

# **Robot Mapping Without A Precise Map**

**Zati Hakim Azizul Hasan**

A thesis submitted to  
Auckland University of Technology  
in fulfilment of the requirements for the degree of  
Doctor of Philosophy (PhD)

**2013**

School of Computer and Mathematical Sciences

# TABLE OF CONTENTS

<b>ATTESTATION OF AUTHORSHIP .....</b>	<b>IX</b>
<b>ACKNOWLEDGEMENTS .....</b>	<b>X</b>
<b>ABSTRACT .....</b>	<b>XI</b>
<b>1 INTRODUCTION.....</b>	<b>1</b>
1.1 The Notion of a Cognitive Map .....	1
1.2 Cross-Fertilisation between Cognitive Mapping and Robot Mapping .....	3
1.3 Yeap's Theory of Cognitive Mapping.....	5
1.4 Work Done .....	8
1.5 A Guide to this Thesis .....	11
<b>2 LITERATURE REVIEW.....</b>	<b>13</b>
2.1 Overview .....	13
2.2 Theories in Cognitive Mapping.....	13
2.2.1 Development Theory .....	14
2.2.2 Neurological Theory .....	19
2.2.3 Computational Theories of Cognitive Mapping .....	21
2.3 Implementing Cognitive Theories on Robots.....	32
2.3.1 Spatial Semantic Hierarchy .....	32
2.3.2 R-PLAN.....	38
2.3.3 Absolute Space Representation .....	41
2.3.4 Neural Cognitive Maps.....	47
2.4 Traditional Robot Mapping.....	57
2.4.1 Metric Maps in SLAM.....	58
2.4.2 Topological Maps in SLAM.....	62
2.4.3 Hybrid Maps in SLAM.....	64
2.5 Chapter Summary.....	67
<b>3 METHODOLOGY .....</b>	<b>69</b>
3.1 MFIS Computation.....	69
3.2 The Robot: Its Sensors and Input .....	70
3.3 An Algorithm for Autonomous Exploration .....	75
3.4 MFIS Computation.....	82
3.5 ASR Computation .....	98
3.6 Conclusion .....	105
<b>4 RESULTS .....</b>	<b>107</b>
4.1 Overview .....	107
4.2 Experiment 1 (Going clockwise).....	108
4.2.1 Computing the MFIS and ASRs .....	108
4.2.2 Closing the loop.....	118
4.2.3 Going around the second time .....	121
4.2.4 Going Home .....	124
4.3 Experiment 2 (Going anti-clockwise) .....	126
4.3.1 Computing the MFIS and ASRs .....	126
4.3.2 Closing the loop.....	128
4.3.3 Going Home .....	130
4.3.4 Going around the second time .....	132
4.4 Experiment 3 .....	135
4.4.1 Computing the MFIS and ASR.....	135
4.4.2 Go To ASR .....	139
4.4.3 Home finding when known route is blocked .....	142

4.4.4	Novel short-cutting .....	144
4.5	Conclusion .....	145
<b>5</b>	<b>CONCLUSION .....</b>	<b>146</b>
	<b>REFERENCES .....</b>	<b>149</b>

## LIST OF FIGURES

Fig. 1.1: The ASR model of the cognitive map. Reproduced from Fig. 1 of Yeap (2007) .....	5
Fig. 1.2: The environment used for testing. Highlighted is the path travelled by the robot which is about 30m by 30m in size. Arrows denote an example of the robot moving in an anti-clockwise manner .....	9
Fig. 1.3: An example of the MFIS computed for the environment .....	10
Fig. 1.4: An example of a network of ASRs computed for the environment .....	10
Fig. 2.1: (a) The egocentric (self-to-object) spatial representation, (b) the allocentric (object-to-object) spatial reference .....	17
Fig. 2.2: Possible novel short-cutting when human configures new routes to get between goals in the environment .....	20
Fig. 2.3: An example of the centroid-slope model. Reproduced from O'Keefe (1991) .....	21
Fig. 2.4: The TOUR computational model of cognitive map. Reproduced from Kuipers (1978) .....	23
Fig. 2.5: The PLAN computational model inspired by Figure 2 and 3 in Chown et al. (1995) .....	29
Fig. 2.6: Example of an indoor map and the ASRs computed .....	31
Fig. 2.7: The logical dependencies and constraints (red arrows) in the SSH Model. Blue arrows denote potential information flow without dependencies. Reprinted from Kuipers (2008) .....	35
Fig. 2.8: (a) Simulated exploration by the NX robot highlighting the distinct paths and places, (b) parts of the topological map which defined the relation between paths and places, (c) the global metric map produced. All images reprinted from Kuipers (2000) .....	36
Fig. 2.10: (a) An environment with multiple nested loop, (b) the LPMs computed at each place and tagged with a number, (c) connecting the LPMs via their gateways. Adapted from Kuipers et al. (2004) .....	38
Fig. 2.11: Computation of a global metrical map (right) from what is initially a topological representation of the environment (left). Reprinted from Kuipers (2008) .....	38
Fig. 2.12: Samples of types of indoor gateways in R-PLAN; (a) left room entrance, (b) room entrance, (c) left corner, (d) right room exit, (e) T-shape gateway, (f) four corners gateway, (g) left opening gateway and (h) right opening gateway. Reprinted from Kortenkamp (1992) .....	39
Fig. 2.13: Storage of visual cues into a scene (ASR). (b) The configuration of scenes for place identification. Reproduced from Kortenkamp & Weymouth (1994) .....	40
Fig. 2.14: (a) Raw laser points separated at 180-angle to each other and (b) computing the corners by breaking raw laser points at any right-angle found. Reprinted from Chown & Boots (2008) .....	41
Fig. 2.15: Example of exit detections. G1 and G2 are the gaps detected because surfaces S1 and S3 are occluding surface S2. G3 is a gap because surface S5 is occluding S4. All gaps have one occluding point (marked as filled circles) and one occluded point (marked as empty circles). Connecting the two occluding points of G1 and G2 creates the first exit E1. Connecting the two occluding points between G2 and G3 creates the second exit E2. ....	43
Fig. 2.16: (a) Example of a network of ASRs computed with robot currently in ASR3 (b) The MFIS produced with robot about to revisit ASR7 from ASR3. Since ASR7 is identified, no new ASR is formed. New link is established between ASR7 and ASR3. Reproduced from Jefferies et al. (2008) .....	45
Fig. 2.17: (a) The large corridor-like environment for testing, (b) map for the outward journey with 10 ASRs computed, and (c) map for the homeward journey with 8 ASRs computed. Reproduced from Wong et al. (2007) .....	46
Fig. 2.18: The neural network structure for visual homing. All weights at the output layer which triangulates home is adjusted using the Hebbian learning. Reproduced from Hafner (2001) .....	48
Fig. 2.19: (a) aMouse with a single camera and real rat whiskers, and (b) neural network with neurons at output layer represents places in the topological map. Reproduced from Hafner (2008) .....	48
Fig. 2.20: (a) and (b) represent the environment in grayscale format; black areas are obstacles and different gray shadings indicate different place fields in the environment, (c) denotes the	

connections between places at random parameters, and (d) the interconnectivity with optimal parameters. Reproduced from Hafner (2005) .....	50
Fig. 2.21: Sketch of the model. From left to right: merging landmarks (Pr) and their azimuth (Ph) in a matrix of neurons called product space, then learning of the corresponding set of active neurons on a place cell (ECs). Two successive place cells define a transition cell (CA). Place cell at time t-1 is in DG. Transitions are used to build the cognitive maps (PF) which are linked with movements (ACC). Diagram and description reprinted from Cuperlier et al. (2007) .....	51
Fig. 2.22: An example of extracting landmarks and its azimuths from an agent's visual input. Reproduced from Cuperlier et al. (2007) .....	53
Fig. 2.23: The cognitive map computed using the transitions information. The triangles denote the robot's localisation and direction at the point of exploration. Reproduced from Cuperlier et al. (2006) .....	53
Fig. 2.24: Building the robot's experience map using pose cells and local view cells. Reproduced from Milford (2007) .....	55
Fig. 2.25: (a) Representation of the pose (position and orientation) in a 3D network with the wrap-around leads to grid-like response, (b) shows the connection between all three representations. Reprinted from Milford et al. (2007) .....	55
Fig. 2.26: An example of the RatSLAM results in outdoor environment. Picture and results reproduced from Milford et al. (2008) .....	56
Fig. 2.27: (a) Two sonar beam readings, (b) both readings projected onto same grid map, (c) and (d) show readings on individual probabilistic map, (e) conflicting regions on grid when probabilistic maps are compared, and (f) final map with removal of conflicting region. Reproduced from Thrun (2003) .....	61
Fig. 2.28: Grid mapping in large environment. (a) Raw sonar data over 50m corridor-like environment, (b) map with missing walls and doors in traditional grid map, (c) map with completed walls and doors in advanced grid maps. Reproduced from Thrun (2003) .....	61
Fig. 2.29: Example of a GVG topological solution in performing robot's localization. Readings from 8 range-finders are used to estimate the nearest obstacles with H and C the two lowest values in the sensor array (A is not considered as it is not a local minimum). When two adjacent sensors have similar values, the closest obstacle is assumed to be in between the two sensors. Reprinted from Choset & Nagatani (2001) .....	64
Fig. 2.30: Example of a hybrid map. The environment is represented with global topological nodes and local metric places. When travelling from one node to another, the system switches from topological to metric and vice versa. Reproduced from Tomatis et al. (2003) .....	66
Fig. 3.1: The laser taking samples from the environment at $0.5^\circ$ angular resolutions with $180^\circ$ scanning field. Drawings of the narrow laser beams are an approximate and not to scale .....	72
Fig. 3.2: Processes involved in extracting surfaces from the environment .....	75
Fig. 3.3: Gaps (G1-G7) found in a view. The robot is at (0, 0). Circles indicate small gaps that are ignored .....	76
Fig. 3.4: Creating a new gap for G1 and G2 will produce a gap that is outside the current bounded space. Such new gaps are illegal .....	79
Fig. 3.5: (a) The view, (b) with gaps identified, (c) first iteration with G3 and G4 replaced and gaps renamed, (d)-(e) two final iterations. (f) Removal of G1 and G2 as they are not made of two occluding points .....	80
Fig. 3.6: (a) Two exits are identified in the initial view in addition to the gaps; (b)-(c) show two iterations of the algorithm, and (d) shows the final output .....	81
Fig. 3.7: (a) The view, (b) with gaps computed, (c) first iteration which produces an exit E1, (d)-(e) two final iterations, and (f) the final output .....	81
Fig. 3.8: Processes shaded in orange summarises the flow where the robot pre-process the laser scans, make decision on the next target before planning its movement whereas the process shaded in purple denotes where and when the spatial perception computation occurs in the implementation .....	82
Fig. 3.9: Transferring of landmarks ID from the MFIS to $V_{n-1}$ .....	87

Fig. 3.10: (a) Candidate in green being paired with a smaller comparing surface, (b) perpendicular lines generated on the comparing surface, (c) markers X denote the intersections between perpendicular lines and candidate, (d) match is verified after angle between the lines is found satisfactory, (e) candidate inherit the ID .....	89
Fig. 3.11: Finding landmarks whereby the surface perceived in $V_{n-1}$ (single line in red) has changed its shape drastically in the current view (split into 3 green lines) .....	89
Fig. 3.12: Finding landmarks: (a) $V_n$ (surfaces in green) combined with $V_{n-1}$ (surfaces in red and are labelled), and (b) Landmarks identified (IDs 7, 8, 9, 11, 12, and 13) are assigned the same ID .....	90
Fig. 3.13: Computing the spatial locations of surfaces close to a landmark: S1 is recognised as a landmark and S2 and S3 are coded using two pairs of vectors centred on the right end-point of S1 .....	91
Fig. 3.14: (a) and (b) show which end-point of the landmark surface (ID 7) is chosen as its centre of co-ordinate system respectively in $V_n$ and $V_{n-1}$ , and (c) shows the calculation of the position of the unknown surface, $U_3$ , and (d) shows where it is positioned. In this case, the position is roughly correct .....	93
Fig. 3.15: (a) and (b) show which end-point of the landmark surface (ID 7) is chosen as its centre of co-ordinate system respectively in $V_n$ and $V_{n-1}$ , and (c) shows the calculation of the position of the unknown surface, $U_3$ , and (d) shows where it is positioned. In this case, the position is seriously distorted .....	93
Fig. 3.16: Normalisation of landmark surfaces – Perpendicular lines are generated at the end-points of both surfaces and the two surfaces are normalised to become of equal length. ....	94
Fig. 3.17: Three examples of normalisation in a robot's view: – The left column shows the projection of $V_{n-1}$ onto $V_n$ and landmarks (circled surfaces) are identified, and the right column shows the landmarks that are normalised. ....	94
Fig. 3.18: Examples of choosing the nearest landmarks to localise unknown surfaces. New surfaces $U_1$ , $U_2$ and $U_3$ are closest to landmark ID7, new surface $U_4$ to landmark ID9, and new surfaces $U_5$ , $U_6$ and $U_7$ to landmark ID11 .....	95
Fig. 3.19: (a) Processes shaded in orange summarises the flow where the robot pre-process the laser scans, make decision on the next target before planning its movement whereas the box shaded in purple denotes where spatial mapping takes place. The flowchart diagram in (b) breaks down the spatial mapping processes in (a). Blue boxes are processed in the working memory whereas red boxes indicate updating in the MFIS .....	97
Fig. 3.20: (a) The green line denotes the robot's path, E1 denotes the exit crossed and the arrow points at the exit-path line intersection, (b) the virtual boundary created perpendicular to the path line slope at the exit-path line intersection where surfaces circled are considered beyond the exit thus are not included in the ASR being computed .....	100
Fig. 3.21: (a) Surfaces shown connected to the robot's current position (path point ID 15) via a dashed line can be eliminated according to algorithm #7. These are surfaces perceived after crossing the exit of the previous ASR. (b) The remaining surfaces for ASR computation .....	101
Fig. 3.22: Computing the boundary using the outermost surfaces perceived. ....	101
Fig. 3.23: (a) The initial spatial information for the first local space boundary computation where two surfaces (in yellow circles) fails the corner point test. (b) The large segments are filtered from the MFIS 15 surface set whereas the singular points denote the corner points derived from the smaller surfaces .....	104
Fig. 3.24: (a) The complete robot mapping system developed in this thesis. (b) Where ASR boundary is computed in the robot system .....	106
Fig. 4.1: ASR 1 surfaces (shaded) extracted from MFIS .....	110
Fig. 4.2: Corner points and large surfaces for ASR 1 .....	110
Fig. 4.3: ASR 1 boundaries computed. Red lines denote the large surfaces which have been extracted from the MFIS. The blue lines are boundaries computed as a result of joining adjacent surfaces and corner points together .....	110
Fig. 4.4: ASR 2 surfaces (shaded) extracted from the MFIS .....	111
Fig. 4.5: Corner points and large surfaces for ASR 2 .....	111
Fig. 4.6: Network of ASR showing ASR 1 and ASR 2 .....	111


Fig. 4.7: ASR 3 surfaces extracted from the MFIS .....	112
Fig. 4.8: Corner points and large surfaces for ASR 3 .....	112
Fig. 4.9: Network of ASR showing ASR 1 to ASR 3 .....	113
Fig. 4.10: ASR 4 surfaces extracted from the MFIS .....	113
Fig. 4.11: Corner points and large surfaces for ASR 4 .....	114
Fig. 4.12: Network of ASR showing ASR 1 to ASR 4 .....	114
Fig. 4.13: ASR 5 surfaces extracted from the MFIS .....	115
Fig. 4.14: Corner points and large surfaces for ASR 5 .....	115
Fig. 4.15: Network of ASR showing ASR 1 to ASR 5 .....	116
Fig. 4.16: MFIS computed just after the robot crosses E6 and re-enters ASR1.....	116
Fig. 4.17: ASR 6 surfaces (shaded) extracted from the MFIS .....	117
Fig. 4.18: Surface and corner point for ASR 6 .....	117
Fig. 4.19: Network of ASR showing ASR 1 to ASR 6 .....	118
Fig. 4.20: MFIS surfaces computed inside ASR1. E6 has been computed on surface ID 8 and the robot will go through the exit computed and revisit ASR1.....	119
Fig. 4.21: Comparison between $V_n$ and $V_{n-1}$ before crossing E6. Robot is at (0, 0). (a) U1 matches landmark 172 and (b) the normalisation process .....	120
Fig. 4.22: Updating before the robot crosses E6. (a) Showing U3 being matched to an old landmark ID 22 and (b) the normalisation process .....	120
Fig. 4.23: The MFIS (top) and the network of ASRs (bottom) after looping the environment twice	122
Fig. 4.24: (a) and (b) depict the MFIS and the network of ASRs built in the first round. (c) and (d) are the MFIS and ASRs computed in the second round. Changes is apparent to ASR6, ASR5 and ASR1 .....	123
Fig. 4.25: Example of using direction to navigate home. R1, R2 and R3 are three steps in the navigation. Direction are recalculated at each step until the robot reach closer to home	124
Fig. 4.26: Black path lines indicates the robot's trajectory to home after crossing E6 .....	125
Fig. 4.27: The network of ASRs after looping twice and the robot is instructed to go home .....	125
Fig. 4.28: The MFIS computed from home until the robot crosses the exit E6 .....	127
Fig. 4.29: The network of ASR depicting ASR1 to ASR6 in the second experiment.....	127
Fig. 4.30: Some of the earlier MFIS surfaces computed inside ASR1 .....	128
Fig. 4.31: Comparison between $V_n$ and $V_{n-1}$ before crossing E6. Robot is at (0, 0). (a) Denotes U1 and U5 matching the landmarks 119 and 121 respectively. Results of the normalisation is shown in (b).....	129
Fig. 4.32: Updating the MFIS before crossing the exit E6 .....	129
Fig. 4.33: Updating after crossing the exit E6 where three current surfaces matching the centroids 12, 13 and 27 .....	130
Fig. 4.34: The MFIS when the robot reaches home. Path lines in black denotes the steps towards home .....	131
Fig. 4.35: Changes to ASR1 after the go home activity.....	131
Fig. 4.36: MFIS after traversing the environment the second time. The robot ended the journey at home .....	133
Fig. 4.37: The network of ASR after the second looping .....	133
Fig. 4.38: Three sets of MFIS and its corresponding network of ASRs. (a) and (b) for going about in the first round, (c) and (d) for going home, and (e) and (f) indicate the representations when the robot is instructed to go around again .....	134
Fig. 4.39: The MFIS computed once the robot closes the loop after crossing E6.....	136
Fig. 4.40: Network of ASRs after the robot crosses E6 and into ASR1.....	136
Fig. 4.41: The MFIS as the robot goes home after entering ASR1 from ASR6 .....	137
Fig. 4.42: ASR1 is expanded due to the robot seeing surfaces behind the robot when it first started	137
Fig. 4.43: MFIS after completing the second loop in the opposite direction. 'Robot' indicated where the robot is after completing the second loop .....	138
Fig. 4.44: Network of ASRs after the robot closes the second loop and re-enters ASR1 via E1 .....	138
Fig. 4.45: Strategy to navigate towards E6 and E5 .....	140
Fig. 4.46: MFIS as the robot travels from ASR1 to ASR5 via E6 and E5 (in that order).....	140
Fig. 4.47: Choosing new exit to go to ASR3 .....	141
Fig. 4.48: The MFIS updated with E7 and E8 .....	141

Fig. 4.49: New ASR7 and its location inside the network of ASR .....	142
Fig. 4.50: Locations of new found exits E9 and E10 in the MFIS. Dotted line is where the physical barrier is put up so the robot cannot choose to re-enter ASR1 via E1 .....	143
Fig. 4.51: Network of ASR depicting the new ASR8 .....	143
Fig. 4.52: Utilizing new connection on the network (ASR8) to reach ASR2 from home when known exit E1 is inaccessible. Dotted line is where the physical barrier is put up.....	144



### **ATTESTATION OF AUTHORSHIP**

“I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.”

Signature	:	
Name of Candidate	:	Zati Hakim Azizul Hasan
Student ID	:	0815129
Date	:	16 May 2014

## **ACKNOWLEDGEMENTS**

I am most grateful to the Almighty for inspiring and guiding this humble being

To my supervisor, anything that moved me leaps and bounds ahead has come from you. Thank you, Albert, for this gift of knowledge. You're the best mentor one could hope for. To my co-supervisor, you have provided invaluable perspective to my research whenever needed. Thank you, Chee Kit for all your support.

My CAIR mates; Bobby, Thamilini, Tommi, Siew Cheng and Hossain, have without exception been most supportive. To Jordan, thank you for always being there when I needed advice particularly the time spent in Penrose. All of you have been partial to this journey and may the friendship built long lasting.

Special thanks to Mohammad Hafizullah and Robin Cyril who came in to help with the earlier testings. You guys were a joy to work with.

To my parents; Azizul and Nora, and siblings; Ainul, Rizal, Ida and Uniza, my deepest appreciation for your understanding, prayers and endless love. You reminded me why I chose this journey and to you I dedicate this thesis.

To my in laws; Karim and Salmah, and aunt Rosnah, thank you for the care and support especially during the writing of this thesis. I am forever indebted to you with the care of little Adam.

Last but not least, my better half, Shah, thank you for sharing every step of this journey with me. You were a beacon in the dark. I could have not done it without you, love.

The work presented in this thesis was funded by the SLAI Fellowship Scheme from the University of Malaya and the Ministry of Higher Education Malaysia

## ABSTRACT

In this thesis, I took two key ideas of cognitive mapping developed in Yeap's (1988) theory of cognitive mapping and developed a robot mapping system that maps without having a precise map.

Yeap argues that in a cognitive mapping process, it is important to compute a local space representation that affords boundedness and a global map that tells one *roughly* where things are located with respect to the self. While these two representations appear to be similar to the global map and a topological network of local spaces that robotics researchers compute for their robots, there are two major differences. First, Yeap's global map is a transient, inexact map and second, the local space computed is often inexact and incomplete. Computing such representations meant that one does not need to correct errors due to sensors and generate an exact map.

I have successfully developed one such algorithm and tested it successfully on a mobile robot equipped with laser and odometer sensors in a large office environment. The journey through the environment before loop closing is about 30m x 30m. The robot went round the environment twice and in a clockwise and anti-clockwise direction. It also finds its way from one part of the environment to another.

There are three key steps in my approach. The first step is to constantly detect "landmarks" in two consecutive views (the current view and the previous view). Having two consecutive views meant that any errors due to the sensors are not accumulative.

Furthermore, one gets two copies of the landmark – one currently in view and the other in memory. Consequently, their position in space need not be absolute. The second step is to use the landmarks identified to provide a frame of reference to localize unknown surfaces that appear in the current view. The third is to enter those unknown surfaces into its global map using its own landmarks.

The development of such an algorithm has led to better insights into cognitive and robot mapping. From a cognitive standpoint, what is important is that we now have an algorithm that computes an inexact map by attending to recognizable surfaces (referred to as landmark surfaces) in successive views rather than dependent on continuous tracking of one's position and orientation in the environment. Furthermore, it does not require continuous updating of the map as long as there are some overlapping surfaces between views. Both are characteristics of the human cognitive mapping process. From a robot mapping standpoint, my new algorithm shows that it is possible to compute and utilize an inexact map for navigation. This could be a new paradigm for robot mapping.

# 1

---

## Introduction

---

### 1.1 The Notion of a Cognitive Map

As we move, our interactions with the environment allow us to encode the what and where of things seen. With it, we could retrieve the remembered locations of objects even though they are now out of sight or recognise them if they are in view again. Consequently, we know what lies immediately behind us and can predict what might appear in front as we move forward. These activities of sensing, encoding, storing, decoding and deploying the spatial information are all part of one's spatial process. Tolman (1948) first inferred that a map-like representation emerges in this process in animals other than humans. He demonstrated this by recording the spatial behaviour of maze-running rats that were able to take short cuts to their final destination. He called this act of mental structuring process leading to the creation of the rat's internal spatial model as *cognitive mapping*, and coined the internal model produced as *cognitive map*.

Recognition of this place learning activity stimulated multidisciplinary research in the human spatial knowledge acquisition. In city planning, geographers Kevin Lynch (1960) used sketch maps to reveal human knowledge of large-scale complex environments. He emphasised spatial visibility as the means by which people extend their knowledge about the city landscape as well as engaging in way finding. The environmental image is said to be the product of immediate sensation and the memory of past experience, and it is used to interpret information and to guide action. In brain studies, neurophysiologists studied the role of the hippocampus in navigation (O'Keefe & Nadel, 1978) and continued their in-depth research in breaking down the anatomy of memory into quantitative models by reproducing detailed computational representations based on the specific changes in cellular firing rates after complex environmental manipulations (Zipser, 1985; Burgess et al., 1994; Touretzky & Redish, 1996).

In cognitive psychology, the nature of mental map revolves around understanding how human process information. The Multi Store Model (Atkinson & Shiffrin, 1968) is a classic model studying the internal process of the mind by describing the human memory in terms of information flowing through a system. The model suggests that the human memory is made up of a series of stores; the sensory memory (SM), the short-term memory (STM) and the long-term memory (LTM), but the lack of a mechanism to prove encoding of memory occurred when memories are being rehearsed, retrieved and transferred between the three stores led to newer memory models eliminating the SM (Raaijmakers & Shiffrin, 1981; Baddeley, 2003).

## 1.2 Cross-Fertilisation between Cognitive Mapping and Robot Mapping

Advances in artificial intelligence research that led to the development of the first integrated mobile robot Shakey (Nilsson, 1969) presented a new and exciting paradigm for studying the mapping problem. When faced with the problem of creating a map of the robot's environment, these researchers quickly realised that there is a need to *simultaneously* locate and map. Otherwise, one cannot integrate successive views to form a map. Researchers investigating how humans/animals compute their map often overlook this part of the problem. For example, a popular early theory of cognitive mapping (Siegel & White, 1975) suggests that what is remembered initially are landmarks only. Robotics researchers thus refer to the mapping problem as SLAM, simultaneous localisation and mapping (Leonard & Durrant-Whyte, 1991).

In robot mapping, the robot localises itself via updating its Cartesian position in the map and then using its known position at each step to overlay, re-align, and add subsequent views to the map. This is a fundamental step in the mapping process. However, it turns out that there is much noise present in the process and this distorts the localisation step. Errors accumulate quickly and the map created will soon be rendered useless. This problem, correcting the errors to get a precise metric map, became the focus of robotics research and many probabilistic solutions (such as those using Bayesian formula (Majumder et al., 2000), the extended Kalman filter or EKF-SLAM (Newmann et al., 2002), Rao-Blackwellized particle filters (Murphy, 1999; Doucet et al., 2000) were proposed. Rigorous testing and development were done on wheeled mobile robots with either sonar and/or laser sensors (Chong & Kleeman, 1999; Castellanos & Tardos,

1999; Diosi et al., 2005; Newmann et al., 2006; Zhao et al., 2008; Kretzschmar et al., 2011). The robotics researchers have had great success which meant that the problem is now well understood (Thrun, 2002; Durrant-Whyte & Bailey, 2006; Bailey & Durrant-Whyte, 2006; Aulinas et al., 2008) and the field is advancing to develop robots mapping outdoor environments and using other sensors, notably vision (Jensfelt et al., 2006; Nuchter & Surmann, 2007; Milford & Wyeth, 2008; Maddern, Milford & Wyeth, 2012; Wendel & Bischof, 2013). The current success of robotics researchers, however, highlights a major point of difference between robot mapping and cognitive mapping, namely the latter is unlikely to produce and use a precise map.

As Jefferies and Yeap (2008, p.1) noted, “it is not surprising that the cognitive and robot mapping problems share some common core problems” and “one would reasonably expect some cross-fertilisation of research between the two to have occurred”. To date, the lessons learned in robot mapping provide significant insights into the nature of the low-level mapping problem and observations made about cognitive mapping provide significant insights into the nature of the map computed. The availability of more powerful robots for autonomous exploration and mapping provide a platform for testing cognitive mapping theories, and, in turn, the richer cognitive mapping theories being developed lead to the development of more powerful mapping algorithms for robot use.

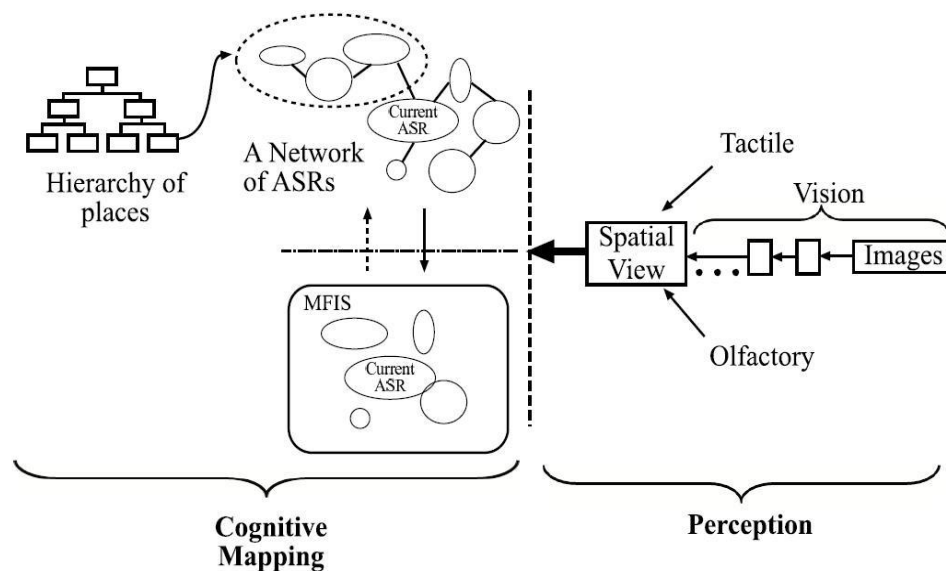
This thesis contributes to the cross-fertilisation of ideas between the two fields by developing a cognitive mapping algorithm (Yeap, 1988; Yeap & Jefferies, 1999) on a mobile robot that computes a key characteristic of cognitive maps, namely that such a



map is inexact. Given that robotics researchers are currently focusing on developing algorithms that compute a precise map, developing such an algorithm could lead to a paradigm shift in robotics. An earlier student's work (Wong, 2008) has shown that an inexact map extracted directly from the robot's sensors could still help the robot to find its way home albeit in a limited corridor-like environment. In this thesis, I ask: Can a robot compute an inexact map itself and how useful is such a map? For instance, can the robot use the map to perform various spatial tasks such as taking a short-cut, recognizing that one is returning to a familiar place, etc. in its environment?

### 1.3 Yeap's Theory of Cognitive Mapping

Briefly, in Yeap's theory of cognitive mapping, three key representations are computed, namely a network of ASRs (which stands for an Absolute Space Representation), an MFIS (which stands for Memory For one's Immediate Surroundings), and a hierarchy of place representations (see Fig. 1.1).



**Fig. 1.1:** The ASR model of the cognitive map. Reproduced from Fig. 1 of Yeap (2007)

The first representation, an ASR, is a representation that makes explicit the local environment that one is in. Its significance is that while we live in a universe that is ever expanding (Carroll et al., 1992), we are, due to the limit of our own perceptual apparatus, always bounded in a local space that limits where we can go and what we can see. To us, such a space is absolute and forms the cornerstone element for building our cognitive map.

The second representation, an MFIS, is a representation that makes explicit one's immediate surroundings within a single co-ordinate system. Our ability to use visual cues to remain oriented to our surroundings and to reorient ourselves when we get lost (e.g. Mou et al., 2004; Cheng & Newcombe, 2005) supports the presence of such a representation. Note that an ASR is inadequate as a representation of one's immediate surroundings for two reasons. First, an ASR is a representation that indicates where one is but not necessary where one has been (except within the confine of that local space). Second, an ASR does not hold information beyond what is in one's local environment and therefore things perceived outside its boundary are not represented.

The third representation is a hierarchy of place representations (Hirtle & Jonides, 1985); each place formed represents one's conceptual view of a part of the environment. Each place, if it is not an imaginary one, will be grounded to its physical environment as a collection of ASRs computed. Humans have shown that some form of hierarchical organisation is in order when they learn about their environment. For example when human subjects are asked to relay their experiences traversing a known route, they tend

to organise their spatial representation in clusters of landmarks (Reitman & Reuter, 1980). Humans are also shown to learn an unstructured environment such as a map layout containing uniformly distributed objects by encoding the environment into a hierarchy of place by associating each place with the spatial proximity to the neighbouring landmarks (McNamara, Hardy & Hirtle, 1989). Although these studies present evidence about the organisation of place representations in our spatial knowledge, they do not provide information on how the hierarchical structure is built and used.

The theory argues that an ASR is computed immediately as one enters a new local environment but given one has only a limited view on entry; it is not possible to identify its complete boundary without exploring the space further. If so, during exploration, how does one distinguish one ASR from another? Jefferies (1999) argued that “exits” of an ASR play an important role in both identifying and computing an ASR. Exits are not doorways but any perceived gaps that the individual believes will allow them to “escape” out of the current bounded space. Wong (2008) showed that the spatial extent of each ASR and in particular between exits, though imprecise, could provide very useful information for traversing an environment using a network of ASRs.

