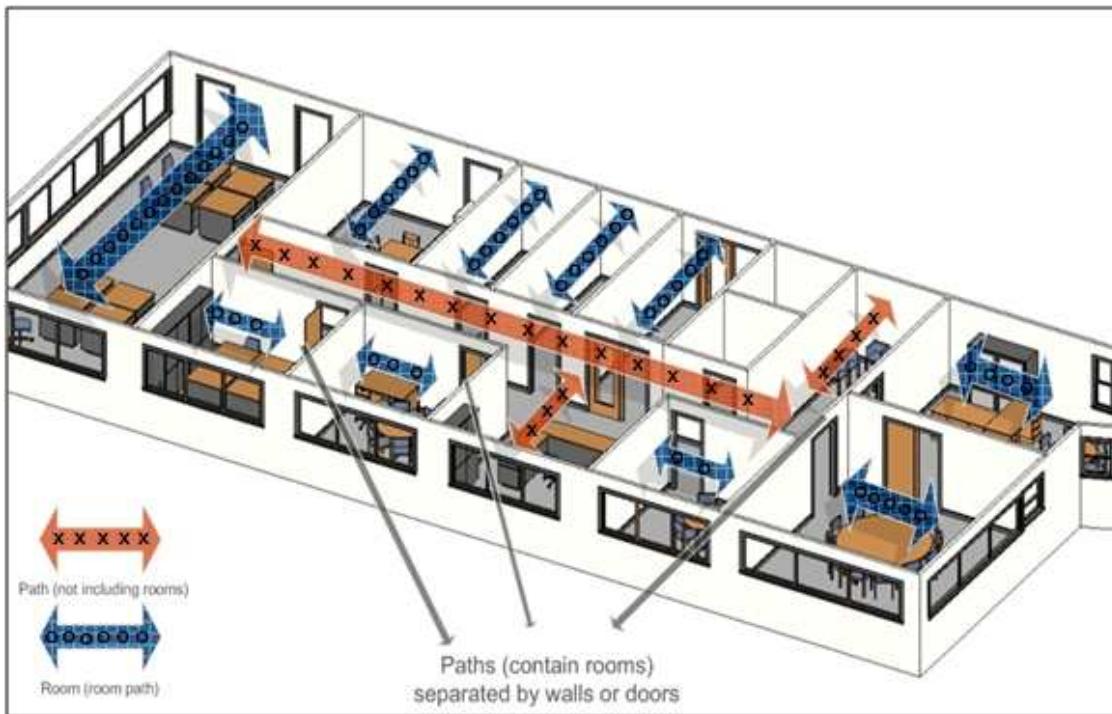


Figure 3-5 : Main Path in a Building

3.8 Rooms

In buildings any closed space that has a floor, ceiling, walls, at least one door and might by windows, but not the main path, is considered as a room. So for example, at home; the kitchen, hall, child or parents rooms, bathrooms ... etc, are all categorized under the name of room. In this thesis, which deals with architectural blueprint maps, rooms are any white shapes sized more than a specific value (the value does not matter here, because it is mentioned for the reader just to give him the knowledge about the

architectural blueprints), that is not the main path. So path is mentioned as a room, if it is not the main path.

Rooms could also be counted as paths, but for the classification purpose, it is considered as a room. Figure 3-5 shows rooms (lines with o symbols).

3.9 Turnings

The path could contain turnings, a turning is a result of two corridors or more meeting each other in a point, this point is the turning. The turning on the real world is a crossroad where the navigator needs to make a choice to continue his way (left, right, go ahead).

In the map, the turning is two straight corridors meeting each other in a point within the main path; turnings cause specific shapes that are used in the thesis algorithms to recognize them, after discarding the corridors.

The degree of the turning is specified by the angle between the two corridors. Since not all turnings have the same degree, describing the turning in the goal of building the cognitive map contains a simple specification for the degree, (hard left, semi straight, normal right ... etc).

3.10 Vertical and Horizontal Corridors

A straight corridor in the real world might be vertical or horizontal line in the architectural blueprints. Sometimes, these lines are not so vertical nor so horizontal.

In this thesis, semi vertical lines (corridors) are considered as vertical corridors. However, semi horizontal lines (corridors) are considered as horizontal corridors. As shown in Figure 3-6, a certain degree separates vertical from horizontal corridors.

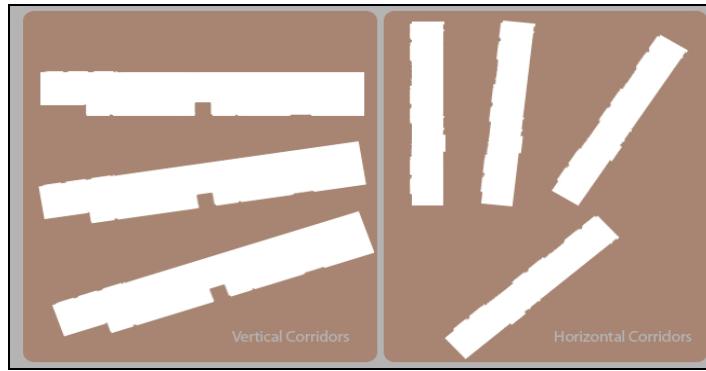


Figure 3-6 : Horizontal (right) & Vertical (left) Corridors

3.11 Start and end points

The point that the robot will be placed on the building is called the start point. The end point is the target that the robot must reach as a result of its job.

It is assumed in the thesis that start point is on the main path with only one door which separates it from the end point, where it is a room or another path. While current work deals with surfing one floor on a building, then these two points should be on the same floor.

As an example of the start and end points, let's assume that you have entered a university building, intending to reach the registration office in that building, the first step you make inside that building is the start point, and the registration office is the end point.

3.12 Labeling Maps

Many methods can be used for labeling an image or map, marking certain Figure, text or color, on an area on the image. For example, if we want to label a room on a map as the kitchen, then as seen in Figure 3-7, “kitchen” text might be printed on the map on the kitchen room, kitchen sign placed on the map in the kitchen room, or a colored shape centered on the kitchen room (box with k symbol), as agreed before with the person reading the map, that colored shape (box with k symbol) is the kitchen.

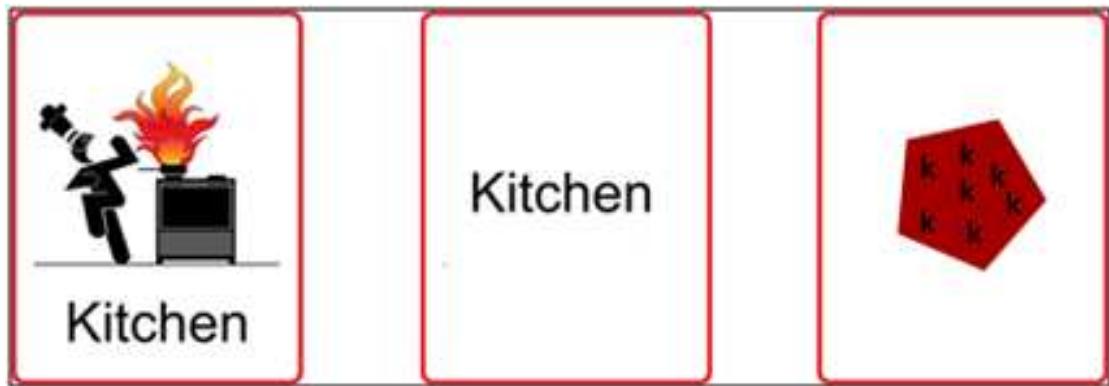


Figure 3-7 : Different Labeling Techniques

In this thesis, red and blue shapes were used to label respectively start and end points on the map. Labeling person must be accurate when specifying the position of labels, as labels must be in the same place in the reality. Also labels must be clear, using a computer program for labeling or putting previous prepared stickers, and avoiding the use hand drawings.

3.13 Medial Axis (skeleton)

Skeleton or medial axis on the image processing field is finding the core pixels of 2D shapes, as Figure 3-8, or 3D shapes, as Figure 3-9, where this core (Skeleton) can be the identifier of that shape. This matches the reality, which is that finding the skeleton of a dead creature can help finding the species this creature is related to.

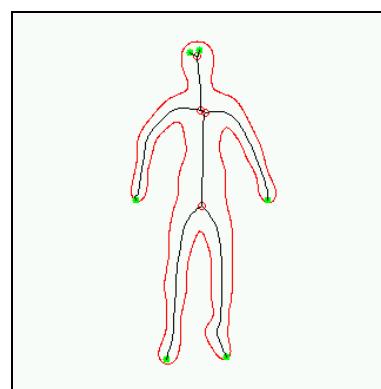


Figure 3-8 : 2d Skeleton Example (Sundar et al. 2003)

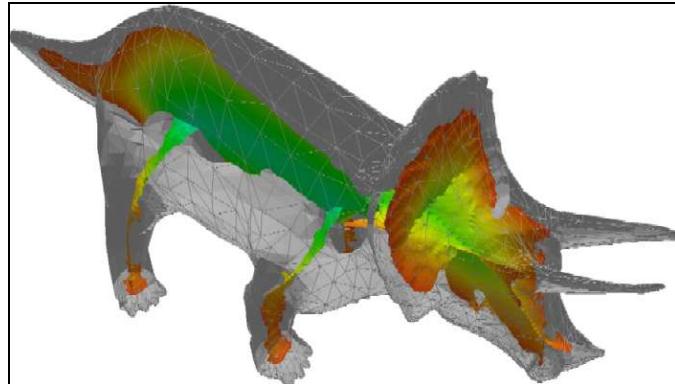


Figure 3-9 : 3d Skeleton Example (Sundar et al. 2003)

General steps as described by Sundar et al. (2003) “The steps in this process include: obtaining a volume, computing a set of skeletal nodes, connecting the nodes into a graph, and then indexing into a database.”

In this thesis, medial axis is used to find the graph that presents the path that the intelligent robot can take during its navigation.

3.14 Cognitive Mapping

The Intelligent Robot should have the capabilities that is declared by Owaied (2009), which he says: “behaving like a human being, smart, problem solver of unstructured and complex problems as human does, understands languages, learner, and is able to reason and analyze data and information, and so on”. However, human does work as a knowledge-based system as described earlier. It is constructed as shown in the Figure 3-10 from: the communication with the environment unit, human inference engine, and long term memory.

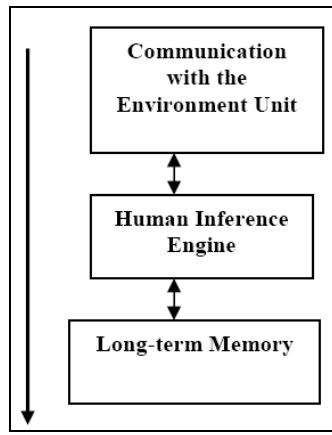


Figure 3-10: Functional Model of Human System Owaied (2009)

On the other hand, robot knowledge based system is built as Figure 3-11 shows, where as described Owaied (2009) by “the left side arrow presents the direction of implementation of knowledge-based system as computer-based software and the most important part is the knowledge base and the inference engine and user interface will be implemented according to the knowledge base scheme used”.

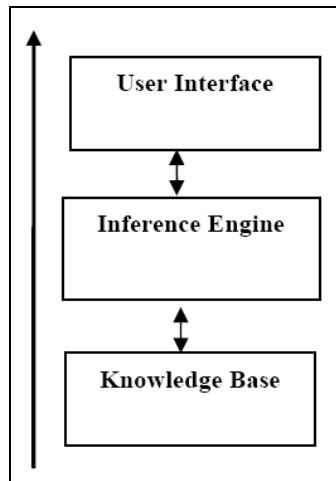


Figure 3-11 : Intelligent Robot Knowledge Based System Owaied (2009)

A detailed view of the architecture of Intelligent Robot knowledge based system is shown in Figure 3-12.

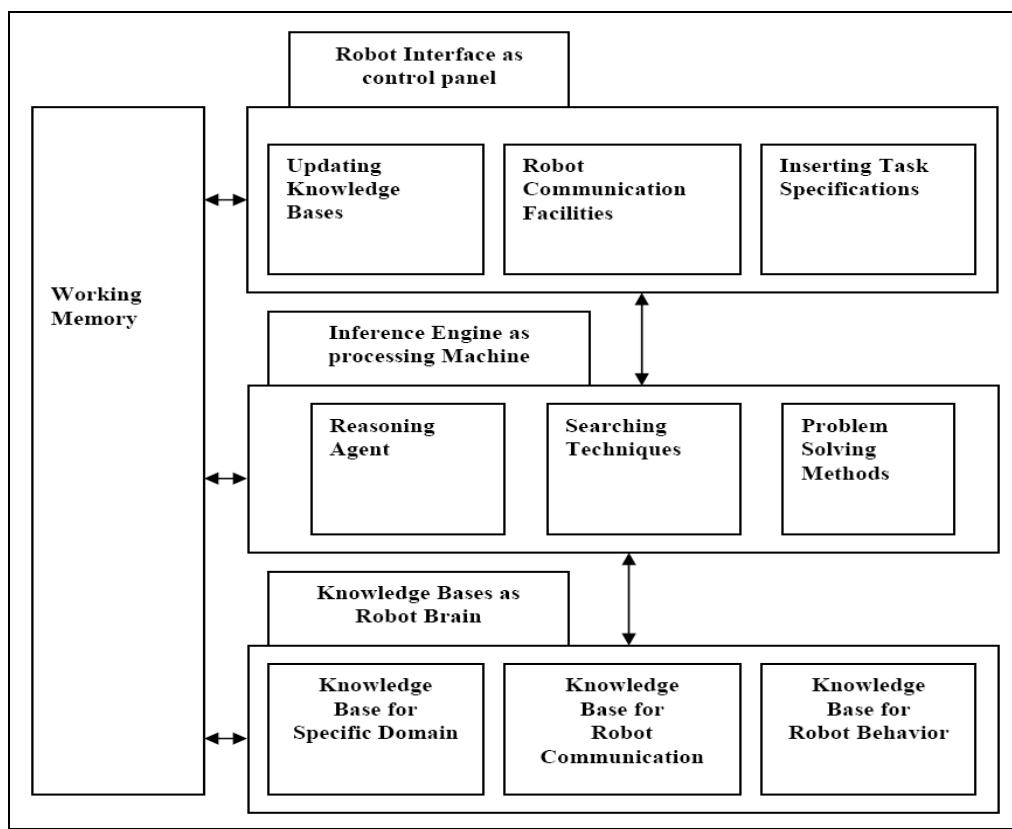


Figure 3-12 : The Architecture of Intelligent Robot as Knowledge-Based System Owaied (2009)

As shown in Figure 3-13, the intelligent robot takes the image using the communication facility in the robot interface, which is a part of the knowledge based system. Then it updates the knowledge in the knowledge base by producing a cognitive map to be stored in the knowledge base for the robot behavior, to help the intelligent robot on navigating and mapping in real time. These steps are declared in detail in the next chapter.

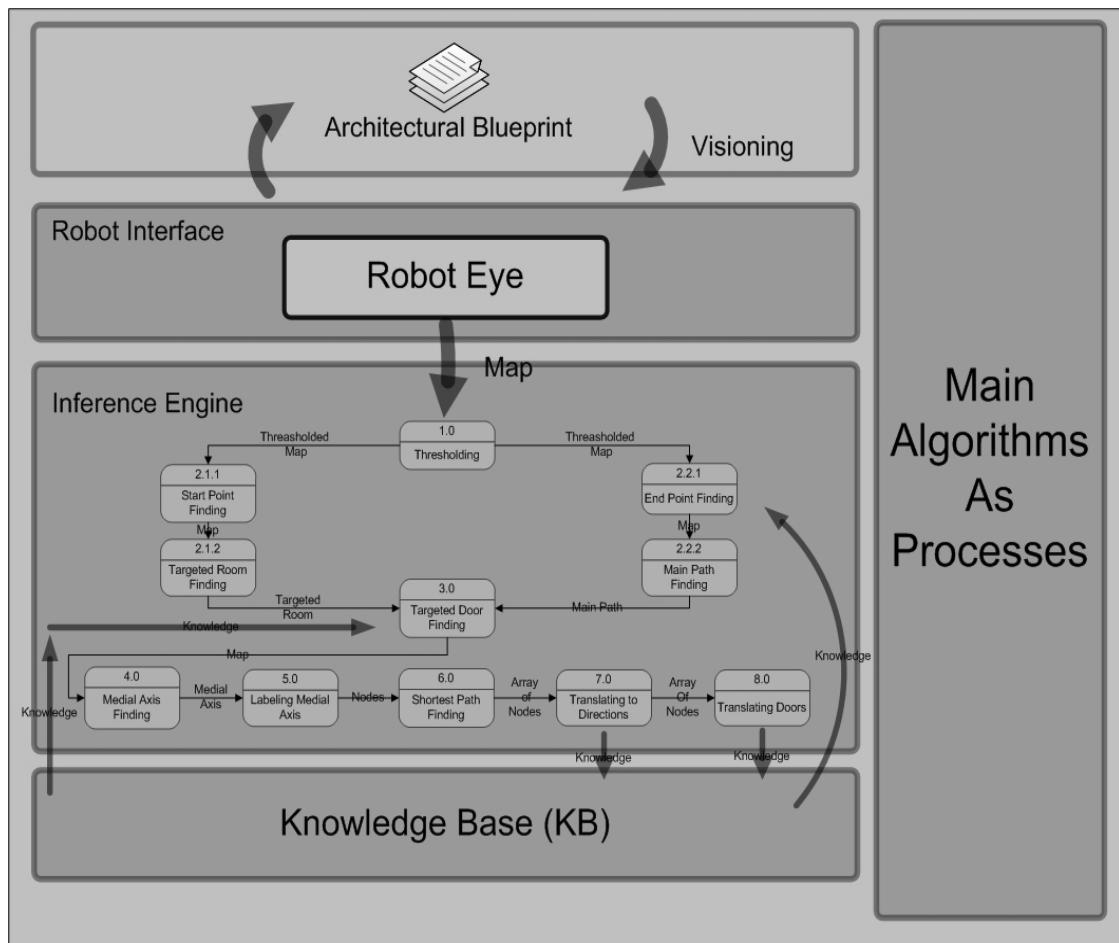


Figure 3-13 : Main Algorithms as Processes as a Knowledge-Based System

Chapter Four

Chapter Four

The Implementation of Proposed Methodology

4.1 Overview

This chapter presents the methodology and implementation of creating the cognitive map using a blueprint map. The methodology consists of many algorithms; these algorithms are used for gaining knowledge from the architectural blueprint maps so as to build the cognitive map. They will be used by the inference engine of the knowledge based-system of the intelligent robot. These algorithms will be explained in the next section.