According to Yeap’s theory of cognitive mapping, there are thus three key representations, namely, a network of ASRs, an MFIS, and a Hierarchy of Place representation. The MFIS and a network of ASRs bear a strong resemblance to the global map and a topological network representation computed in robotics research.

However, there are two major differences. First, the MFIS is a transient inexact map and second, the ASR is often an inexact and incomplete description of local spaces. The advantage of having an exact map is that the robot knows exactly where things are but the disadvantage is that such a map is not easy to maintain in the real world. The physical environment could often be altered significantly and the robot could be seriously displaced during its journey through an environment. Nature has shown that we could live with an inexact representation and it must have discovered other ways to cope with errors present in the sensors. In this thesis, I developed one such algorithm and tested it on a mobile robot equipped with laser sensors.

#### **1.4 Work Done**

In this thesis, I implemented a complete robot mapping system that explores its environment autonomously and computes both an inexact map and a topological network of incomplete and inexact local spaces without error corrections. Few robotics works, if any, generate such representations for their robots since they claim that without error corrections, the map will be distorted.

The robot has been tested going through the environment as shown in Fig. 1.2 several times. The experiments conducted demonstrate successful loop closing and successful use of the inexact map for performing various spatial tasks such as returning home, find short cuts, and re-plan new routes. An important feature of the algorithm developed is that it does not correct errors. Rather, it detects “landmarks” that are present from one view to another and via them, creates multiple frames of reference that could be used to

update new surfaces to the MFIS. An example of the kind of map generated is as shown in Fig. 1.3. As can be seen from the output, the map is neither exact nor precise but it is good enough for the robot to use for finding its way in the environment. The latter is aided with a network of ASRs extracted from the MFIS as shown in Fig. 1.4. An ASR is extracted from the MFIS each time the robot crosses an exit in the journey. To compute a boundary for the ASR, the algorithm follows the outermost surfaces starting from one end-point of the crossed exit to its other end-point, thereby merging the gaps in between these surfaces to enclose the space traversed by the robot. The ASRs are then connected to one another using their common exits. Using both the MFIS and a network of ASRs, the robot successfully traversed to different parts of the environment, re-visit places, find alternative routes and can take short-cuts.



**Fig. 1.2:** The environment used for testing. Highlighted is the path travelled by the robot which is about 30m by 30m in size. Arrows denote an example of the robot moving in an anti-clockwise manner

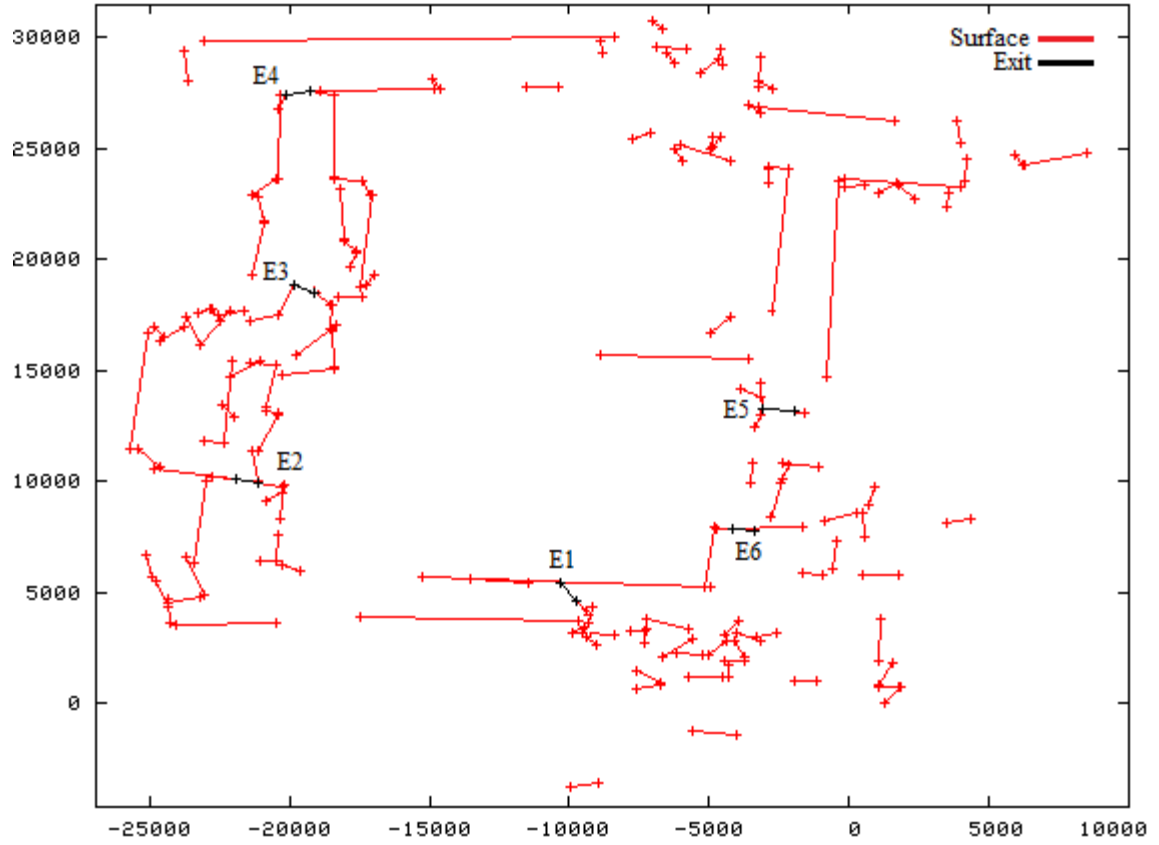


Fig. 1.3: An example of the MFIS computed for the environment

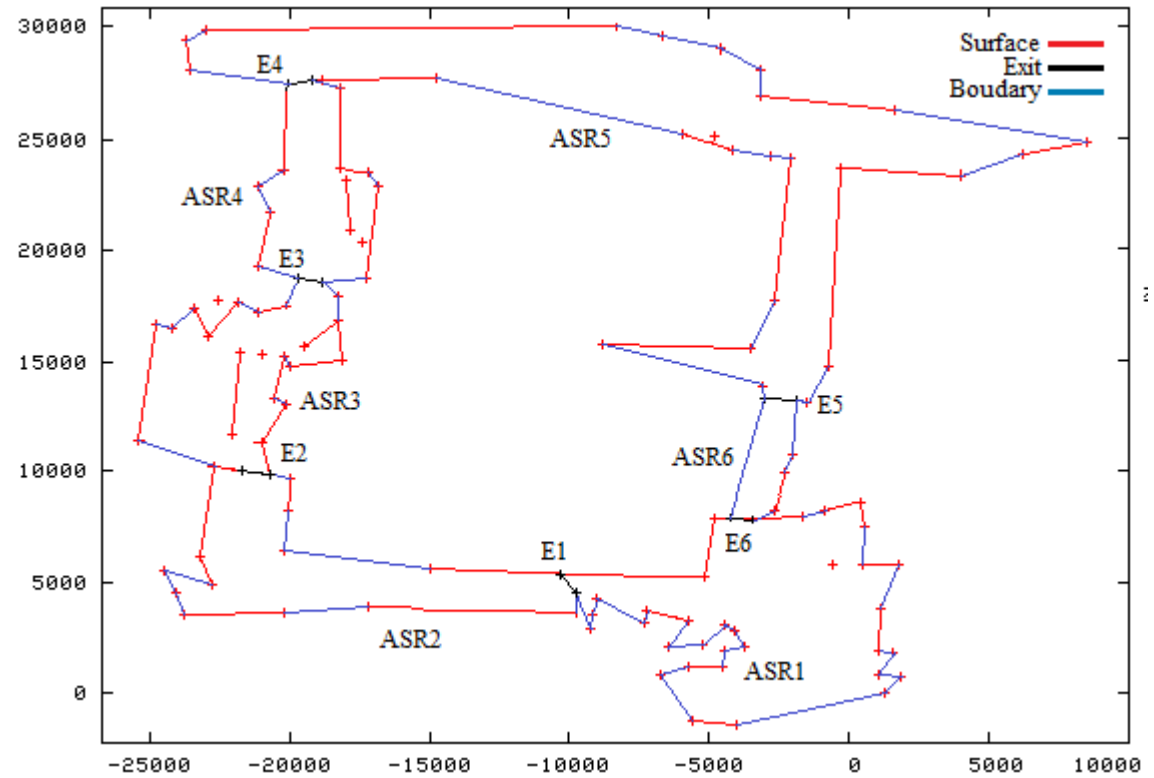


Fig. 1.4: An example of a network of ASRs computed for the environment

## 1.5 A Guide to this Thesis

Chapter 2 provides a review of the relevant literature. While this thesis focuses on robot mapping, the review begins with a brief discussion of ideas in cognitive mapping found in three different major research areas, namely psychology, neurophysiology, and computational studies. Then, the review continues with works that develop some interesting algorithms for robot mapping based on cognitive ideas. For completeness, the review ended with a discussion on traditional SLAM approaches that compute exact metrical maps. The latter would provide a contrast to the implementations done using cognitive ideas.

Chapter 3 focuses on the development of the eight new algorithms for robot mapping without a precise map. These algorithms include:

1. An algorithm for autonomous exploration;
2. An algorithm for computing minimal bounded space;
3. An algorithm for MFIS computation;
4. An algorithm for finding landmarks in view;
5. An algorithm for transferring surfaces in the current view to the MFIS;
6. An algorithm for computing ASRs;
7. An algorithm for eliminating new surfaces coming into view after crossing an exit;
8. An algorithm for identifying surfaces and corner points for boundary computation;

Chapter 4 presents the results of three major experiments using the robot embedded with the seven new algorithms described in chapter 3. The first experiment sees the robot build the MFIS and the network of ASRs while exploring the environment twice in a clockwise direction. Going round the environment twice tests the robot's ability to close loop and the effect on the representations computed when re-visiting. In the second experiment, the robot is assigned similar tasks to the first experiment but the direction of travel is reversed. This ensures that the representations computed are not direction dependent. In the third experiment, the robot explores the environment in a clockwise direction and then in an anticlockwise direction. It is also instructed to find its way from one ASR to another and in two occasions, the known routes are blocked.

Chapter 5 concludes the thesis with a general discussion of the lessons learned, a summary to major contributions and points towards future work.



---

# Literature Review

---

## 2.1 Overview

This chapter reviews literature related to this study. It has three sections. Section 2.2 discusses ideas about cognitive mapping found in cognitive and neurosciences and the computational models that have been proposed based upon these studies. Section 2.3 discusses studies that implement cognitive ideas/models using robots. For completeness, Section 2.4 provides a review of work done in traditional robotics.

## 2.2 Theories in Cognitive Mapping

The mental representation of the spatial environment is referred to as a cognitive map. Tolman (1948) first coined the term. He observed the behaviour of rats in a maze-like environment and noticed that rats did not respond to stimulus while navigating but also produced something similar to a field map of the environment in their brains. This internal map is what helped the rats choose alternative correct routes when known

routes are inaccessible leading to assumptions that rats remembered their environment and able to perform novel connections between learned routes and routes they have never travelled before. For humans, a cognitive map is a more complex representation and computed from “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 his everyday spatial environment” (Downs & Stea, 1973).

The subject of humans’ and animals’ spatial knowledge has also been the interest of other researchers from various other backgrounds; all keen to understand how cognitive agents sense and acquire a representation of their environment. Developmental psychologists studied how spatial cognition develops from children to adulthood. Neuropsychologists studied how the brain in general and the hippocampus in particular computes its map. Behavioural scientists focus their study on how humans and animals find their way in the environment given the diverse capacities of their sensors and survival needs. Even architects, inspired by the famous work of Lynch (1960) studied how people form images of their city.

### **2.2.1 Development Theory**

The development theory (Piaget & Inhelder, 1967) is an early work based on observing children’s developmental sequence in understanding the environment. The theory recognised cognitive maps to constitute more complexity than just a spatial layout of things in the environment as suggested by Tolman (1948). The theory expanded the

cognitive maps description by including mode of travel, preferences, attitude, and past experiences as additional factors to influence one's mental representation. In general, researchers building from this theory mainly delivered their findings based on the three recognised stages of development, i.e. the landmarks, the route map and the survey map. There is also evidence that these sequence of developmental stages occur similarly in adults traversing new environments (Golledge, 1987).

When breaking down children way-finding strategies, it has been observed that children begins by recognising objects and are able to distinguish them. Thus landmarks identification is considered an object-based approach that occupies the first stage in the developmental sequence. According to Kaplan (1973), landmarks must be unique entities that function as indexes that mark places in the environment. With such distinct features, landmarks identification is assumed to be the easiest way-point reference for children to determine their locations and orientations in the environment. However in reality, identifying or extracting distinct objects or features from the environment as landmarks is not an easy task and remain the biggest challenge in this primary stage.

Children are then observed to connect landmarks identified via route selections. When asked to navigate in the neighbourhood, children were selecting routes between landmarks based on their direct (egocentric) experience during travel. At this point in the stage, children are yet to form abstract representations of the environment; meaning a route does not represent a global direction in the map but are steps in achieving the

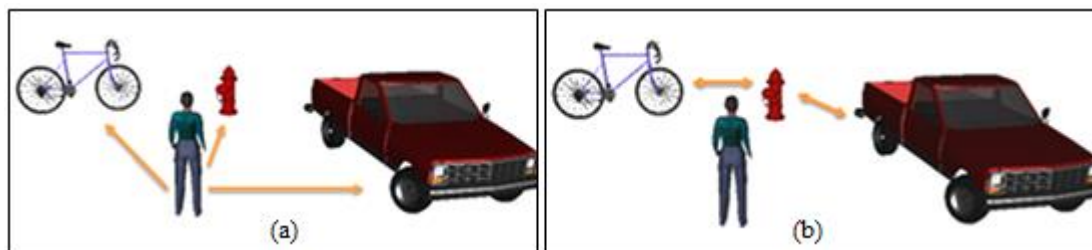
navigation target. These series of individual routes that will lead to the goal form connections that are topological in structure.

The survey map can be defined as an abstract environmental overview which basically a further generalisation of the route map concept. In this stage, the quality of the spatial knowledge is upgraded from the egocentric representation (of route map) to an allocentric representation of the environment, giving birth to the notion of reference frames in mental representations (Gallistel, 1989 & 1990). If one were to travel extensively in a particular environment, it would be useful to have a logical overview (or a bird's eye view) of the entire environment. Rather than dealing with routes individually, one may get information required for traversing the environment from such abstraction. In addition, this overview would make large-scale reasoning about the environment simpler. However, although the main function of these overviews is to increase the effectiveness of way finding, it does not mean one may need them all the time.

The notion of frame of reference has shaped the way the robot in this thesis collects spatial inputs and organise them into building an imprecise map for the robot. For this reason, a sub-section on the encodings in the human spatial memory which underlies the idea of survey maps is added here.

### 2.2.1.1 Encodings in the Human Spatial Memory

It has been argued that since humans live in a geometrical world, humans should be locating objects in the environment by means of reference to the geometrical features. Plenty of works have adopted this notion of frames of reference as a means to represent the location of entities in space (Wang & Spelke, 2000; Wang & Spelke, 2002; Mou & McNamara, 2002; Mou et al, 2004). These researchers believed that different frame of reference is used for different navigational activities. For instance, navigating through closely spaced trees requires accurate self-to-object (egocentric) judgement else one could bump into the obstacles (Andersen et al., 1997), but planning a distant goal and maintaining a sense of orientation in large environment requires one to judge how objects are allocentrically related to one another (Loomis & Beall, 1998). The following illustrations showcase how the two frames of reference configure. Fig. 2.1(a) denotes the egocentric frame of reference where locations of objects are encoded in relation to own body (e.g. left-right, front-back, or up-down). Fig. 2.1(b) depicts the allocentric frame of reference where the locations of objects are encoded relative to other objects surrounding the traveller.



**Fig. 2.1:** (a) The egocentric (self-to-object) spatial representation, (b) the allocentric (object-to-object) spatial reference

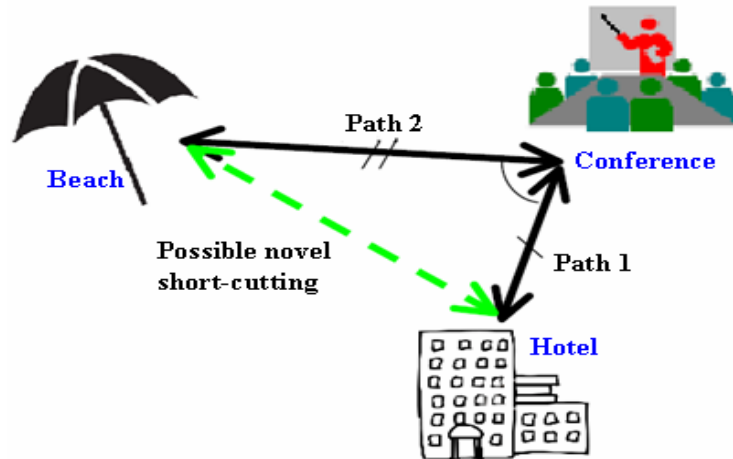
In Sholl (2001; see also Holmes & Sholl, 2005; Schmidt & Lee, 2006; Ruggiero et al., 2011), different roles and representations are identified for each reference frames. The egocentric representations always measure distance and orientation between the *ego* and the surrounding objects. Since this viewpoint-dependent system provides a framework for spatially directed motor activity such as walking, reaching and grasping, it requires constant updating as the human moves in the environment. In the case of an allocentric representation, the information is referred to the *space* independent to the human. The location of any objects is measured using its spatial relations (distance and orientation) to other objects in the same space. Holmes & Sholl (2005) postulated that this type of spatial framework takes time to develop because in the early stage of learning a new environment, the human is observed to rely more on the egocentric reference system to identify objects in the environment. Throughout the course of the journey, there may be occasions where the human's self-to-object spatial relations get disrupted by disorientation resulting in a temporary loss of his current orientation with respect to the surroundings.

Therefore, the enduring allocentric representation which has been maintained independently is still intact in the human's memory, making it a more stable memory option for the human to correct his orientation by referring to recognised objects or landmarks available in the allocentric system. Once the orientation issue is resolved, the human is considered to switch back to the egocentric system to resume the journey.

### 2.2.2 Neurological Theory

The notion of cognitive maps used in neurological studies (O'Keefe & Nadel, 1978) often pay little attention to the use of spatial information about distances and directions as evidenced in the other psychological studies of cognitive maps. This is because of their discovery of place-coded neurones in the hippocampus where a specific firing activity among these neurones corresponds to a stimulus reading from a particular place cells. When tested the theory on primates and humans, researchers discovered that the firing activities are parallel to the subject's sensory input from environmental cues rather than to where the subject is physically located at (Burgess et al., 1994).

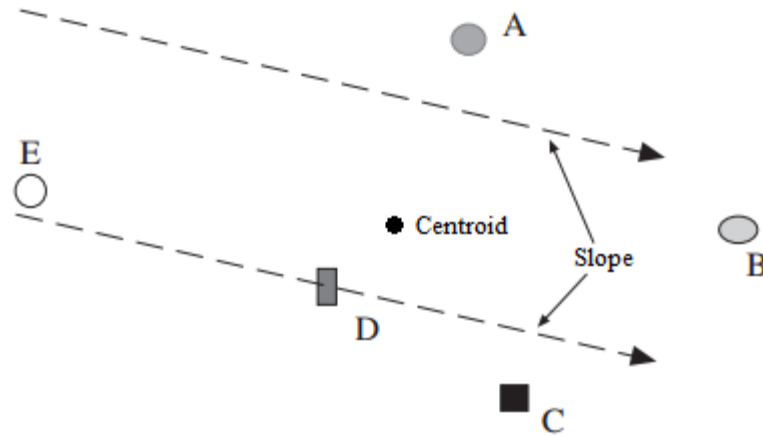
Later, Nadel (1991) reiterated that in support of Tolman's (1949) theory that the cognitive map has multiple learning systems. O'Keefe & Nadel (1978) proposed two classes from the systems as *locale* and *taxon*. A taxon is a navigation strategy which involves heading towards a landmark at or very near the goal. A locale is a navigation strategy which is used when a taxon strategy is disrupted, for instance the landmark is removed from the route. Therefore the taxon is similar to a route map whereas the locale is an allocentric-hippocampally representation since the cognitive map contains larger information about the environment and allows alternative routes towards a goal. With respect to their roles, it is assumed that learning within the locale system occurs during exploration and novelty-directed behaviours and that spatial learning is achieved by updating the locale system with new routes, landmarks and short-cuts over time.



**Fig. 2.2:** Possible novel short-cutting when human configures new routes to get between goals in the environment

O'Keefe (1990 & 1991) proposed an extension to the allocentric model by defining the *centroid* and *slope* measurements. The centroid is referred to as the object which is fixed and does not move with the animal, whereas the slope is essentially the line that follows through the centroid at an orientation that fixes the orientation of the allocentric frame; unchangeable regardless how the animal moves around. In this model, an animal is said to construct its mental representation in two stages; first, they must identify a notional or a fixed point in the environment (i.e. the centroid, and second, they must identify the gradient (i.e. the slope) for the environment to fix the direction. Usually the slope is depicted as the gradient that is similar to how objects in the environment are widely spread. The animal's cognitive map is then a result of encoding objects in the surrounding using vectors whose lengths are distances from the centroid.





**Fig. 2.3:** An example of the centroid-slope model. Reproduced from O’Keefe (1991)

### 2.2.3 Computational Theories of Cognitive Mapping

This section discusses three major cognitive mapping models developed to date, namely the Tour Model, the Plan Model, and the ASR/MFIS model.

#### 2.2.3.1 The TOUR Model

The first comprehensive computational model of the environment was Kuipers’ (1977) Tour model. This model was influenced by Lynch’s (1960) work on “The Image of the City” and studies concerning the development of spatial knowledge among children (Piaget & Inhelder, 1967). What distinguishes this model from earlier ones is its inclusion of several important representations for large-scale space navigation. For example, it has a representation which allows users to select correct routes to get from one place to another, a topological representation that keeps connection between traversed routes, and a different metrical representation for each local place visited. Other spatial knowledge models beforehand only proposed the mental map as a single global reference frame for the entire environment.

The following depict how the TOUR model divides the spatial knowledge into five different groups, each with its own representation:

1. *Routes*

A representation of a sequence of action which guides users along a particular route through the map

2. *Topological*

A network of routes with emphasise on the order of the places connected by the routes. This representation also makes explicit the intersections where these series of routes meet.

3. *Frame of Reference*

This representation describes each local space as a local metric map of its own. Objects inside a local metric map are positioned according to a particular orientation to the user. Different places are built from different orientation preference thus they cannot be compared to each other unless they share similar orientations.

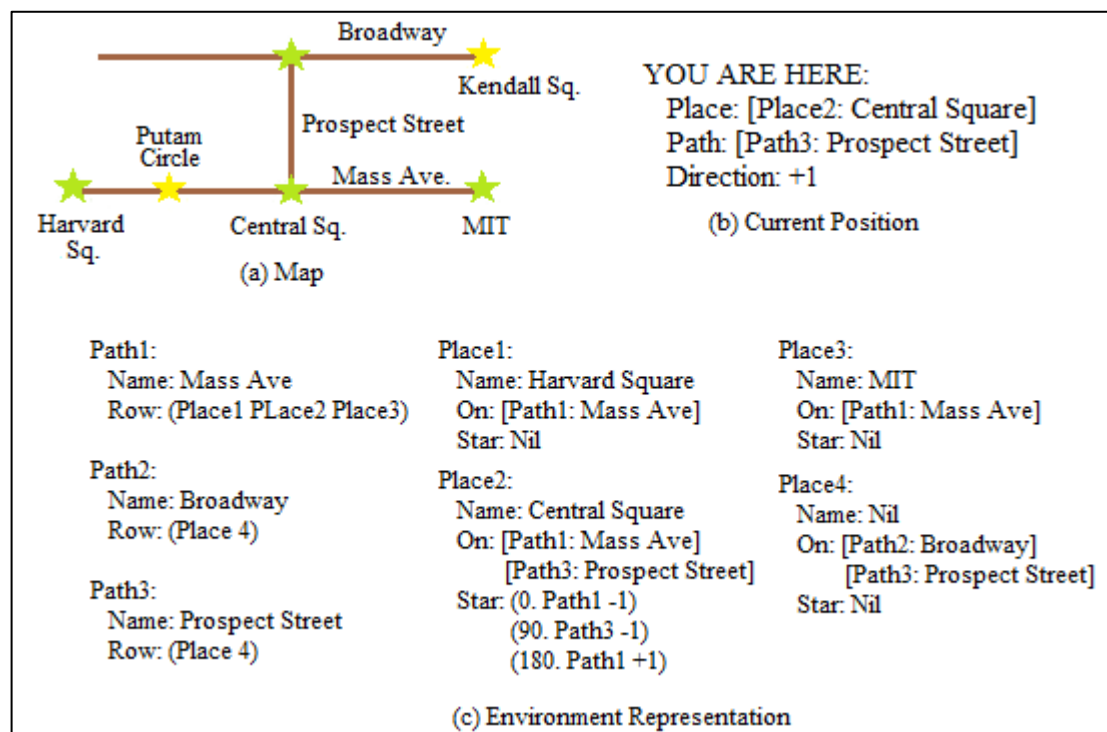
4. *Dividing Boundaries*

This representation supplies the knowledge of where a place is located relative to other places in the environment. The boundaries are not physical barriers which may disrupt travel; their main purpose is to provide a qualitative measurement of positions.

5. *Regions*

Regions are defined as a structure with levels of abstraction use for stating properties of their elements.

To function properly, the TOUR model requires the navigational problem to be broken down into three different representations. The first is usually a map of the area (see Fig. 2.4(a)), followed by a description of where the traveller is located at present in the environment (Fig. 2.4(b)), and lastly a set of inference rules to manipulate the knowledge of two previous classes (Fig. 2.4(c)). The performance of the model depended on how much information is available in the last two since these representations may be incompletely described.



**Fig. 2.4:** The TOUR computational model of cognitive map. Reproduced from Kuipers (1978)

The set of rules in the last representation is inferred as a result of manipulating and combining different parts of knowledge from the other two spatial representations. However it is not the representation which is accessed to learn about the environment. The only accessed environmental descriptions from the TOUR machines are the ones

referred by the “You are here” pointer in the second representation, the representation which depicts the traveller’s current position. This allows the model to work quite efficiently since the route programs that drives it do not have to search for environmental elements.

The inference rules are made of simple modules which triggers some simple actions when pass certain thresholds. In general these rules are categorised according to the following:

1. Rules which compare the route instructions in the “You are here” pointer to the topological descriptions of the environment. If a part of the topological representation is incomplete, these rules can patch up the missing gap with information from the others
2. Rules that fix the orientation to adequate the current heading of the frame. This is achieved by comparing the “You are here” pointer to knowledge of the current place and path
3. Rules which detect structural features that can be defined as dividing boundaries
4. Rules that utilises the hierarchy of regions, boundaries and referential frames to solve route-finding and position-finding problems

Kuipers & Levitt (1988) discussed that the TOUR model keeps a topological map of the environments but has a separate representation for the route knowledge. This is the disadvantage of the model since the route knowledge on its own is not reliable in

discovering new routes and shortcuts or cope without guidance in unknown environments. The model however is able to cope with the problem to some extent by depending on information from its topological representation.

#### **2.2.3.2 The PLAN Model**

PLAN is short for Prototypes, Location and Associative Networks (Chown et al., 1995) which derived inspirations from the developmental theory. They argued that the human cognitive mapping process is about solving how humans find their ways in the environment. Basically the model identified four functions of the humans' cognitive maps which are defined as follow:

1. *Landmark Identification*

With powerful sensory system like the vision, humans are able to identify distinct features among objects in the environment quite easily. For this reason landmarks are considered the most basic component in the human way-finding. By associating a unique object or landmark to a particular area (Kaplan, 1973), humans can later use the information to recognise where they are in the environment and to plan their routes to chosen destination.

2. *Path Selection*

The main task here is to choose a route to achieve the navigation target. A route is considered as a series of paths that will lead the human to the goal. The common concept when dealing with paths is that they are actually sequences of landmarks. In this model, gateways used to enter and leave a city are depicted as landmarks. So to travel along a path means to follow from one landmark to

another (in this case, gateways) until one reaches the target. With many alternative paths to choose from the environment, selecting the correct ones is a problem which requires some reasoning skills.

### 3. *Direction Selection*

As the name suggests, this function deals with selecting a direction to guide one's travel. If one can see the target without obstructions, the practical solution is to turn towards the target. The problem arises when one has to deal with a faraway target which is beyond one's visual capacity at the time. Since one may have to perform constant turning to reach a distant target, selecting a direction when one begins a journey is unlikely sufficient to guide the entire journey (Golledge & Garling, 2003).

### 4. *Abstract Environmental Overviews*

When required to explore in a large-scale environment, a logical overview of the entire environment is a useful spatial knowledge to have. The abstract structure would aid in reasoning tasks and can overcome problems posed by dealing with only route knowledge. However acquiring such knowledge is not straightforward and often involves the combination of each of the other three solutions.

The PLAN model includes three major modules; prototypes, locations and a simple associative network in its development.

## **Prototypes**

Chown et al. (1995) argued that landmarks are no different to other objects in general and thus proposed a category called prototypes which do not define landmarks as a single distinct entity. To address landmarks identification, the most common features among the landmarks will be included in the prototype representation. The prototypes are then ordered as a hierarchical representation with individual feature samples at the bottom and the more common features sampled at higher levels. The following conditions further defined landmarks in the model:

1. Landmarks must be distinguishable even when perceived from different point of views. Also, it is required for the prototypes to function even with partial view of the landmarks
2. The number of landmarks which can be processed at any time is limited to  $5 \pm 2$  similar to the number of objects human processes mentally at a time
3. Landmark must be explicitly influenced by the background. A good landmark for a particular area in the environment may not be a good choice to other areas

Due to the above reasons, it can be said that not every object can pass as a landmark, only those that stood out and distinctively recognisable.

## **Location**

Later, Chown et al. (1995) also proposed that local maps are not computed at landmarks but rather at a choice point in the journey. The observation leads to the construction of another network since choice points are not represented in the network of landmarks.

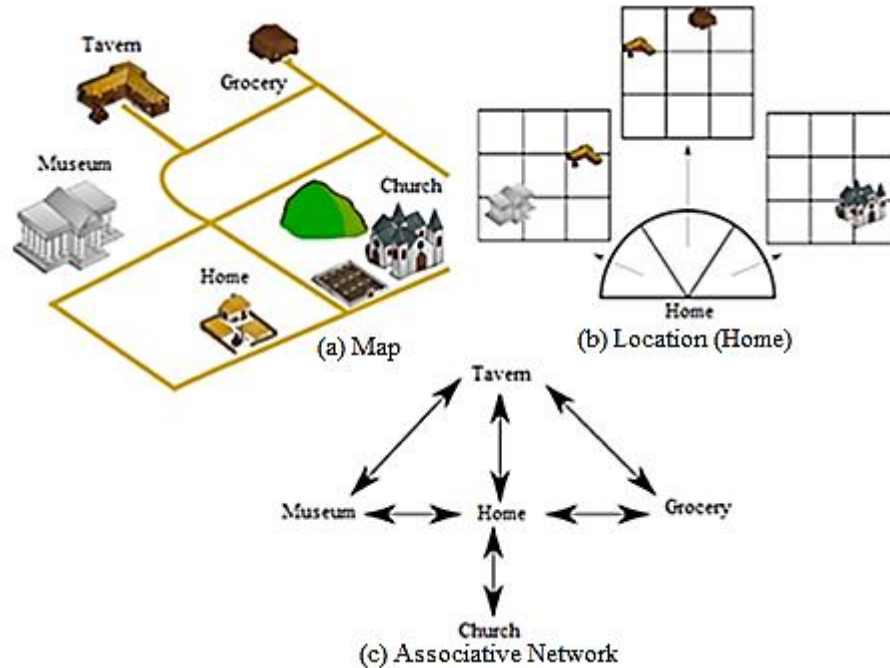
The new network is called the R-Net which stands for the network of local maps. The R-Net is similar to NAPS but rather than storing landmarks (in the form of prototypes), R-Net stores local maps or regions which are separated using gateways. The regional maps function like a survey map with the overview or abstraction from the R-Net and the lower level information from the network of landmarks. As a higher level representation, the regional maps optimise the amount of information in them making the maps the primary structure to reason and solve way-finding issues. When requiring further information, they can be extracted from the lower level representation.

### **Associative Network**

To test the idea of using an associative network, an implementation known as NAPS (Network Activity Passing System) is carried out (Levenick, 1985; Levenick, 1991). The goal here is to encode a topological system into the associative network so the nodes in the network correspond to landmarks and links that connect the nodes correspond to distance between the landmarks. The establishment of such network made explicit paths as they are accessible by tracing the series of links between the landmarks. To solve the path searching problem, the nodes are activated from both ends; the starting point and the goal location. Wherever the signals *meet* at some point in the middle of the network, the meeting point (which is also a node) is computed as a sub-goal. The process is repeated until the path is completed. In the model, each landmark is treated like a local map which stores all links that connects it to other landmarks. Having local maps is useful to orient oneself to a nearby landmark. The



following figure denotes an example of how landmarks are extracted onto directional and locational grids which form the local maps.