4.2 The Algorithms

Before proceeding to the algorithms declaration, the algorithms are listed below, then the structure of the algorithms will be defined to view both the input and output of each algorithm and its position in the whole program.

The algorithms respectively are:

- 1- **Threshold algorithm:** filters and simplifies the architectural blueprint map (changing the map from all color ranges to four colors only, which will be mentioned in the algorithm).
- 2- **Flood Fill Algorithm:** masks pixels with the same color in a closed region.
- 3- **Start Point Finding:** finds the start point label in the map.
- 4- **End Point Finding:** finds the end point label in the map.
- 5- **Main Path Finding:** finds the main path (the path to be taken to navigate from the start point to the end point).

- 6- **Targeted Room Finding:** finds the room that must be reached as a result of the navigation job to be done by the intelligent robot.
- 7- **Similarity Algorithm:** searches the map for a certain pattern to find a certain objects on it (here it is used to find the doors in the map).
- 8- **Medial Axis:** finds the center pixels of the corridors and turnings in the map and connects them together to make the medial axis of the main path.
- 9- **Labeling the Medial-Axis:** converts the set of the nodes of the turnings and the start and end points, together and finds the distance between each of the two neighbors together.
- 10- **Finding the Shortest Path:** finds the shortest path array starting from the start point and ending with the end point.
- 11- **Translating Path to Directions:** translating the shortest path to directions, finding the angle of each three nodes together, and then converting these angels to a direction to be stored in the knowledge base.
- 12- **Cognitive Mapping Storing Algorithm:** builds the cognitive map and stores each direction in a new node (frame) that establishes the cognitive map as a network of nodes (graph).

To declare the whole process, the intelligent robot takes the blueprint map as an image and, simplifies it using the threshold algorithm. Then it finds the start and end points based on their labels on the map. After that, it masks the main path and the target room based on the start and end points using the flood fill algorithm. Then it finds all the doors in the map including the target room door. Then the usage of the medial axis algorithm comes, where it finds the medial axis of the path, then labels it and connects each node with its neighbor node to be sent to the shortest path algorithm, which returns an array of the shortest path nodes. Then the intelligent robot uses a translating path to directions algorithm, where each turn is translated to a direction and sent to the cognitive mapping algorithm. Meanwhile, the cognitive map is initiated and updated with each new

direction translated. Finally, the cognitive map is committed and stored in the knowledge base. Figure 4-1 shows the data flow diagram for the whole program.

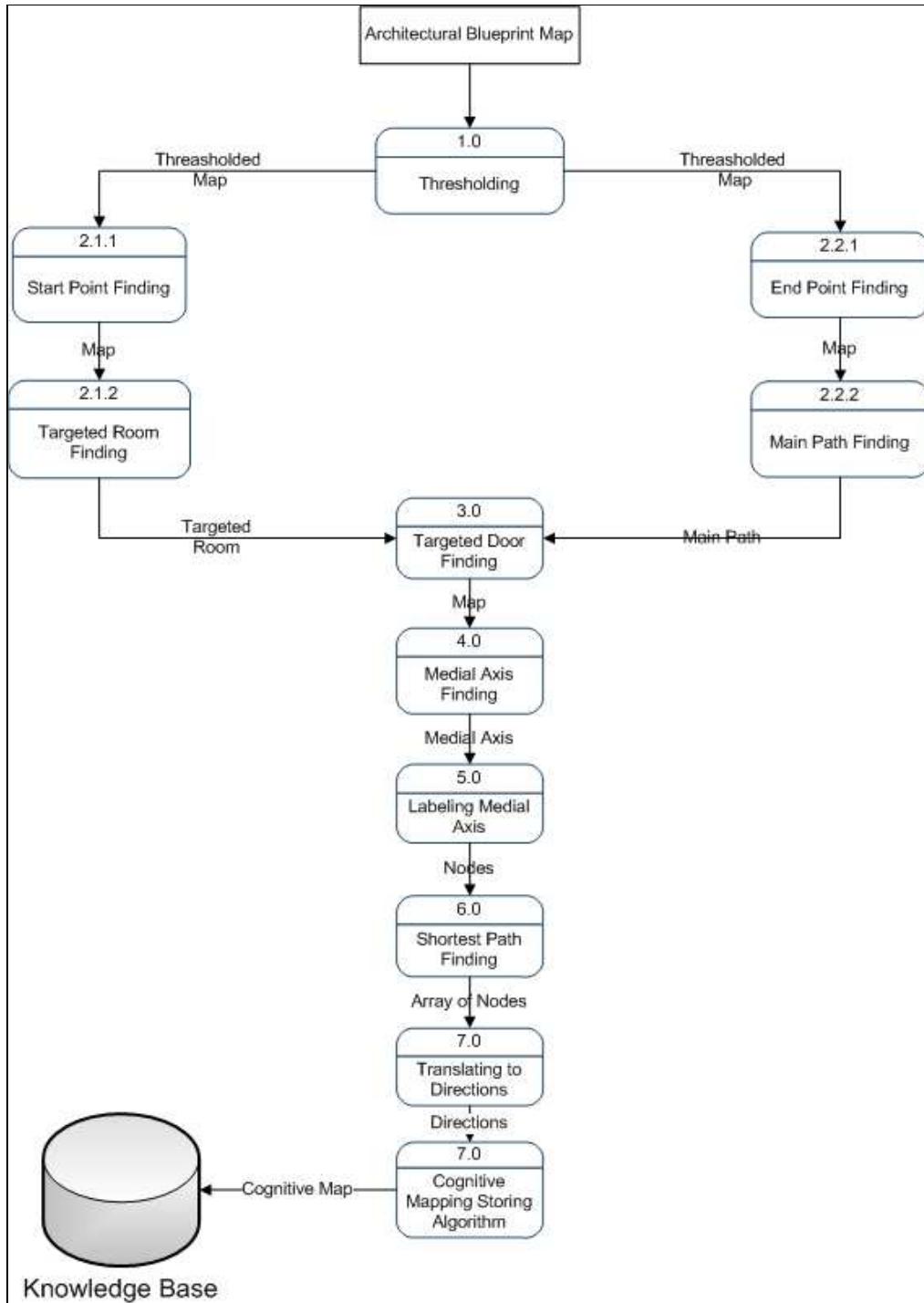


Figure 4-1: The Data Flow Diagram for the Algorithms inside the Program.

For the purpose of demonstration, testing and declaring the algorithms, Architectural blueprint map for offices building at Regus Whitefields Centre, Bangalore, India is used. Figure 4-2 shows the building itself.



Figure 4-2 : Offices Building at Regus Whitefields Centre, Bangalore, India (India Property, 2008).

Figure 4-3 presents the labeling architectural blueprint process for the center floor of the offices building at Regus Whitefields Centre, Bangalore, India. Some algorithms were proved using MATLAB R2009b.

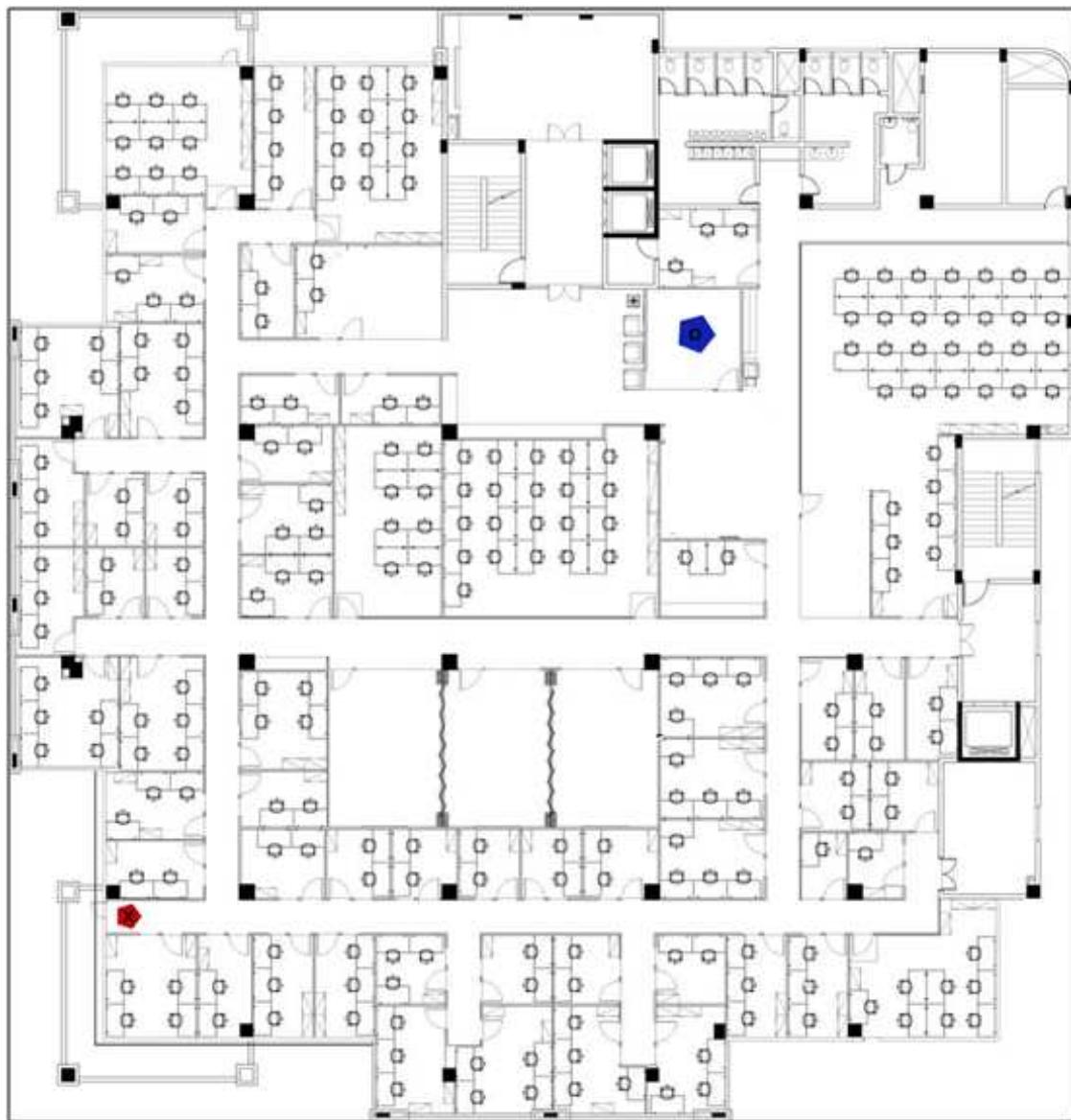


Figure 4-3 : Labeled Architectural Blueprint for the Center Floor at Regus Whitefields Centre, Bangalore, India.

All algorithms will be respectively declared below, with an example on it and its pseudocode:

1- Threshold algorithm

Architectural blueprint maps most likely have a lot of noise Figure 4-4, especially if they are scanned from a hard copy or old architectural maps, gray scale colors. Hence, they need to have a threshold to certain colors in order to decrease and utilize the processing time.

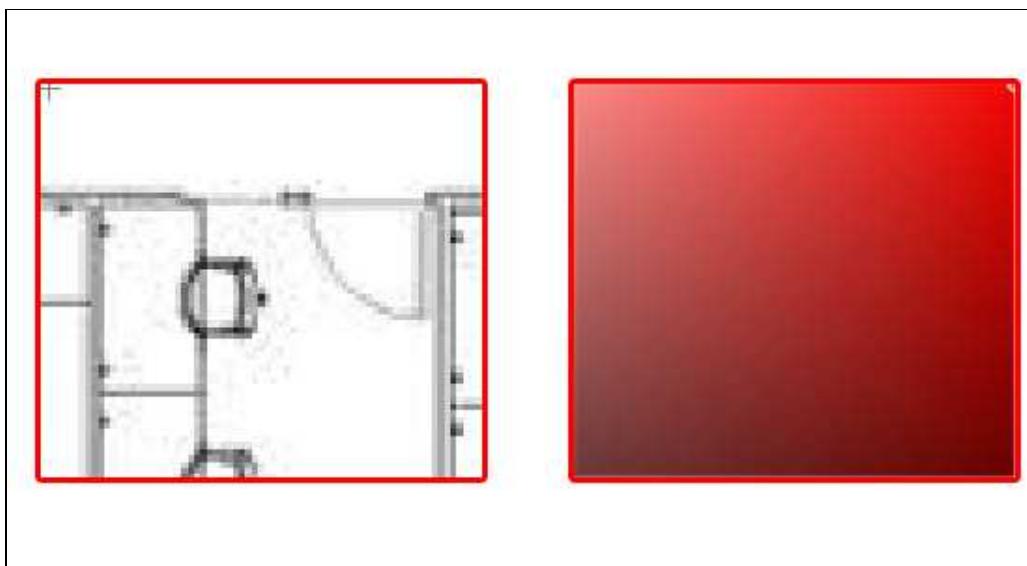


Figure 4-4 : Scanned Blueprints have a lot of Noise.

While the need of a simple, dedicated threshold algorithm is necessary to simplify working on the architectural map, a simple enhanced threshold algorithm has been developed in this work, to match color ranges (black, white, red, blue) expected by the robot inference engine, where black color presents the elements inside the map (walls, doors ...), the white pixels present the paths available, the red pixels present the start point and the blue pixels present the end point.

As shown in Figure 4-5, starting from the first pixel in the Architectural blueprint map (0,0), threshold algorithm checks if the pixel color sets in the range of the white color, if so, it converts the pixel to perfect white (FFFFFF). If not it checks the pixel

whether color sets in the range of the black. If yes, it converts the pixel to perfect black (000000), if not, it checks if the pixels color sets in the range of the red color. If so, it converts the pixel to perfect red (FF0000) (shape with x symbol), if not, it checks if the pixels color sets in the range of the blue. If so, it converts the pixel to perfect blue (0000FF) (shape with o symbol). If none of the above ranges matches the color, then it converts the color of the pixel to perfect black (000000). Since all other colors will present elements in the map and tell the robot that these are elements using the algorithms, it must be changed to black pixels (robot considers black pixels as elements).

The mentioned color ranges sets on the knowledge-base of the robot based on RGB starting from perfect black (000000), ending with perfect white (FFFFFF).

Previous step is repeated for each pixel in the Architectural blueprint map, narrowing the colors of it to the four colors mentioned above, Figure 4-6 shows the output after applying the code on the map using MATLAB R2009b.

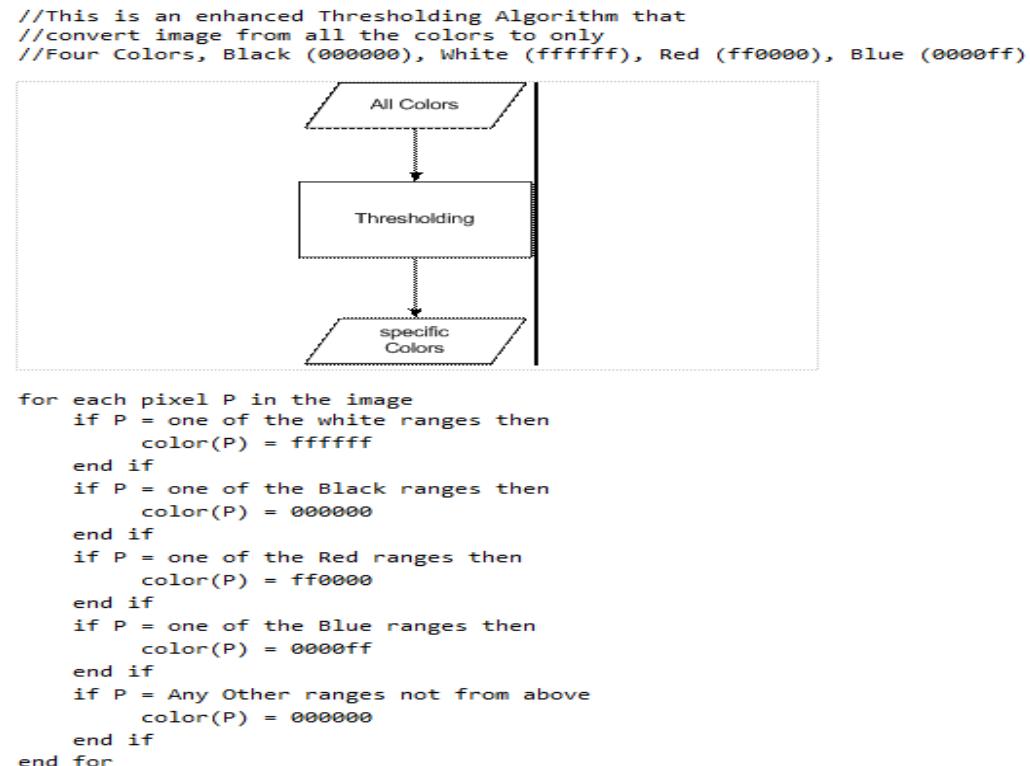


Figure 4-5 : Thresholding Pseudo code



Figure 4-6 : Blueprint after Thresholding, Using MATLAB R2009b.

2- Flood Fill Algorithm

Flood Fill, is an algorithm that is commonly used by pictures editors. It is mostly used in selecting a set of connected pixels with the same color within a bordered figure, and the first picture editor starts using it was Adobe Corporation, the pseudo code mentioned here is the code for 8-Way Recursive Method from Lode (2004).

“The 8-Way Recursive Method works like this: at the start position, you "plant a seed". Each seed gives the pixel at its position the new color, and then plants a new seed at its 8 neighbors. Each of these new seeds will draw a pixel again and plant even more seeds, but only if fulfills the following conditions:

- The pixel is inside the screen: edges of the screen also count as edges of the area to be filled.
- The pixel has oldColor: if it hasn't, it's either a border of the area we're filling, or a pixel that has already been given the newColor.
- The pixel doesn't have a newColor: this condition is only needed if oldColor is the same as oldColor. Otherwise, it'll keep running forever since newly drawn pixels will again have oldColor and thus, seeds will be planted again and again forever.

The function will keep calling itself more and more until eventually all pixels are filled”.

If the newColor is not spacific, then it is the same as oldColor.

The aim of this algorithm is to put the selection in a non symmetry matrix, so that each pixel in the mask represents a pixel in the Figure with the same coordinates. The non symmetry matrix is called a mask.

In the algorithm, masks are used as a tagging method side by side as a sub image within the architectural blueprint map, while working with the sub image will not reflect the pixel in the main map. Figure 4-7 shows the algorithm Pseudocode.

```

//Recursive 8-way floodfill, crashes if recursion stack is full
function floodFill8(x, y, int newColor, int oldColor)
    if x >= 0 && x < w && y >= 0 && y < h && screenBuffer[x][y] == oldColor
        && screenBuffer[x][y] != newColor then
            screenBuffer[x][y] = newColor //set color before starting recursion!
            floodFill8(x + 1, y,      newColor, oldColor)
            floodFill8(x - 1, y,      newColor, oldColor)
            floodFill8(x,      y + 1, newColor, oldColor)
            floodFill8(x,      y - 1, newColor, oldColor)
            floodFill8(x + 1, y + 1, newColor, oldColor)
            floodFill8(x - 1, y - 1, newColor, oldColor)
            floodFill8(x - 1, y + 1, newColor, oldColor)
            floodFill8(x + 1, y - 1, newColor, oldColor)
    end if
    return screenBuffer
end function

```

Figure 4-7 : Flood Fill Algorithm Pseudo code

3- Start Point Finding

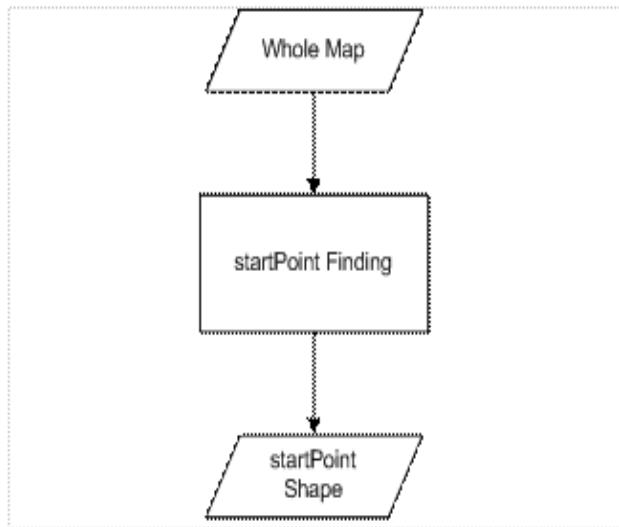
As mentioned on the scenario, intelligent robot owners should specify start and end points (Figure 4-10) to the intelligent robot, start point should be a red label (shape with x symbol) that represents the intelligent robot start position in the targeted building.

To recognize a start point, an algorithm was made (As shown in Figure 4-8) to find the only red pixels set in the architectural blue print map. Thus, starting from the first pixel on the map if the pixel is red (FF0000) (shape with x symbol), the algorithm put it on the startPointMask which is a mask that represents the set of all red pixels in the map (the red label (shape with x symbol)). After that, the red pixel (shape with x symbol) in the map is converted to a white pixel (FFFFFF). This step decreases later calculations.

```

//startPoint Finding
//search red pixels where pixel eight neighbors are red
//this is the start point startPoint

```



```

for each pixel P in the image
    if color(P) == 0000ff then
        color(p) = ffffff
        startPointMask[i] = P
    end if
end for

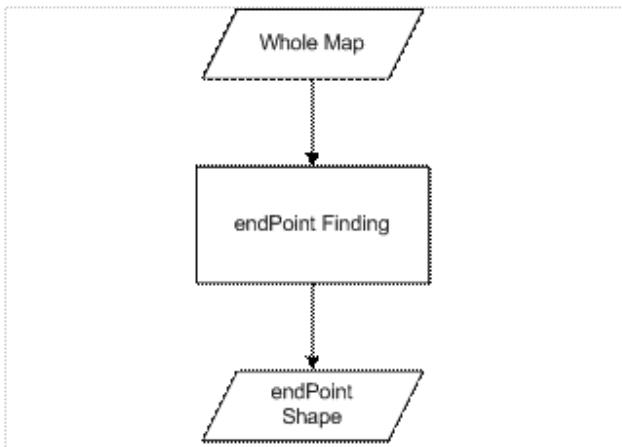
```

Figure 4-8 : StartPoint Finding Algorithm Pseudo code

4- End Point Finding

This algorithm is pretty similar to the previous one, which was used to recognize the end point, an algorithm was made (As shown in Figure 4-9) to find the only blue pixels set in the architectural blue print map. Thus, starting from the first pixel on the map if the pixel is blue (0000FF) (shape with o symbol), the algorithm put it on endPointMask which is a mask that represents the set of all blue pixels in the map (the blue label (shape with o symbol)). After that, the blue pixel in the map is converted to a white pixel (FFFFFF). This step decreases later calculations.

```
//endPoint Finding
```



```
for each pixel P in the image  
    if color(P) == 0000ff then  
        color(p) = ffffff  
        endPointMask[i] = P  
    end if  
end for
```

Figure 0-9 : endpoint Algorithm Pseudo code

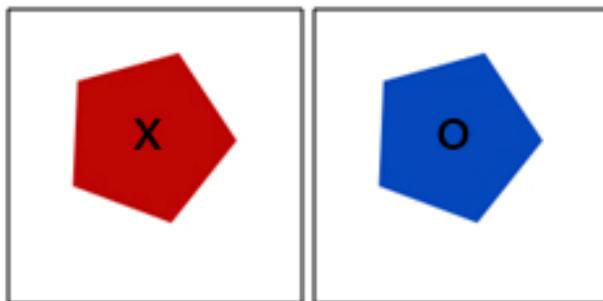


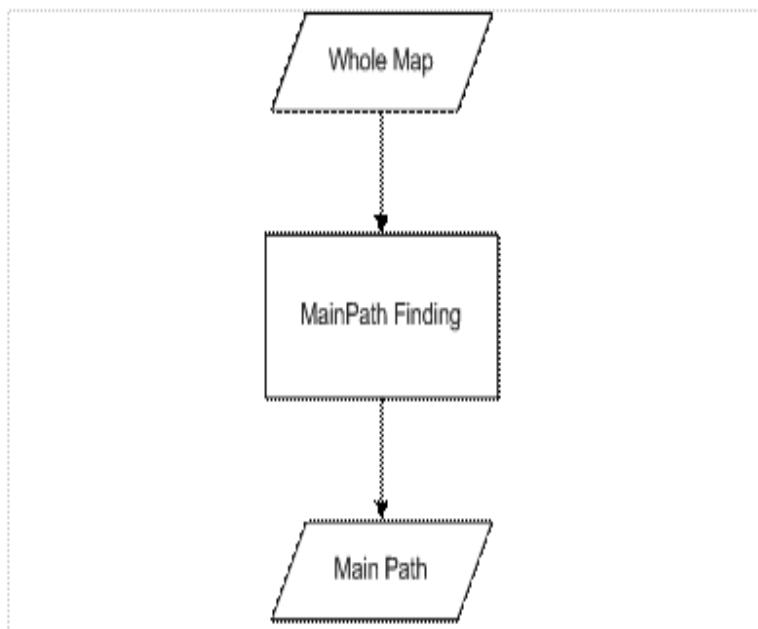
Figure 4-10 : Start (right (shape with x symbol)) & End (left (shape with o symbol)) Point Labels.

5- Main Path Finding

As in the scenario, the start point should be in the main path of the targeted room. While the start point relays on the main path, then all of its pixels are part from the main path pixels.

As shown in Figure 4-11, to find the main path, the algorithm needs to get any point from the mask startPointMask where the positions of all start point pixels are stored; this position is then sent along with the main path color (white), to the floodFill8 function which will capture all the main path pixels, which are white pixels disclosed inside the walls, doors and windows of this main path. The function returns a mask that is captured by the main path finding algorithm as a mask called mainPathMask. Figure 4-11 displays a declaring shape for the mask.

```
//MainPath Finding
//Selecting all white pixels inside the boarder starting from
//startPoint Pixel
```



```
while startPoint != ffffff
    x(startPoint)++;
    y(startPoint)++;
end while
mainPathMask = floodFill8(x(startPoint),y(startPoint),ffffff)
```

Figure 4-11 : Main Path Finding Algorithm Pseudo code.

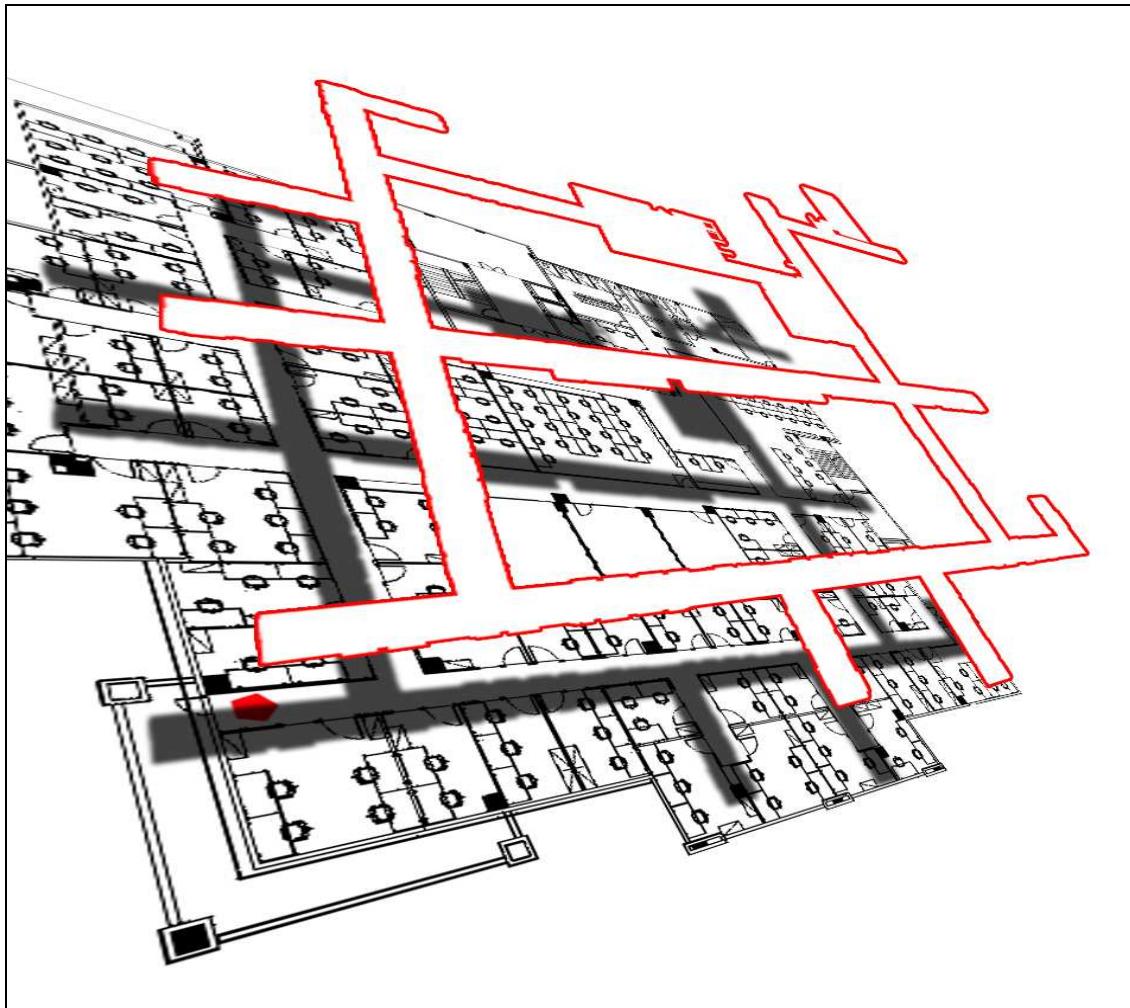


Figure 4-12 : Main Path Extracted from the Blueprint after Applying the Algorithm.

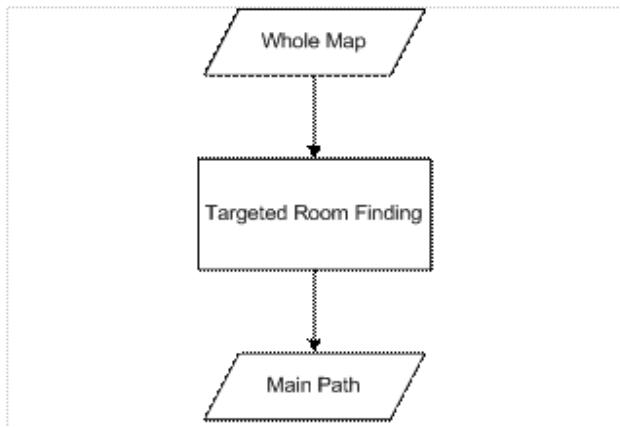
6- Targeted Room Finding

Scenario has specified that the end point should be in the targeted room, which is the room that is separated from the main path by only one door. Since the end point relies on the targeted room, then all of its pixels are part from the targeted room white pixels.

To find the targeted room, the algorithm (Figure 4-13) needs to get any point from the mask endPointMask, where the positions of all end point pixels are stored; this position is then sent along with the targeted room background color (white), to the

floodFill8 function which will capture all the targeted room background pixels, which are white pixels disclosed inside the walls, doors and windows of this targeted room. The function returns a mask that is captured by the targeted room finding algorithm as a mask called targetedRoomMask. Figure 4-14 shows a declaring shape for the mask.

```
//targetedRoom Finding
```



```

while endPoint != ffffff
    x(endPoint)++;
    y(endPoint)++;
end while
targetedRoomMask = floodFill8(x(endPoint),y(endPoint),ffffff)
  
```

Figure 4-13 : Targeted Room Finding Algorithm Pseudo code

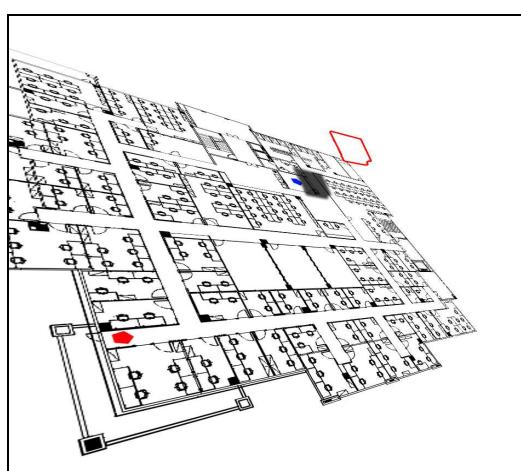


Figure 4-14 : Targeted Door Extracted from the Blueprint after Applying the Algorithm.