**Fig. 2.5:** The PLAN computational model inspired by Figure 2 and 3 in Chown et al. (1995)

### 2.2.3.3 Yeap's Computational Model

In developing his model, Yeap (1988) questioned how information perceived by the sensors is processed into a cognitive map. This emphasis on the input follows from Marr's (1982) work on his theory of vision. Humans can rarely remember every detail perceived from the environment. Based on this observation, Yeap suggested that an important question about cognitive mapping is: "What is remembered first and how is that information be useful later?" While other studies at that time have emphasised that cognitive mapping is a complex process that involves one's perception and conception of the world, it is important, and following Marr's idea, that our understanding of how

conceptual ideas are derived be grounded with our understanding of what is delivered at the perceptual end. In his model, four important representations were proposed (see Fig. 1.1), namely:

1. *The Absolute Space Representation (ASR)*

An ASR is the most basic description for each local space visited. It functions to capture the spatial extent of the space where one is in. The task in computing an ASR is to find the boundary surrounding the viewer and this comes from the surfaces perceived relative to the viewer. Another important feature of the boundary is exits (Yeap & Jefferies, 1999). Exits are path options which allow one to escape the current local space and move into another.

2. *The Memory for One's Immediate Surroundings (MFIS)*

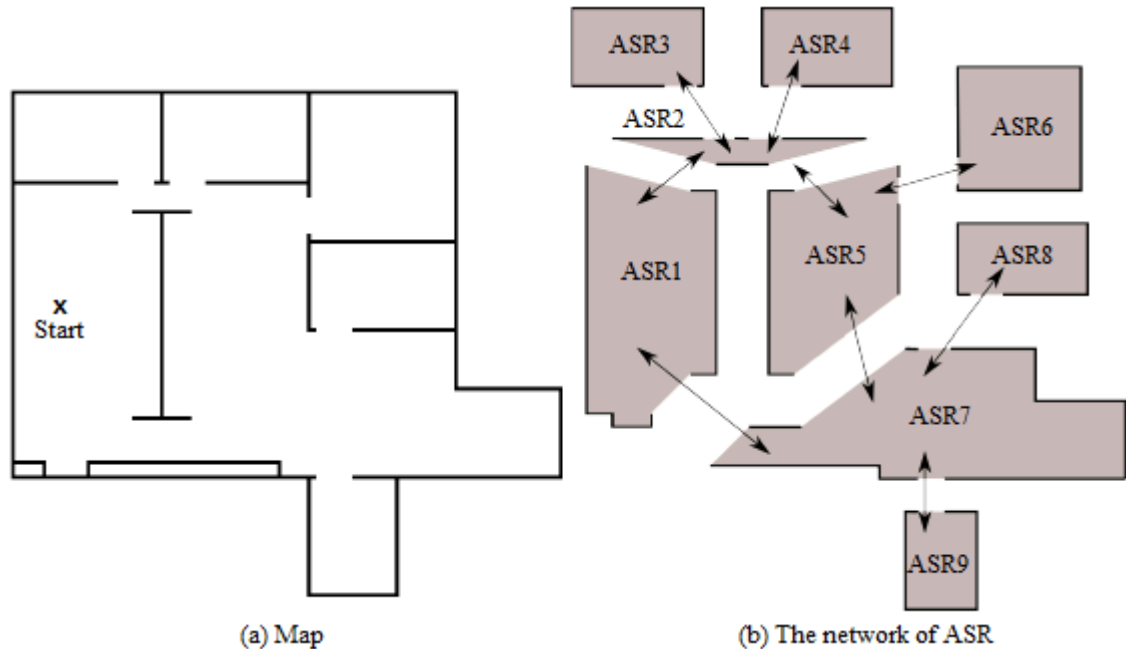
The MFIS is a global map of a viewer's immediate surroundings. Using a global co-ordinate system, it depicts the network of ASRs and has information on the current location of the viewer. It is often used as an extension to what an ASR is so the viewer is aware of the spatial layout of their immediate surroundings.

3. *The Raw Cognitive Map*

The raw cognitive map is a network of simplified ASRs which are loosely connected. It is termed 'raw' because it produces a representation which is dependent on one's sensory perception. So the more one explores and visits new places, the larger one's map will be. The linking between one ASR to the other is done by connecting the common exits between them. The network, if well connected, enables the viewer to navigate with ease in the environment. Fig. 2.6 depicts the raw cognitive map built based on a sample environment.

#### 4. *The Full Cognitive Map*

The full cognitive map identifies places in the environment by organizing several ASRs into groups. These groups of ASRs can also be combined and organised into a hierarchy of place representations. This type of representation is highly dynamic as the groupings may change if one perceives a place differently over time. At the inter-level, this map explains how to connect the places in the hierarchy and the possibility of joining new places to build new hierarchies. Similar information is available at the intra-level so one can extract or modify the connections between members inside a particular group. Another advantage of the structure is one can examine exits which connects any two or more ASRs without having to search the entire hierarchy. This allows fast searching of exits (basically from anywhere) as long as one knows which ASR one wants to go to and which ASR one is leaving.



**Fig. 2.6:** Example of an indoor map and the ASRs computed

## **2.3 Implementing Cognitive Theories on Robots**

The goal of this section is to describe robotic works that attempts at mapping the environment deriving inspiration from any of the cognitive theories presented in the previous section. Much of the works in cognitive inspired robot mapping are done in static, structured and small-scale environments.

Briefly, in developing autonomous navigation behaviour, the robot is required to have an internal representation that adequately resembles an overview map of the environment. To acquire the map, mobile robots are equipped with sensors such as vision (camera, CCD, 3D laser, panoramic), range finders (sonar, laser, radar and infrared), and advanced positioning sensors (digital compass and GPS). These perceptual sensors are noisy with each one having their own measurement errors and range limitations. With the robot's limited capacity sensing the environment, it is of interest to see for example, how object-based robots work with landmark representations or how would a perceptually-driven architecture be fitting to develop a robot's spatial knowledge. Discovering new routes and performing novel short-cuttings are advanced way-finding tasks for robots, thus one would wonder if mapping algorithms based on the cognitive structures are useful to solve these problems.

### **2.3.1 Spatial Semantic Hierarchy**

The SSH (Kuipers, 2000) is a further extension of the TOUR model (see section 2.2.3.1). However, unlike the TOUR model, the SSH proposed one's perceptions while

experiencing the environment as reference to how one builds one's cognitive maps. The SSH is hierarchical by structure and offers five different levels of knowledge which influences one's cognitive maps.

1. *The Sensory Level*

This is the interface to the robot's sensors and effectors. Here, the environment is perceived as a black-box process which returns symbols of images from the sensor views. These symbols are then matched for recognition or used as retrieval keys.

2. *The Control Level*

At the control level, the agent and its environment are described as part of continuous dynamical system. Two behaviours are detected to define the agent's actions; the trajectory-following and hill-climbing control laws, so the agent-environment system moves correctly towards an attractor. The stable attractor of a hill-climbing control law is called a distinctive state.

3. *The Causal Level*

Here, the agent and its environment are described as a partially known finite-state automaton with views, actions and schemes representations. The views correspond to sensor readings at distinctive states identified at the control level. The actions are responsible for the commands to move where each movement matches several sequences of the control laws. The schemes are extracted from the interaction between the views and actions.

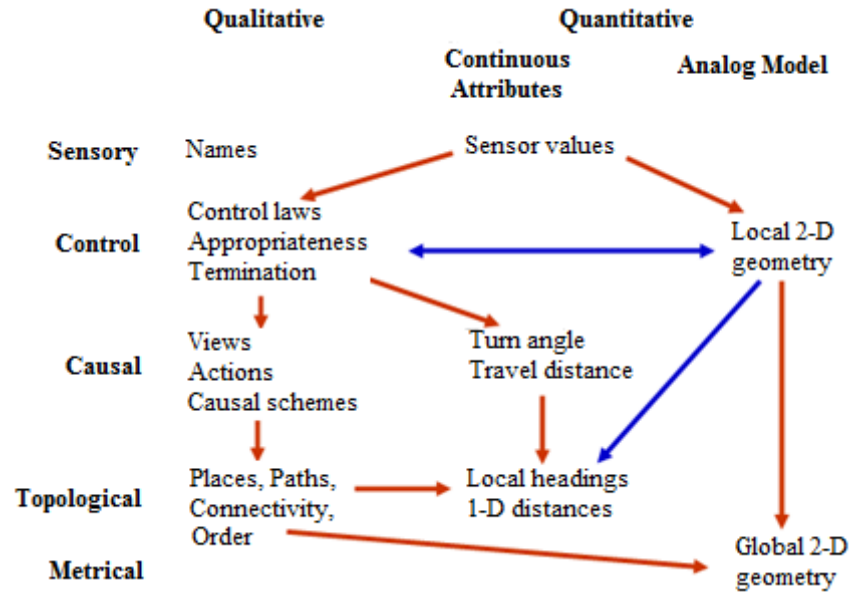
#### 4. *Topological Level*

This interface deduce two relationships; the first one between the distinctive states and movements which are calculated at the control level, and the second between the views and actions at the causal level. These relationships allow the interface to relay information about places, paths and regions and how they are connected to one another in the form of nodes and edges. The map produced here is responsible to solving way-finding problems.

#### 5. *Metrical Level*

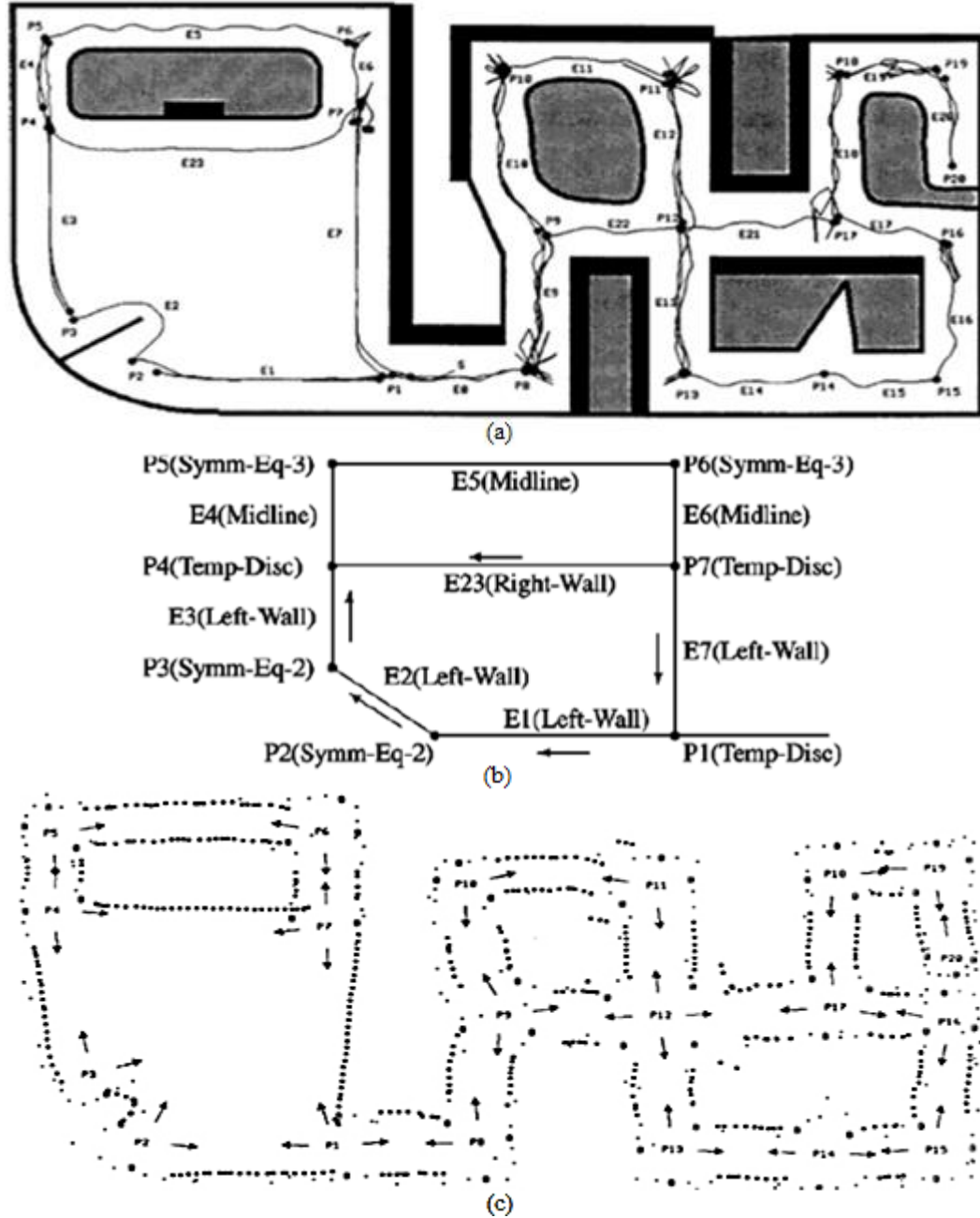
A 2-D global topological map is defined here. The map consists of properties like distance between places and angles between paths. These properties correspond to the nodes and edges of the topological graph.

Even though the five levels in the SSH model relies on one another, the reliance are not without constrained. This control makes each combination of more than one level a unique representation on its own as every level defines different SSH knowledge. This model, when used to describe an agent's cognitive map, divides the map into representing knowledge at different SSH levels (see Fig. 2.7).



**Fig. 2.7:** The logical dependencies and constraints (red arrows) in the SSH Model. Blue arrows denote potential information flow without dependencies. Reprinted from Kuipers (2008)

The earlier implementations of the SSH model were done in Kuipers & Byun (1987, 1988 & 1997) using a simulated sonar robot with a digital compass called the NX (see Fig. 2.8). The compass is used for global orientation so the robot could reach distinctive states in the environment using simple control laws. They were also interested to see how the topological and metrical maps can be abstracted from the states. Then Lee (1996) implemented this algorithm on a real mobile robot called Spot. Spot is also a sonar robot with 3-legged wheels. A more recent implementation using the SSH model can be found in Remolina & Kuipers (2004).

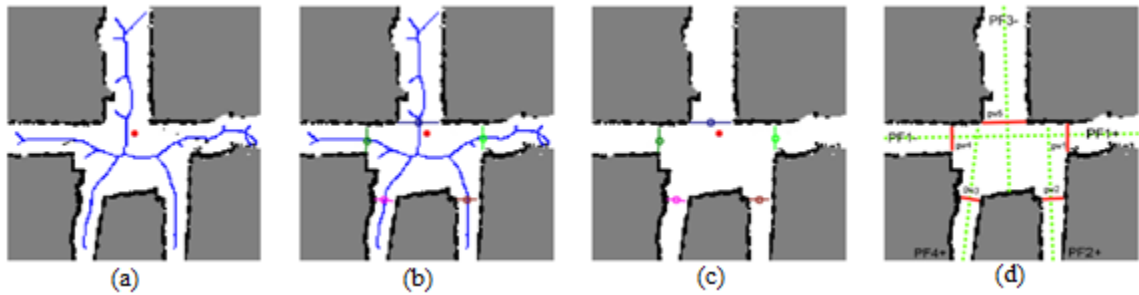


**Fig. 2.8:** (a) Simulated exploration by the NX robot highlighting the distinct paths and places, (b) parts of the topological map which defined the relation between paths and places, (c) the global metric map produced. All images reprinted from Kuipers (2000)

Later implementations see an upgrade to the SSH architecture when Kuipers and his co-workers include the robot's *local perceptual map* and termed it as the Hybrid SSH (Kuipers et al., 2004). The popular SLAM methods are used to compute the local

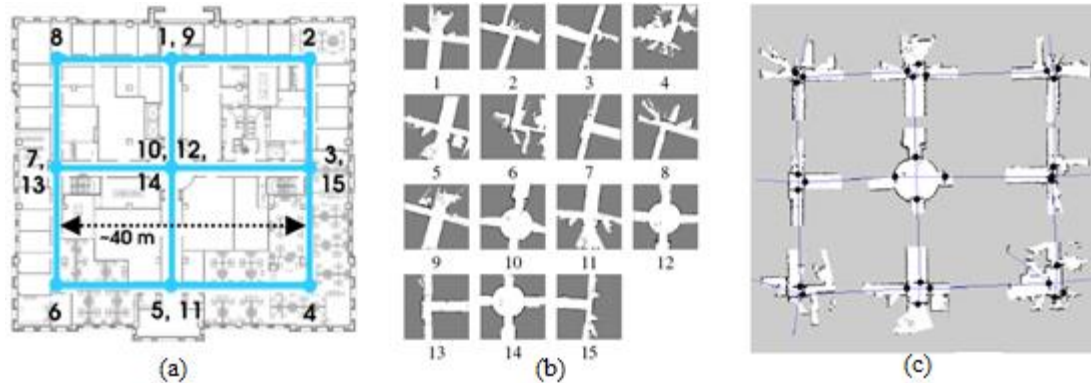


perceptual map (LPM) which is a precise overview of the distinctive places available. The change sees the atomic representations of views of the SSH model is replaced by the LPMs in building the topological map. The LPMs provide precision in way-finding and thus avoid having to deal with the loop closing problems. Fig. 2.9 shows how the HSSH computes first the gateways, followed by the paths and distinct states.

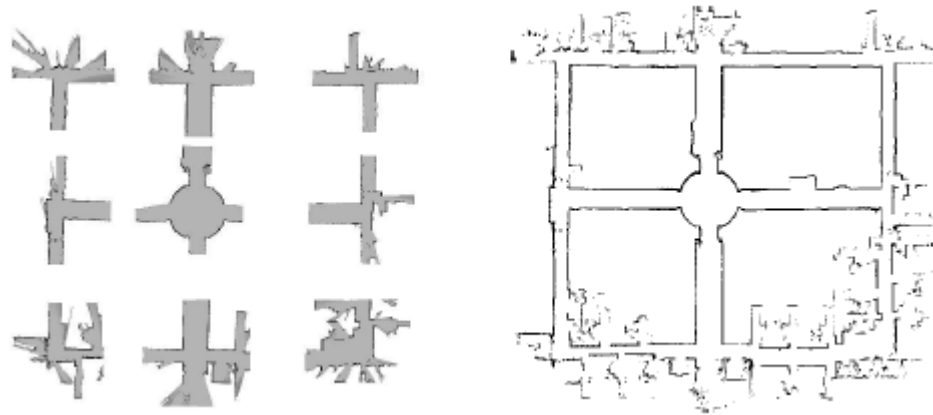


**Fig. 2.9:** (a) Robot (red) in the middle of an LPM, (b) and (c) show 5 gateways recorded where the corridors just about to merge into a large common area, (d) described how 4 paths are computed and 8 distinct states detected. Reproduced from Kuipers et al. (2004)

The LPMs are built on occupancy grids where each grid having their own local Euclidean representations. Testing the connections between the LPMs has been the main focus in Kuipers et al. (2004). To test the model's performance, the robot was made to travel in an environment with multiple nested loops (Fig. 2.10(a)). Each time the robot passes through a place, an LPM is computed for the place and a number is given to tag the LPM. The numberings accord to the order in which the robot travels (Fig. 2.10(b)). They observed that when revisiting the same LPM, the gateways are the key in solving the perceptual aliasing problems (see Fig. 2.10(c)). Once the loop problem is solved they were able to produce a global metrical map of the environment which originated from a topological representation in Kuipers (2008). See Fig. 2.11 for results.



**Fig. 2.10:** (a) An environment with multiple nested loop, (b) the LPMs computed at each place and tagged with a number, (c) connecting the LPMs via their gateways. Adapted from Kuipers et al. (2004)



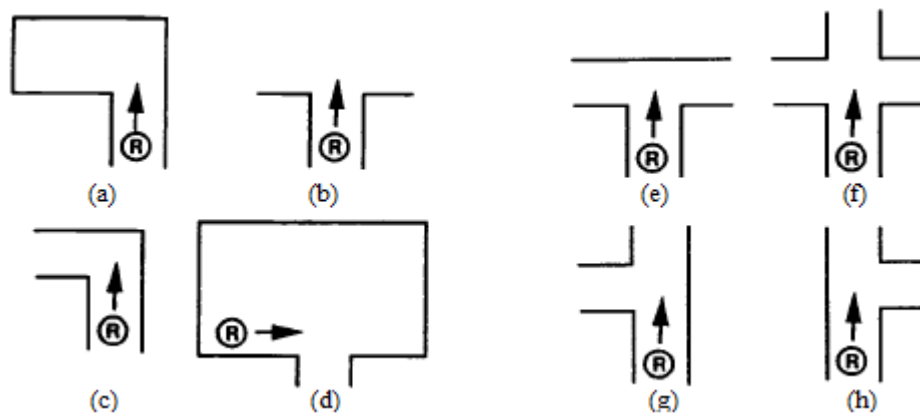
**Fig. 2.11:** Computation of a global metrical map (right) from what is initially a topological representation of the environment (left). Reprinted from Kuipers (2008)

### 2.3.2 R-PLAN

In Kortenkamp (1992), the cognitive theory PLAN (see 2.2.3.2) is implemented on a mobile robot equipped with 16 rings of sonar sensors and a single camera. The implementation is called the R-PLAN (robot PLAN) and the focus is to investigate the integration of both sensors to develop for the robot the notions of gateways and scenes respectively. The gateways in theory are places of choice points where the robot may potentially perceives new landmarks in its exploration. In the cognitive model (PLAN) these landmarks are an abstraction of objects such as buildings and trees. However,

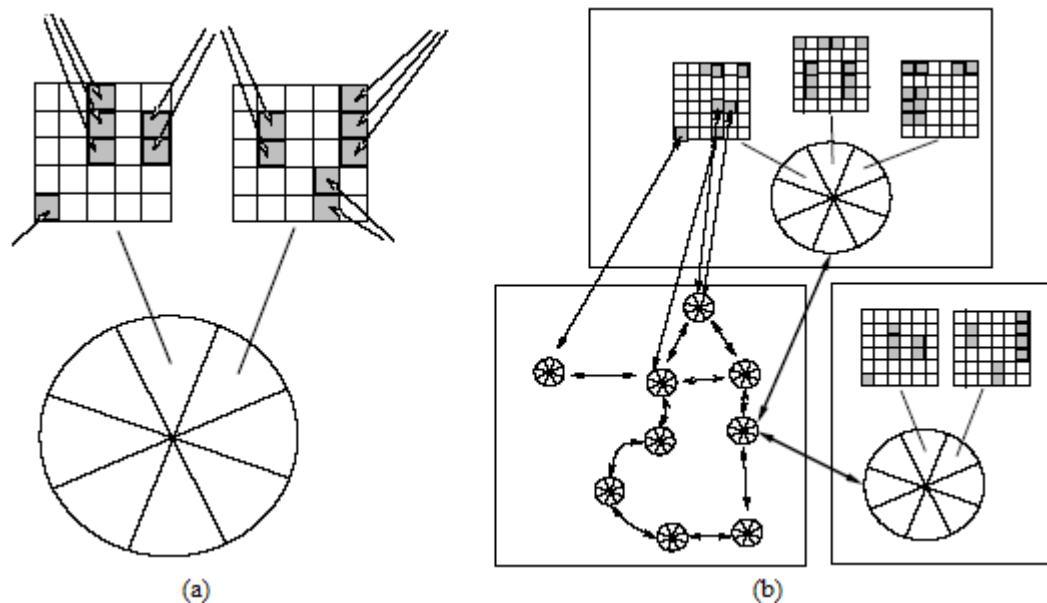
these objects are too complex to be recognised as such using their robot senses and as such, in the implementation, they extracted vertical edges of objects instead. The latter is done using vision.

For the gateways, doorways are seen as good options to see more of the environment because they offer the sense of visual narrowing then opening as the robot move towards and cross them. Sonar readings were used to detect them by sensing the spatial changes leading to gateways. The scenes are composed of visual images captured at the gateways and each composition is specific to identify different gateways. Fig. 2.12 illustrates several types of gateways identified for the robot. These gateways are useful to overcome the perceptual aliasing problems by reasoning about the type of gateways identified when looping occurs in the regional map. For instance, a T-shape gateway could not be a four-corner gateway on a route, but it could be a left or right opening on another.



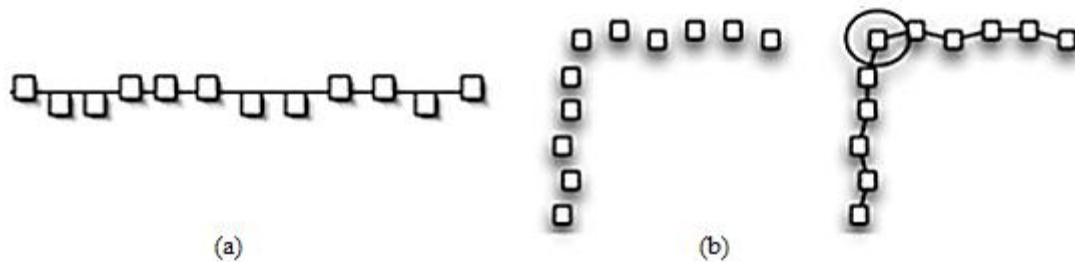
**Fig. 2.12:** Samples of types of indoor gateways in R-PLAN; (a) left room entrance, (b) room entrance, (c) left corner, (d) right room exit, (e) T-shape gateway, (f) four corners gateway, (g) left opening gateway and (h) right opening gateway. Reprinted from Kortenkamp (1992)

In Kortenkamp & Weymouth (1994), the scenes are used to identify places in the environment. The scenes store the robot's visual cues in an abstracted scene representation which is a  $5 \times 5$  grid. Each cell of the scene represents a visual cue; direction, distance or length of the cue. Fig. 2.13 shows the cues stored for a scene where each scene serves as a basic component for a larger representation. There were four algorithms to match the scenes for place identifications; (1) feature-to-feature matching using distance and direction, (2) cell-to-cell matching using distance and direction, (3) cell-to-cell matching using only direction, and (4) cell-to-cell matching using only occupancy. These algorithms however, are limited to highly structured and orthogonal environments due to the expensive computational resources to process the visual scenes.



**Fig. 2.13:** Storage of visual cues into a scene (ASR). (b) The configuration of scenes for place identification. Reproduced from Kortenkamp & Weymouth (1994)

Recent implementations of the PLAN theory have seen a shift in the way the basic component is implemented. In Chown & Boots (2008), corners are detected using laser sensors which they termed the implementation as C-PLAN (corner PLAN). Each corner is annotated as node and the path and direction used to traverse between corners is recorded. The nodes and their spatial relations form a topological representation of the environment. Apart from corners, the use of laser sensors allows Chown & Boots to also perceive wall information and suggested that walls can be of influence to humans' perceptions of an indoor environment. They further defined corners to be readings of two surfaces separated at 90-degree angle and walls to have 180-degree angle. These corners and walls were influential in their depiction of the environmental representation (see Fig. 2.14 for example).



**Fig. 2.14:** (a) Raw laser points separated at 180-angle to each other and (b) computing the corners by breaking raw laser points at any right-angle found. Reprinted from Chown & Boots (2008)

### 2.3.3 Absolute Space Representation

Yeap and his students (see Yeap & Jefferies, 1999, 2001; Jefferies et al., 2004; Jefferies et al., 2008; Wong, 2008) have also attempted to use robots to test their theory of cognitive mapping. Their goal is to use the robotic platform to advance their theory for explaining cognitive mapping. As such the implementation for the ASR theory has so

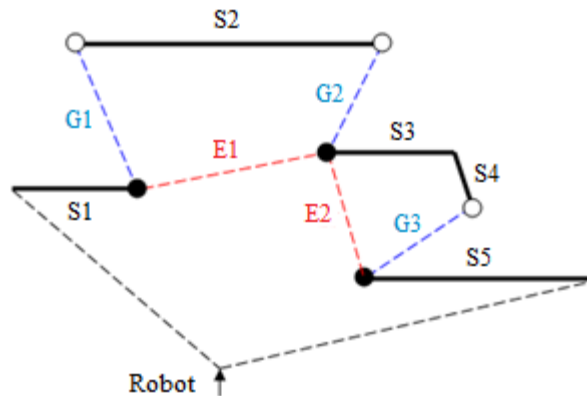
far never been attempted *for* robot mapping, which is the aim of this thesis. However, much can be learned from their implementations and for that reason two of their contributions are forwarded here.

Briefly, Jefferies contributed the idea of exit and the role of topological hierarchy in cognitive mapping. She tested using a laser robot and reported the importance in connecting network of local spaces (topology) particularly for loop closing. Then later, Wong tested the ASR on a sonar robot to see what kind of map is produced when the sensing is poor. He showed that distance and orientation is an important aspect in cognitive mapping.

According to Yeap (1988), representing the local environments is the most basic requirement in building a robot's cognitive map. The robot must then describe the boundary of the local space by forming suitable connections between the surrounding surfaces, which gives the rough shape of the space the robot currently resides in. But Yeap & Jefferies (1999) highlighted that determining where a local space should start and end is not a clear-cut decision. They proposed the notion of exits as a cognitive solution in guiding the robot to leave the current local space and move into a new one. In Yeap & Jefferies (1999), an exit is defined as a narrow opening in the surrounding boundary which exists between local spaces. It acts as an entryway with enough space for the robot to pass through and it allows the robot to move into another local area to continue exploration. Exits are normally detected at *occlusion points*. Technically, when

a surface is perceived being occluded by another surface, then gaps are present in between these surfaces (see Fig. 2.15).

These gaps are a good indication that there are areas beyond which are hidden from robot's direct line of sight. By connecting the occluding points of these gaps, a virtual edge is created. If the virtual edge is wide enough for the robot to pass through, it is a good candidate exit for the current local space. Sometimes a false exit can be calculated. But once the robot is closer towards a false exit, feedback from the sensors should be able to verify spurious data and eliminates any discrepancies detected (Jefferies et al., 2001). Once exit information is available, the robot will be able to decide how it is going to leave the current local space.



**Fig. 2.15:** Example of exit detections. G1 and G2 are the gaps detected because surfaces S1 and S3 are occluding surface S2. G3 is a gap because surface S5 is occluding S4. All gaps have one occluding point (marked as filled circles) and one occluded point (marked as empty circles). Connecting the two occluding points of G1 and G2 creates the first exit E1. Connecting the two occluding points between G2 and G3 creates the second exit E2.

Fig. 2.15 shows that regardless whether the robot leaves via the first exit (E1) or the second exit (E2), S1-E1-E2-S5 are used to compute the rough shape of current local space. Surfaces and gaps beyond an exit; such as S2, G1, and G2 for exit E1, or, S3, S4

and G3 for exit E2, are not included in the current local boundary computation unless the exits are deemed false and robot is not able to cross them. In the case where robot decides to use E2 but discovers upon crossing it that G3 is an actual surface hidden from robot's previous position, E2 will then be the only exit for that local space and robot has to reuse E2 to escape the area. E1 will then be useful to continue its exploration in the environment.

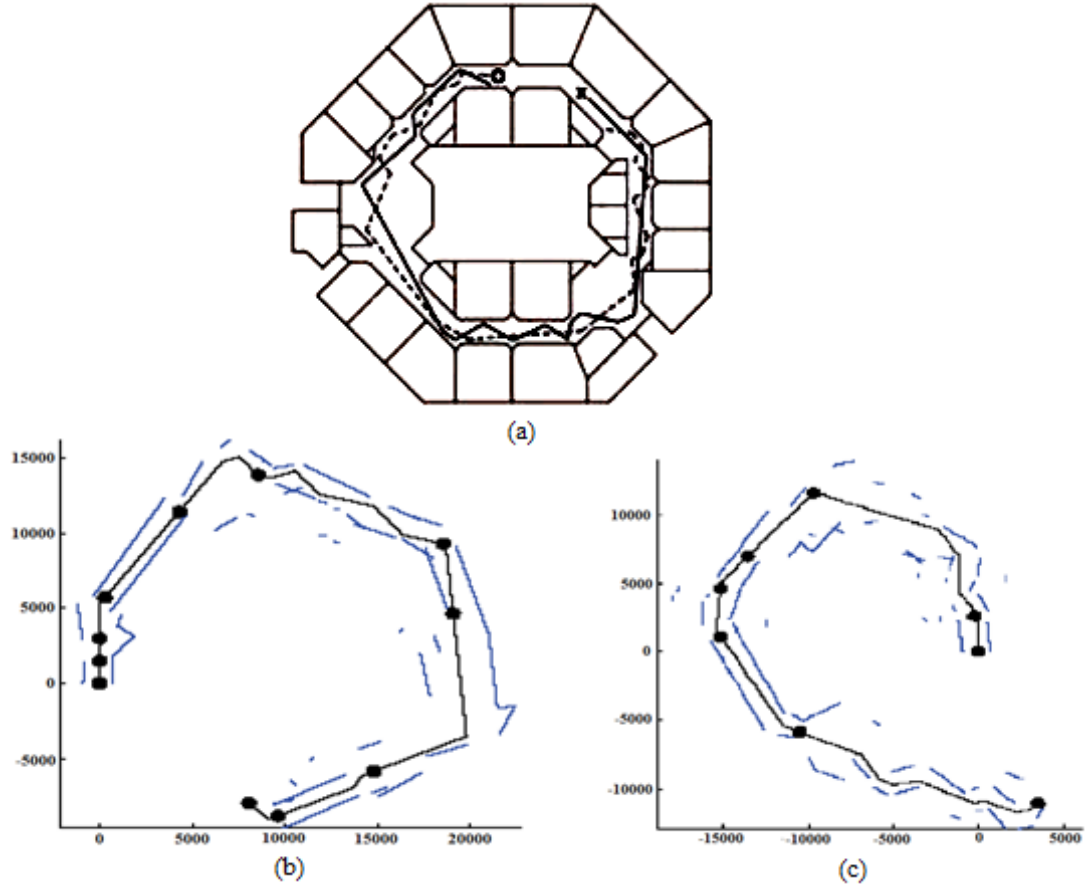
Exploiting the sequence in which these exits are crossed allows for one to build up the connection between the different local spaces. With a network of the local spaces established, the robot will have another view of its environment. However, when using the network of ASRs to navigate, identification of a previously visited ASR particularly from the opposite direction remains a challenge. From different perspectives, the same local environment may have significant variations in terms of the shape or key feature descriptions and this makes matching them difficult. To solve the place recognition problems, Jefferies et al. (2008) proposed the computation of an MFIS representation whereby the recent ASRs visited are described are placed in the MFIS and thus described using a single global frame of reference (see Fig. 2.16). In this way, if it has re-entered any of the recently visited ASRs, it will still be able to identify them even though the route chosen is a novel one. However, distortions and the limited size MFIS meant that the problem is only solved partially.





**Fig. 2.16:** (a) Example of a network of ASRs computed with robot currently in ASR3 (b) The MFIS produced with robot about to revisit ASR7 from ASR3. Since ASR7 is identified, no new ASR is formed. New link is established between ASR7 and ASR3. Reproduced from Jefferies et al. (2008)

Later, Wong et al. (2007) purposely implemented the ASR on a lower sensing robot. With only 8 sonar rings, he collected noisy readings and used them to compute what he called a network of fuzzy local places. His implementation showcased that an ASR could allow such representation and still be of use for navigation. In home-finding experiments, Wong et al. computed two separate maps; one for the journey outward and another for the journey homeward. Fig. 2.17 shows the results from the experiment. Each node in Fig. 2.17 (a) and (b) represents an ASR computed in the journey. It has been observed that the total number of the ASRs and its start and end points are different in both maps, which is due to the robot moving in opposite directions. This means that direct comparison of features from both maps; such as their rough shapes and sizes, is futile to perform place recognition.



**Fig. 2.17:** (a) The large corridor-like environment for testing, (b) map for the outward journey with 10 ASRs computed, and (c) map for the homeward journey with 8 ASRs computed. Reproduced from Wong et al. (2007)

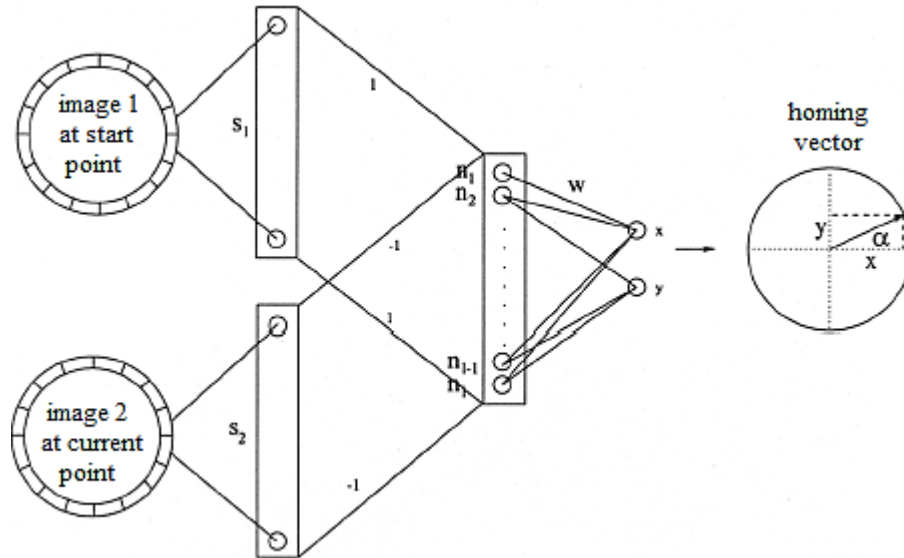
For this reason, Wong et al. (2007) came out with the strategy to utilise distance and orientation in performing home-finding. The robot is instructed to calculate the distance between exits from the local spaces in the homeward journey and projected it onto the local space network created in the outward journey (Wong et al., 2007). The distance projection is then remembered and utilised for localisation and navigation towards home. Another important feature of the strategy is it also enables the robot to maintain orientation with its home throughout the return journey, which has been successfully demonstrated in several homecoming attempts (Wong, 2009).

### **2.3.4 Neural Cognitive Maps**

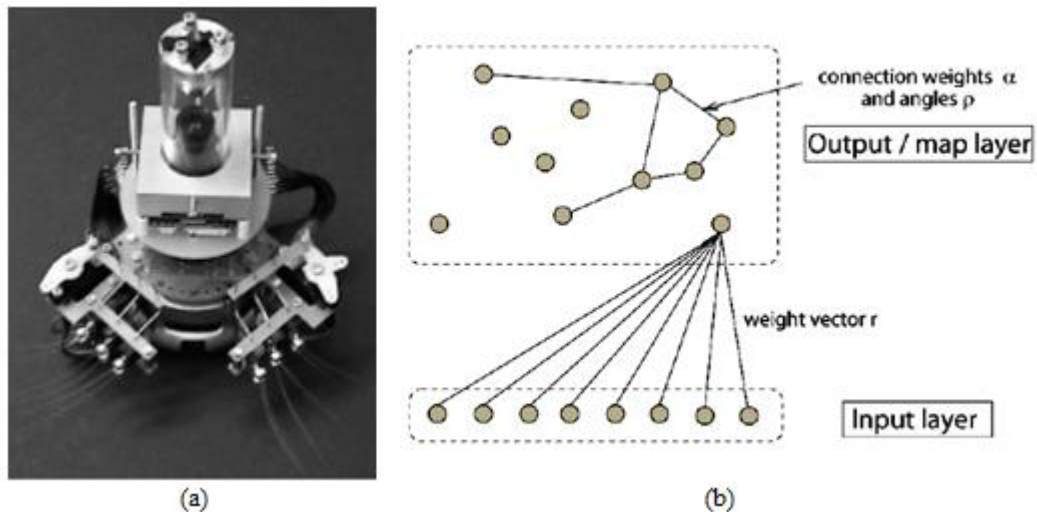
Many robotics implementations are inspired by the neurological theory of cognitive mappings (briefly described in section 2.2.2). These implementations differ from those based on symbolic models. In general, they are concerned about the connection between activated place cells whereas symbolic approaches are concerned with how information from the environment influences the acquisition, processing, development and use of one's cognitive maps.

#### **2.3.4.1 Hafner's Works**

Hafner (2001) introduced a visual homing neural network model which is trained using a normalised version of the Hebbian learning. See Kempter (1999) and Gerstner & Kistler (2002) for formalisms. The neural network model is implemented on a mobile robot called Samurai with sensors (omni-directional camera and magnetic compass) resembling the desert ants' Cataglyphis. The main task of the robot is to perform visual homing but without any feature extractions or landmark segmentations. The strategy is to capture two snapshots of the environment; one at the beginning of the journey and the second at current position. The compass aligns the two images according to an external reference direction. The result is a homing vector pointing toward the position where the initial snapshot was taken (see Fig. 2.18).



**Fig. 2.18:** The neural network structure for visual homing. All weights at the output layer which triangulates home is adjusted using the Hebbian learning. Reproduced from Hafner (2001)

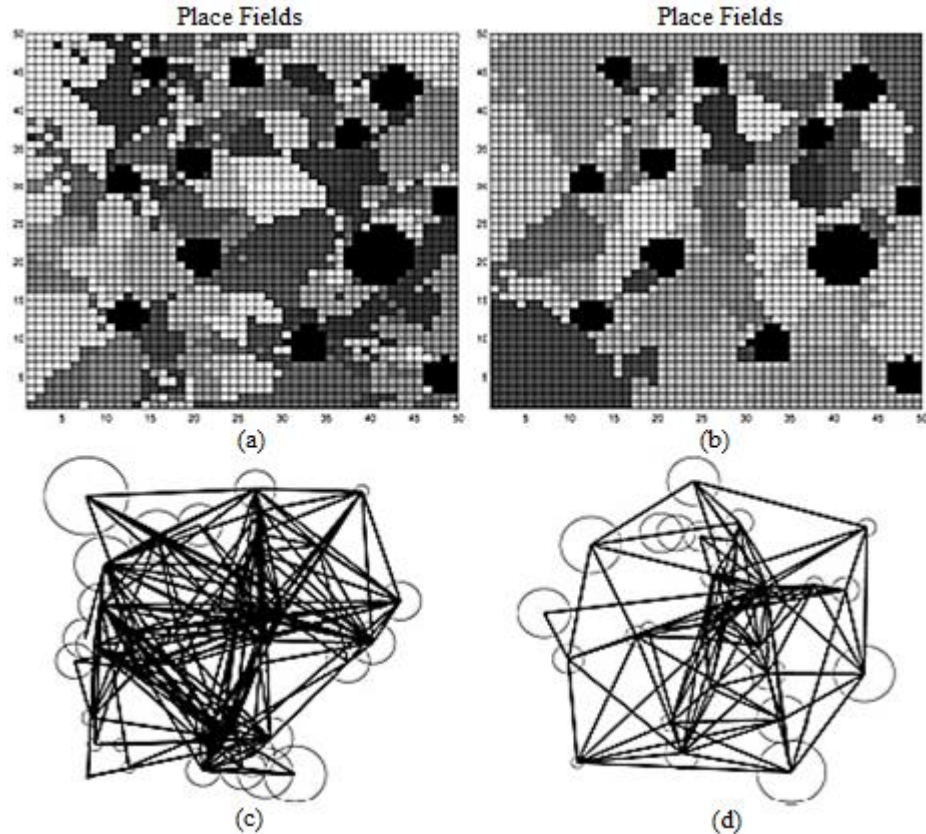


**Fig. 2.19:** (a) aMouse with a single camera and real rat whiskers, and (b) neural network with neurons at output layer represents places in the topological map. Reproduced from Hafner (2008)

Later, Hafner (2005) created her version of artificial mouse or *aMouse* which is a mobile robot that has omni-directional vision and active whiskers (see Fig. 2.19(a)). The visual field of the robot is similar to the large visual field of rats and mice and its whisker is made of real rat whiskers for realistic texture recognition. The whisker sensors of aMouse are attached to microphone membranes in order to produce high

resolution sensor data. The implementation is an effort to test if the added tactile sensor (whisker) on top of visual information aids Hafner's neural network model in performing place recognition.

Inspired by a review of map building and path-planning strategies by Meyer & Filliat (2003), Hafner attempted to compute a topological model based on the output layer of her normalised Hebbian neural network. Instead of storing images only for visual homing, each neuron in the output layer stores an image of the place visited by the robot (see Fig. 2.19(b)). These images are processed into lower resolutions before Gaussian filtering is used to reduce the noise so that the intensity curves of images captured using the omni-vision can be extracted. After aMouse randomly explored its environment, the intensity information would reveal a range of grayscale values and areas of similar values is classified as the same place cell. Fig. 2.20 denotes the 2-D grayscale image of the simulated environment. It was observed that the robot's visual input changed more rapidly when it reaches closer to the surrounding obstacles (Hafner, 2008). The implication here is that more place cells are recruited for this area, adding new place nodes to the topological graph structure. The metrical map is extracted from the topological structure using the distance and orientation between the places defined by specific weight functions in the network's output layer.

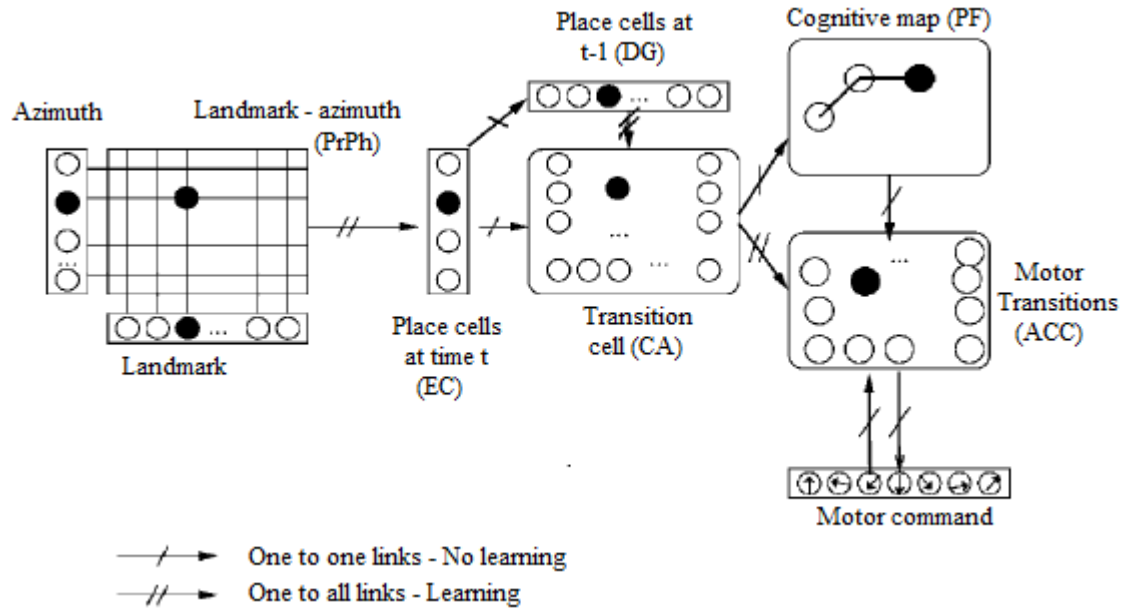


**Fig. 2.20:** (a) and (b) represent the environment in grayscale format; black areas are obstacles and different gray shadings indicate different place fields in the environment, (c) denotes the connections between places at random parameters, and (d) the interconnectivity with optimal parameters. Reproduced from Hafner (2005)

#### 2.3.4.2 Cuperlier et al.

Cuperlier et al. (2007) also propose mapping on a neural network architecture inspired by the interactions between the hippocampus and prefrontal activities. However, the representation of place cells has been redefined to include a new cell type called the *transition cells* (se Fig. 2.21). The transition cells is said to replace the traditional place cells in the hippocampus. The idea is derived from observing humans do not have to ‘see’ the map of the environment in order to use it. For instance, in using place cells to describe paths, one could say; “... at A, turn 10 degree to the left and go straight to reach B. At B, turn 40 degree to the right and go straight to reach C”. If transition cells

are used, the description is changed to “... at A use the transition AB to reach B, next use the transition BC to reach C”. This mean the transition cells holds information (distance and orientation) about the link between two places and in the network, they replaces the place cells as the basic components that make up the neurons.



**Fig. 2.21:** Sketch of the model. From left to right: merging landmarks (Pr) and their azimuth (Ph) in a matrix of neurons called product space, then learning of the corresponding set of active neurons on a place cell (ECs). Two successive place cells define a transition cell (CA). Place cell at time t-1 is in DG. Transitions are used to build the cognitive maps (PF) which are linked with movements (ACC). Diagram and description reprinted from Cuperlier et al. (2007)

With each panoramic image from the environment, the model trains the network to identify a constellation of landmarks and their orientation in the environment using a compass. They defined specific features from the image to be processed as landmarks with a sampling size of  $32 \times 32$  pixels thus making their approach different than the traditional object-based landmark extractions from visual input of the agent. The samples are then processed so its centres are the curve points. The orientation (azimuth)

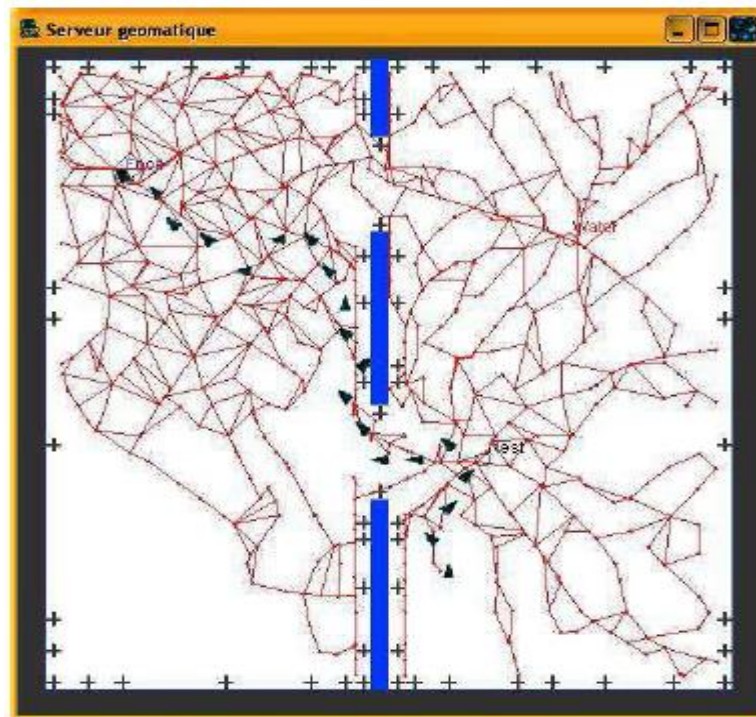
of these curve points are computed relative to North as defaulted by the compass. Both the sample and its corresponding azimuth become the 'landmark' which signifies a particular area in the environment. Fig. 2.22 shows a total of 10 different landmarks were extracted from the panoramic image which is first processed into binary pixels.

Each pairings of the landmark and its azimuth occupies a place cell (at the neuronal level). Once the trained set is setup, a matching function compares the distance between the learned set and the current set. If the current set is not matched, then a new place cell (neuron) is defined for this new location. It has been observed that when an agent traverses near walls or doors, more locations are learned since there is a rapid change in the angular positions of the landmarks. However, as the agent leaves an area the landmarks would begin to disappear from view. In Cuperlier et al. (2006), it is explained that once a transition is used, a new link is added into the cognitive map. This link represents a continuation of the path from the previous transition. These links which are numbered form the connections and denote the successive paths being traversed by the agent. The links can be trained with higher values when it is often revisited and decreased if they are not used much in the exploration. The links and landmarks representation looks like the map generated in Fig. 2.23.





**Fig. 2.22:** An example of extracting landmarks and its azimuths from an agent's visual input. Reproduced from Cuperlier et al. (2007)



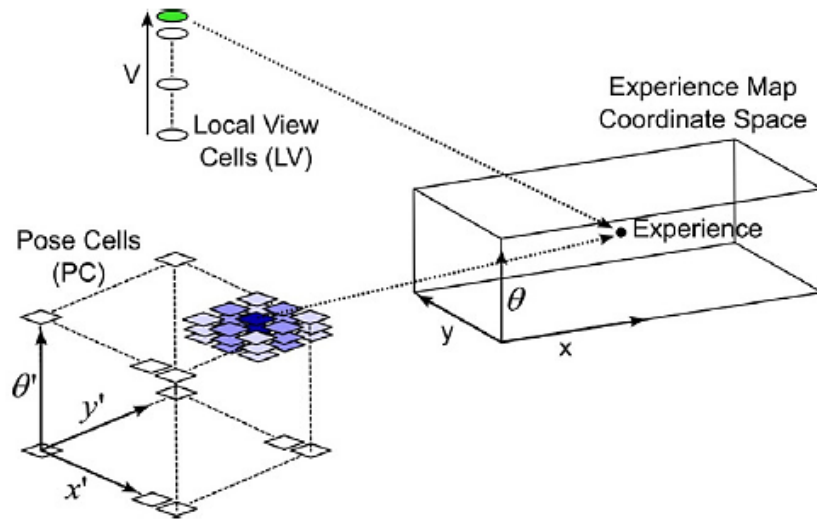
**Fig. 2.23:** The cognitive map computed using the transitions information. The triangles denote the robot's localisation and direction at the point of exploration. Reproduced from Cuperlier et al. (2006)

#### **2.3.4.3 RatSLAM**

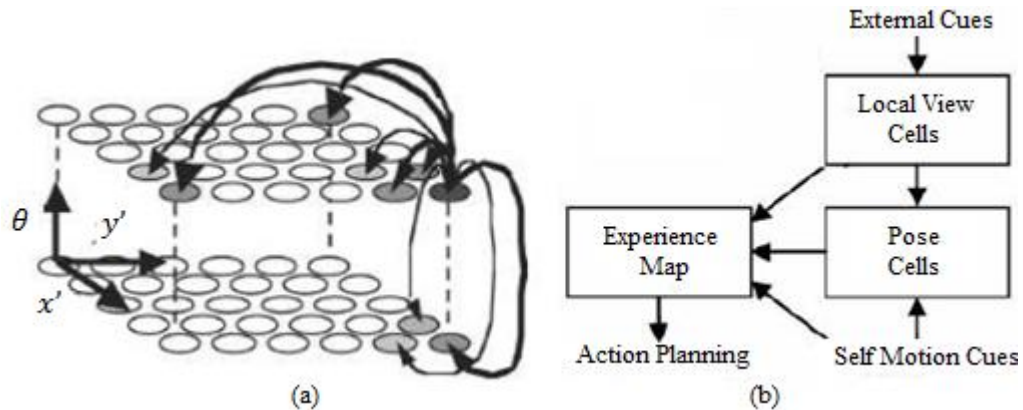
As the name suggests, RatSLAM is a mapping approach which solves the simultaneous localisation and mapping (SLAM) problems. It derives inspiration from observing how rodents build their representations of the environment by studying the patterns of neural activities which correspond to the place fields inside their hippocampus. The place fields are activated when the rodents moves about and collect visual input. Thus at the core of the model, RatSLAM has to deal with integrating odometry readings and visual stimulus in building the map similar to the SLAM's. However, as Milford et al. (2004) noted, the two approaches differ in defining what place cells or place fields are in their mapping components. SLAM treats place fields as Euclidean grids on a Cartesian plane and they could be topological or landmark-based. For the RatSLAM, place fields yield representations that are part grid and part topological. This means they are not rigid in the representation as the grids do not have to be strictly geometric nor the landmarks must be uniquely defined.

Conceptually, the RatSLAM architecture is built on three representations; the pose cells (robot's position and orientation), the local view cells (stores visual information) and an experience map (Milford et al., 2007). As shown in Fig. 2.24, activities on the pose cells and local views cells propel the creation of experiences. In return, each experience represents the activity within the pose cells and local view cells. New experience is created when current experiences cannot be matched to any activity inside existing experience map. For each new addition of the local view and/or pose, an experience node is stored and this node is linked to the previous node by a transition derived from

the robot's self-motion (see Fig. 2.25). These experience nodes are like the rat's place cells in the hippocampus. When revisiting the same place, the place cell is recognised when the same view and pose occur in the experience map. Once the robot knows it is at the same spot again in the environment, loop closing is performed by aligning the transitions and poses. Correcting the geometric errors allows the robot to estimate self-motions or localisations for the entire exploration.

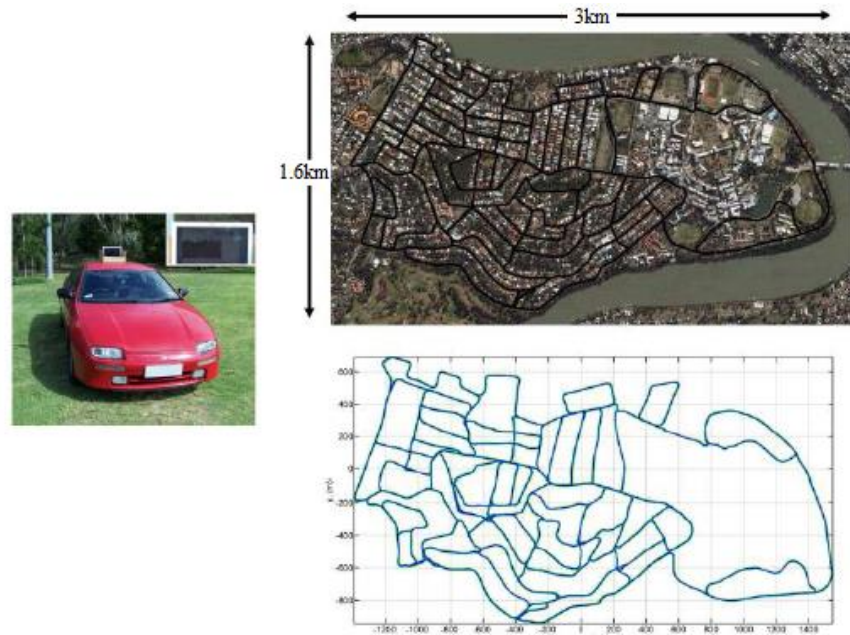


**Fig. 2.24:** Building the robot's experience map using pose cells and local view cells. Reproduced from Milford (2007)



**Fig. 2.25:** (a) Representation of the pose (position and orientation) in a 3D network with the wrap-around leads to grid-like response, (b) shows the connection between all three representations. Reprinted from Milford et al. (2007)

Fig. 2.26 denotes the results of the RatSLAM in outdoor environment. Instead of using a mobile robot, a built-in laptop with webcam is mounted on top of a car and the car was driven for 100 minutes through 3km by 1.6km in an area in the Brisbane. When the experience map is visualised, the map was able to capture a fairly accurate representation of the area traversed which could be useful for navigation. Recently, the RatSLAM (Milford et al., 2008) is augmented with a probabilistic approach to associate visual data called the FAB-MAP (Cummins & Newman, 2008). The FAB-MAP is short for Fast Appearance-based Mapping and uses recursive Bayesian estimations to infer the probability that two images seen at different times are indeed representing the same scene. While a full SLAM system since FAB-MAP has no pose data, the method on its own has successfully associated visual data on road loops as large as 1000km (for more information see Cummins and Newman, 2011). Results of combining both approaches can be seen in details in Maddern et al. (2009).



**Fig. 2.26:** An example of the RatSLAM results in outdoor environment. Picture and results reproduced from Milford et al. (2008)

## 2.4 Traditional Robot Mapping

Traditionally, the robot mapping is divided into two major groups; metric and topological. Metric mapping is an approach that utilises the geometric features of the environment (see Chatila & Laumond, 1985; Arras, 2003). The maps generated this way are also referred to as relative maps as roboticists use the concept to build maps in response to the movement of the robot in the environment (Martinelli et al., 2004). Topological mapping is normally associated with the computation of place-related data and other information that can help robot to get from one place to another (Choset & Nagatani, 2001; Tapus, 2005). There is also a new emerging concept which fuses both conventional methods termed the hybrid mapping (see Thrun, 1998; Tomatis et al., 2003). This approach typically uses a combination of metric map for precision navigation in a local space and a global topological map for moving between places.

All three approaches are similar in one sense; they are all representations centralised to attend to robot navigation which means they are useful for the robot to find its way in the environment. Two key questions being addressed in these approaches are the robot localisation; “where am I?” and the robot mapping; “what does the world look like?” By simultaneously estimating both map and robot location, robot mapping allows robots to be fully autonomous and able to operate in an environment without *a priori* knowledge of a map and without access to independent position information. This widely accepted concept in robot mapping is called the SLAM, and attempts at it are termed solving the SLAM problems.

In solving the SLAM problem, generally roboticists rely on the production of precise and complete map. Unfortunately, this often requires high-end probabilistic measures and is usually computationally expensive particularly in mapping large-scale environments. Furthermore, building such accurate maps has been reported to lack resemblance to the way biological agents computes its cognitive map. However, it is important to understand how roboticists have used different sensors to map the environment because efficiency of the mapping algorithms also depends on the physical capabilities of the robot. Learning the way the robot perceives its environment determines which mapping approach to be used and how the technique should be applied in this work.

#### **2.4.1 Metric Maps in SLAM**

A metric map is the capture of the geometric properties of the environment to build a detailed metrical description of robot's environment from sensor data (Preciado et al., 1990; Durrant-Whyte, 1998; Thrun, 2001). The geometric properties refer to the geometric relations between objects and a fixed frame of reference identified in the map presenting accurately the positions of objects inside the environment, for instance chairs, desks and walls. A common method used in the construction of metric maps is the *occupancy grid*. This technique uses a matrix to store the exact position of objects in the global frame, in which each element of the matrix can be empty or occupied, or can be an unknown area. The metric map is said to be the most explicit map in robot mapping because it only reproduces the spatial state of the environment and carries no

functional information. Also, the precision of the information given by a metric map depends highly on the quality of the robot sensors.

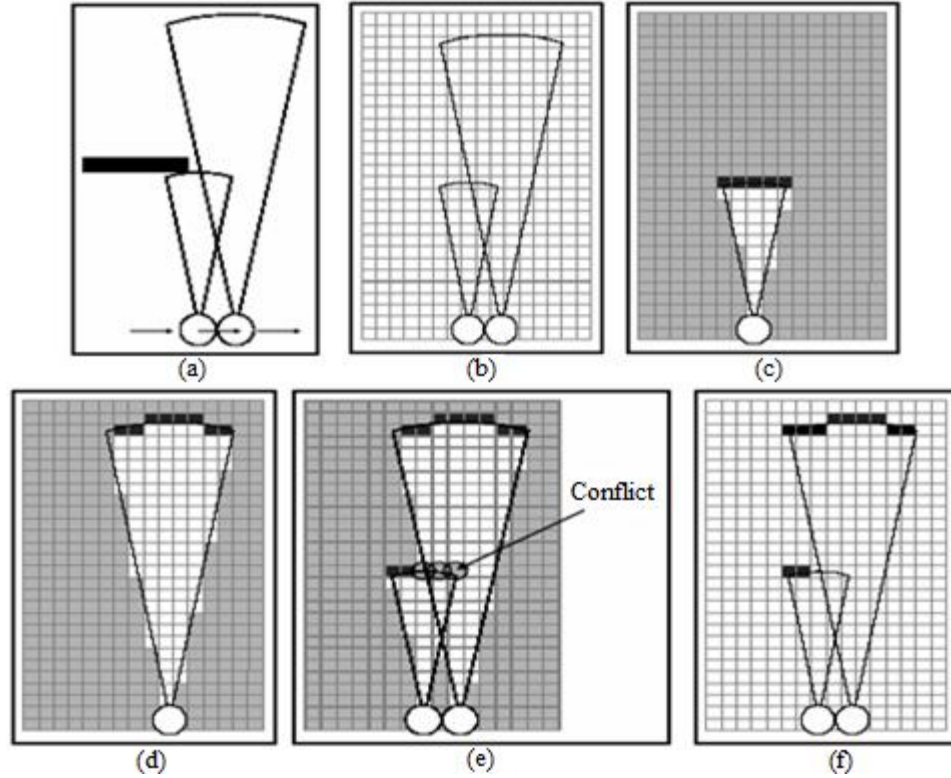
As sensors are often subject to noise, this dependence is a weakness of the metric map and it becomes more prominent as the map increases in size. Many works in space representation are based on metric maps such as the stochastic map technique to perform SLAM (Castellanos & Tardos, 1999; Dissanayake et al., 2001; Leonard & Durrant-Whyte, 1992). Another example of approach is the occupancy grids proposed by Thrun (1998). Thrun (2000) then proposed probabilistic methods that make the metric mapping process faster and more robust. These mapping techniques described above address the mapping problem with unknown robot poses. There is also a simpler version of the problem reported in the literature. It is described as mapping with known robot poses. Some examples of this approach are the occupancy grid maps developed by Moravec & Elfes in the mid-eighties, and one created by Thrun (1998).