7- Similarity Algorithm

As shown in Figure 4-15. For each white pixel, the algorithm finds the white shape that contains it, each shape gets its own mask, then is sent to a function that calculate the size of the mask. If the returned size is bigger or smaller than a specific size, then the mask will be ignored, because as experienced, doors size set between that ranges. If returned size sets within that range, then it might be a door or other architectural symbol. So, in order to make sure that it is a door before tagging it, the algorithm contains pattern matching. Figure 4-16 shows different patterns examples.

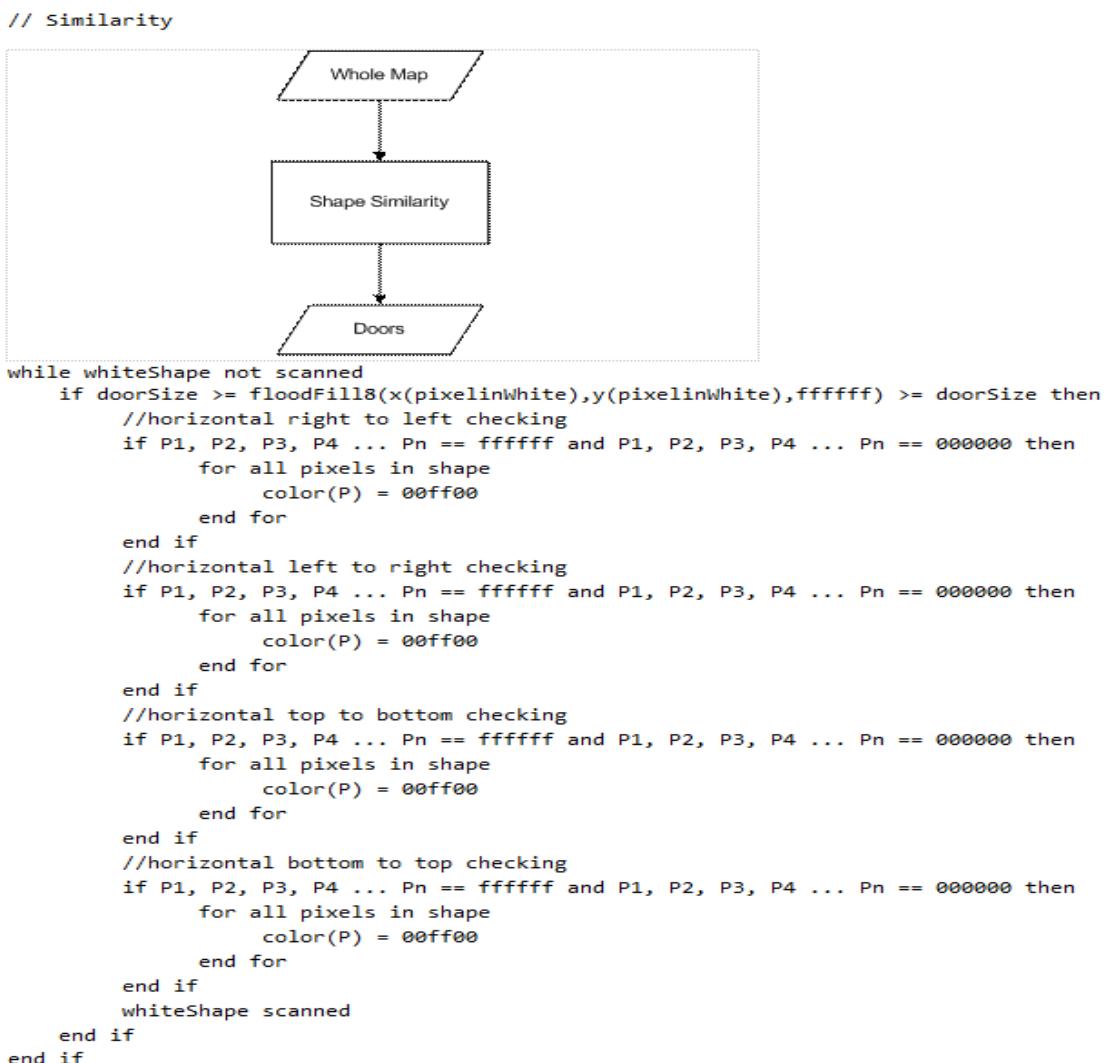


Figure 4-15 : Similarity Algorithm Pseudo Code

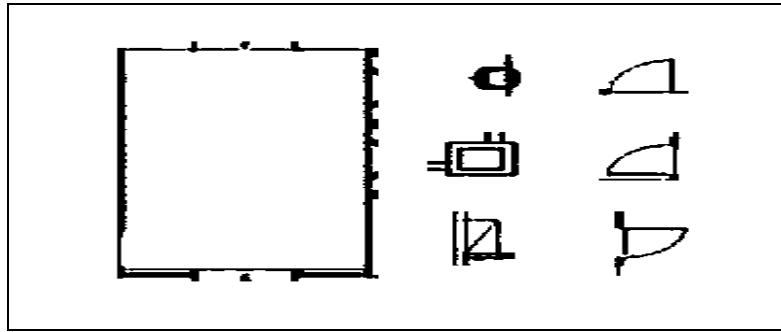


Figure 4-16 : Different Patterns Examples.

a. Pattern Matching

The main idea of pattern matching is checking for specific patterns in images, if patterns almost rely on the checked image, then the image is similar to the patterned image. In this algorithm, pattern matching is enhanced to do a specific job, utilizing processing time. Matching the suspected doors' background with specific pixels (white, nonwhite) for right to left, left to right, bottom to up and top to bottom doors, where Figure 4-17 shows different doors' patterns.

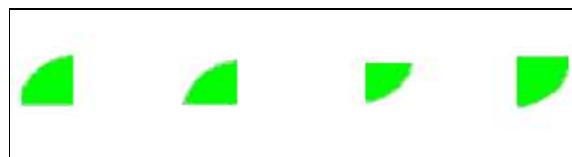


Figure 4-17 : Different Door Patterns

The algorithm takes the current suspected door's mask background and compares it with the mentioned patterns (right to left, left to right ...). If one of the patterns matched the mask, then the mask is a door, the patterns are put to satisfy the heel and the semi heel background door shapes.

For a detailed description for the above, to do the comparison, each pattern contains a matrix for a specific white and nonwhite (no pixel). Let's assume the

following; left to right pattern, has $[Ps, Pg, Pr \dots Pn]$ white pixels, and $[Pw, Pd, Pf \dots Pn]$ non white pixels, if the two conditions are satisfied, where the mentioned pixels are in the door background mask, then the shape is the background of a door. Figure 4.19 shows the Pseudo code of the algorithm.

b. Target room door

A door is said to be a targeted room door, if it is the door that directly separates the targeted room (the room that the intelligent robot reaches it as a result of the cognitive map built on knowledge gained from reading and analyzing the architectural blueprint map) and the main path (the path that the robot start point is set). Figure 4-18 shows the target door position in the map.

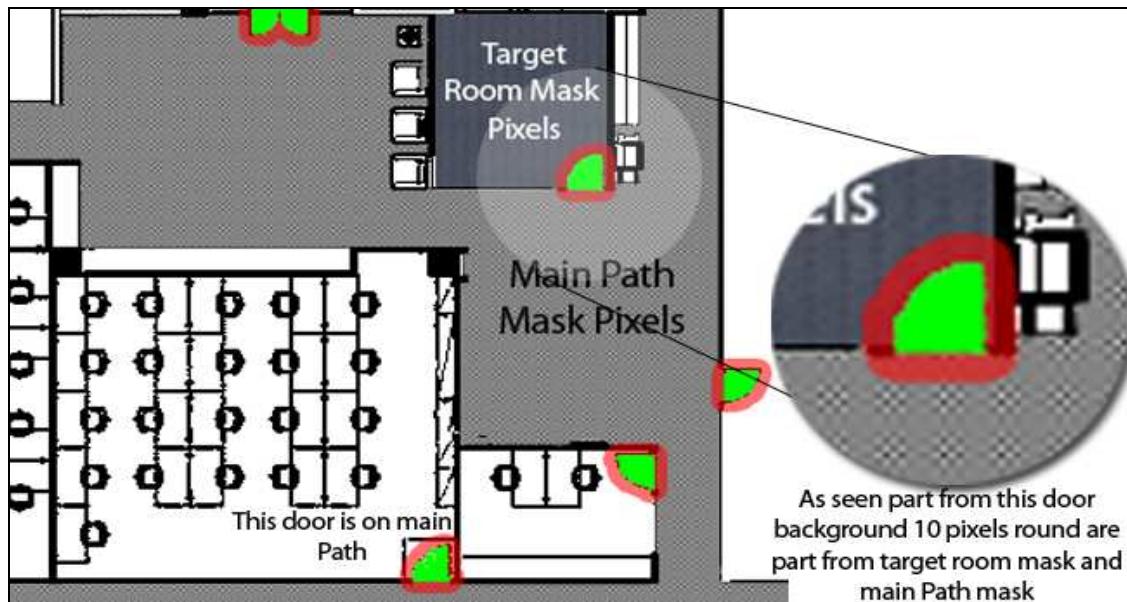


Figure 4-18 : Target Door Round Pixels Set Within Target Room and Main Path Pixels.

Figure 4-19 presents the Pseudo code for performing the previous step, the door is checked if it is a target room door, by checking 10 pixels around the door back into the main map, if there exists a pixel from round pixels is part from the mainPath mask, and there exists another pixel from the same door round pixels is part from the targetRoomMask, then it is the Targeted Room Door.

```

//doors on main path and targeted room
while greenShape not scanned
    for each border pixel in the greenshape
        if x(P) + 4 == MainPath then
            dooronMain = greenShape
        end if
        if x(P) + 4 == TargetedRoom then
            dooronRoom = greenShape
        end if
    end for
end while

for all dooronMain and dooronRoom
    for each pixel in the shape
        if P eight neighbors != 00ff00 then
            if dooronRoom then
                targetDoor = P
            else
                doors[j] = P
            end if
            end the current loop
        else if one of P neighbors == 00ff00 then
            boarderPixel[i] = P
        end if
    end for
    remove boarderPixel array
    re-loop
end for

```

Figure 4-19 : Finding Doors Pseudo code.

Target Room Door shape is stored on a new mask named targetDoorMask. Also, all the other doors background are stored in a new mask named doorsMask. All the door shapes' pixels are converted from white pixels to green pixels.

8- Medial Axis

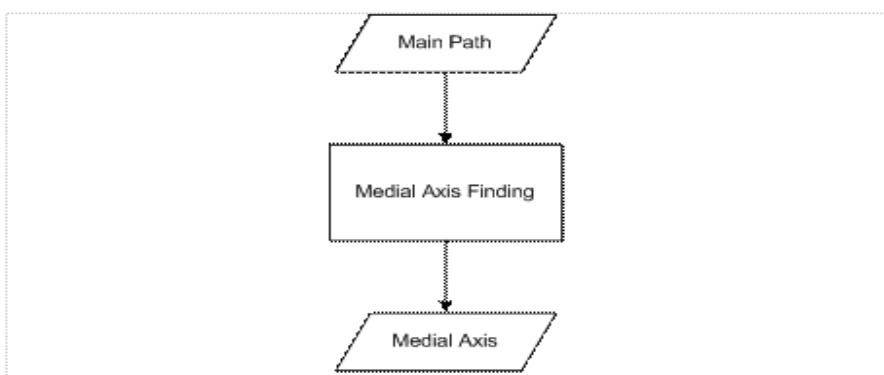
This is an enhanced algorithm for finding medial axis for paths on architectural blueprint maps. The idea for using the medial-axis is to find main path central pixels, and it is worth mentioning that the medial axis is defined by Rudolf (1999) as the following “The medial-axis transformation is useful in thinning a polygon, or, as is sometimes said,

finding its skeleton. The goal is to extract a simple, robust representation of the shape of the polygon”.

Each center pixel represents the center distance between two side boarder pixels; the need for these pixels in the algorithm emerges from the need of building a network of nodes for the path, where each node represents a turning in the real world (left, right ...), the algorithm also ignores finding the center pixels paths that are too small for intelligent robot to navigate into. Figure 4-20 shows finding boarder pixels Pseudo code.

First, it is important to group the mask mainPathMask with the two doors masks, doorsMask and targetedDoorMask. The new mask contains green shapes which are doors from doorsMask and targetedDoorMask, and white shape which is the main path.

```
//Medial Axis Finding
//maxLength and minLength for corridor (minLength the path that is
//not suitable for navigating inside)
```



```
//find boarder pixels and put them inside array

for each pixel P in the MainPath
    if P eight neighbors == ffffff then
        continue
    else
        boarderArray[j] = P
        j ++
    end if
end for
```

Figure 4-20 : Finding Boarder Pixels Pseudo code.

The algorithm starts finding the boarder of the main path, where the boarder is any white pixel that at least one of its neighbors is not a white pixel. All the boarder pixels are put on an array boarderArray.

a. Finding Central Pixels for Corridors

There are two types of corridors for the main path. One of them is horizontal that includes typical horizontal and semi horizontal corridors, and the other one is vertical that includes typical vertical and semi vertical corridors.

The algorithm repeats the grouped mask twice; one for finding center pixels for vertical corridors and the other loop for finding center pixels for the horizontal corridors.

b. Vertical Corridors

As shown in Figure 4-21, starting from the first boarder pixel in the mask, looping mask row by row, for the left boarder pixel (white boarder that its left neighbor pixel is nonwhite colored) the algorithm starts searching for its right boarder, by increasing the x value of the left pixel until it reaches the right boarder (white boarder that its right neighbor pixel is nonwhite colored), if the distance between these two pixels is smaller than minBoarder value (the value that the corridor must be greater than) then all pixels between them including them are removed. And if the distance between these two pixels is greater than maxBoarder value (the value that the corridor must be smaller than), then these pixels are ignored for now. Otherwise, the center pixel of these two side boarders is calculated as equation 1.

$$P_{center} = (Y(P), X(P_{right}) - X(P) / 2) \dots \text{equation 1}$$

The Pcenter is then black-colored.

Then, the algorithm puts the white pixels between the side boarders, including the side boarders on an array named arrayToRemove, in order to delete this array later.

As shown in Figure 4-24, searching 30 pixel on the left of P and 30 pixels on the right of Pright to find if there is a green pixel, which is the doors center pixels, if the green pixel found, then the Pcenter is put on an array that collects all the doors on the main path and their coordinates doorsArray along with the “left” or “right” values, if the green pixel is within the targetDoorMask, then the variable endPointVar is set as Pcenter

along with the “left” or “right” values. Figure 4-22 shows the path after finding center pixels for vertical corridors only using MATLAB R2009b.

```

//for vertical and semi vertical paths
for each boarder pixel in boarderArray
//assure that it is a bottom boarder
    if color(P4) != ffffff then
        for each P in boarderArray
            if y(Pbottom) == y(P) and x(Pbottom) != x(P) and color(Pbottom) != ffffff
                and Pbottom is not scanned and maxBoarder>= x(Pbottom)-x(P) >=minBoarder Then
                    Pcenter = (y(P),(x(Pbottom)-x(P))/2)
                    Pbottom scanned
                //checking for doors
                if any pixel between P and P(x(P)-20) ∈ doors then
                    topDoors[i] = Pcenter
                end if
                if any pixel between P and P(x(P)-20) ∈ targetDoor then
                    targetTopDoors[i] = Pcenter
                end if
                if any pixel between Pbottom and P(y(Pbottom)+20) ∈ doors then
                    bottomDoors[i] = Pcenter
                end if
                if any pixel between Pbottom and P(y(Pbottom)+20) ∈ targetDoor then
                    targetBottomDoors[i] = Pcenter
                end if
                for each pixel between P and Pbottom including them
                    if color(Px) == ffffff then
                        arraytoRemove[j] = Px
                        j ++
                    end if
                end for
                break the current loop
            end if
            if maxBoarder< x(Pbottom)-x(P) then
                for each pixel between P and Pbottom including them
                    bottomMaskShape = Px
                end for
            end if
        end for
    end if
end for

```

Figure 4-21 : Center Pixels for Vertical Corridors Pseudo code.

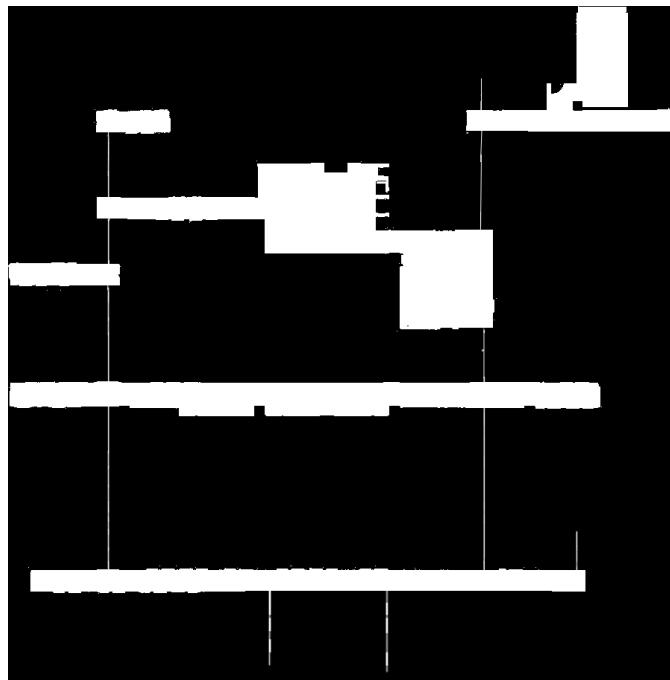


Figure 4-22 : the Path after Finding Center Pixels for Vertical Corridors Only.

c. Horizontal Corridors

The same as vertical scanning, as shown in Figure 4-23, starting from the first boarder pixel in the mask, looping mask column by column, for the top boarder pixel (white boarder that its top neighbor pixel is nonwhite colored) the algorithm start searching for its bottom boarder, by increasing the y value of the top pixel until it reaches the bottom boarder (white boarder that its bottom neighbor pixel is nonwhite colored), if the distance between these two pixels is smaller than minBoarder value (the value that the corridor must be greater than) then all pixels between them including them are removed. And if the distance between these two pixels is greater than maxBoarder value (the value that the corridor must be smaller than) then these pixels are ignored for now. Otherwise the center pixel of these two side boarders is calculated as equation 2.

$$P_{center} = (Y(P), X(P_{bottom}) - X(P) / 2) \dots \text{equation 2}$$

The Pcenter is then black colored.

```

//finding central pixels between two side boarders
//for horizontal and semi horizontal paths

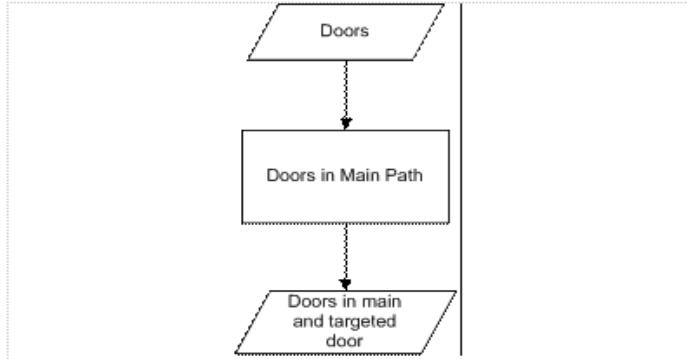
for each boarder pixel in boarderArray
//assure that it is a left boarder
  if color(P4) != ffffff then
    for each P in boarderArray
      if y(Pright) == y(P) and x(Pright) != x(P) and color(Pright) != ffffff and
          Pright is not scanned and maxBoarder>= x(Pright)-x(P) >=minBoarder Then
        //checking for doors
        Pcenter = (y(P),(x(Pright)-x(P))/2)
        if any pixel between Pright and P(x(Pright)+20) ∈ doors then
          rightDoors[i] = Pcenter
        end if
        if any pixel between Pright and P(x(Pright)+20) ∈ targetDoor then
          targetRightDoors[i] = Pcenter
        end if
        if any pixel between P and P(x(P)-20) ∈ doors then
          leftDoors[i] = Pcenter
        end if
        if any pixel between P and P(x(P)-20) ∈ targetDoor then
          targetLeftDoors[i] = Pcenter
        end if
        Pright scanned
        for each pixel between P and Pright including them
          if color(Px) == ffffff then
            arraytoRemove[j] = Px
          end if
        end for
        break the current loop
      end if
      if maxBoarder< x(Pright)-x(P) then
        for each pixel between P and Pright including them
          rightMaskShape = Px
        end for
      end if
    end for
  end if
end for

```

Figure 4-23 : Center Pixels for Horizontal Corridors Pseudo code.

Then the algorithm puts the white pixels between the side boarders including the side boarders on an array named arraytoRemove, to delete this array later.

```
//connect doors with medial axis
```



```
for each pixel in medial axis
    if x(P)+maxBoarder == dooronMain then
        if pixel(x(P)+maxBoarder) == dooronRoom then
            for each pixel from P to x(P)+maxBoarder
                Px = 000000
            end for
        end if
        remove dooronMain
        save pixel P, left to the short memory
    end if
    if x(P)-maxBoarder == dooronMain then
        if pixel(x(P)+maxBoarder) == dooronRoom then
            for each pixel from P to x(P)+maxBoarder
                Px = 000000
            end for
        end if
        remove dooronMain
        save pixel P, right to the short memory
    end if
    if y(P)+maxBoarder == dooronMain then
        if pixel(x(P)+maxBoarder) == dooronRoom then
            for each pixel from P to x(P)+maxBoarder
                Px = 000000
            end for
        end if
        remove dooronMain
        save pixel P, bottom to the short memory
    end if
    if y(P)-maxBoarder == dooronMain then
        if pixel(x(P)+maxBoarder) == dooronRoom then
            for each pixel from P to x(P)+maxBoarder
                Px = 000000
            end for
        end if
        remove dooronMain
        save pixel P, top to the short memory
    end if
end for
```

Figure 4-24 : Connecting Doors with Medial Axis Pseudo code.

As shown in Figure 4-24, there is a process of searching 30 pixel on the top of P and 30 pixels on the bottom of Pbottom to find if there is a green pixel, which is the doors center pixels. If a green pixel is found, then the Pcenter is put on an array that collects all the doors on the main path and their coordinates doorsArray along with the “top” or “bottom” values, if the green pixel was within the targetDoorMask, then the variable endPointVar is set as Pcenter along with the “top” or “bottom” values. Figure 4-25 shows the path after finding center pixels for vertical and horizontal corridors.

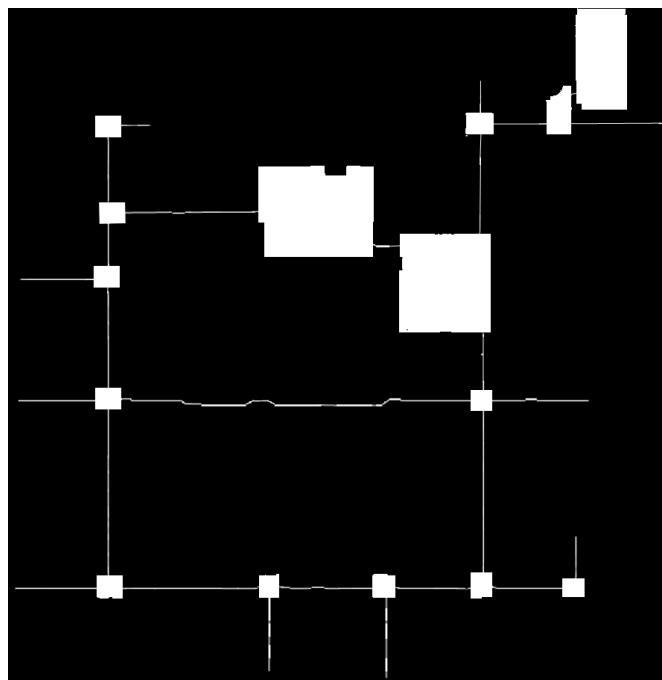


Figure 4-25 : Path After Finding Center Pixels For Vertical & Horizontal Corridors.

d. **Finding Turnings Centers**

From applying the two above steps, almost all white pixels from the mask are removed, except white shapes that connected horizontal and vertical corridors. These shapes present the turning in the real world, and the nodes in the path graph, since any pixel in these shapes cannot be either a center pixel or mid pixel (pixels between two boarders), since these shapes were ignored by the conditions in the above two sections.

```

//turns from the previous step didn't included in its condition
//represent pixels that are in bottomMaskShape rightMaskShape
//and with black pixels touching its boarder from two sides at least
//repeat this step for all turnings
for all shapes
    for each boarder pixel
        if one of P neighbors == 000000 then
            curFigNeig[i] = Pneig
            i ++
        end if
    end for
    for each pixel in the shape
        if P eight neighbors != ffffff then
            for each pixel from P to curFigNeig[i]
                Px = 000000
            end for
            turnings[j] = P
        else if one of P neighbors == ffffff then
            boarderPixel[i] = P
        end if
    end for
    remove boarderPixel array
    re-loop
end for

remove all pixels arrayToRemove

```

Figure 4-26 : Finding Turns Pseudo code.

The color of all these shapes is white, with at least two black pixels touching its boarder from more than one side, why? Because if there is one black pixel touching its boarder, then the corridor would not be ignored by the conditions, (it is not a turning shape), and it has been already skeletonized in previous steps.

Using a code in Figure 4-26, each turning shape is selected using the floodfill8 function, then for each pixel in the shape if one of its neighbors (not all of them) is a nonwhite colored pixel, then the pixel is a boarder pixel, and it is removed. The algorithm repeats removing the turning boarder until finding a pixel that its eight neighbors' pixels colors are nonwhite, then this is a center pixel for the turning, that should be black colored and added to an array that contains all nodes of the main path named nodesArray.

The black pixels touching the turning shape are saved in an array for the current shape, before finding the center pixel for this shape. So when the center pixel is allocated, the algorithm connects these pixels with the center pixel. Figure 4-27 shows the original medial axis after applying the Algorithm on MATLAB R2009b.

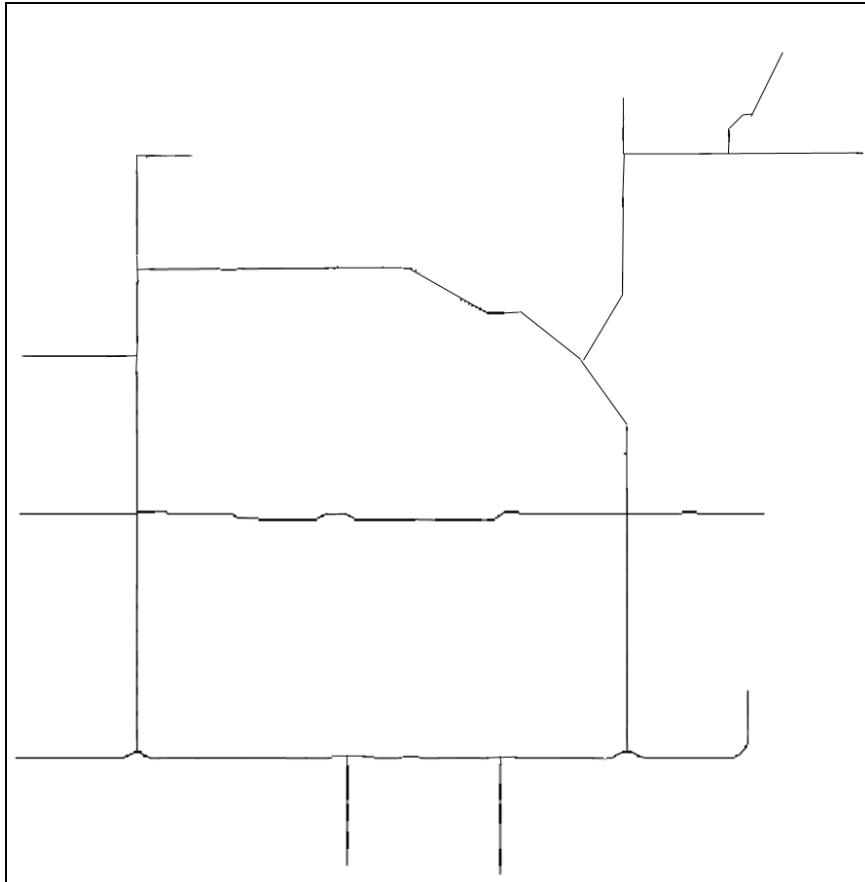


Figure 4-27: Original Medial Axis after Applying The Algorithm on MATLAB R2009b.

9- Labeling the Medial-Axis

The previous algorithms labeled the doors in the medial-axis. Also, the end point variable was found, so the algorithm (Figure 4-28) searches for the start point in the medial-axis from the set of pixels in startPointMask. These values are put on a variable startPointVar. Figure 4-29 shows the labeled medial axis with nodes on red for a simulation purpose.

```

//Finding start point node in Map
For each pixel in MainPath
    If P ∈ startPointMask then
        startPointNode = P
        return
    end if
end for

```

Figure0-28 : Start Point Node in the Map.

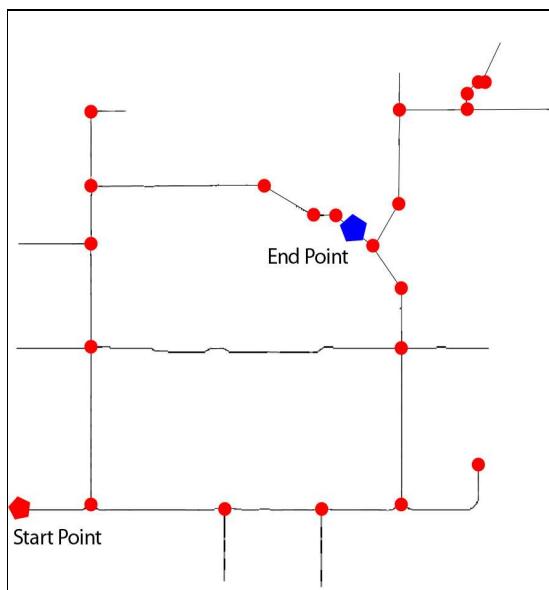


Figure 4-29 : Labeled Medial Axis with Nodes on Red for a Simulation Purpose.

10- Finding the Shortest Path

The Idea of this algorithm is to find the possible shortest path between the start and end points. In order to give the intelligent robot the required knowledge from the blueprints to sweep a building, the shortest path starting from startPoint ending with endpoint should be found in order to decrease the time that is needed for sweeping the targeted building..

This Algorithm contains connected sub algorithms, listed below:

a. Creating Graph (Connecting Nodes)

As a result of the previous step, a collection of nodes have been extracted from the blueprint map. Each node must be connected to its neighbors, so that the nodes are in sets as a network, each node with its neighbor (i.e. (node1, node3), (node1, node4), (node2, node3) ...).

In order to find the neighbors, as shown in Figure 4-31, each node boarder pixel labeled previously (including start and end points) is put in a function along with the node itself as a pixel that returns a node neighbor from one direction. Algorithm takes the current pixel (initially it is the node itself), put it in a variable, then take one of its eight neighbors. If one of these pixel's eight neighbors is a node, then it finishes its job and returns the node. Otherwise, it takes the other neighbor from the eight neighbors of the pixel. If this pixel is not as the same as the variable, then it is set as the new variable, and checked for next neighbor again until the next node is reached or a node with only one neighbor that is tagged in the variable, this means that a closed path is reached. Two Cases are shown in Figure 4-30.