The occupancy grid mapping addresses the problem of producing a consistent metric map from noisy or incomplete sensor data. Even with predefined robot poses, the inaccuracy of the sensor data usually makes it very hard to mark whether a place in the environment is occupied or not. More often than not, range finder sensors such as sonar and laser are favoured in occupancy grid applications as both type of sensors are characterised by noise. However, there is an extra weight in using the sonar sensors as each sonar rings usually cover a cone-like area of the space. A single sonar reading set is insufficient to tell where exactly in the cone the object is. Both sensors are also

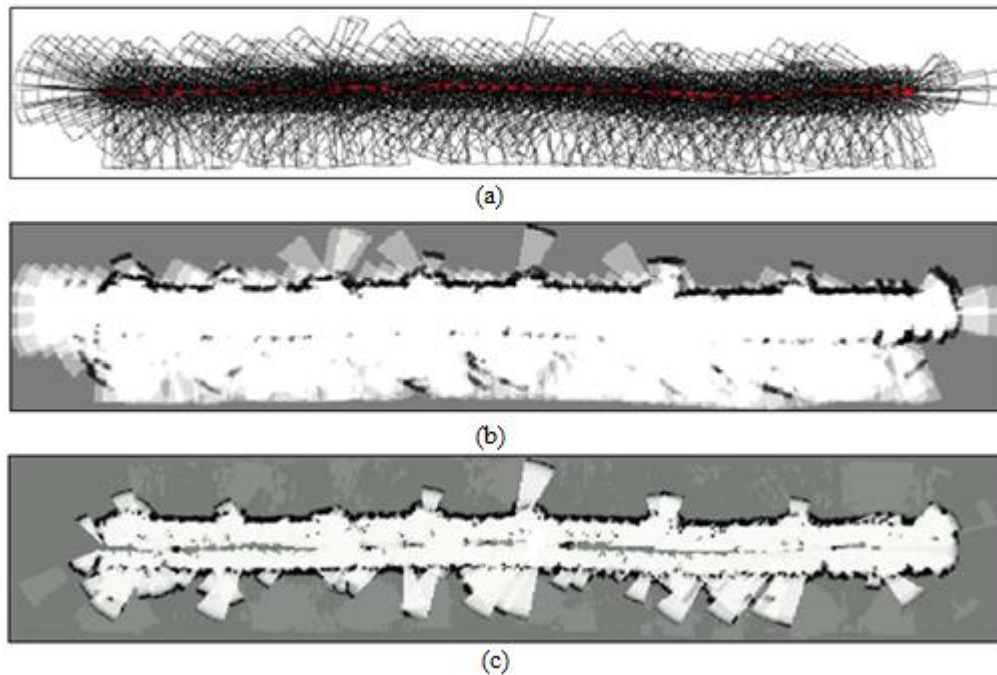
sensitive to the angle of an object surface relative to the sensor and the reflective properties of the surface (absorption and dispersion of signal).

The occupancy grid maps take care of such problems by generating probabilistic maps. Each occupancy maps are represented by grids, which can be two-dimensional or three-dimensional. The standard occupancy grid mapping algorithm normally uses Bayes filters which are used to calculate one set of occupancy grid map given the occurrence of another set of occupancy grid map. Fig. 2.27 displays the standard grid mapping process when a robot equipped with a set of sonar sensors passes by an open door. Such cluttered environment often leads to the sonar beams getting conflicting measurements about the area of the doorway, sometimes resulting in the opened doorway being closed in the final map. The conflicting regions are usually accommodated by averaging to avoid wrong interpretation of the physical data. Fig. 2.28 is an example of the real world data being mapped when a mobile robot is made to pass through a long corridor. The result in Fig.2.28(c) is used for robot's localisation, path planning, navigational strategies, obstacles avoidance and landmarks recognition (Thrun et al., 2005). However, like many other metric SLAM approaches reported in the literature, this method can become computationally very expensive for large environments.





**Fig. 2.27:** (a) Two sonar beam readings, (b) both readings projected onto same grid map, (c) and (d) show readings on individual probabilistic map, (e) conflicting regions on grid when probabilistic maps are compared, and (f) final map with removal of conflicting region. Reproduced from Thrun (2003)



**Fig. 2.28:** Grid mapping in large environment. (a) Raw sonar data over 50m corridor-like environment, (b) map with missing walls and doors in traditional grid map, (c) map with completed walls and doors in advanced grid maps. Reproduced from Thrun (2003)

Often, the uncertainties from the measurement errors must be controlled using probabilistic calculations. For instance, the final map shown in Fig. 2.28(c) is a solution called Forward Models derived from the Bayesian theory. Other solution known from the literature includes the vision-based metric approaches utilizing the Scale Invariant Feature Transform (SIFT) algorithm which detect and describe local features in images. Example of this approach can be seen in Se et al. (2002). More recent work highlights the widely popular approach called the Extended Kalman Filter (EKF) where Kong and his colleagues (Kong et al., 2006) implemented a localization system which detect features inside the surroundings such as corners and flat surfaces.

#### **2.4.2 Topological Maps in SLAM**

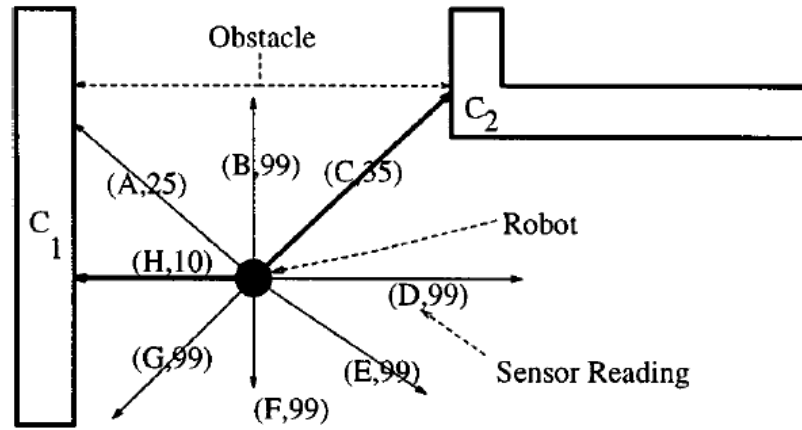
The main drawback of the metric maps approach is the rigidity in which they are computed. Researchers solving the map using purely metric maps face the challenge in controlling the odometry error of the robot. Such inaccuracy of the robot's wheel trajectory makes it difficult to maintain the global consistency of the map even if all relationships between the environmental features and the robot itself has been attended to. Also, most metrical maps do not have information about the objects and places of the environment which limits its flexibility to be extended to incorporate higher level reasoning or symbolic reasoning. In contrast to geometric mapping, the topological approach does not build a representation that is visually comparable to the environment. Rather, it is a feature-based map that uses symbolic representation that records the relationship between geometric features known as landmark (Engelson & McDermott, 1992; Fabrizi & Saffiotti, 2000). Examples of landmarks include junctions, corners and

dead ends, and they can either be natural or artificial. Since the published works of Kuipers (1978 & 1983), a lot of progress has been made by researchers to tackle the shortcomings of the geometric method. A review of Kuiper's topological SSH can be found in section 2.3.1.

One of the more popular strategies was to model the space using graphs. Unlike the metric map which is an absolute representation, the topological graph represents the environment as a list of significant places which carry information on how a robot can travel from one place to another. Two key properties of the graph; the nodes and the arcs, correspond respectively to the places and the paths from the environment. When two nodes are connected without any obstacle in the graph, it means that they are adjacent places in the real environment.

In Choset & Nagatani (2001) a new method using Generalized Voronoi Graph (GVG) to solve SLAM is proposed. The main goal is to exploit the topology of the robot's free space on a partially constructed map. Using a graph matching process, their robot is able to localise itself by tracing two points from the walls so it is always positioned in the middle part of the pathways. When it is no longer able to find the two points, meaning it reaches some point where the distance threshold is met; the robot will choose to follow the obstacle boundaries to perform localization. Fig. 2.29 is a sample of localisation strategies worked using the GVG algorithm. Lisien et al. (2003) extended this graph matching concept to H-SLAM (i.e. Hierarchical SLAM) in which a high-

level topological map organises a collection of low-level feature-based maps creating a hierarchical approach to the topological SLAM.



**Fig. 2.29:** Example of a GVG topological solution in performing robot's localization. Readings from 8 range-finders are used to estimate the nearest obstacles with H and C the two lowest values in the sensor array (A is not considered as it is not a local minimum). When two adjacent sensors have similar values, the closest obstacle is assumed to be in between the two sensors. Reprinted from Choset & Nagatani (2001)

### 2.4.3 Hybrid Maps in SLAM

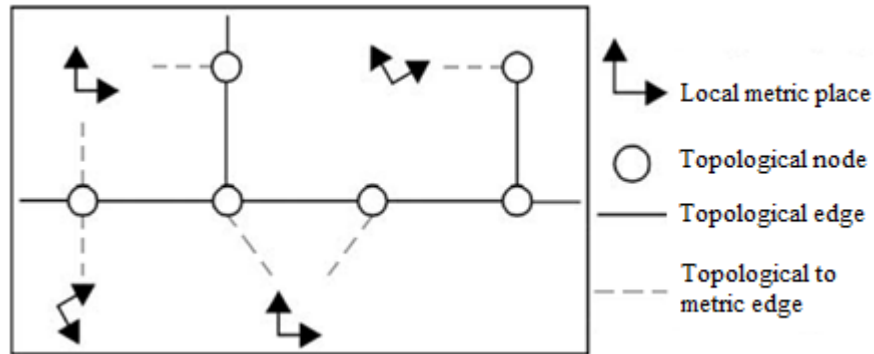
It has been a common assumption that two distinctive places can be easily recognised, but in robot reality, it is not so easy to differentiate between the two places. The absence of compact and dense metric data in topological maps, by principle, should make the coarse, graph-structured topological maps fare better in recognizing an already visited place. However, closing the loop remains a fundamental problem in topological mapping because these techniques often ignore valuable metric information which affects its ability to describe individual objects in an environment, a feat that is easily solved with richer sensory information and unique features abstraction offered by the grid techniques.

The lack of one-to-one relationship between the matrix grid and the geometry of the represented environment proves to be computationally inadequate in practice when dealing with place recognition particularly in large-scale surroundings. Moreover, even though purely landmark-based approaches are fairly robust against sensor noise and small environmental changes, they also rely heavily on the global consistency of the topology. This proves to be a challenging task as the map lacks recognizable landmarks which make minimizing the number of topological nodes in the network difficult as it contains redundant information that is hard to remove (Kuipers et al., 2004).

Since metric and topological maps focus on different aspects of the same environment, their integration is useful in producing a robust map which contains both the quantitative and qualitative information of that environment. In a hybrid representation, the environment is described using a global topological map and local metric maps. When traversing from one place to another, the path within the whole environment is planned using the global topology so the robot is able to find the connecting route to a targeted place. Once it reaches the target, navigating inside that place is done via the local metric map to ensure precise localization and safe navigation.

Works which are prominent in the concept of enriching the topological map with metric information include these references (Habib & Yuta, 1993; Tomatis et al., 2001; Tomatis et al., 2003). Their hybrid models describe the requirement to switch from the topological to the metric paradigm each time their robot traverses between the local metric maps. Since the local metric maps are independent from one another, it allows

for a compact environment modelling which does not require global metric consistency and permits both precision and robustness. Fig. 2.30 shows how hybrid maps are used.



**Fig. 2.30:** Example of a hybrid map. The environment is represented with global topological nodes and local metric places. When travelling from one node to another, the system switches from topological to metric and vice versa. Reproduced from Tomatis et al. (2003)

Advancement of vision techniques in indoor SLAM has allowed production of visual hybrid map of a mobile robot's working environment. The construction of a hybrid visual map has been done on image based navigation along the lines of appearance based mapping (Cummins & Newmann, 2008) and topological SLAM (Angeli et al., 2008a). Earlier, the hybrid visual map was proposed using two levels of representation; an absolute map representation of distinct visual planes using the Extended Kalman Filter (or EKF-SLAM), and a relative map representation of dense visual features for each visual place done using sparse information filter update (Ahn et al., 2007). However, it has since been a common technique to use semantic approaches in reducing the number of nodes in the topological graph through a combination of local and global decision making strategies (Krishnan et al., 2010).

Utilizing images, these vision based explorations provide dense range information (Sim & Little, 2006) or as a classification of the surrounding area around the robot pose (Santosh et al., 2008; Krishnan et al., 2010). Given that it is possible to extract a lot more information about the objects and spatial layouts from an image, researchers are able to obtain the semantic construction. The knowledge is then applied in robot operations where higher level understanding can be used to formulate an effective exploration and mapping strategy in tackling visual homing (Filliat, 2008) as well as loop detection (Angeli et al., 2008b). Nonetheless, these semantic approaches sometime require user interventions where the robot is either guided to collect the images for learning (Filliat, 2008) or made to move along predefined paths (Angeli et al., 2008b) limiting the robot's autonomy operating in its own surroundings.

## **2.5 Chapter Summary**

This chapter shows that there has been a significant progress particularly in the development of computational models for cognitive mapping. Significantly, implementation and testing of these theories have been expanded on the mobile robot platform. The current thesis investigates how a cognitive theory such as the ASR can be borrowed *into* robot mapping. However, the current implementations of the computational models including the SSH and PLAN are basically using the mobile robots as test-bed to get more insights on the cognitive mapping process itself. None of these approaches have been adopted into a *robot mapping* work. This work is the first attempt at doing so and in return the work presented a novel approach to using an inexact map for a robot. The following Chapter 3 discusses how a cognitive theory such

as the ASR can inspire a novel robot mapping work that computes and uses an inexact representation of the environment without any error corrections.



---

## The MFIS & ASR Computation

---

### 3.1 MFIS Computation

The MFIS is a representation that describes the spatial layout of one's immediate surroundings using a single global co-ordinate system. The MFIS is viewed here as similar to the global map computed using the traditional robotics SLAM approach except that (i) it is not an exact map, and (ii) it is not meant to be a map representing the entire environment that the robot has experienced. Psychologists often describe the implementation of such a map using an egocentric frame of reference (Klatzky, 1998; Wang & Spelke, 2000; Mou et al., 2004). That is, the spatial layout of surfaces in one's immediate surroundings is described using a co-ordinate system centred on the self (see Yeap, Jefferies, & Naylor (1991) for an alternative approach). As one move, the co-ordinates of these surfaces are updated but at some point, surfaces that are far away

from the self are simply removed from it. The extent of such a map has little been studied.

In this chapter, we present the algorithms needed for the robot to compute its MFIS and from it, extracts a topological network of local environments (ASR). The transient nature of the MFIS will not be studied here (if readers are interested, see Jefferies, 1997) and consequently, the MFIS is computed for the entire test environment. The robot explores its environment autonomously and computes its map.

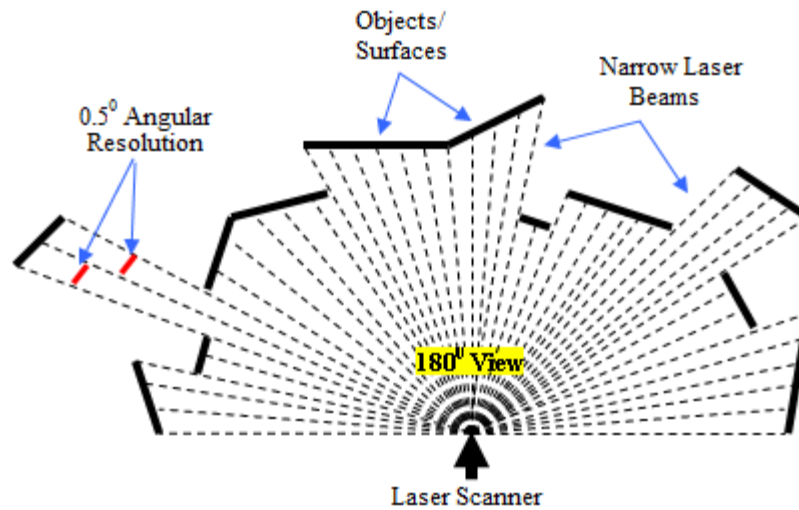
Section 3.1 discusses the setup of the robot and the input it gets from its laser. Sections 3.2, 3.3, and 3.4 discuss the algorithms developed in this thesis for autonomous navigation, MFIS computation and ASR computation respectively. Section 3.5 presents a summary of my approach to robot mapping.

## **3.2 The Robot: Its Sensors and Input**

The robot used is a mobile robot of the model Pioneer 3DX from *MobileRobots Inc.* commonly used in indoor environments and by default its base is factory manufactured with 8 rings of sonar sensors for data collection. However, for the purpose of this work, the base has been fully integrated with a set of SICK LMS-200 laser rangefinder instead and the sonar sensors are ignored. Even though such a robot is selected for the work, the algorithms developed in this thesis are not strictly dependent on the particular robot hardware and sensors.

The laser rangefinder used on the robot is a non-contact optical device which emits pulsed laser beams. When set up to collect data, the laser beams are fired to hit objects in the scanning field where part of those beams are reflected back to the sensors. Data are gathered by calculating the distance between the sensors and the objects, which according to the time of flight (TOF) principles applied, is directly proportionate to the time between the transmission and reception of each pulsed signal.

The sensor's scanning range can be modified according to the resolution setting but in order to capture the most out of the scanning field in an indoor environment, it has been left to scan at the maximum range of approximate 30-32m. Separated at  $0.5^{\circ}$  angular resolutions, each laser beam travels a long distance in a straight line, maintaining a narrow beam which covers a wide  $180^{\circ}$  field of view for a total of 360 laser point collections at any single scan. During each scan, the laser pulse is diverted sequentially using an internal rotating mirror which results in a fan-shaped 2D scan of the surrounding area (see Fig. 3.1). Laser sensor is chosen for the robot mainly because of its accuracy and high sampling density but the beam readings may sometimes face multiple reflections during scanning. The environment used for testing in this work is a standard office environment with a combination of carpeted flooring in the rooms and tiles surfacing in most of the connecting corridors.



**Fig. 3.1:** The laser taking samples from the environment at  $0.5^\circ$  angular resolutions with  $180^\circ$  scanning field. Drawings of the narrow laser beams are an approximate and not to scale

For this robot to map the spatial extent of an indoor office environment, the ideal candidates for surfaces have to be the walls enclosing and partitioning the spaces or the rooms. However, given its size, the robot has its sensors positioned at approximate 45-50cm height from the ground. The sensor cannot be panned, tilted or elevated and can only emit long-range laser beams horizontally in a straight line. With these limitations, it is impossible for the robot to avoid scanning the office furniture such as desks, chairs, cupboards, bins, pots and boxes scattered around in the office rooms as well as in the kitchen, lounges and larger corridors while searching for the limits of the space (i.e. the walls).

Furthermore, due to limited recognition abilities of the laser sensor, the robot is not able to distinguish between these objects or choose the ones that are most likely to be useful to assist the spatial mapping tasks. The only decision that the robot has control over is what to do to the surfaces once they are computed. Using size of the surfaces as

reference, those that are long are regarded as important since they have the highest chance to be the walls or part of the walls or some major obstacles to avoid during exploration. Tiny surfaces computed may not be as important to the robot and they can be dismissed as junks.

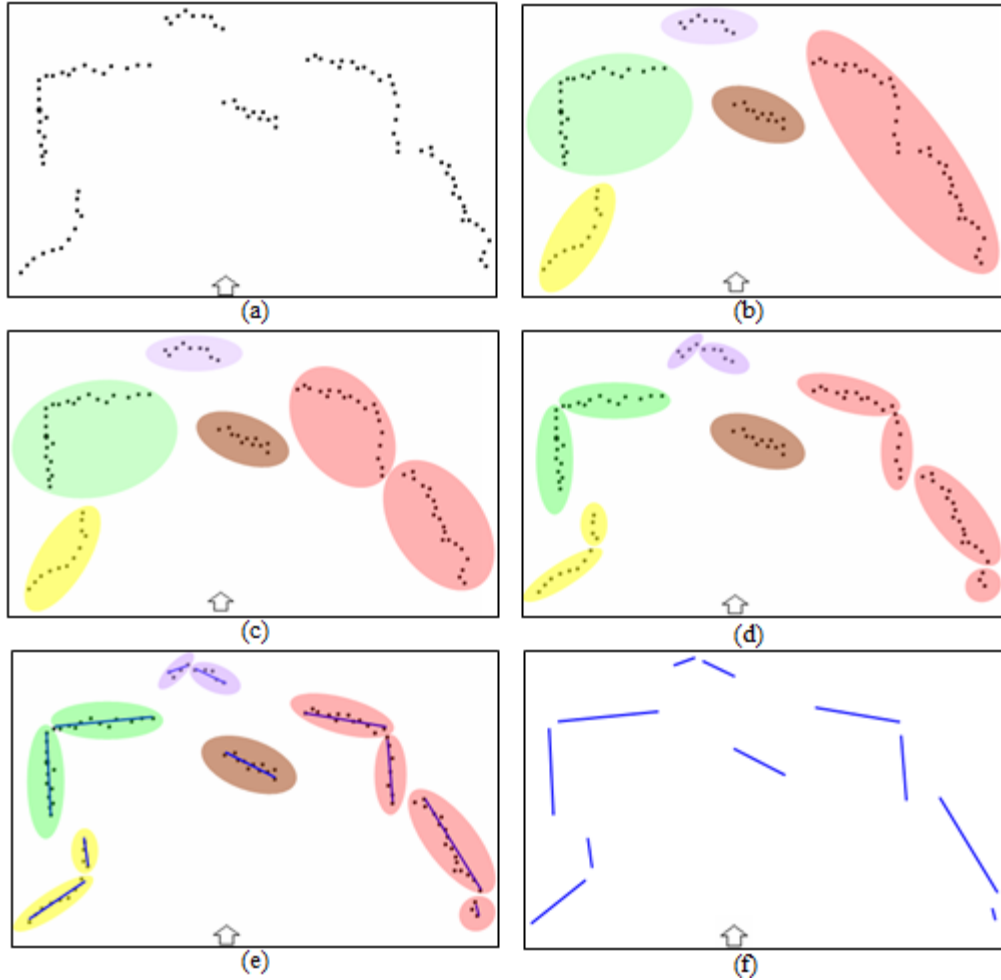
Given that laser beams are in the form of light and light penetrates through transparent materials, all glass walls in the office are covered using cardboards throughout the implementation. This is important because if the robot does not realise it is scanning through a glass wall, the area where the glass wall lies will be computed as an empty space. If the robot decided that the empty space is an interesting location for the robot to explore, this could lead to property damage should the robot manoeuvred right through it. Thus, covering all forms of highly transparent materials with something solid like cut out cardboards is necessary for safety reasons. Note that this is the only physical modification done to the environment. Everything else (position of things, opening or closing of doors to the rooms, etc.) is left as it is. After making sure only solid objects accommodate the robot's scanning field, the environment can then be treated as experiment-ready. All geometrical properties scannable as surfaces to the robot are decoded as straight lines in the implementation.

With the scanning range set at maximum, it is very rare for the laser sensors to return a no object detected reading since 30-32m is often enough to cover the distance to the closest objects in the structured indoor spaces. Nonetheless, if the scanner does return valid readings with values exceeding the maximum detection distance, they are filtered

out as spurious data. The 2D range data are then processed using line segmentation algorithm to generate planar surfaces so they would correspond to the geometrical properties scanned from the environment.

There are many sophisticated algorithms such as the popular split-and-merge algorithm (Borges & Aldon, 2000; Zhang & Ghosh, 2000; Castellanos & Tardos, 1996), the line regression algorithm (Arras & Siegwart, 1998), the incremental algorithm (Vandorpe et al., 1996; Taylor & Probert, 1996) and the Hough transform algorithm (Jensfelt & Christensen, 1998; Pfister et al., 2003) to perform line extraction from points; all interested in providing an accurate polygonal model of the environment. However, since we do not need to build an exact map, precision is not of utmost important and hence a simple line approximation from points based on the spatial relationship of neighbouring laser information suffices.

A straightforward method for computing lines from laser points is thus implemented. First, the laser points are grouped into different clusters. This is done by going through the laser readings one after another in a clockwise sequential manner and calculating the Euclidean distance between them. If the distance between any of them exceeds a set threshold (currently set at 1.2m), a new cluster is formed. Second, for each cluster, the exact shapes of the lines in it are recursively computed using the average gradient descent between neighbouring points. Points on the same slope are grouped as a line representing a surface (see Fig. 3.2). Note that, for simplicity, small surfaces (of size < 500mm) in view are simply ignored.

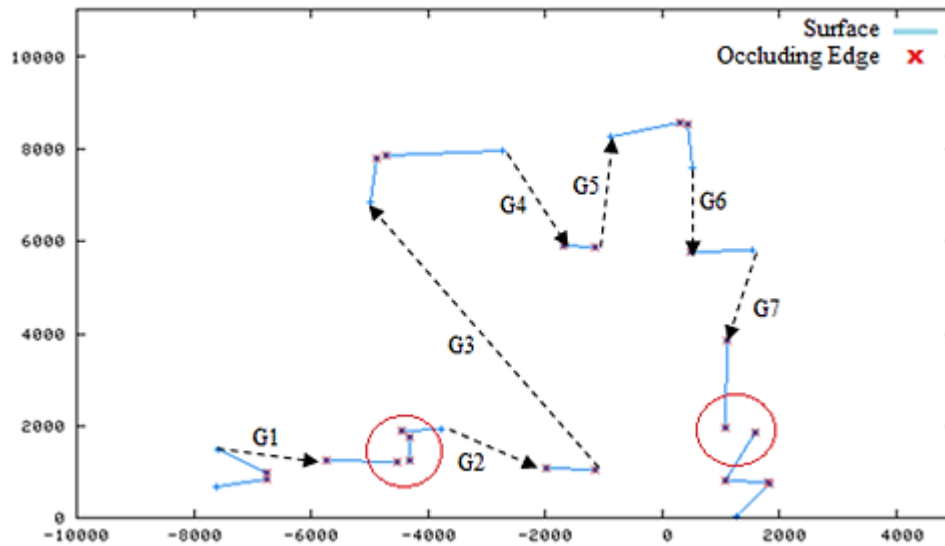


**Fig. 3.2:** Processes involved in extracting surfaces from the environment

### 3.3 An Algorithm for Autonomous Exploration

For exploring the environment autonomously, the robot must decide where to go next and how to get there. This is very different than say allowing the robot to simply move about by avoiding obstacles in the environment since the latter may not guide the robot to get out of a room. The algorithm here is about getting the robot to explore as much of the environment as possible on its own. Therefore, the basic strategy opted for autonomous exploration is for the robot to move towards a gap, selected at random,

found in each view (see Fig. 3.3). A gap is initially defined as an empty space (large enough for the robot to cross i.e.  $> 0.6$  meters) between two adjacent surfaces in view, scanned in a clockwise or anti-clockwise direction. However, this strategy was found, after rigorous testing, to be inefficient for two reasons. First, by definition, one could not see what is beyond the gap from where one is. Thus, using such an algorithm, one often arrives at a gap that does not open up well to other parts of the environment. Second, the robot is often unable to move straight to, say, the mid-point of the gap targeted. This is because the robot could easily drift and bump into an obstacle along the way. These gaps are usually far enough that any drifts from the robot could be significant. For example, going to the mid-point of G3 in Fig. 3.3, the robot could easily bump onto the surface immediately to its left due to drifts.



**Fig. 3.3:** Gaps (G1-G7) found in a view. The robot is at (0, 0). Circles indicate small gaps that are ignored

Consequently, to reach any target gap, the robot has to move incrementally towards it, taking a small step at a time. In the implementation, each step of the robot is limited to a maximum of 3 meters and each turn,  $30^\circ$ . Doing so meant that the robot would



necessary pause and could take a new view of the environment. If so, it is unnecessary to target a gap in view unless it is close to the robot itself. Rather, one should calculate a suitable point in space for the robot to move to next. My algorithm calculates such a point by finding the minimal bounded space for the robot.

The minimal bounded space is defined as one that contains no gaps that can be covered by another gap in view. Yeap & Jefferies (1999) introduced the notion of *covering* by a gap as a space in which an individual must cross in order to reach another part of the environment that is currently in view. They used the idea for computing ASRs. I have used it here to compute the minimal bounded space for the robot. My algorithm for autonomous exploration can now be described:

**Algorithm #1: Autonomous exploration:**

1. Identify the gaps in view (gaps are defined above)
2. Compute the minimal bounded space
3. Select a gap on the new boundary as target
4. Move towards the gap
5. Repeat

Here I introduce the notion of an exit gap. In my test environment, a typical office environment, one finds many doorways, a gap of a certain size (defined here as between 0.6 to 1.2 meters). It is useful to recognise such gaps as special because they often lead from one room/corridor to another, thus enabling the robot to “escape” the current local space (Yeap & Jefferies, 1999; Jefferies et al., 2001). These gaps, when detected in a view, are labelled as *exits* and they are important targets (as opposed to other gaps) to

be remembered. The robot prefers moving towards exits than other gaps. Consequently when found, they will be remembered in the view. The algorithm for computing the minimal bounded space (step 2 above) is given below (this algorithm includes remembering exits found in the current view):

**Algorithm #2: Computing the minimal bounded space:**

Input:  $G = \{ \}$  (a list of gaps,  $G_1 \dots G_n$ , obtained from the current view in a clockwise manner)

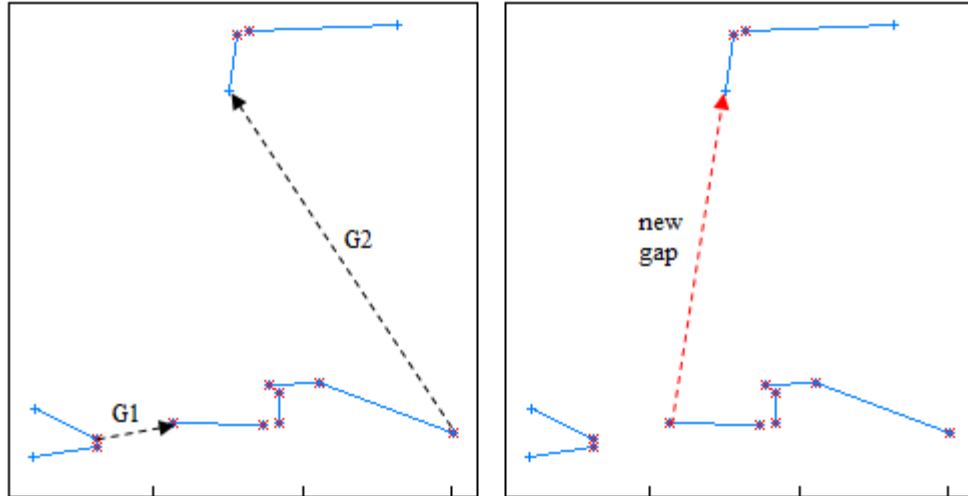
$E = \{ \}$  (a list of exits,  $E_1 \dots E_n$ , i.e. gaps measured between 0.6 to 1.2 meters)

1.  $I = 1$ ;  $G_{\text{new}} = \text{nil}$
2. While  $i < n$  do:
  - 2.1 If the start-point of  $G_i$  and the end-point of  $G_{i+1}$  are both occluding point do:
    - 2.1.1 Create a new gap joining the start-point of  $G_i$  and the end-point of  $G_{i+1}$ .
    - 2.1.2 Create a new line connecting the mid-point of the new gap and robot
    - 2.1.3 If this new line does not intersect any surfaces in view, do:
      - 2.1.3.1 If this line is an exit, save it in  $E$  else save it in  $G_{\text{new}}$
      - 2.1.3.2 Delete  $G_i$  and  $G_{i+1}$  from  $G$
      - 2.1.3.3  $i = i + 1$
  - 2.2  $i = i + 1$
3. If  $G_{\text{new}}$  is not empty, add  $G_{\text{new}}$  to  $G$ , re-order the gaps in it and goto step 1
4. Finish

Output:  $G = \{ \}$  (a list of gaps computed)

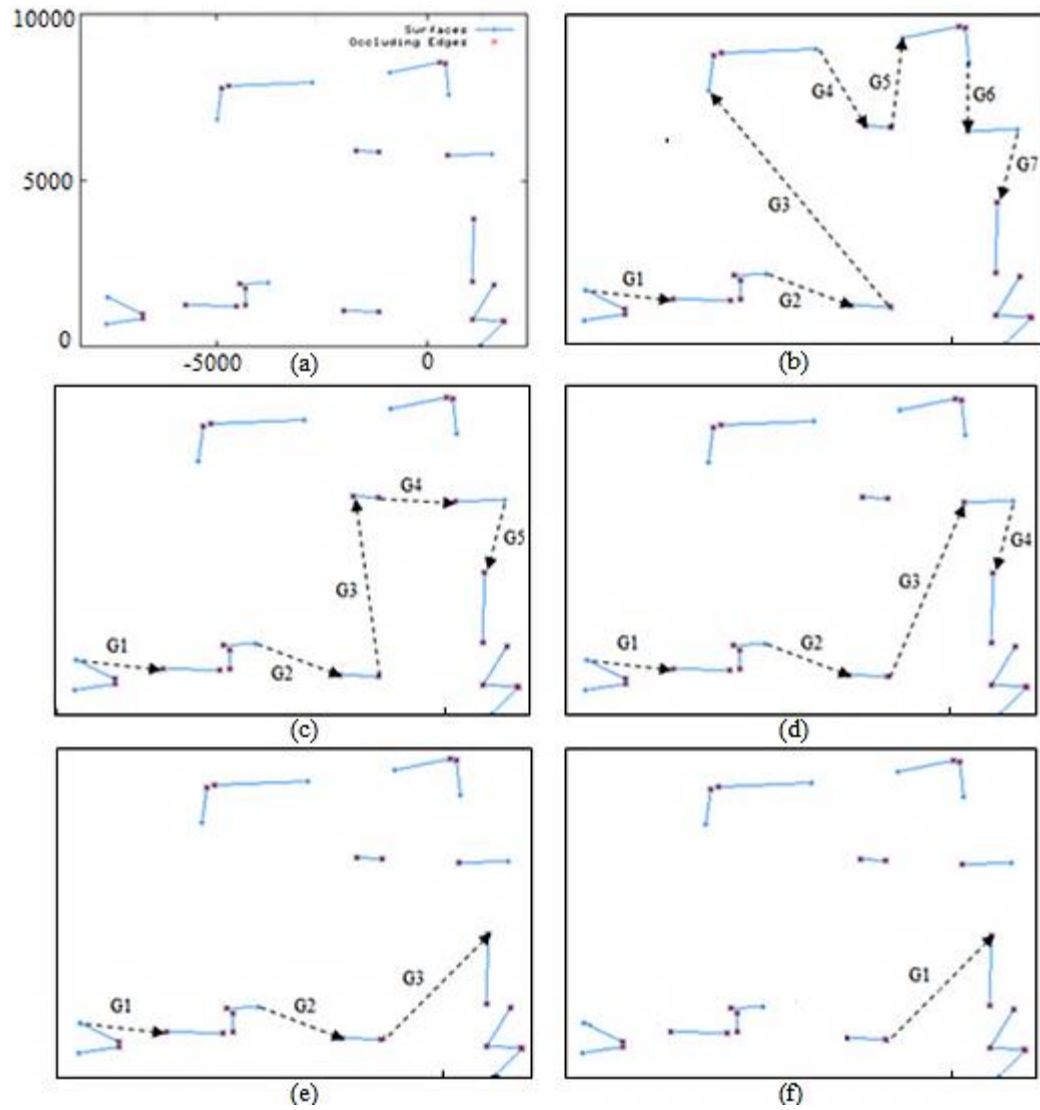
$E = \{ \}$  (a list of exits computed)