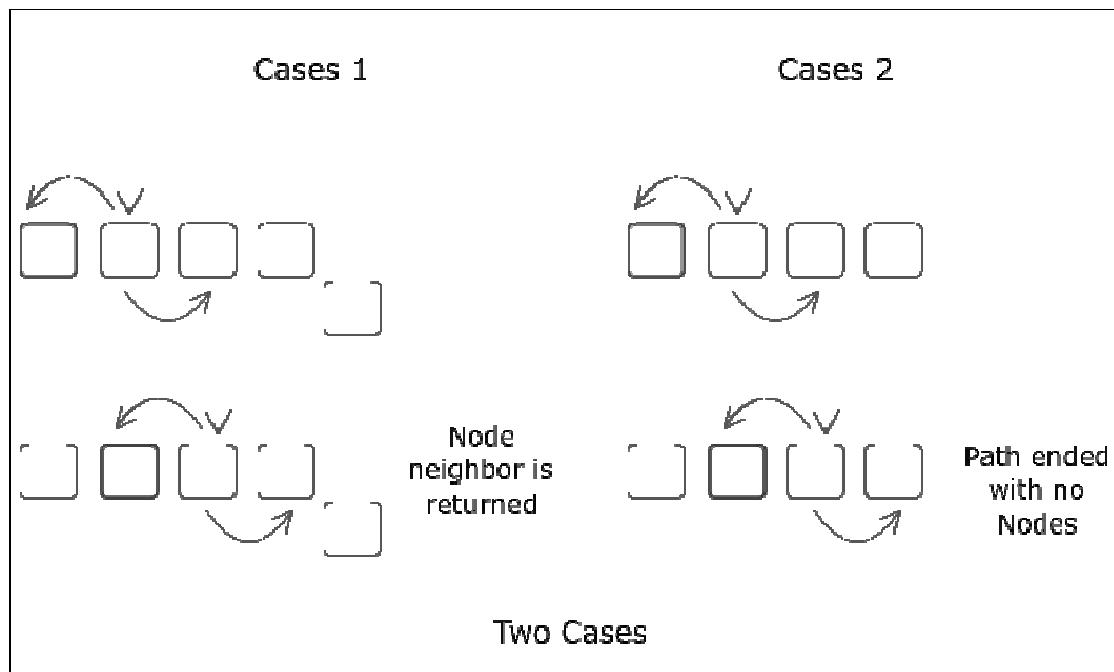
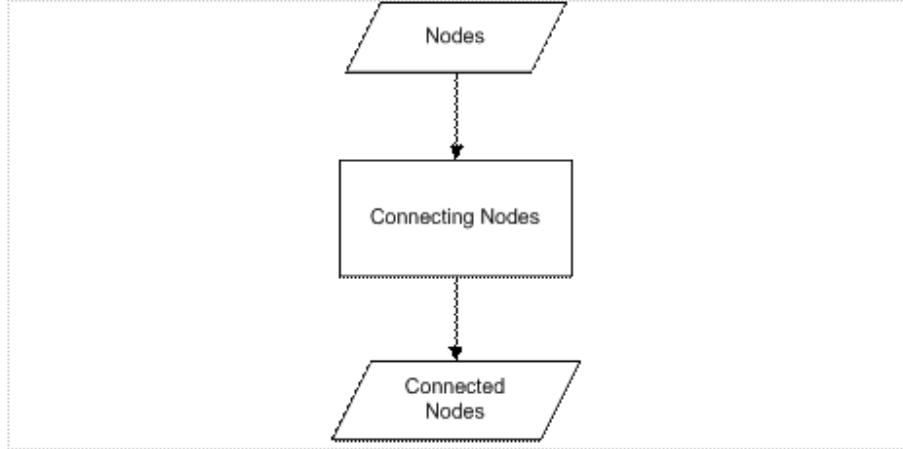


Figure 4-30 : Two Cases Appear on Connecting Nodes.

```

//finding shortest path
//nodes with its neighbors (connecting Nodes)



```

 Nodes
 |
 Connecting Nodes
 |
 Connected Nodes

```



```

//For each node in the array of nodes find its neighbors
//of nodes and the distance between them
for each node
 for each node neighbor
 if color(neighbor) = 000000 then
 if getNextNode(node, neighbor) not in connectNodes[i] then
 connectNodes[i] = getNextNode(node, neighbor)
 end if
 end if
 end for
end for

//get next node function get the node and
//which node pixel neighbor number (1,2,3,...)
function getNextNode(node,neighbor)
 currentPixel = node
 nextPixel = neighbor(neightbor) where neighbor is not currentPixel
 while nextPixel is not node && nextPixel is not nothing
 currentPixel = nextPixel
 nextPixel = neighbor(nextPixel) where neighbor is not currentPixel
 end while
 Return node & nextPixel
End function

```


```

Figure 4-31 : Connecting Nodes Pseudo code.

In the case of finding the neighbor node, the distance between the two nodes is calculated and returned within the set (i.e. (node1, node3, 50), (node1, node4, 150), (node2, node3, 70) ...).

b. Floyd's Algorithm

This algorithm has been previously declared by Taha (2007): “It determines the shortest route between any two nodes in the network. The algorithm represents an n-node network as a square matrix with n rows and n columns. Entry ($\sim j$) of the matrix gives the distance d_{ij} from node i to node j, which is finite if i is linked directly to j, and infinite otherwise”. The algorithm finds the shortest path in a network of nodes. As it is built it takes the sets of nodes, where each node is put within its neighbor node along with the distance (weight) between the two nodes. Also, the start and end points must be specified. Figure 4-32 shows the algorithm Pseudo code.

The algorithm has been edited in this study to match on going algorithms, because the original one assumes directed paths, where no directed paths are considered in navigation.

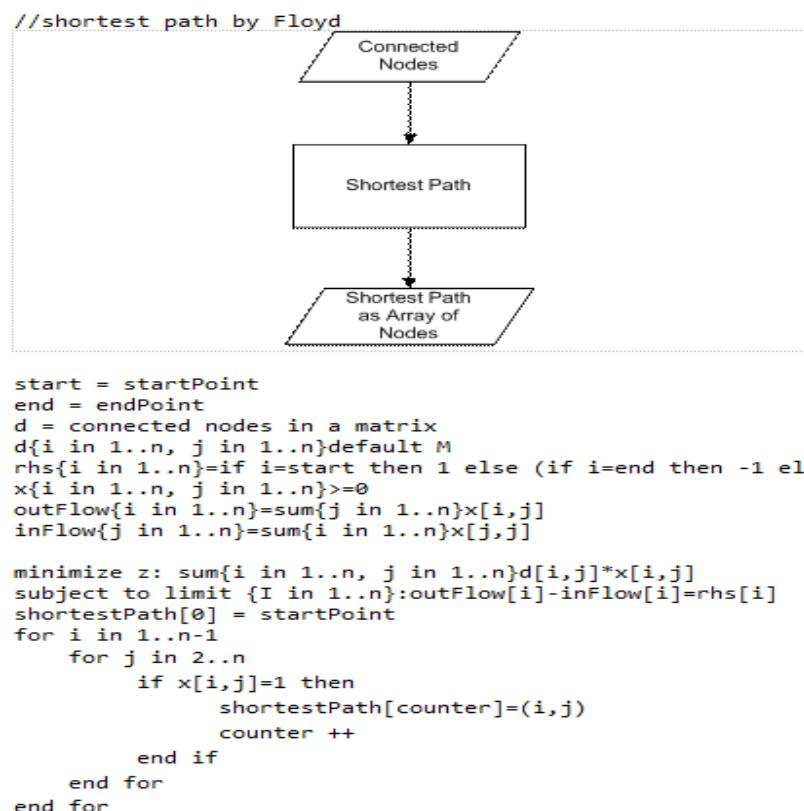


Figure 4-32 : Shortest Path by Floyd Algorithm Pseudo Code.

11- Translating to Directions

Finding Shortest Path Algorithm returns the path as an array of nodes. This array needs to be translated as understandable directions to be built as a cognitive map.

As shown in Figure 4-34, back to the labeled map, starting from startPoint, the angle between each three nodes is calculated in order to find the direction which three nodes presents, as specified previously in the knowledge-base, every angle presents a specific direction (i.e. 10-30: Hard right, 31-50: normal right, 180-200 right left and so on).

If the direction is found, it is stored in the knowledge-base within the analyzed map knowledge section, and then next three nodes should be taken within the shortest path, till reaching endpoint. Figure 4-35 shows how angels are translated into directions.

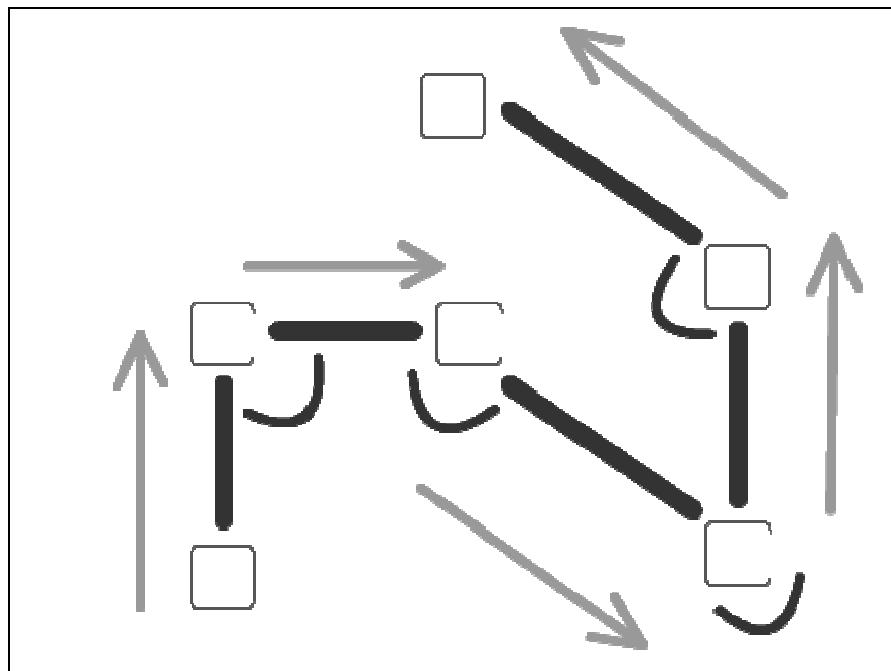


Figure 4-33 : Angels Translated into Directions.

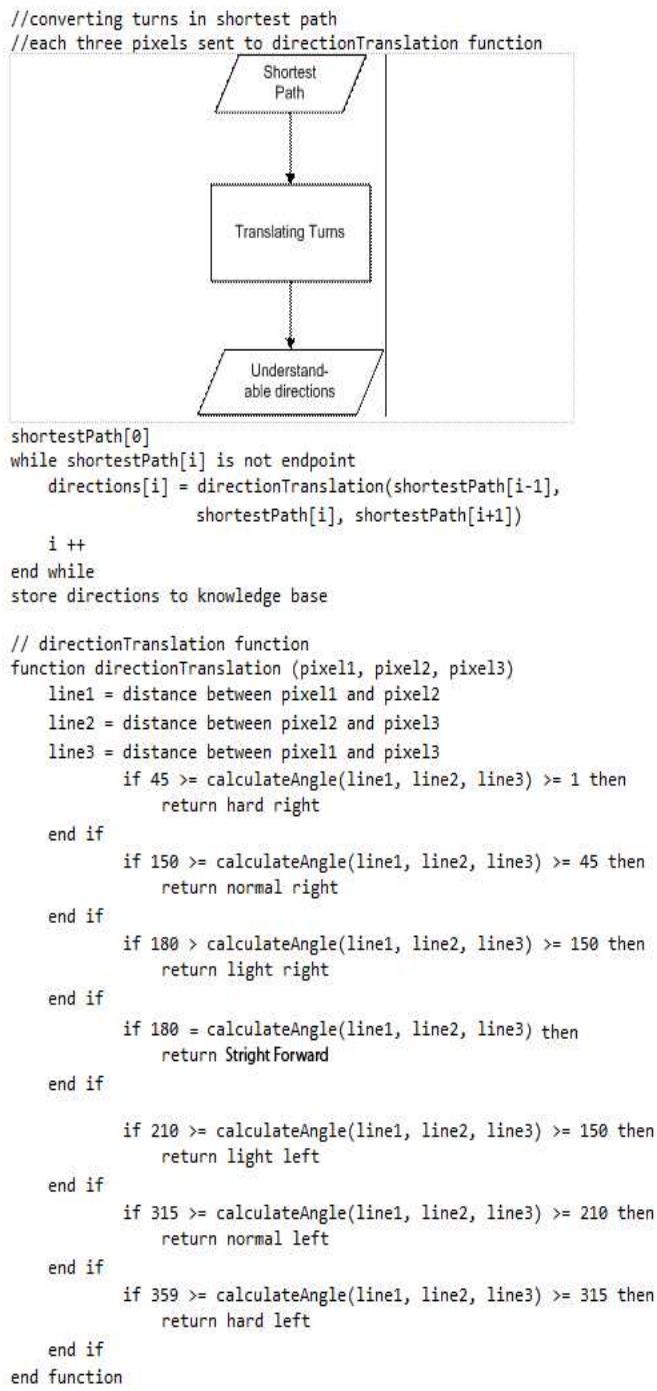


Figure 4-34 : Converting Nodes to Directions Algorithm Pseudo code.

a. Finding the Angle

In order to find the angle between each three nodes, the three nodes are presented in the Cartesian plane then the three line length could be found measuring the distance between each node from the another (the three lines are: one line from the first node to the second node, the second line is from the second node to the third node, and the third line is from the first node to the third node). These three lines present a triangle that its three lines are known, which means that it can be solved (knowing other elements values which are the three angles of the triangle).

The important angle is the angle number one which presents the angle or the complementary angle that is translated to direction.

At the end of this algorithm a knowledge base category is stored for this architectural blue print map, but a little knowledge is still missing. This knowledge is in regards of knowing which door is meant to be the targeted door (endpoint). Whereas more than one door could be located on the mainPath, especially in the corridor where the targeted door exists. For example, three doors on the left and four doors on the right could exist, which will be somehow confusing for even the human to find the targeted door among these doors, so the following algorithm was created to solve this issue.

b. Allocate Doors in Directions

As doors were previously allocated in the mainPath medial axis, they were not much accurate to be used in this stage, where they were allocated considering the optimal sweeping (sweeping from right to left or from down to top). In order to solve this issue, new doors positions should be applied using this algorithm.

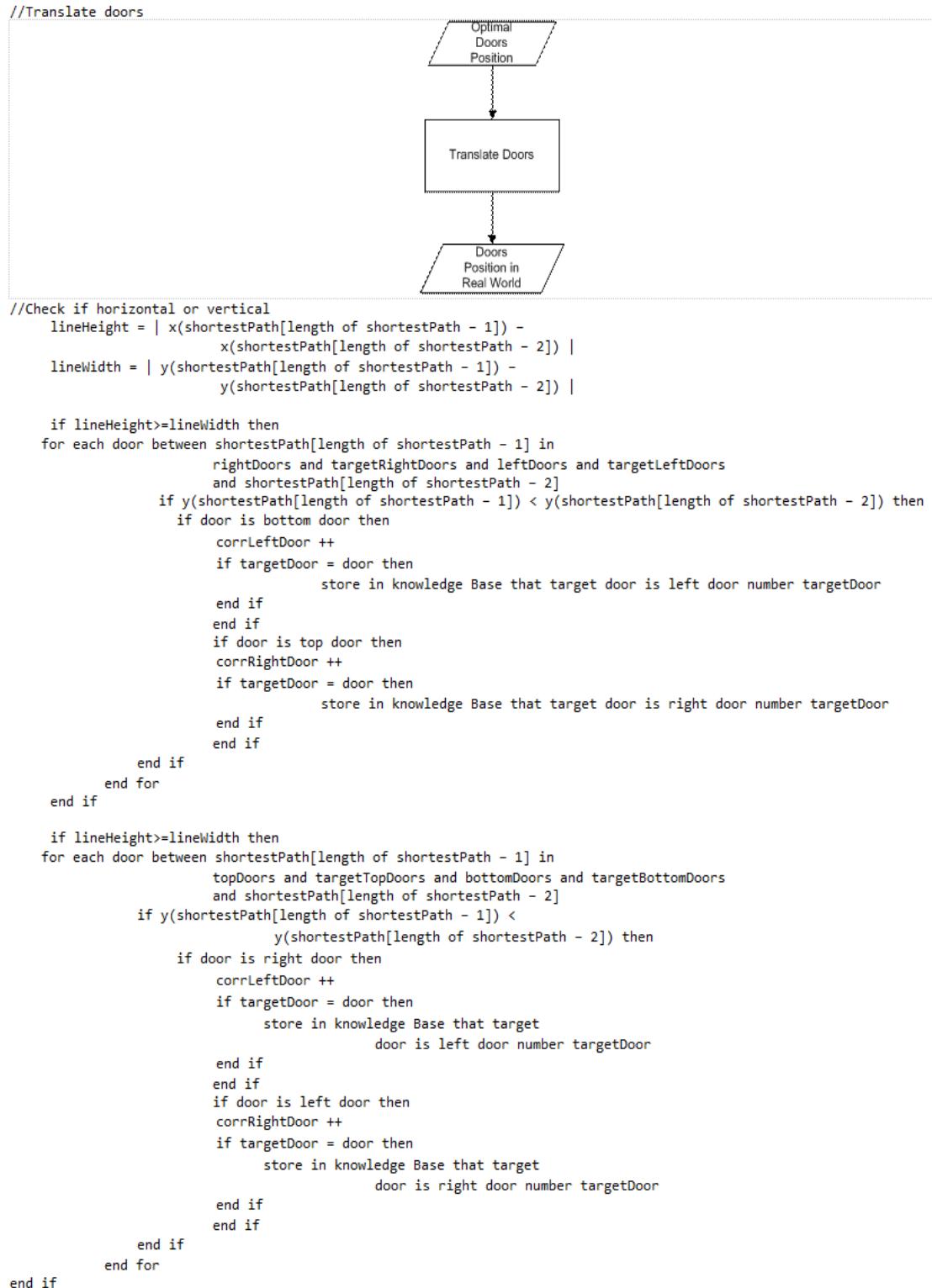


Figure 4-35 : Translating Doors Algorithm Pseudo Code.