Note that the test condition 2.1.3 is needed to ensure that the gap identified is “inside” the bounded space (Fig. 3.4)

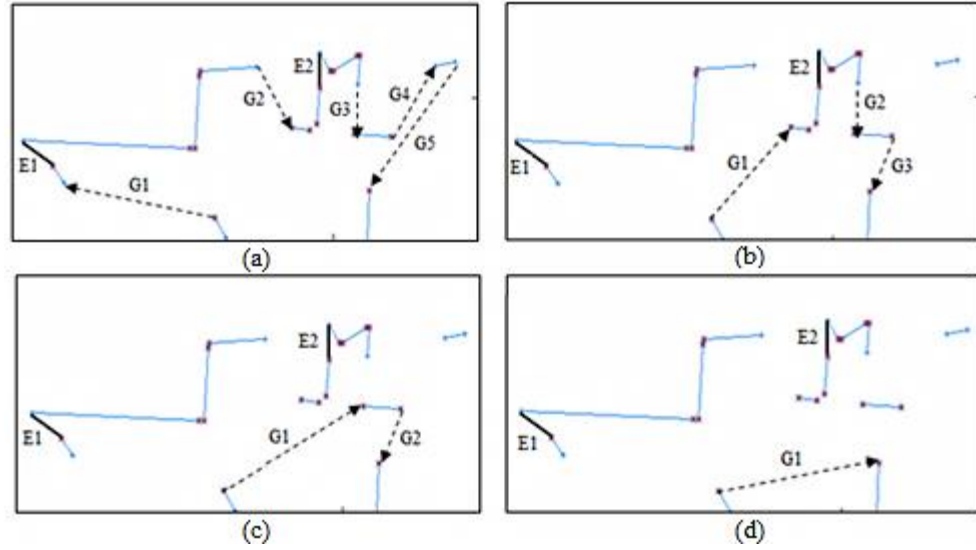


**Fig. 3.4:** Creating a new gap for G1 and G2 will produce a gap that is outside the current bounded space. Such new gaps are illegal

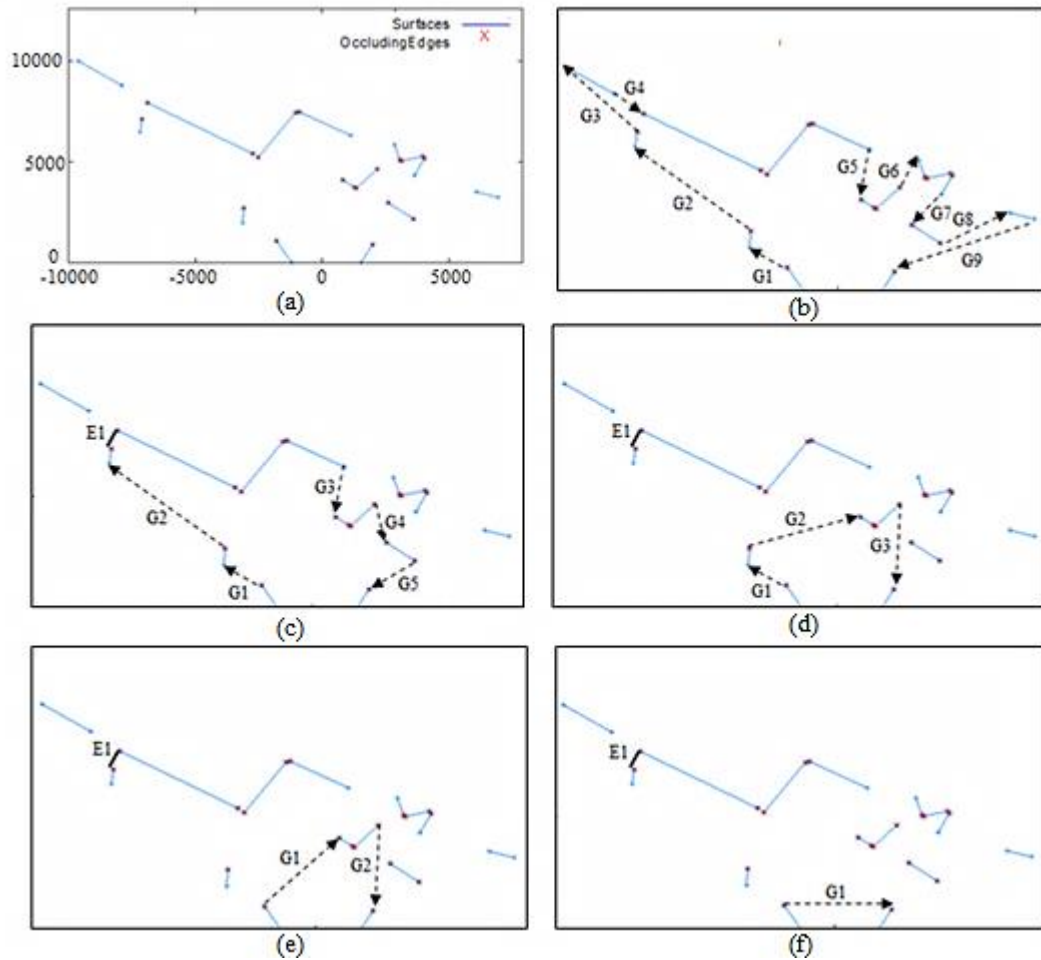
Fig. 3.5 to 3.7 presents three examples showing how algorithm #2 works. Fig. 3.5 shows the initial view and the iterations of the algorithm to produce the final gap for the example. Figs. 3.6 and 3.7 show the robot in the same position but with different viewing angles. In the former, two exits are identified in the initial view and in the latter, an exit is discovered when computing the minimal bounded space. Although in all three examples, the robot moves to the nearest boundary,  $G^1$ , the robot is aiming for  $E^1$  in the latter two examples. The robot is programmed to choose a potential exit that is furthest away; thus, giving it an opportunity to explore the current space. If the exit chosen turns out to be a dead-end (a false exit), the robot is simply instructed to rotate backward and re-scan the environment. If no exit is present, the robot moves to the gap that is directly in front or to the largest available gap.



**Fig. 3.5:** (a) The view, (b) with gaps identified, (c) first iteration with G3 and G4 replaced and gaps renamed, (d)-(e) two final iterations. (f) Removal of G1 and G2 as they are not made of two occluding points



**Fig. 3.6:** (a) Two exits are identified in the initial view in addition to the gaps; (b)-(c) show two iterations of the algorithm, and (d) shows the final output

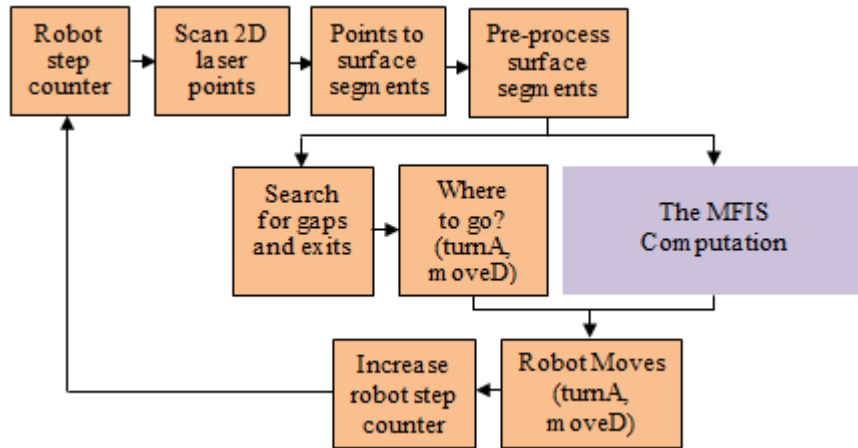


**Fig. 3.7:** (a) The view, (b) with gaps computed, (c) first iteration which produces an exit E1, (d)-(e) two final iterations, and (f) the final output

### 3.4 MFIS Computation

The robot explores the environment autonomously and computes a map of its immediate surroundings. Every time the robot moves and perceives a new view, it initiates two different processes. One is to decide where to go next and the other is to update its MFIS (see Fig. 3.8). This section describes how the latter is done. Like each view, the MFIS is implemented using a single Cartesian co-ordinate system to describe the surfaces perceived in its immediate surroundings. It thus consists of a list of surface descriptors and each descriptor consists of its own unique name and spatial co-ordinates:

$$\text{MFIS: } [(S_1 (x_{11} \ y_{11}) (x_{12} \ y_{12})) (S_2 (x_{21} \ y_{21}) (x_{22} \ y_{22})) (S_3 \dots)]$$



**Fig. 3.8:** Processes shaded in orange summarises the flow where the robot pre-process the laser scans, make decision on the next target before planning its movement whereas the process shaded in purple denotes where and when the spatial perception computation occurs in the implementation

The MFIS is a structure built from integrating successive views as the robot explores its environment. Initially, it is initialised with the robot's first view,  $V_1$  and will use the co-ordinate system of the robot's first view to describe the spatial layout of its surfaces. To add subsequent views to it, a straightforward method is to transform the co-ordinates of the new surfaces or parts thereof in each subsequent view to that of the MFIS using the standard co-ordinate transformation formula as shown below:

$$x' = x \cos\theta - y \sin\theta + \delta x$$

$$y' = x \sin\theta + y \cos\theta + \delta y$$

Where

$x'$  is the transformed  $x$ -coordinates

$y'$  is the transformed  $y$ -coordinates

$\theta$  is the robot's turn angle and is always negative in the calculation

$\delta x$  is the translation in  $x$ -direction which is always zero, and

$\delta y$  is the translation in  $y$ -direction which is always negative

To transform surfaces in the current view so that it appears as if the robot is looking at them from its new position in the subsequent view, the surfaces must be rotated first *opposite* the direction the robot took in the move. This explains why  $\theta$  is always negative in the formula. Due to the fact that the distance travelled by the robot is measured as a result of the robot traversing in a forward manner, the surfaces are then translated *backward* using the distance traversed by the robot in the move. This explains why  $\delta x$  is always zero and  $\delta y$  is always negative; the transformation must shift the surfaces backward as if the robot is leaving them behind.

Using the above formula, and as noted in the Introduction, errors in measuring the  $\theta$ ,  $\delta x$  and  $\delta y$  will cause distortions in the map computed and robotics researchers have produced algorithms to correct the errors and produce an accurate map. However, to maintain such a map for a large environment in practice is not easy (see reviews of such work in chapter 2).

My goal therefore is to compute an approximate map that is good enough for one to orient in it. To do so, I implement a strategy that has been commonly observed in cognitive species i.e. the use of landmarks for orientation. However, the idea of a landmark as interpreted in the cognitive sciences often refers to objects found in the environment that are perceived as salient, unique and/or special. Recognising them informs one's whereabouts but these objects are typically found few and far between in the environment. In contrast, we argue that surfaces that could be seen in consecutive views can serve the role of a landmark. This is because recognising them in the later views allows one to localise one's position and the spatial arrangement of other objects in the current environment even though one has moved. For example, if, say, the same table or doorway is seen in consecutive views, one could use the table as the common reference point for describing the objects seen in these views.

To use such an algorithm, one must assume that there will always be overlapping surfaces between consecutive views of the robot. This is generally true, however, by not allowing the robot to take too large a step forward or turn too far to its left/right. My algorithm for computing an MFIS is described below:



### Algorithm #3: Computing MFIS:

Input:  $V_{n-1} = \{S'_1, S'_2 \dots S'_n\}$

$V_n = \{S_1, S_2 \dots S_n\}$

MFIS initialised to  $V_1$

1. Transform surfaces in  $V_{n-1}$  onto  $V_n$  using the transformation formula described above.
2. Remove all  $S'$ , in  $V_{n-1}$  that cannot be seen in  $V_n$
3. Identify and if necessary normalise landmarks in  $V_n$ .
4. For each surface,  $S$ , of  $V_n$  that is not recognised as a landmark, transfers it to the MFIS.
5. Remove all  $S'$  in  $V_n$  and  $V_n$  becomes the next  $V_{n-1}$

Output:  $L_n = \{L_1, L_2 \dots L_n\}$

$L_{n-1} = \{L_1, L_2 \dots L_n\}$

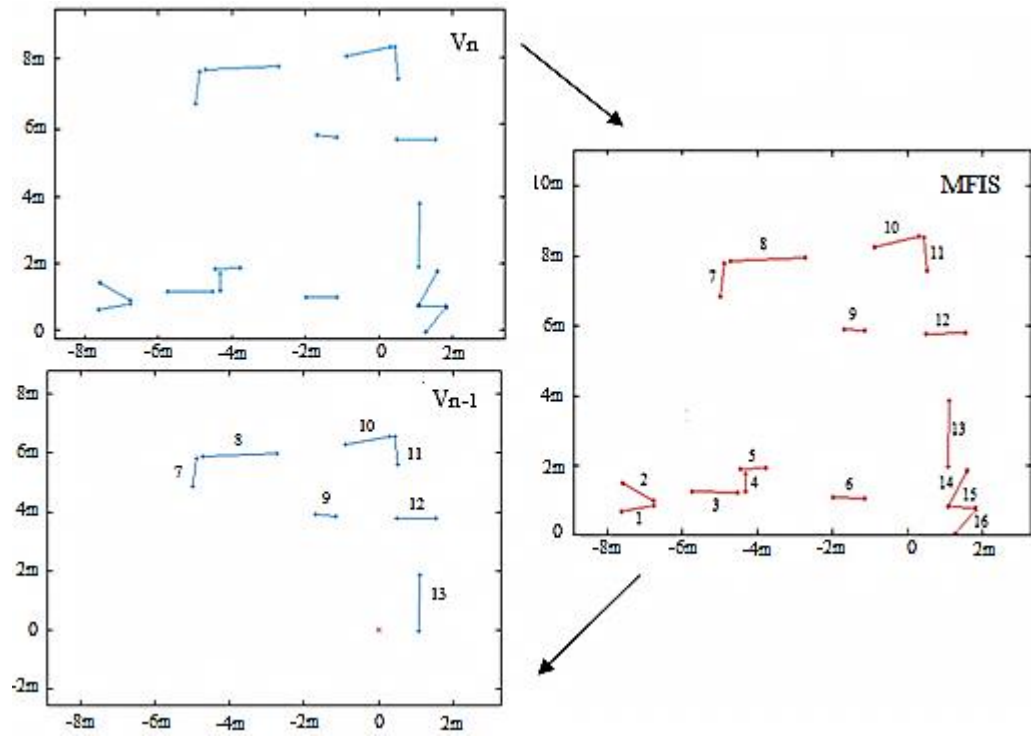
$V_{n-1} = \{S'_1, S'_2 \dots S'_n\}$

The initialisation of the MFIS using  $V_1$  involves transferring all the surfaces in  $V_1$  into the MFIS and giving each surface a unique ID (a number). Doing so also meant that the MFIS uses the same co-ordinate system of  $V_1$  to describe the surfaces in it.

For every move, landmarks are identified in the current view and using them, any new surfaces that have appeared in the current view will be updated in the MFIS. This is the basic loop for getting the MFIS up to date. However, since my robot can only “see” the surfaces as lines, it cannot use a more powerful landmark recognition algorithm (i.e. one based on richer features of objects such as colour, texture, shape, etc.). Consequently, we implemented the traditional transformation approach (above) to transform surfaces in the previous view to the current view and surfaces from both views that are found to be in close spatial proximity are evaluated as landmark surfaces

(steps 1 and 2). Note that while this landmark recognition algorithm employs the same transformation method that robotics researchers used to compute their global map, they are not the same process. The latter process is about integrating what is new with what was remembered whereas the former process is about “recognising” landmarks. Thus, in the former process one is concerned with correctly matching all surfaces and computing their exact locations while the latter is concerned only with the identity of the surfaces. It is also not essential to find all matches (i.e. landmarks) and this is important given that the method provides only a crude means of finding landmarks. It is alright that some will be missed and others that are doubtful will be ignored.

Landmarks are identified in every two consecutive views and these two views,  $V_n$  (current view) and  $V_{n-1}$  (previous view), thus form part of the input for computing the MFIS.  $V_{n-1}$  is the in-between representation that provides a linkage between what is in view with what is in the memory. Based upon the assumption we made earlier that there are overlapping surfaces between two consecutive views, some of the surfaces in  $V_{n-1}$  are found in  $V_n$ . At the same time, some, if not all, of the surfaces in  $V_{n-1}$  have already been transferred into the MFIS and therefore surfaces in  $V_{n-1}$  are also found in the MFIS. Furthermore, these surfaces in  $V_{n-1}$  are assigned the same ID as their corresponding surfaces in the MFIS. Consequently, if surfaces in  $V_{n-1}$  are identified as landmarks with those in  $V_n$  after transforming  $V_{n-1}$  onto  $V_n$ , then one will also know how  $V_n$  is linked to the MFIS. Fig. 3.9 shows the initial set up of the process prior to exploring the environment. Note that  $V_n$ ,  $V_{n-1}$ , and the MFIS are the same except surfaces in the latter two have unique IDs.



**Fig. 3.9:** Transferring of landmarks ID from the MFIS to  $V_{n-1}$

The algorithm for finding landmarks (step 3 of algorithm #3) after transforming surfaces in  $V_{n-1}$  onto  $V_n$  is described below:

**Algorithm #4: Finding landmarks in view:**

For each surface,  $x$ , in  $V_n$  do:

For each surface,  $y$ , in  $V_n$  but transformed from  $V_{n-1}$  do:

1. If there is strong evidence that  $y$  is close to  $x$ , then  $y$  is a candidate match for  $x$
2. If there is weak evidence that  $y$  is close to  $x$ , then
  - If  $y$  has a similar orientation as  $x$ , then  $y$  is a candidate match for  $x$ .
3. If neither (1) nor (2) is true then  $y$  is not a candidate match for  $x$ .

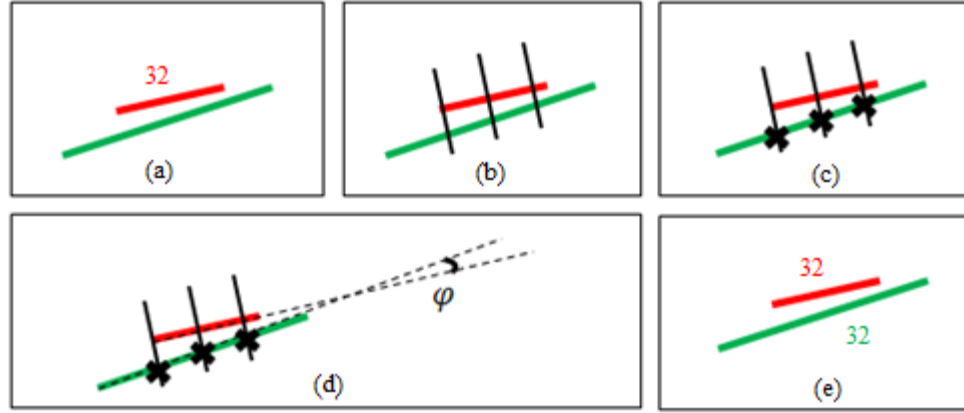
For each surface,  $x$ , in  $V_n$  that has matching candidates do:

Select the best candidate for  $x$  (i.e.  $x$  is thus recognised as the matching surface) and assign its ID to  $x$

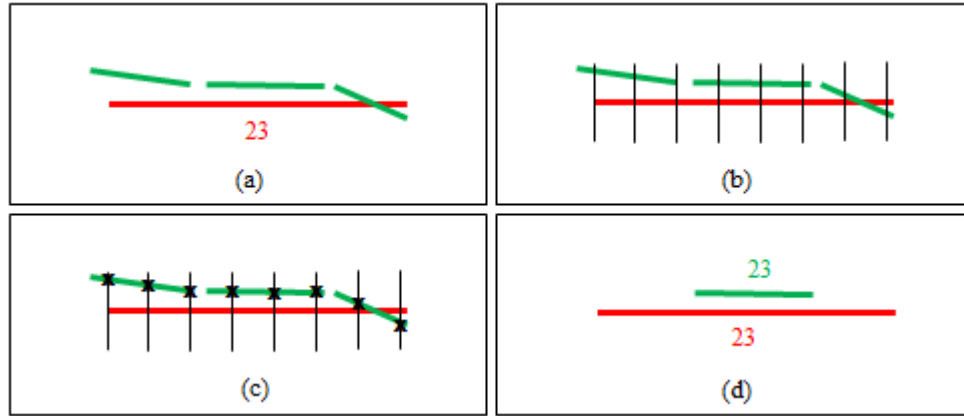
Note that after identifying a landmark in  $V_n$ , there are now two copies of a landmark surface – one in  $V_n$  and one from  $V_{n-1}$  (also in  $V_n$ ). We will refer to the landmarks identified as  $L^1, L^2 \dots L^n$ , to refer to both surfaces collectively. To identify its unique copy, we will use the notations:  $L^1_{vn}$  and  $L^1_{vn-1}$ . The former refers to the landmark surface originally found in  $V_n$  and the latter refers to the one transformed from  $V_{n-1}$ .

Evidence of closeness (re: step 1 of algorithm #4) is obtained by generating imaginary lines of 1-meter length (on both sides of the line), drawn perpendicular along each  $x$  beginning at one of its end points and at regular  $\frac{1}{2}$ -meter intervals (see Fig. 3.10). If any of these lines intersect  $y$ , then there is weak evidence that  $y$  is close to  $x$ . If more than 5 such lines are found intersecting  $y$ , then there is strong evidence that  $y$  is close to  $x$ . Two surfaces are considered to be of the same orientation if their orientation does not differ by more than 10 degrees. If the above algorithm produces more than one matching surfaces, then the surface that has the most similar orientation would be chosen as the matched surface.

Fig. 3.11 shows another example of how a surface is matched to another using the above algorithm. This example is quite commonly observed in an office environment. Whenever there is a significant change in the robot's position and/or orientation, the robot's view could change quite significantly from one view to another. In this example, the robot saw a single large surface in its previous view and then sees the same surfaces as three separate surfaces in the current view. The algorithm treats these surfaces as competing surfaces and selects the best possible option.



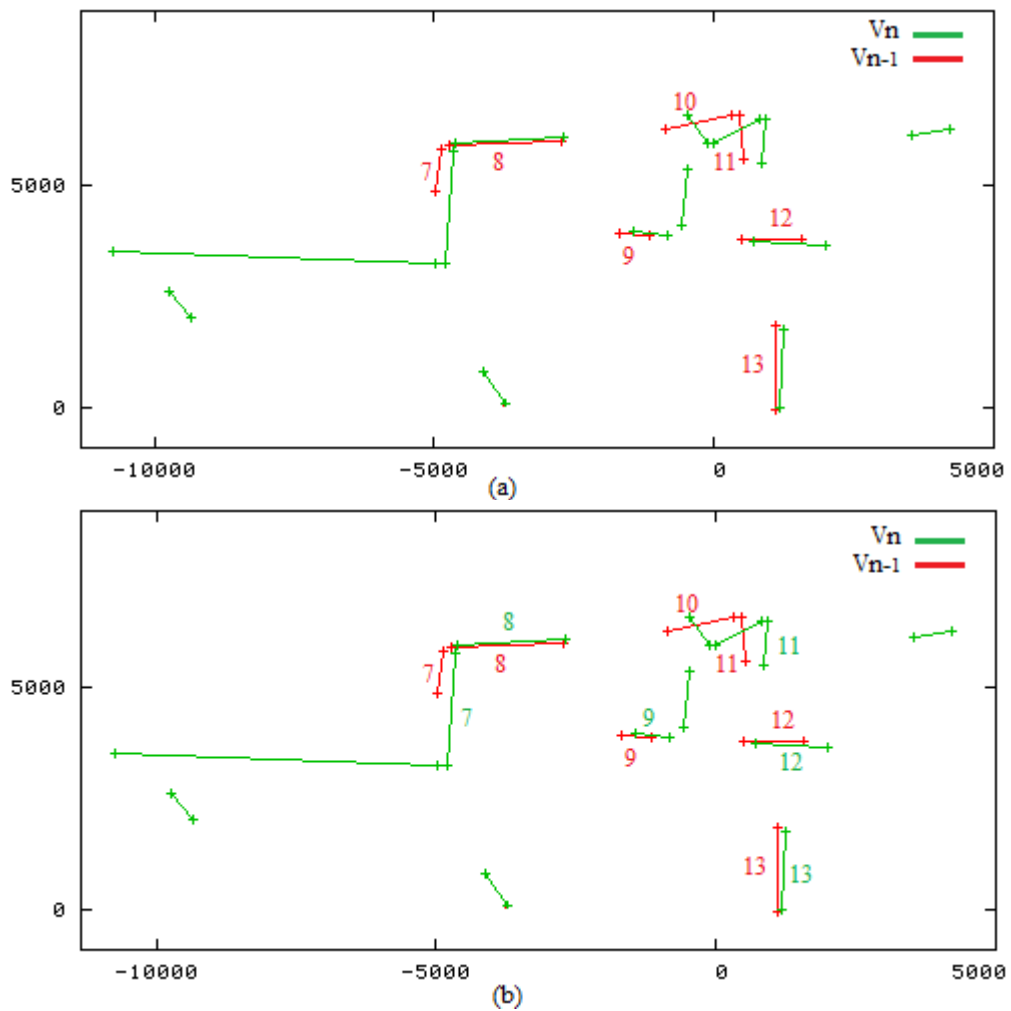
**Fig. 3.10:** (a) Candidate in green being paired with a smaller comparing surface, (b) perpendicular lines generated on the comparing surface, (c) markers **X** denote the intersections between perpendicular lines and candidate, (d) match is verified after angle between the lines is found satisfactory, (e) candidate inherit the ID



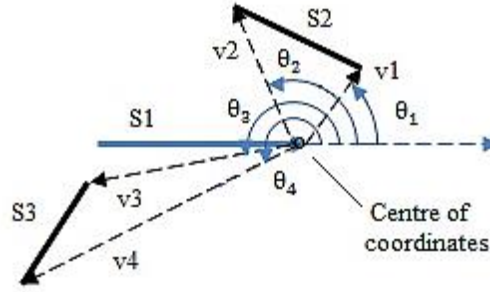
**Fig. 3.11:** Finding landmarks whereby the surface perceived in  $V_{n-1}$  (single line in red) has changed its shape drastically in the current view (split into 3 green lines).

Fig. 3.12 shows the identification of landmarks by the robot in two successive views. Once landmarks in the current view are identified, the remaining surfaces are either new or unrecognised. The last step in MFIS computation is to add those new surfaces to the MFIS so that the robot gets a reasonably accurate map. To do so, we compute their spatial locations relative to one of the landmarks identified, say,  $L^x_{vn}$ . Note that one could choose the closest landmark or the best-matched landmarks. Given the latter could be quite far away from the unknown surfaces, we have decided to use the closest

landmark. Locations of nearby surfaces can be coded as a vector whose length is the distance from the chosen end-point and whose angle is its angular displacement from the surface slope (Fig. 3.13). Such coding has been suggested for cognitive species. An example is O’Keefe’s (1991) “slope-centroid” model. By using the same landmark found in the MFIS (i.e. using  $L^x_{v_{n-1}}$ ), we can enter these surfaces into the MFIS in their positions relative to the same landmark.



**Fig. 3.12:** Finding landmarks: (a)  $V_n$  (surfaces in green) combined with  $V_{n-1}$  (surfaces in red and are labelled), and (b) Landmarks identified (IDs 7, 8, 9, 11, 12, and 13) are assigned the same ID



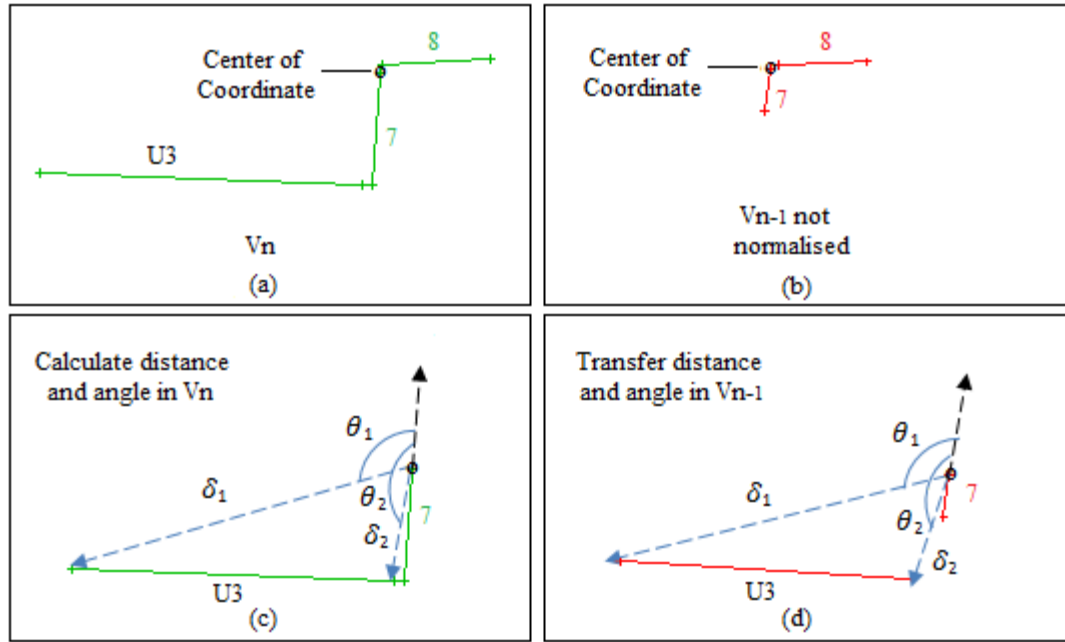
**Fig. 3.13:** Computing the spatial locations of surfaces close to a landmark: S1 is recognised as a landmark and S2 and S3 are coded using two pairs of vectors centred on the right end-point of S1

Finding the landmarks and using them to describe the spatial layout of nearby surfaces meant that one has transformed an egocentric representation into an object-centred representation. With two different surfaces for the same landmark,  $L^x_{vn}$  and  $L^x_{vn-1}$ , one in each view, we have now established a common frame of reference to describe the spatial layout of surfaces in view surrounding the landmark in two different “worlds”. Note that it is not important that  $L^x_{vn}$  and  $L^x_{vn-1}$  are not located exactly in  $V_n$  since  $L^x_{vn-1}$  is just a projection of that surface into  $V_n$ . It is the spatial layout of surfaces surrounding the landmark that matters. However, what is important is that centre of the reference frame established needs to be reasonably close. Otherwise this could cause serious distortion to the map computed in the MFIS. Given the 2D nature of my surface description, this could happen if the two surfaces differ significantly in length.

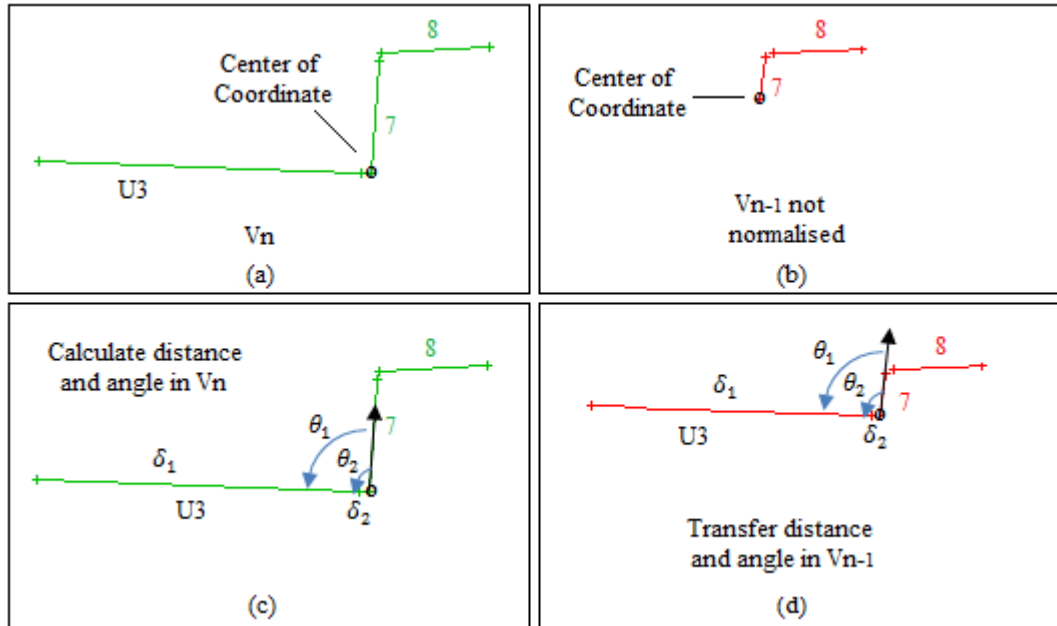
For example, consider ID 7 in Fig. 3.12. If we establish the centre of the reference frame at the end-point of ID7 nearest to ID8 (see Fig. 3.14), then U3 will be copied close to its relative position in the MFIS. If we establish the centre of the reference frame at the opposite end-point of ID7 (see Fig. 3.15), then U3 will be copied incorrectly in the MFIS. To overcome this problem, the length of  $L^x_{vn}$  and  $L^x_{vn-1}$  are “normalised” i.e. make them of equal length and choose the end-points that are closest together as the centre of the reference frame.

For normalisation, we again use the technique of generating perpendicular lines from the end-points of both surfaces. If they intersect with the other surfaces, then the end-point of the surface will be extended to match the end-point of the other surface (see Fig. 3.16). After normalisation, the two surfaces are identical in length. If during normalisation, the surface being extended is  $L^x_{vn-1}$ , then its corresponding surface in the MFIS needs to be extended too. Fig. 3.17 shows 3 examples of identifying and normalising of landmarks using the robot’s actual view of the environment.

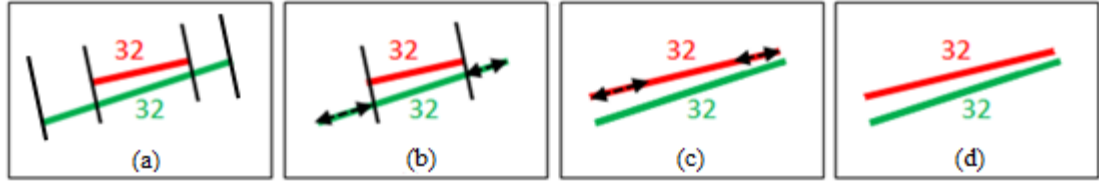




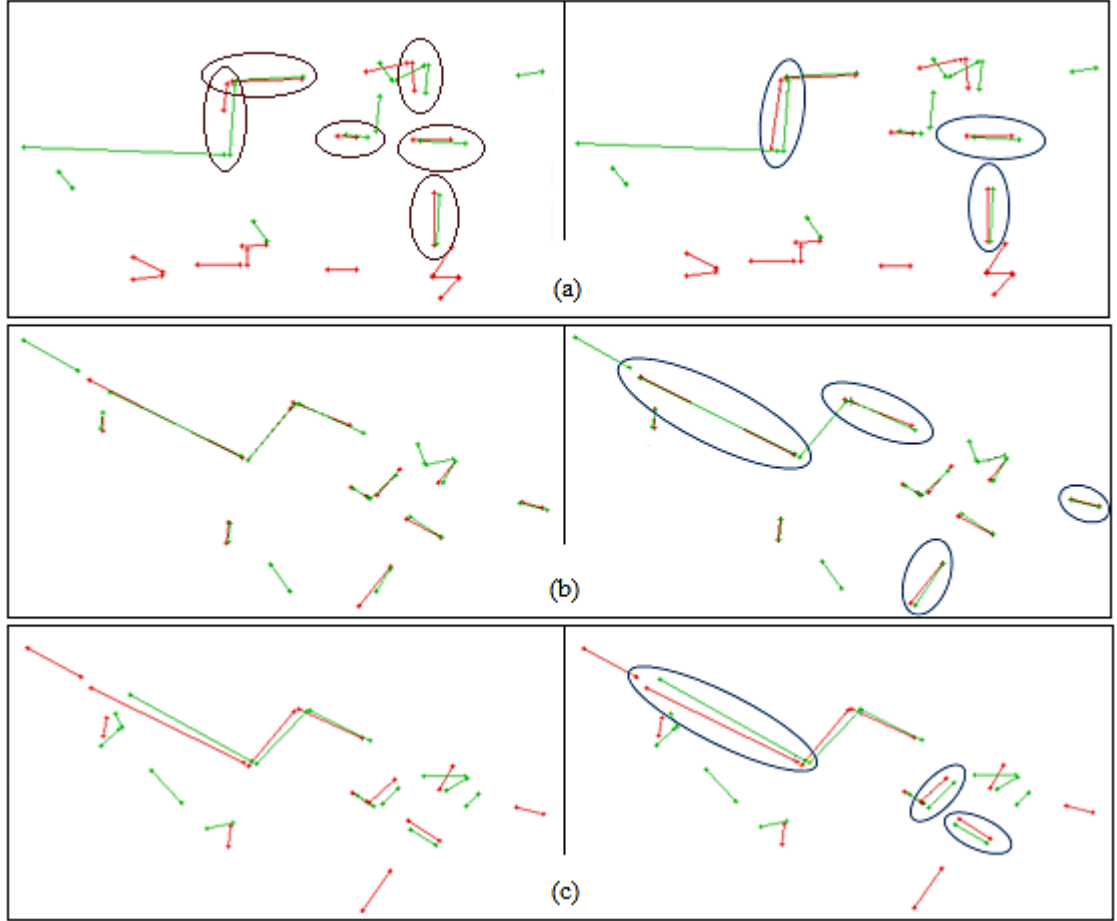
**Fig. 3.14:** (a) and (b) show which end-point of the landmark surface (ID 7) is chosen as its centre of coordinate system respectively in  $V_n$  and  $V_{n-1}$ , and (c) shows the calculation of the position of the unknown surface,  $U_3$ , and (d) shows where it is positioned. In this case, the position is roughly correct.



**Fig. 3.15:** (a) and (b) show which end-point of the landmark surface (ID 7) is chosen as its centre of coordinate system respectively in  $V_n$  and  $V_{n-1}$ , and (c) shows the calculation of the position of the unknown surface,  $U_3$ , and (d) shows where it is positioned. In this case, the position is seriously distorted.



**Fig. 3.16:** Normalisation of landmark surfaces – Perpendicular lines are generated at the end-points of both surfaces and the two surfaces are normalised to become of equal length.



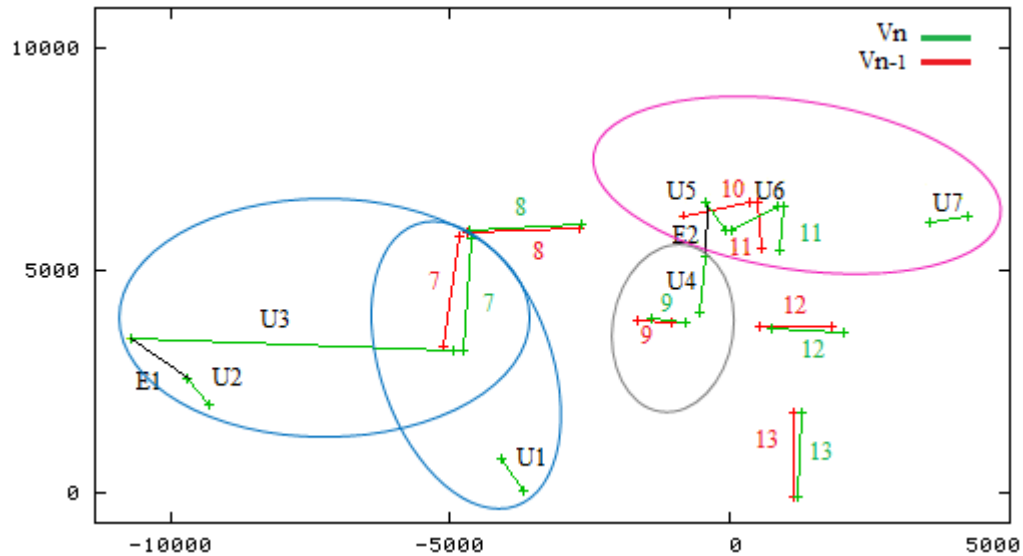
**Fig. 3.17:** Three examples of normalisation in a robot's view: – The left column shows the projection of  $V_{n-1}$  onto  $V_n$  and landmarks (circled surfaces) are identified, and the right column shows the landmarks that are normalised.

Once the landmarks are identified and normalised, we need to transfer the surfaces to the MFIS. The algorithm for this step is described below:

**Algorithm #5: Transfer surfaces to the MFIS:**

For each surface,  $S$ , not recognised as a landmark in  $V_n$ , do:

1. Find the nearest landmark for  $S$  in  $V_n$
2. Compute the vector and angular displacement of  $S$  from  $L_{vn}^x$
3. Create  $S'$  of similar length of  $S$
3. Using (2), compute the position of  $S'$  in the MFIS using the equivalent of  $L_{vn-1}^x$  in the MFIS and put  $S'$  in the MFIS
4. If  $S$  intersects with another surface in the MFIS, remove it and skip step 5.
5. Using algorithm #4 (for finding landmarks), find if there is a “landmark” surface for  $S'$  in the MFIS. If there is, normalise it with  $S'$ , assign its ID to  $S$  and delete  $S'$  from the MFIS. If there isn't, create a new ID and assign it to both  $S'$  and  $S$ .

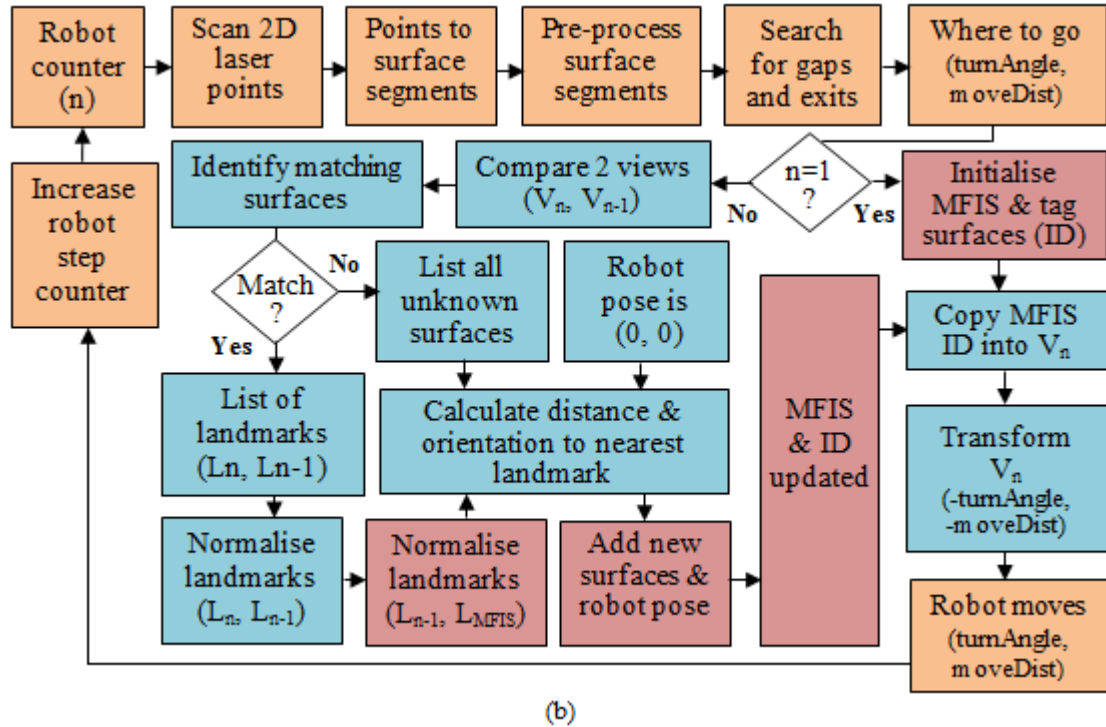
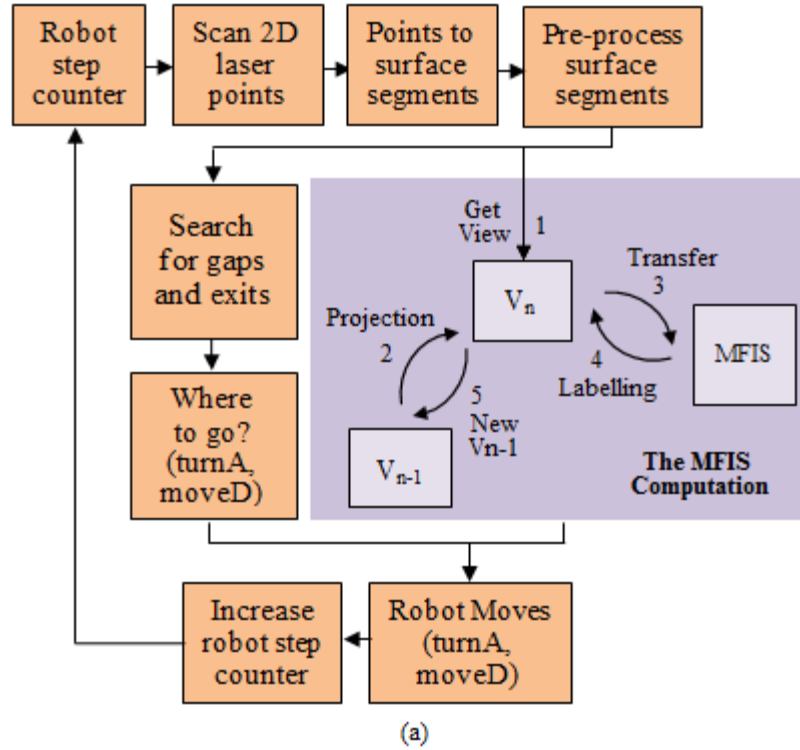


**Fig. 3.18:** Examples of choosing the nearest landmarks to localise unknown surfaces. New surfaces U1, U2 and U3 are closest to landmark ID7, new surface U4 to landmark ID9, and new surfaces U5, U6 and U7 to landmark ID11

When transferring a surface to the MFIS, one uses its position with respect to its nearest landmark. This means that different reference frames are used to localise different sets of unknown surfaces and if errors were introduced in the calculation, its effects could be contained. Fig. 3.18 shows an example of how different reference frames are used to describe unknown surfaces present in the view.

Not all surfaces transferred to the MFIS are new to the MFIS and therefore the last two steps in the algorithm check if the incoming surface is not already known in the MFIS. If the new surface intersects with another surface in the MFIS, it usually indicates an area with several cluttered surfaces. A slight change in perspective can often produce different descriptions of the surfaces in view. As such, it is best to ignore such new surfaces and not update the MFIS. If the new surface is positioned close to another surface in the MFIS, the two could be the same surface. We use the same test for identifying landmark to check if this is the case and if so, that surface is not new and has an ID already assigned. The corresponding surface in  $V_n$  is then marked with the same ID. However, these two surfaces may not be of the same shape and therefore they need to be normalised; surfaces having the same ID must be of the same shape.

In summary, we presented an algorithm for computing an inexact MFIS that utilises two consecutive views to track where new surfaces are appearing and help position them in their relative positions in the MFIS (see Fig. 3.19(a)). Fig. 3.19(b) is a flowchart showing the connection between all algorithms presented in this chapter.



**Fig. 3.19:** (a) Processes shaded in orange summarises the flow where the robot pre-process the laser scans, make decision on the next target before planning its movement whereas the box shaded in purple denotes where spatial mapping takes place. The flowchart diagram in (b) breaks down the spatial mapping processes in (a). Blue boxes are processed in the working memory whereas red boxes indicate updating in the MFIS

### 3.5 ASR Computation

Many robotics researchers compute a topological network out of their global map. The former is a more efficient representation for spatial planning. Each node in the network represents a well-defined empty space, captured using a fixed polygonal structure. The network is often computed offline after the global map is learned and loops are closed. For my robot, I also need to compute a network of “places” visited so that it could use its network to plan paths and find short cuts. However, the MFIS is an inexact and incomplete representation and therefore, computing a network of structured empty spaces might not be useful. The network could be unstable as the inexact nature of the representation meant it could change shape drastically during later explorations of the environment. Furthermore, from a cognitive standpoint, one is not interested in how empty spaces are partitioned but rather the affordance of each local space remembered (Gibson, 1966). Yeap (1988) argued that each local space remembered (or ASR) affords boundedness i.e. it is a bounded region whereby one could move into and out of. Hence, two pieces of information are important when computing an ASR, namely its exits and its boundary. Interestingly, its boundary could be imprecise and incomplete and if so, this indicates the part of the ASR that could be explored further in subsequent visits. In this section, we developed the algorithms needed for the robot to compute a network of ASRs from its MFIS.

As defined above, an ASR is a bounded space with identified exits that one could move in and out of it. Recall that when developing an autonomous exploration algorithm for the robot in section 3.3 above, exits are recognised in the environment as perceived gaps

of a certain size (defined here as between 0.6 to 1.2 meters). Hence, when the robot crosses an exit, it “knows” that it is leaving an ASR and entering a new one. At this moment, it computes an ASR just left and assigned it to the same number as the exit identified. The algorithm for computing an ASR is as shown below:

**Algorithm #6: Computing ASR**

1. Copy what is in the MFIS to ASR
2. Assign an ID to this ASR
3. Identify surfaces in the ASR that do not belong to this ASR:
  - a. Eliminate all surfaces that have already been assigned to an ASR
  - b. Eliminate new surfaces that come into view after crossing the exit
4. Remove all surfaces in the ASR that will not be considered as part of the boundary for this ASR
5. Compute the boundary for this ASR
6. Add the exit used to this ASR and returns it as a new ASR

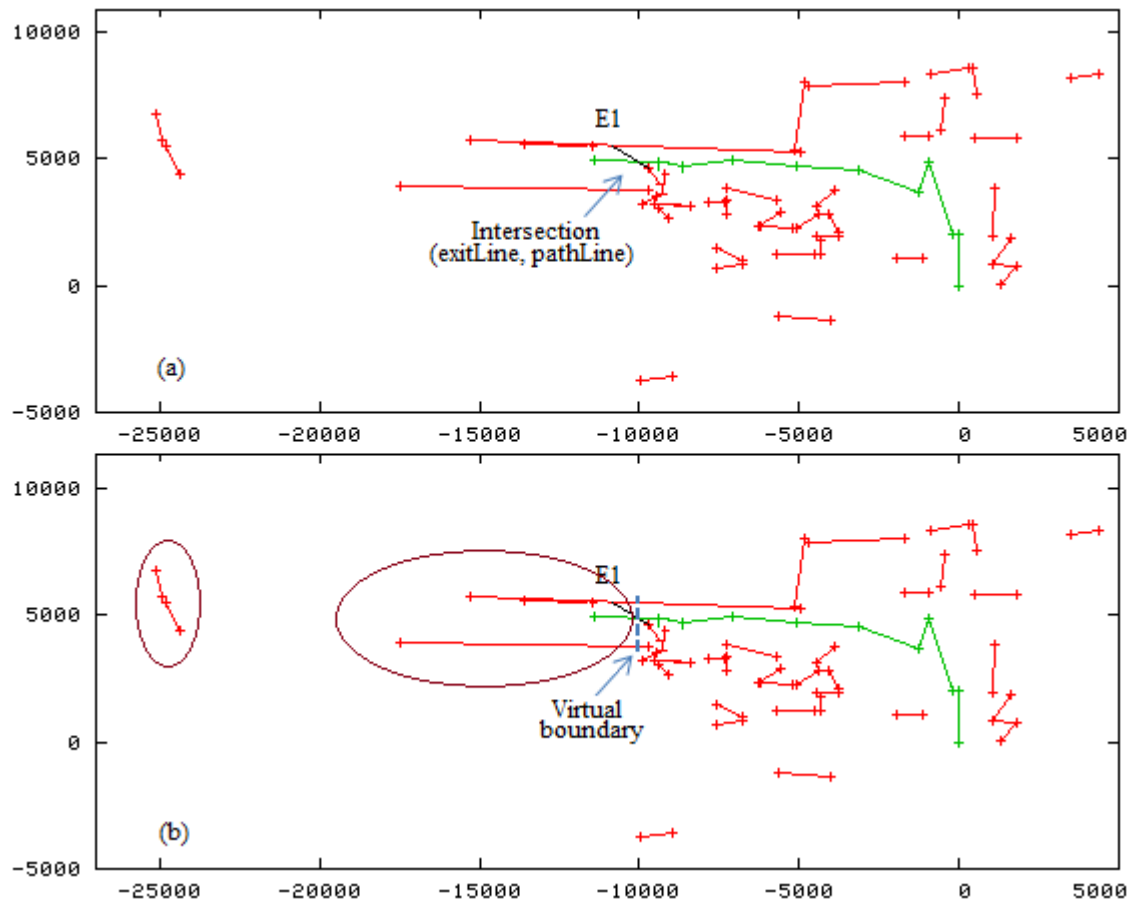
The representation of an ASR is thus similar to that of an MFIS i.e. it consists of a list of surfaces descriptors and each descriptor consists of its own unique name and spatial co-ordinates. These surfaces are the boundary surfaces for the ASR. In addition it has an exit descriptor to describe where an exit can be found. Step 3 of the algorithm is straightforward except for those new surfaces that come into view right after the robot crosses the exit (step 3b). These surfaces belong to the next ASR and should be eliminated when considering surfaces for computing the boundary of the ASR just passed. To do so, I introduce a virtual exit line that is perpendicular to the robot’s path at the point where the path intersects the exit crossed (see Fig. 3.20). This virtual boundary is used as a reference line for separating the MFIS surfaces seen prior to

crossing the exit and those surfaces seen after crossing the exit (see Fig. 3.21). The algorithm for eliminating such surfaces is described below:

**Algorithm #7: Eliminate new surfaces coming into view after crossing an exit**

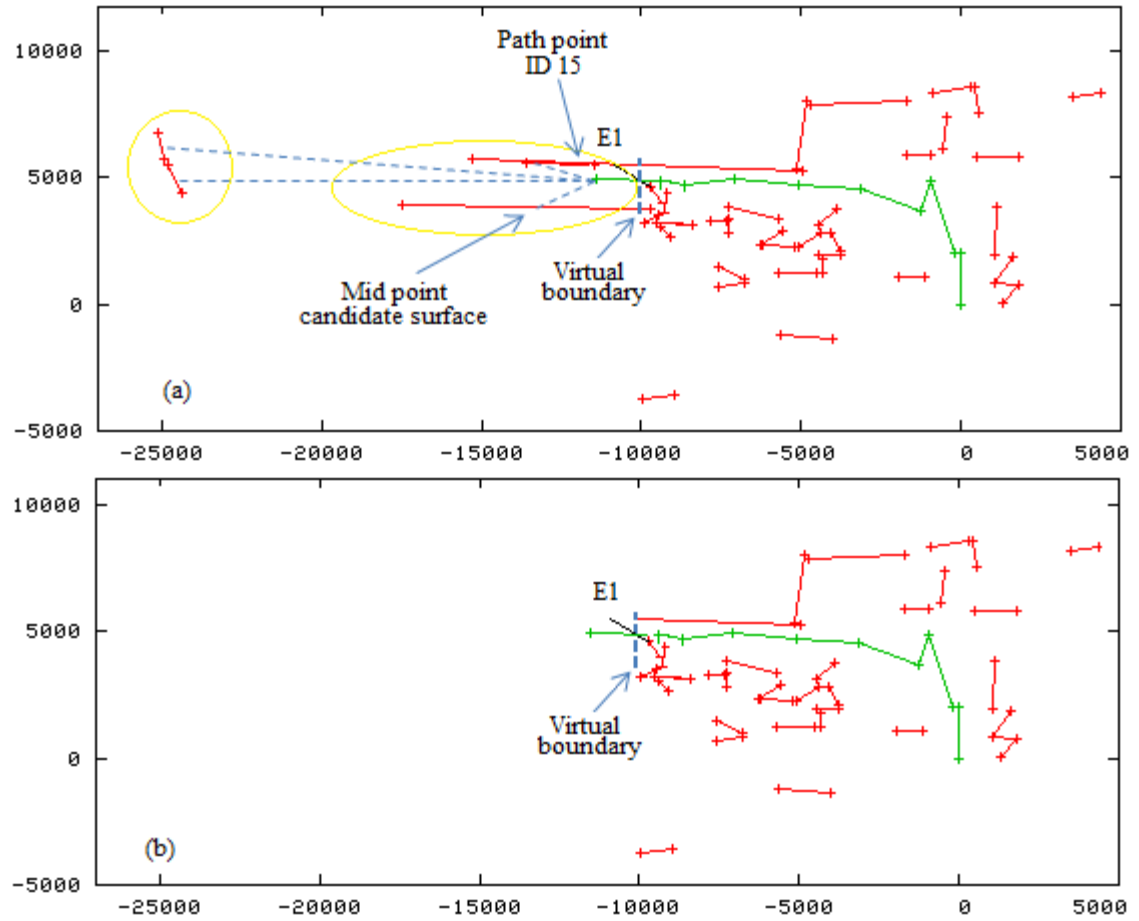
Input: MFIS with all surfaces assigned to an ASR removed but with robot's path added

1. Find the point where the robot's path intersects the exit crossed
2. Create a virtual exit line perpendicular to the robot's path at the intersection point
3. From the robot's current position, draw a line to the mid-point of all remaining surfaces. If the line does not intersect (a) any other surfaces, or (b) the virtual exit line, or the robot's path, then eliminate this surface.

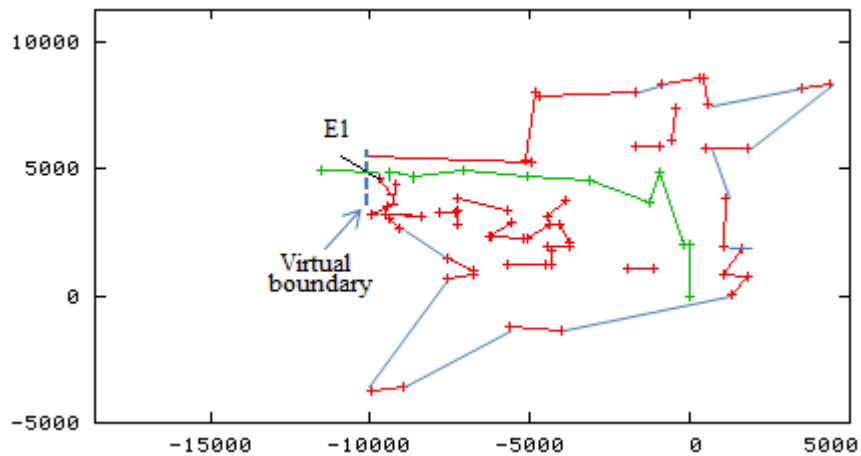


**Fig. 3.20:** (a) The green line denotes the robot's path, E1 denotes the exit crossed and the arrow points at the exit-path line intersection, (b) the virtual boundary created perpendicular to the path line slope at the exit-path line intersection where surfaces circled are considered beyond the exit thus are not included in the ASR being computed





**Fig. 3.21:** (a) Surfaces shown connected to the robot's current position (path point ID 15) via a dashed line can be eliminated according to algorithm #7. These are surfaces perceived after crossing the exit of the previous ASR. (b) The remaining surfaces for ASR computation



**Fig. 3.22:** Computing the boundary using the outermost surfaces perceived.

Intuitively, to compute a boundary for the ASR, one could follow the “outermost” surfaces starting from one end-point of the crossed exit to its other end-point, merging the gaps in between these surfaces so that one forms a boundary enclosing the robot’s path (see Fig. 3.22). However in doing so, one ignores much of the details that are available and one often ends up with a space larger than is required. The latter is because one could often have a glimpse of surfaces at the edge that are not part of the current ASR. To overcome this problem, I need to have more options on how to connect from one surface to another as part of the boundary.

In designing my algorithm, I utilise two heuristics to decide what options one has in boundary formation. In other words, I first select all surfaces that could be part of the boundary and then use the above algorithm that starts from one end-point of the crossed exit to its other end-point, merging the gaps between two *nearest* surfaces so that one forms a boundary enclosing the robot’s path.

These two heuristics concern the use of large surfaces and small surfaces in boundary computation. The first heuristics is that large surfaces are often a part of the boundary. This is because they provide significant barriers to movements and perception of what is on the other side. Thus, these surfaces are always selected as possible candidates for boundary formation. The second heuristics is that while small surfaces are often not a part of the boundary, a close group of them could become a boundary. Such groups of lines are commonly observed in this environment and in places where several small objects were placed alongside a perimeter wall. However, grouping them into a “large

surface” is difficult due to their different orientations and using a heuristics on what is “large” could inevitably rule out groups of some smaller surfaces.

It is observed that corners are detected in early vision and they play an important role in our ability to recognise objects that are partially occluded (e.g. Shevelev, Kamenkovich & Sharaev, 2003). Consequently, for small surfaces, I made explicit corner points as options for boundary formation. Corner points are defined as points whereby two small surfaces intersect or close enough that they are considered as intersecting. Note that this definition allows two collinear lines to create a corner point. While it is not, strictly speaking, a corner, such points could be useful for boundary formation too and especially if there are several such lines close together. My algorithm for identifying surfaces and corner points in the MFIS for boundary computation can now be described as follows:

**Algorithm #8: Identify surfaces and corner points in the MFIS for boundary computation**

1. For each small surface (i.e.  $< 2$  meters) do:
  - 1a. If it intersects with another small surface, create a corner point where the two surfaces intersect.
  - 1b. If it finds another small surface less than 0.5 meter away, create a corner point where the two surfaces would have intersected.
2. Remove all small surfaces
3. For each large surface (i.e.  $> 2$  meters) do:

If it intersects with another large surface then

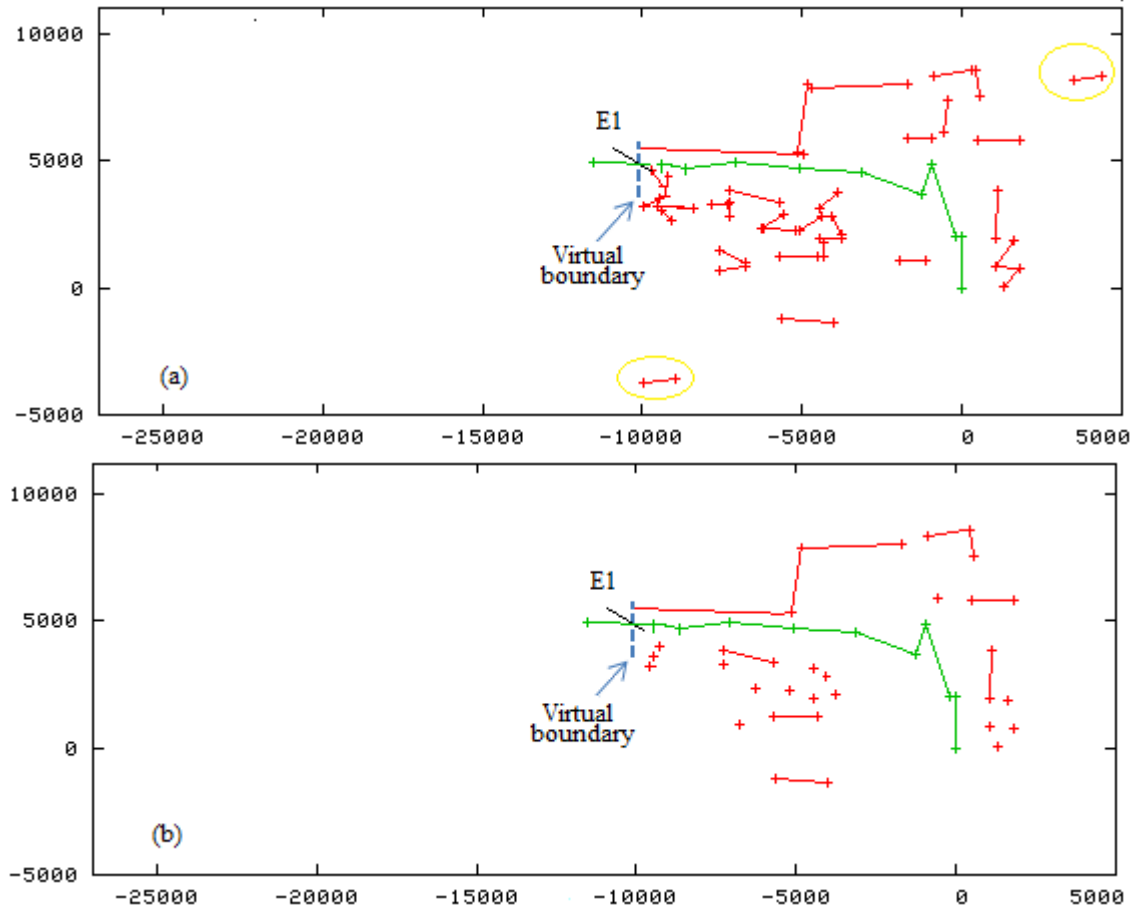
For both surfaces do:

  - 3a. Find length of intersection point to the start point

3b. Find length of intersection point to the end point

3c. If  $(3a) < (3b)$  shorten surface start point to intersection point else shorten surface end point to intersection point.