As long as the corridor, where the targeted door exists, is the most important corridor in this stage, the algorithm (Figure 4-36) takes it (this corridor is presented as the node that comes before endpoint in the shortest path array). Applying this fact on the algorithm this corridor (node to endpoint) is taking, if the position of the node is on the right of the endpoint (in case of a horizontal corridor), or on top of the endpoint (in the case of vertical corridor), then door positions are inverted (if a left door, then it is a right door, and if a right door, then it is a left door). Otherwise, the door position remains as its default value (if a left door, then it is a left door, and if a right door, then it is a right door). Figure 4-37 declares that.

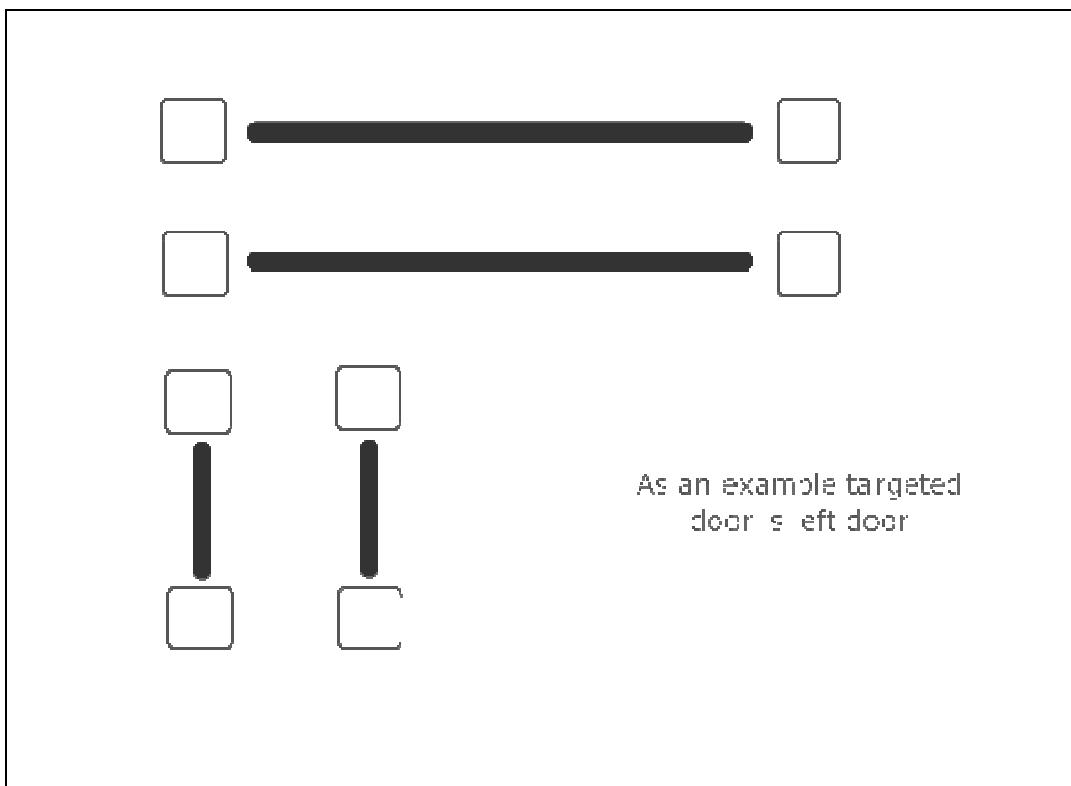


Figure 4-36 : Door Position Translation.

After that, the algorithm counts doors that come in the same side of the targeted door, locates the target door position (endpoint) and places this knowledge in the knowledge base.

12- The Cognitive Mapping Algorithm

Before proceeding a good description of designing, the cognitive maps on intelligent robot were written by Benjamin (1983), who says: ‘Could the human cognitive map, with all of its peculiarities, be structured in any other way and still perform the useful functions it does? We can perform a thought-experiment by imagining that we must design the cognitive map for a robot, operating under limited cognitive resources, which must assimilate and use knowledge about its large-scale environment acquired from observations during travel.’

He also declares than: “When we observe the peculiarities of the human cognitive map, we are tempted to ask, ‘Could it have been any other way?’ We can conduct a thought-experiment to study this question, by viewing it from the point of view of the computer scientist and system designer rather than that of the psychologist”.

From that point of view, it has been decided in this thesis to write the cognitive map in the knowledge base in a way that is near to the human cognitive mapping, but using intelligent robot knowledge base structure.

Based on the knowledge, which was gained from the architectural blueprint maps, the intelligent robot starts building cognitive map for that building (the building that the intelligent robot was supplied with its map) the cognitive map is built as a graph where each node presents a certain knowledge. Figure 4-37 shows a simple example of the directions extracted from the architecture blueprint map presented in the cognitive map.

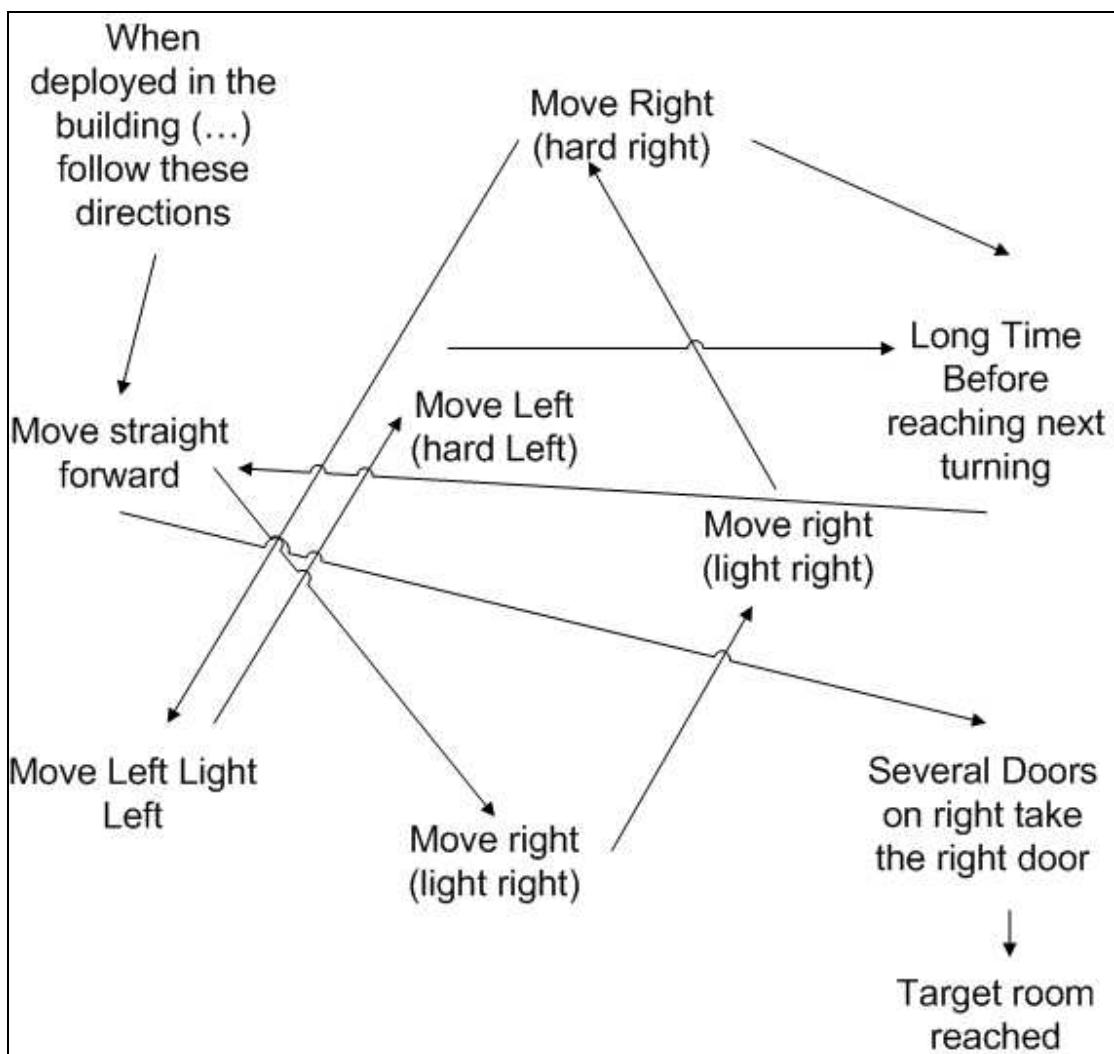


Figure 4-37 : Directions Cognitive Map

The cognitive map shown in the Figure 4-37 is simplified in order to declare the idea of cognitive mapping. In real time it is a more sophisticated map, containing relations with the definition of concepts used in each node (left, right, hard, light, normal, turning ...).

a. Representing knowledge in the knowledge base

There are different ways to represent knowledge in the knowledge base, including semantic networks, case-base, hierarchical structure, frame structure ... etc.

The frame structures are among the ways to represent knowledge in the knowledge-base, where the frames are like records in the file system, where each frame consists of slots (slot 1, slot 2, slot 3, ...), as shown in Figure 4-38. Each slot can have different types of data on it like (fixed data, variable data, range of data ...).

Frame 1					
Slot 1	Slot 2	Slot 3	Slot 4	Slot n

Figure 4-38 : Frame Structure.

b. Cognitive Mapping Structure in the knowledge base

Back to the cognitive mapping, the frame structure is used in the thesis to represent the cognitive map in the knowledge base, where each node in the cognitive map is a frame that contains the following slots:

- 1- The node position: the position of the node in the shortest path array. This field is a number; it is used to determine the ID of the frame.
- 2- The direction to be taken i.e. (hard right, light left ...). This field is the actual turning translated before in the previous algorithm.
- 3- Next frame number: this field is set to tell the robot of the next frame to be taken and to in order to decrease the cognition of the robot of his current status in the real time.
- 4- Truthiness of this direction to be used during the robot navigation (so if there is no such direction, i.e. the turning is blocked, the truthiness percentage of the whole map will be decreased).

- 5- Progress percentage: how many nodes still remaining, also this is to decrease the cognition of the robot of his current status in the real time with the final point.
- 6- The current direction with the end point: also this is to decrease the cognition of the robot of his current status in the real time with the final point.

The nodes are connected with each other as a network (graph of nodes) as shown in Figure 4-39. The design came to satisfy the general idea of cognitive mapping, giving the ability to extend the cognitive map when this extension is needed (based on knowledge update job) or removing nodes (frames) from the cognitive map.

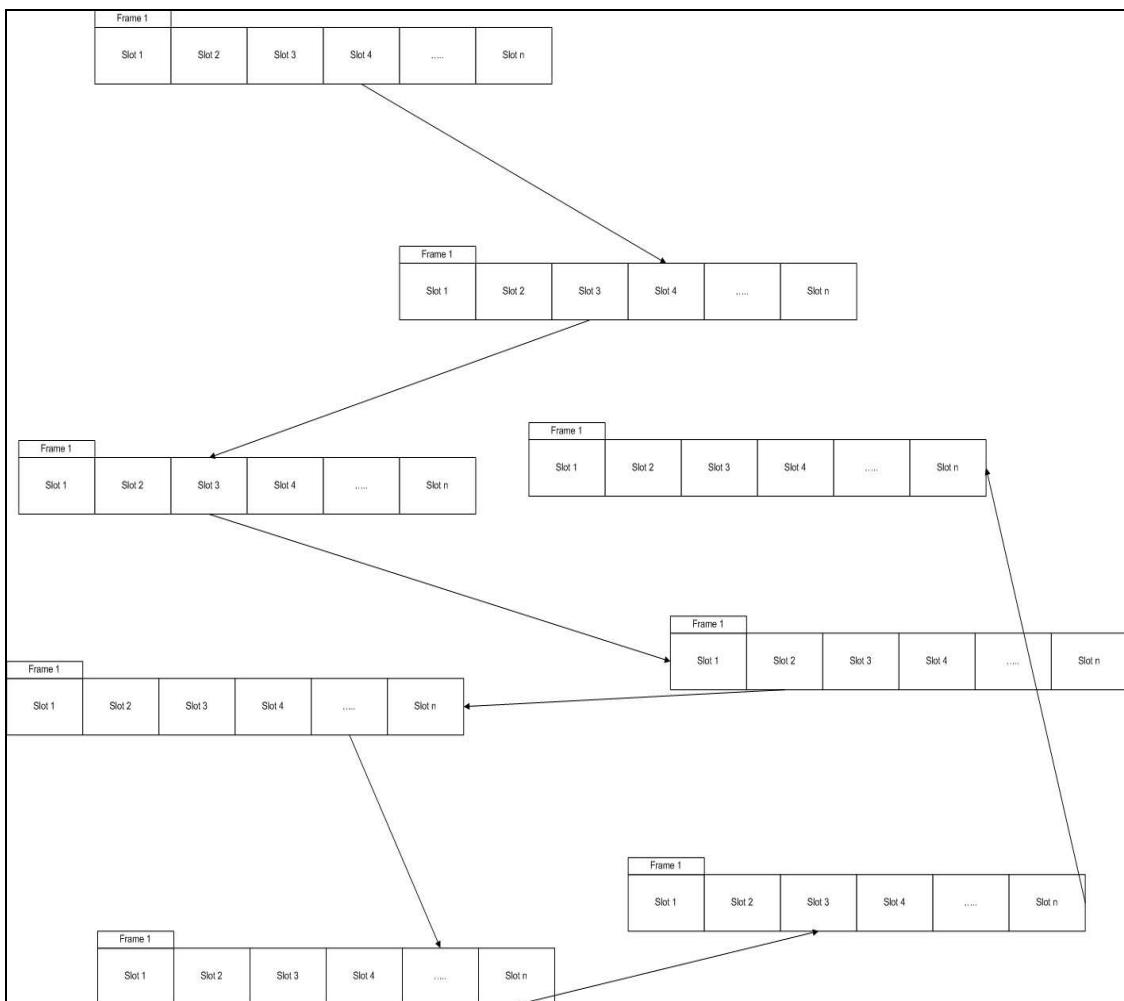


Figure 4-39 : The Cognitive Map, Network of frames.

c. Cognitive Mapping Storing Algorithm

After specifying the structure of the cognitive map, it's time to declare how the algorithm stores the knowledge in the knowledge base.

As shown in the Figure 4-40, before receiving any direction from the previous algorithms, the algorithm initiate the cognitive map in the knowledge base, specifying the startPoint node as the first frame in the Cognitive map. After that, it starts receiving the direction adding each direction as a new frame connected to the network (the cognitive map) and filling the frame with the data received along with the direction, as specified before. After the whole shortest path array is translated, reaching to the final element on it (the endPoint), the algorithm stores it as a new frame in the cognitive map, along with the data required returned from algorithms. Finally, the algorithm commits the cognitive map on the knowledge base storing it with the name of the building and the date and time of the process.

```
// Cognitive Mapping

init cognitive map
startPoint as the first frame in the new cognitive map

for each new direction received

    init new frame
    fill frame with data from the algorithms
    add frame to the cognitive map

end for

store endPoint
commit cognitive map in the knowledge base
```

Figure 4-40 : Cognitive Mapping Storing Algorithm Pseudocode

Chapter Five

Chapter Five

Conclusion, Discussion & Future Work

5.1. Overview

This chapter includes the conclusion of the thesis, the discussion about the directions in cognitive mapping and the future work proposed.

5.2. Conclusion

In this thesis, algorithms for building cognitive map for intelligent robot based on architectural blueprint maps have been presented.

The main aim of this thesis was to build algorithms that take the architectural blueprint maps and analyze them on the basis of the importance of the information needed to be gained from the map. The mentioned algorithms were created to make the robot understand the architectural blueprint maps and convert it to knowledge as a cognitive map inside the knowledge based system of the intelligent robot.

The thesis focused on showing how these cognitive maps could be generated from architectural maps. The main features of these algorithms are to help intelligent robots work without the need of a direct human interaction. In addition to that, these algorithms would decrease the speed of sweeping targeted buildings. Moreover, studies (Shannon, 1988) showed that depending on such maps, even a human could know and navigate buildings for the first time better than people who did not see the architectural blueprint maps, even if they lived in the same building for years.

For the same reason, intelligent robot that knew the directions of the shortest path needed to reach its target in a new building work faster than intelligent robot that did not have any sort of external guidance. In fact, the latter needed to scan room by room and corridor by corridor to reach its target. From this point, it is possible to generalize that the need of such abilities have the high priority for mobile intelligent robots.

5.3. Discussion

As discussed in chapter two, based on studies comparing different kinds of robot maps techniques used, the summary came with the need to establish new intelligent robot navigation and mapping techniques based on what scientists thought about techniques used by human in order to solve the navigation problems in his ordinary life.

From the bases of the cleverness of human that could not be found in any creature on earth, the need of an intelligent robot that thinks like a human being appears, where Chandler (2009) discusses that: “The field of artificial-intelligence research (AI), founded more than 50 years ago, seems to many researchers to have spent much of that time wandering in the wilderness, swapping hugely ambitious goals for a relatively modest set of actual accomplishments. Now, some of the pioneers of the field, joined by later generations of thinkers, are gearing up for a massive “do-over” of the whole idea”.

He also continues declaring that: “This time, they are determined to get it right — and, with the advantages of hindsight, experience, the rapid growth of new technologies and insights from the new field of computational neuroscience, they think they have a good shot at it”.

So it is highly recommended to start building on those same bases, even if it takes a huge amount of time and research rebuilding the techniques developed for robotics in the past, without leaving studying other techniques that might be used by other creatures to face the life tasks and survive in its difficulties.

5.4. Future Work

As a future work, it is highly recommended to start thinking about more algorithms that extend the job of the algorithms created or developed in this thesis to cover multi floored buildings dealing with each floor as a layer connected with the other layers (floors), by connecting points (leaders, stairs, Elevators ... etc), as Figure 5-1 shows.

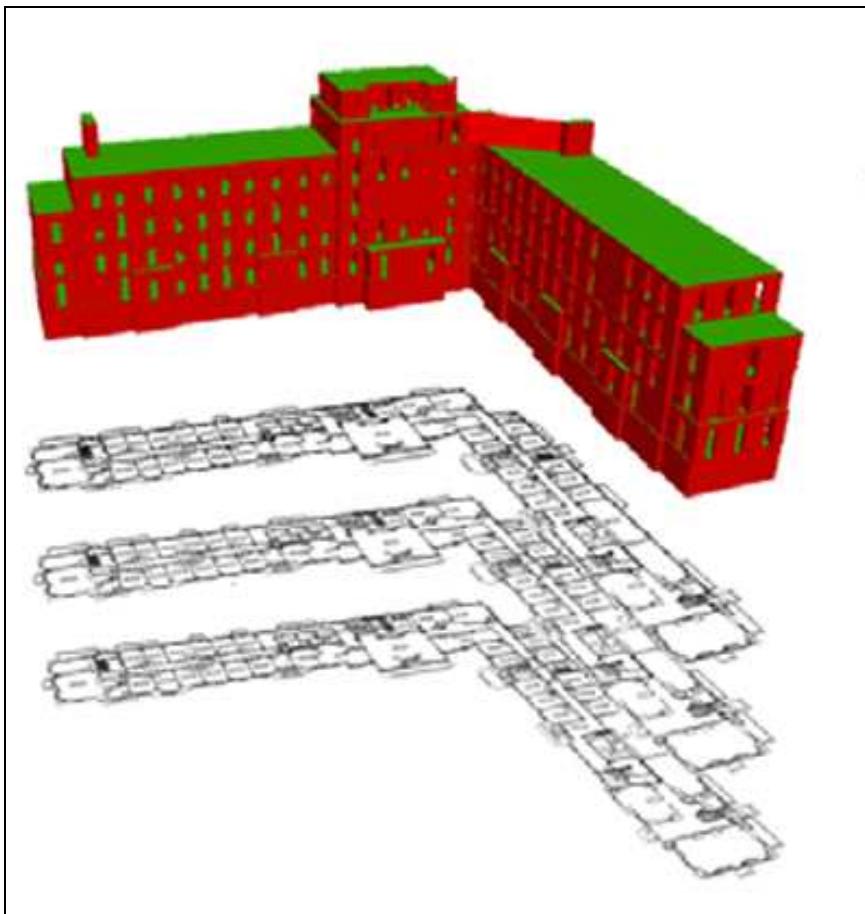


Figure 5-1 : Multi Floored Building Architectural Map (Alex et al. 2009)

As another future work, algorithms need to be developed to be more dynamic dealing with more sophisticated buildings with more sophisticated job routers (separated paths, targeted room inside another room).

Figure 5-2 shows different kinds of doors (internal doors, external doors, sliding doors ... etc). These types of doors cover the typical door types. Considered as a future work, these doors need to be covered by the research algorithms trying to find better and simpler matching and shape recognition algorithms.

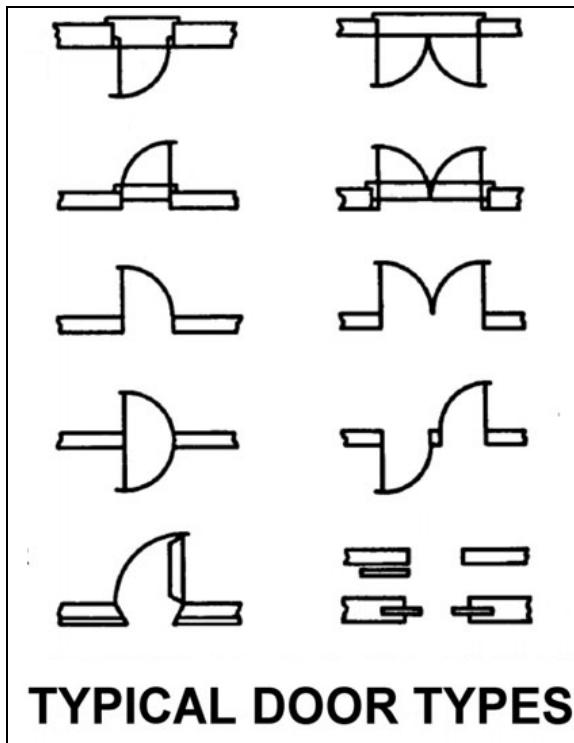


Figure 5-2 : Different Kinds of Doors (Sam, 2009).

The future work may also include a machine learning techniques in order for the intelligent robot to be able to deal with the environment and building on-time cognitive maps, so when the real time comes after being deployed, the intelligent robot starts updating the cognitive map based on reality. Adding more nodes based on what it sees and recognizes from both objects and avoided obstacles...

furthermore, it is demanded as a future work to make the intelligent robot take the angle with the directions taken from the architectural blueprint maps, in order to be more accurate in specifying the angle that should be taken during navigation.

References

- A.Ashbrook, N.A.Thacker, 1998, ‘Algorithms For 2-Dimensional Object Recognition’, Imaging Science and Biomedical Engineering Division, Medical School, University of Manchester, Stopford Building, Oxford Road, Manchester, M13 9PT.
- A. Koutamanis, 1989, ‘Recognition of Locations in Architectural Plans’, *In Proceedings of 6th Scandinavian Conference on Image Analysis*, Oulu (Finland), pp. 800–803.
- A. Koutamanis, September 1990, ‘Development of a Computerized Handbook of Architectural Plans’, PhD thesis, Technische Universiteit Delft, Delft, Netherlands.
- A. Koutamanis, 1995, ‘Recognition and Retrieval in Visual Architectural Databases’, *Visual Databases in Architecture*, chapter 2, pp. 15–42, Avebury.
- A. Koutamanis and V. Mitossi, 1992, ‘Automated Recognition of Architectural Drawings’, *In Proceedings of 11th International Conference on Pattern Recognition*, Den Haag (Netherlands), vol. 1, pp. 660–663.
- Adriana Tapus , Roland Siegwart, 2006, ‘A Cognitive Modeling of Space using Fingerprints of Places for Mobile Robot Navigation’, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA’06)*.
- Alex Levishtein, Adrian Stere, Kiriakos N. Kutulakos, David J. Fleet, Sven J. Dickinson, Kaleem Siddiqi, 2009, ‘TurboPixels: Fast Superpixels Using Geometric Flows’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE Computer Society Washington, DC, USA.

Alexander Gehre, Peter Katranuschkov, 2002, 'Human Centered Knowledge-Based Model Access Service for Engineers', *International Council for Research and Innovation in CIB w78 conference*.

Avron Barr, Paul R. Cohen, and Edward A., Feigenbaum, 1990, "The Handbook of Artificial Intelligence".

Bettina Berendt, Thomas Barkowsky, Christian Freksa, and Stephanie Kelter, 1998, 'Spatial Representation with Aspect Maps', Vogt-Kölln-Str. 30, D-22527 Hamburg, Germany.

Benjamin Kuipers, 2000, 'The Spatial Semantic Hierarchy', Qualitative Reasoning Group at the Artificial Intelligence Laboratory, *Artificial Intelligence*, vol. 119, pp. 191-233.

Benjamin Kuipers, 1983, 'The Cognitive Map: Could It Have Been Any Other Way?', *Spatial Orientation: Theory, Research, and Application*, pp. 345-359, Plenum Press.

Benjamin Kuipers, Yung-Tai Byun, 1993, 'A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations', *Toward learning robots*, MIT Press Cambridge, MA, USA.

Chandler David, Dec 2009, *Rethinking artificial intelligence*, Broad-based MIT project, Massachusetts Institute of Technology University, US, viewed 1 January 2010,
[<http://web.mit.edu/newsoffice/2009/ai-overview-1207.html>](http://web.mit.edu/newsoffice/2009/ai-overview-1207.html) accessed in 13/12/2009.

Christian Ah-soon , Karl Tombre, 1997, ‘Variations on the Analysis of Architectural Drawings’, *In Proceedings of Fourth International Conference on Document Analysis and Recognition*, pp. 347-351.

D. Coeurjolly, Y. Gérard, J.-P. Reveillés, L. Tougne, 2004, ‘An elementary algorithm for digital arc segmentation’, *Discrete Applied Mathematics*, Vol. 139, pp. 31-50.

Dirk Hähnel, Rudolph Triebel, Wolfram Burgard and Sebastian Thrun, 2003, ‘Map Building with Mobile Robots in Dynamic Environments’, *In Proc. of the IEEE International Conference on Robotics and Automation*.

Downs RM, Stea D., 1973, *Cognitive Maps and Spatial Behavior: Process and Product*, In Downs RM, Stea D. (Eds.) *Image and Environment*, Aldine Publishing Company, Chicago.

E. L. Hall, B. C. Hall, 1985, ‘Robotics: A User-Friendly Introduction’, Saunders College Publishing, Holt, Rinehart and Wilson, Orlando, FL.

Emily J. Whiting, 2006, Geometric, 'Topological & Semantic Analysis of Multi-Building Floor Plan Data', Degree of master of science in the architectural studies, MIT.

Emilio Remolina Benjamin Kuipers, 2003, ‘Towards a general theory of topological maps’, *Artificial Intelligence*, vol. 152, pp. 47-104.

Eysenck, M.W., 1990. *The Blackwell Dictionary of Cognitive Psychology*, Cambridge, Massachusetts: Basil Blackwell Ltd.

H. Sundar, D. Silver, N. Gagvani, S. Dickinson, D. Silver Í, 2003, Skeleton Based Shape Matching and Retrieval, the Vizlab at the CAIP Center, Rutgers University.

Hiroshi Ishiguro And, Hiroshi Ishiguro, Katsumi Kimoto, 1997, ‘Town Robot - Toward social interaction technologies of robot systems’, *Proc. Int. Conf. Field and Service Robotics*.

India Property, 2008,
<<http://www.indiaproPERTY.com/index.php?option=upcoming&page=projectview&id=1105583>> accessed Feb 2010.

J. Llad'os, J. López-Krahe, and E. Martí, August 1996, ‘Hand Drawn Document Understanding Using the Straight Line Hough Transform and Graph Matching’, *In Proceedings of the 13th International Conference on Pattern Recognition*, Vienna (Austria), vol. 2, pp. 497–501.

Jay P McCormack, Jonathan Cagan, 2002, ‘Supporting designers' hierarchies through parametric shape recognition’, *Environment and Planning B: Planning and Design*, pp. 913-931.

João Xavier, Marco Pacheco, Daniel Castro, António Ruano, 2005, ‘Fast line, arc/circle and leg detection from laser scan data in a player driver’, *Proc. of the IEEE Int. Conference on Robotics & Automation*.

Ji-Xiang Du, De-Shuang Huang, Xiao-Feng Wang, Xiao Gu, 2006, 'Shape recognition based on neural networks trained by differential evolution algorithm', *Neurocomputing*, Vol. 70, pp. 896-903.

Jochen Schmidt, Chee K. Wong, Wai K. Yeap, 2007, 'Spatial Information Extraction for Cognitive', *Proceedings of the 8th international conference on Spatial information theory*, pp. 186-202.

K. Mikolajczyk, A. Zisserman, C. Schmid, 2003, 'Shape recognition with edge-based features', *Proceedings of the British Machine Vision Conference*, Norwich, UK.

K. Ryall and S. Shieber, August 1995, 'Semi-Automatic Delineation of Regions on Floor Plans', in *Proceedings of Third International Conference on Document Analysis and Recognition*, Montréal (Canada), pp. 964–969.

Liu Wenyin, Dov Dori, April 1998, 'Incremental Arc Segmentation Algorithm and Its Evaluation', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 4, pp. 424-431.

Lode Vandevenne, 2004, 'Lode's Computer Graphics Tutorial', LEUVEN University.

Motilal Agrawal and Kurt Konolige, 2006, 'Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS', SRI International Ravenswood Ave Menlo Park, CA.

LUGER G., 1999, "Artificial Intelligence: Complex Problem Solving Methodologies", Addison-Wesley, Fourth Edition.

Owaied Hussein H., 2010, 'Frame Model for Intelligent Robot as Knowledge-Based System', *Special Issue of Ubiquitous Computing Security Systems*, 15/1/2010.

Owaied Hussein H., Abu-Arr'a Mahmoud, Farhan Hazim, 2010, "An Application of Knowledge-based System", *International Journal of Computer Science and Network Security*, VOL.10 No.3.

Owaied Hussein H., Abu-Arr'a M. Mahmoud, 2007, 'Functional Model of Human System as knowledge Base System', *The 2007 International Conference on Information & Knowledge Engineering*, p.p 158-161.

Owaied Hussein H., Sa'ab Saif M., 2008, "Modeling Artificial Bees-Colony System", *proceeding of The 2008 International Conference on Artificial Intelligence*.

Pavel Dimitrov Carlos, Carlos Phillips, 2000, 'Robust and Efficient Skeletal Graphs', *In Conference on Computer Vision and Pattern Recognition*.

Puneet Goel, Stergios I. Roumeliotis and Gaurav S. Sukhatme, 1999, 'Robot Localization Using Relative and Absolute Position Estimates', Department of Computer Science Institute for Robotics and Intelligent Systems University of Southern California Los Angeles, CA.

Remco C. Veltkamp, Remco C. Veltkamp, 2001, ‘Shape Matching: Similarity Measures and Algorithms’, *Proceedings of the International Conference on Shape Modeling & Application*, pp. 188-197, IEEE Computer Society Washington, DC, USA.

Rudolf F. Graf, 1999, *MODERN DICTIONARY of ELECTRONICS*, 7th edn, British Library Cataloguing-in-Publication Data, London,

Sam A. A. Kubba, 2009, Blueprint Reading: Construction Drawings for the Building Trades, McGraw-Hill, US.

Schönberg, M. Ojala, J. Suomela, A. Torpo & A. Halme, 1995, ‘Positioning an Autonomous Off-Road Vehicle, By Using Fused DGPS and Internal Navigation’, *In 2nd IFAC Conference on Intelligent Autonomous Vehicles*, pp. 226-231.

Sebastian Thrun, 2002, ‘Robotic Mapping: A Survey’, *Exploring Artificial Intelligence in the New Millennium*, Morgan Kaufmann.

Sergios Theodoridis, Konstantinos Koutroumbas, 2006, Pattern Recognition, 3ed edn, Academic Press.

Shannon Dawn Moeser, 1988, 'Cognitive Mapping in a Complex Building', Environment and Behavior, Vol. 20, No. 1, pp. 21-49.

Shrihari Vasudevan, Stefan Gächter, Viet Nguyen and Roland Siegwart, 2007, ‘Cognitive maps for mobile robots—an object based approach’, *Robotics and Autonomous Systems* 55, pp. 359–371, Zürich, Switzerland.

Steven S. Skiena, 1997, *The Algorithm Design Manual*, Springer-Verlag, New York.

Stuart-Hamilton, 1995, *Dictionary of Cognitive Psychology*. Jessica Kingsley Publishers, London.

Taha Hamdy, 2007, *Operations Research: An Introduction*, 8th edn, Prentice-Hall if India.

Tolman EC., 1984, 'Cognitive Maps in Rats and Men', *The Psychological Review*, vol. 55, pp. 189-208.

W.K. YEAP1 and M.E. JEFFERIES, 2000, 'On early cognitive mapping', *Spatial Cognition and Computation*, no. 2: pp. 85–116.

Winston, P.H., 2009, Artificial Intelligence, Addison Wesley, 3rd Edition.

Y. Aoki, A. Shio, H. Arai, and K. Odaka, August 1996, 'A Prototype System for Interpreting Hand-Sketched Floor Plans, in *Proceedings of the 13th International Conference on Pattern Recognition*, Vienna (Austria), volume 3, pp. 747–751.

Yoichiro Endo, Ronald C. Arkin, 2003, 'Anticipatory Robot Navigation by Simultaneously Localizing and Building a Cognitive Map', *Proc. Int'l Conf. Intelligent Robots and Systems*, pp. 460-466.