Figure 3.23(b) shows the result of applying Algorithm #8.

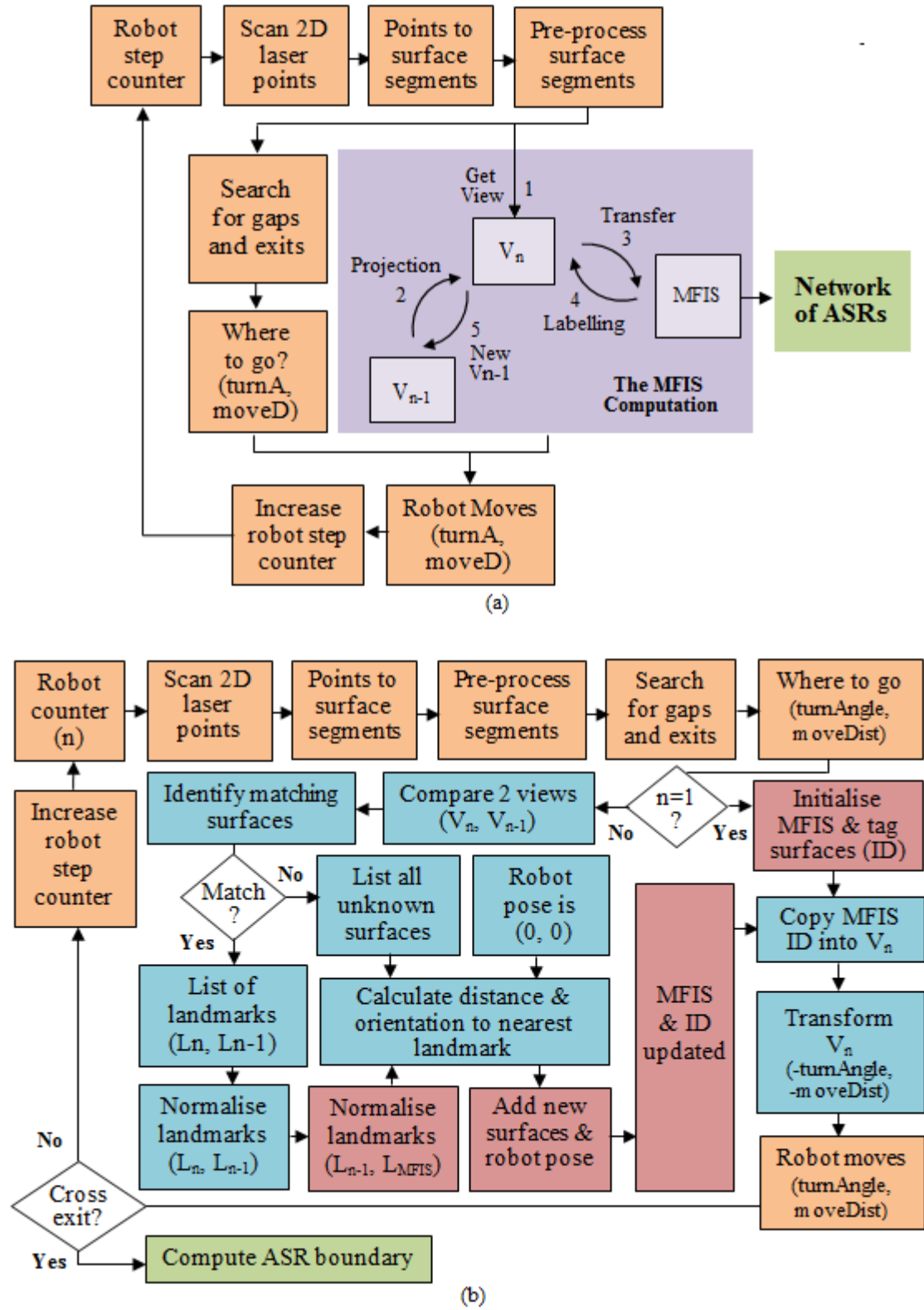


**Fig. 3.23:** (a) The initial spatial information for the first local space boundary computation where two surfaces (in yellow circles) fails the corner point test. (b) The large segments are filtered from the MFIS 15 surface set whereas the singular points denote the corner points derived from the smaller surfaces

To compute the boundary, and as mentioned earlier, I start with one end of the exit and connect it to its nearest surface or corner point and from that to its nearest surface or corner point and so forth. The connection is not allowed to cross any of these: (a) the robot's path (b) the large MFIS surfaces, (c) the exit line and (d) the ASR boundary computed so far. Note that one could form the boundary either in a clockwise manner or in an anticlockwise manner. It was found that at times one gets stuck in forming the boundary i.e. reach a point that one of the connection rules could not be true. It was discovered that the problem can easily be solved by then doing the boundary in the opposite direction. Consequently, and for simplicity, I apply the algorithm twice, once in a clockwise and the other in an anticlockwise direction, and pick the boundary that is successfully completed.

### **3.6 Conclusion**

In this chapter, I presented eight new algorithms to create a robot mapping system that perform mapping without a precise map. Fig. 3.24 shows the complete robot mapping system developed. In the next chapter, I present the results of some experiments performed using my robot.



**Fig. 3.24:** (a) The complete robot mapping system developed in this thesis. (b) Where ASR boundary is computed in the robot system

---

## Results & Analysis

---

### 4.1 Overview

In this chapter I present three experiments on robot mapping using the algorithms developed in the previous chapter. The goal is to demonstrate that the algorithm is robust, at least for mapping in a reasonably large office environment (30m x 30m) and the map produced is of a reasonable shape that the robot could use to find its way around the environment. In each experiment, the robot performs an exploration in the indoor environment (as illustrated in Fig. 1.2) where it computes an MFIS and a network of ASRs in real-time. Apart from demonstrating how the robot builds these representations, the robot also shows how it uses its map to perform spatial tasks like finding its way home and finding its way to different ASRs. In one of the latter experiments, some parts of the environment are blocked so the robot could not use a known route to travel to its target. The robot then re-plans and finds alternative routes.

Section 4.1 presents an experiment whereby the robot explores its environment twice in a clockwise manner before returning home. Section 4.2 presents an experiment whereby the robot explores its environment in an anti-clockwise manner and then finds its way home. It then explores its environment again in an anti-clockwise manner. Section 4.3 presents an experiment whereby the robot explores its environment in a clockwise manner, then in an anti-clockwise manner and then performs several spatial tasks. Throughout this chapter, I will use a diagrammatic approach to present the results to the reader.

## **4.2 Experiment 1 (Going clockwise)**

In this experiment the robot is allowed to wander from its home (0, 0) to explore its environment twice in a clockwise manner and then return to its home. The way the robot is forced to explore its environment in a clockwise manner is by closing the door that leads to the other direction. The goal of this experiment is to see how well the robot computes its MFIS and its network of ASRs and how well it closes loops when returning to familiar places.

### **4.2.1 Computing the MFIS and ASRs**

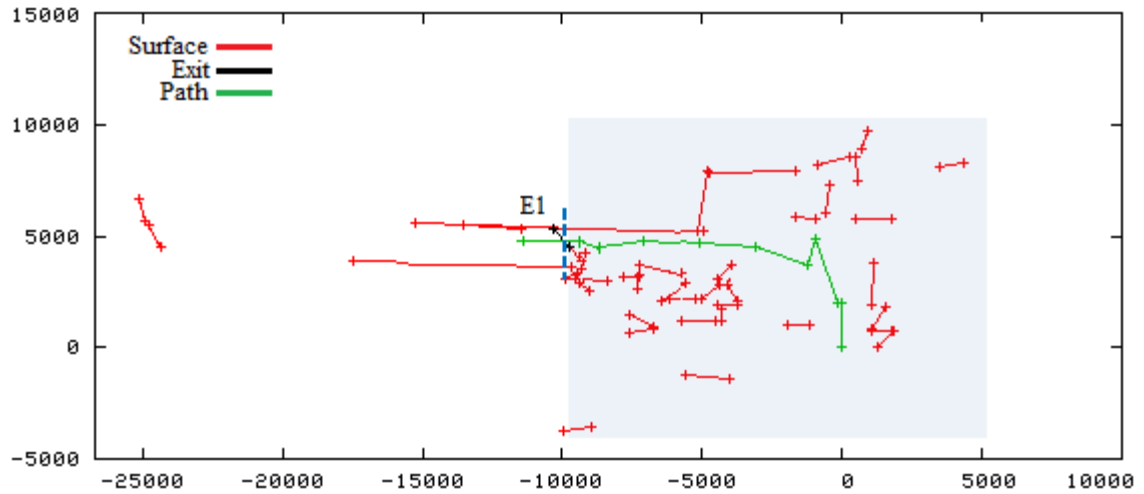
The following figures denote how the MFIS and the ASRs are built. Each time the robot crosses an exit in the environment, the local space it leaves behind is quickly computed as a new ASR. The algorithm to build individual ASRs is as described in Algorithm #6 (section 3.5). In general, what is required to compute the ASR boundary are the surfaces



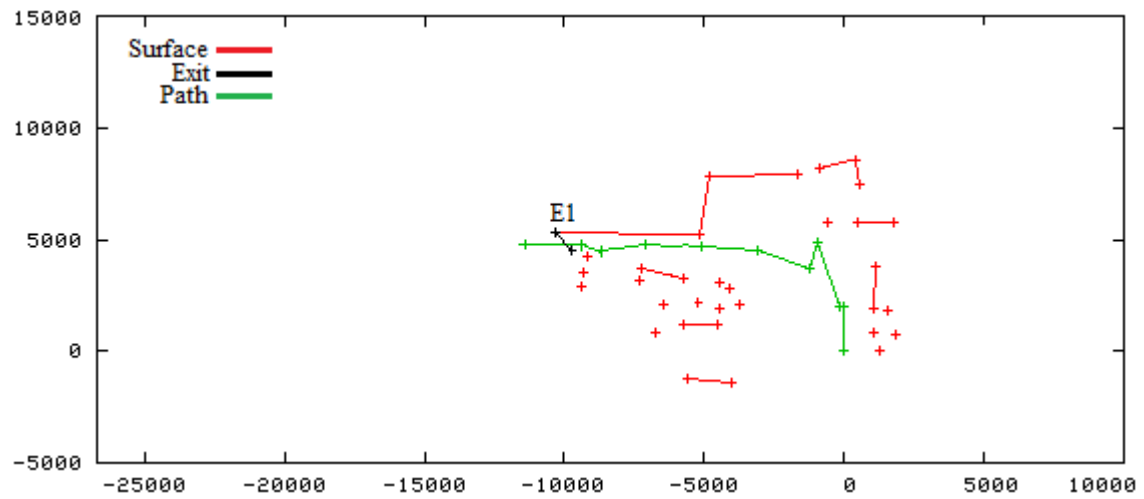
which belongs to the local space the robot just left, as well as the path line computed from the beginning of its journey. Fig. 4.1 to 4.3 shows how the first ASR (ASR1) is computed. Fig. 4.1 highlighted the surfaces and path lines extracted from the MFIS to compute ASR1. Fig. 4.2 shows the results of extracting corner points and large surfaces for the boundary and Fig. 4.3 shows the boundary computed for ASR1. ASR1 then initialises the network of ASRs for this environment.

The robot continues to explore the environment and crosses another exit, E2. Next, the process is repeated for the second ASR (ASR2). Fig. 4.4 shows the surfaces extracted from the MFIS for ASR computation. Fig. 4.5 shows the extraction of corner points and large surfaces. Fig. 4.6 shows the ASR2 computed. I will display the ASRs together using a single global co-ordinate system to reveal their overall shape rather than a topological network of nodes.

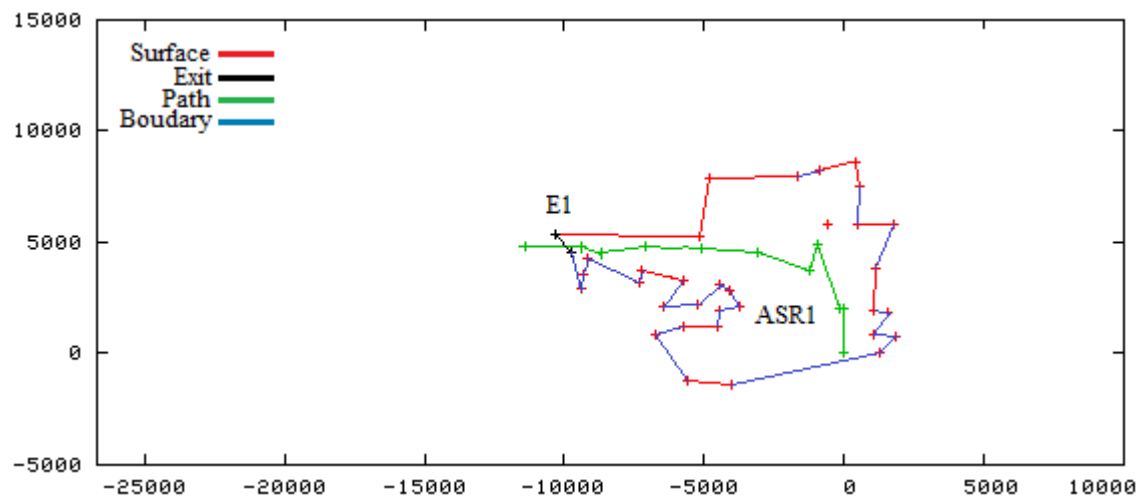
The series of figures after that denote the computation of ASR3 to ASR6 for the environment. All of the ASRs on the network (so far) have two exits on its boundary description. Each time a new ASR is built, they are represented immediately in the network so the robot can grow both the MFIS and the network as it experiences the environment.



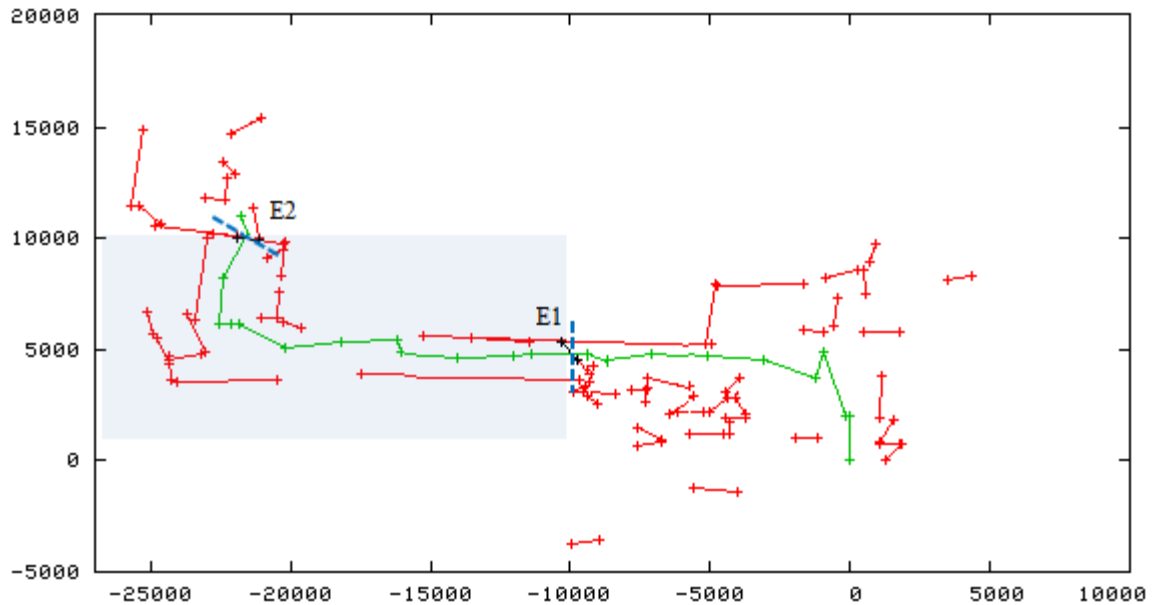
**Fig. 4.1:** ASR 1 surfaces (shaded) extracted from MFIS



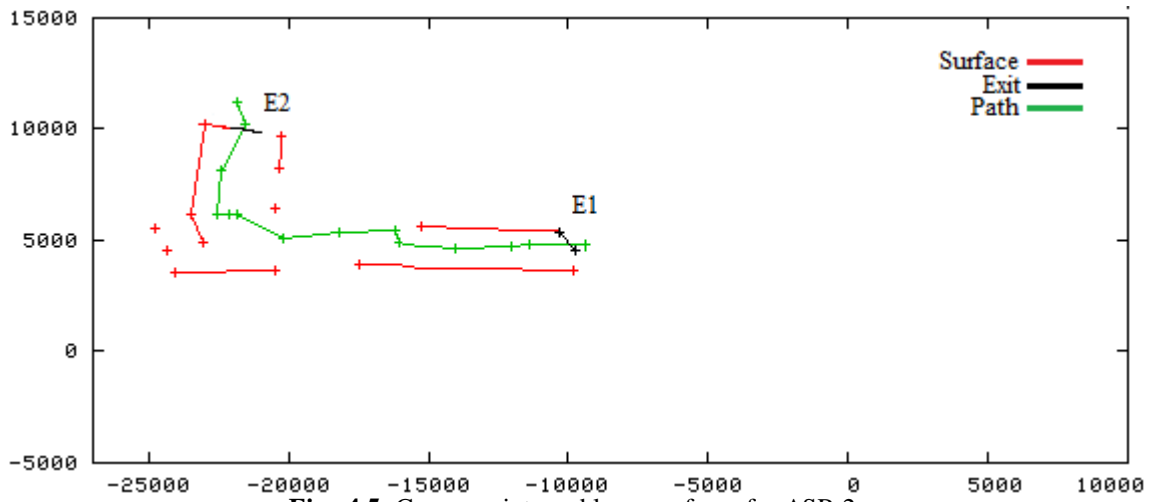
**Fig. 4.2:** Corner points and large surfaces for ASR 1



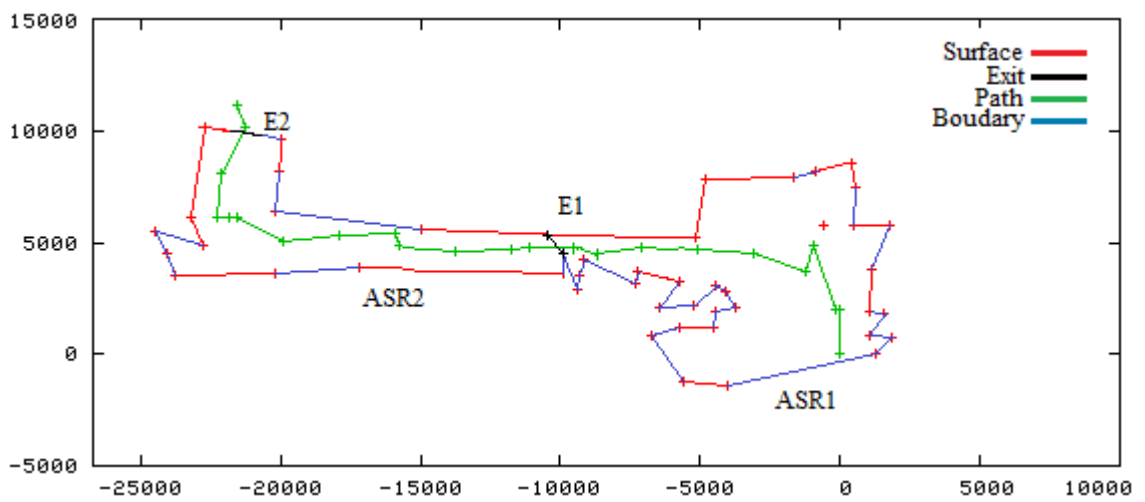
**Fig. 4.3:** ASR 1 boundaries computed. Red lines denote the large surfaces which have been extracted from the MFIS. The blue lines are boundaries computed as a result of joining adjacent surfaces and corner points together



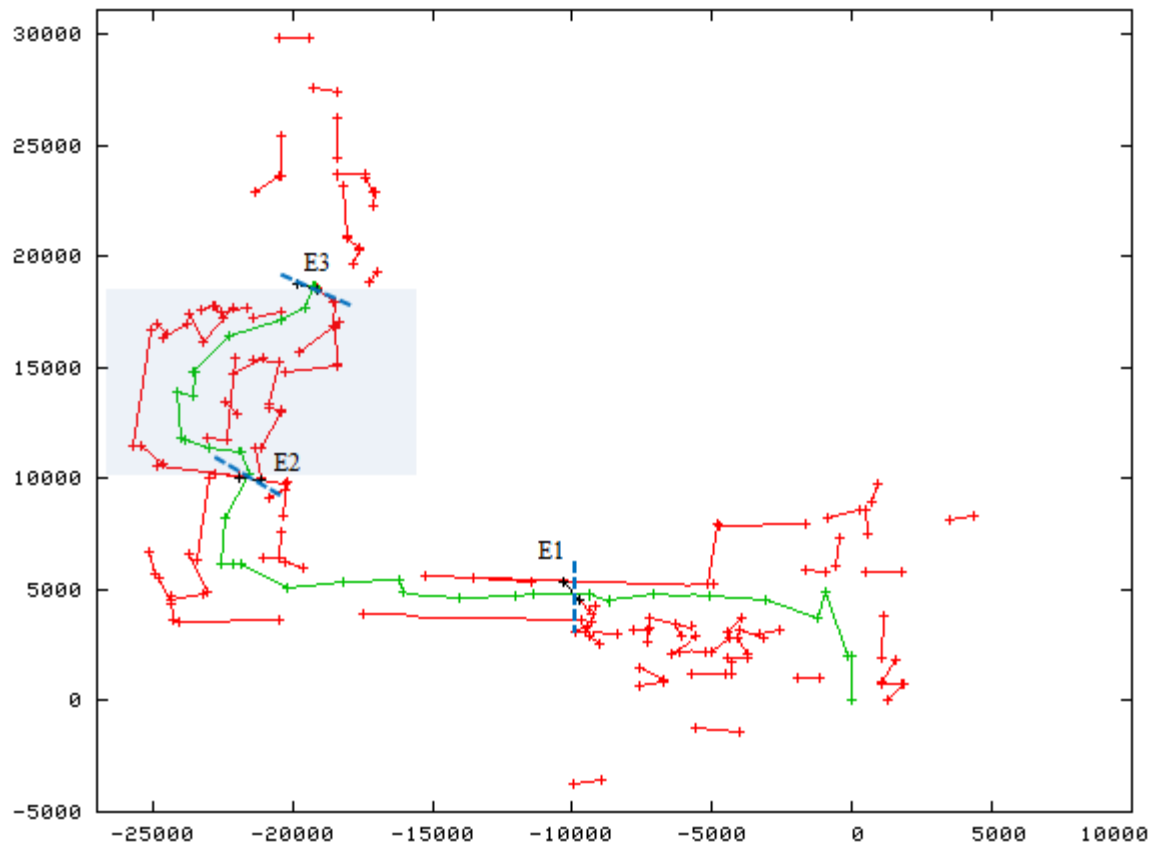
**Fig. 4.4:** ASR 2 surfaces (shaded) extracted from the MFIS



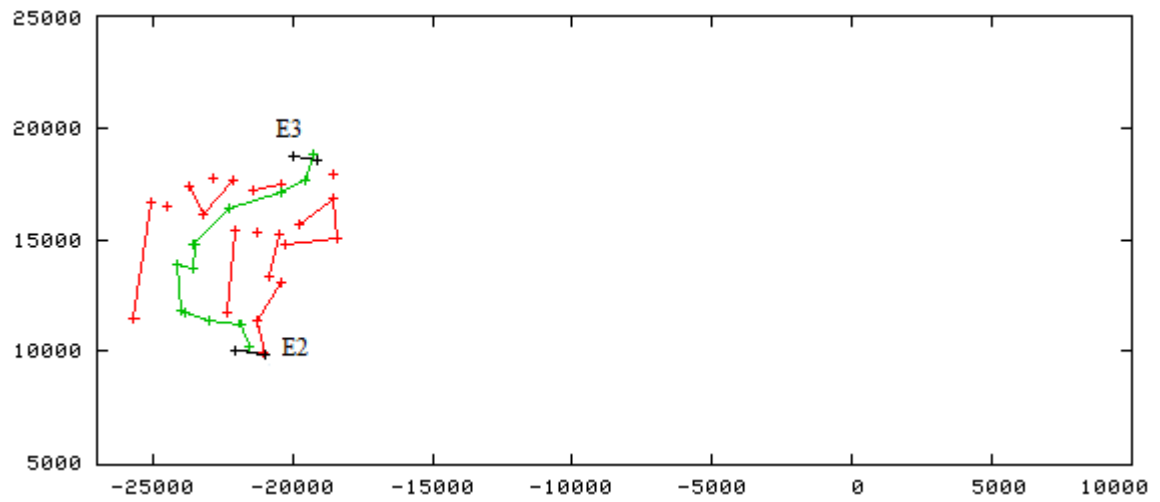
**Fig. 4.5:** Corner points and large surfaces for ASR 2



**Fig. 4.6:** Network of ASR showing ASR 1 and ASR 2



**Fig. 4.7:** ASR 3 surfaces extracted from the MFIS



**Fig. 4.8:** Corner points and large surfaces for ASR 3

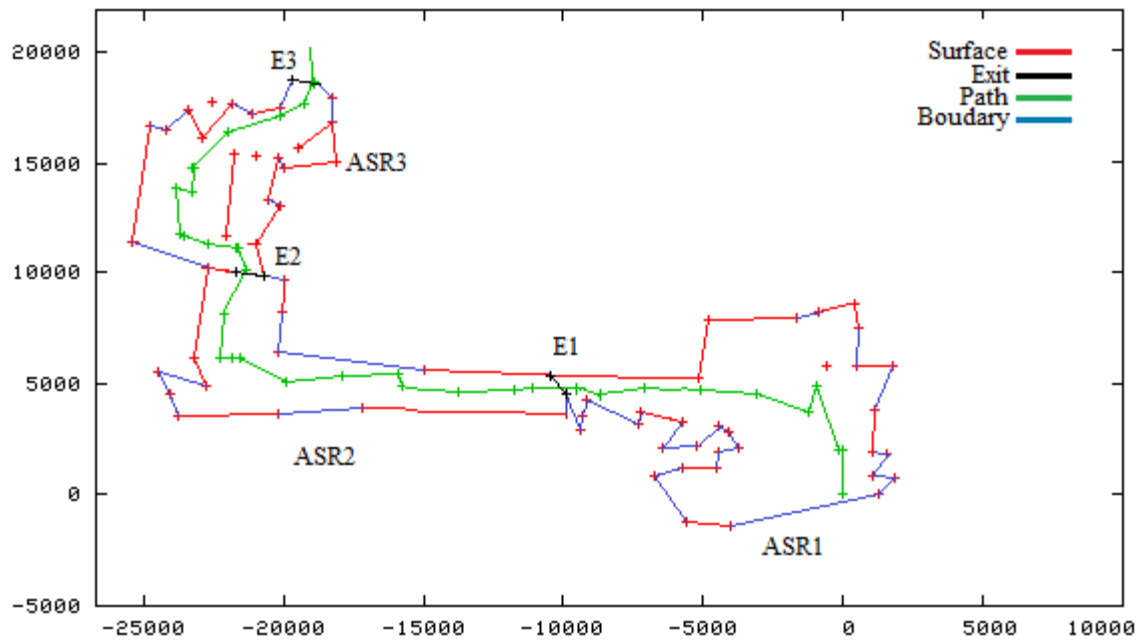


Fig. 4.9: Network of ASR showing ASR 1 to ASR 3

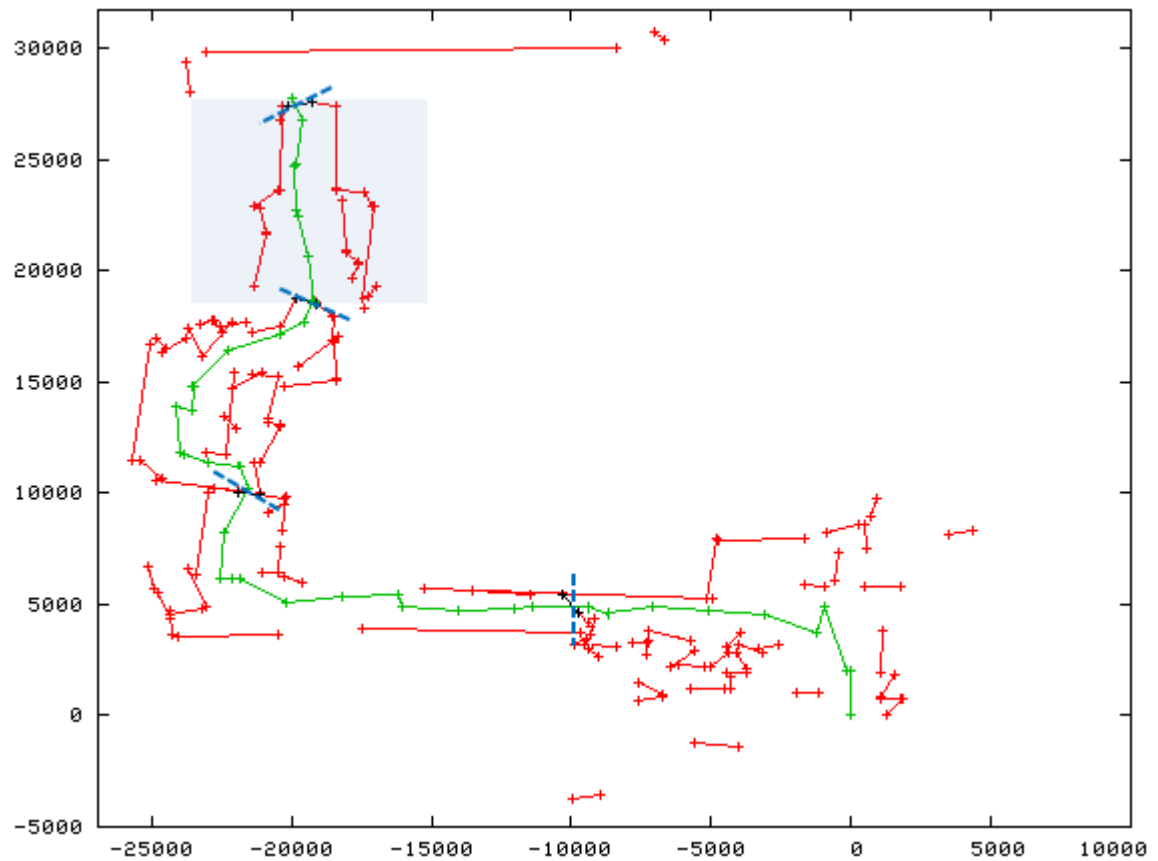
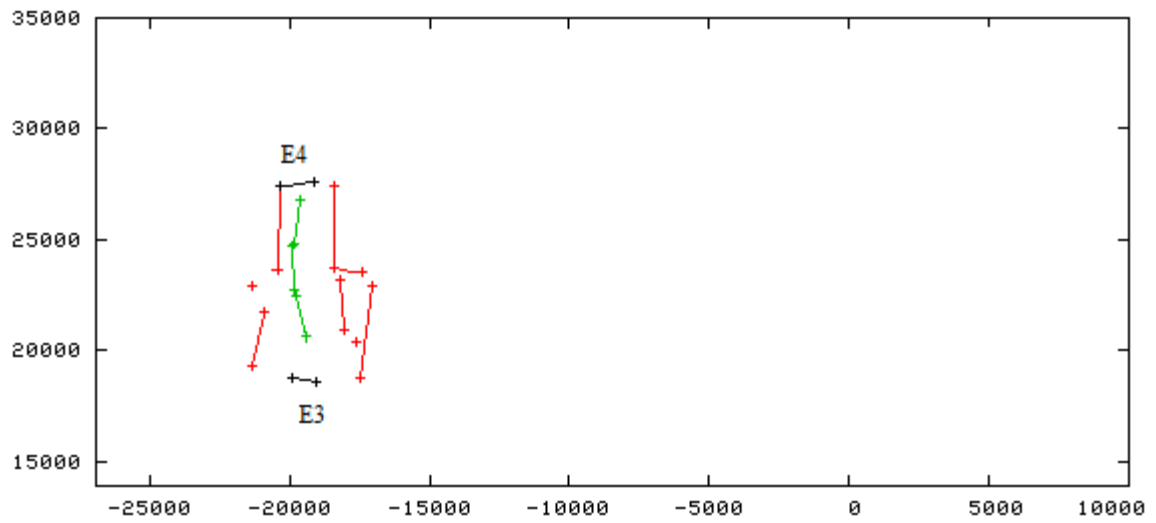
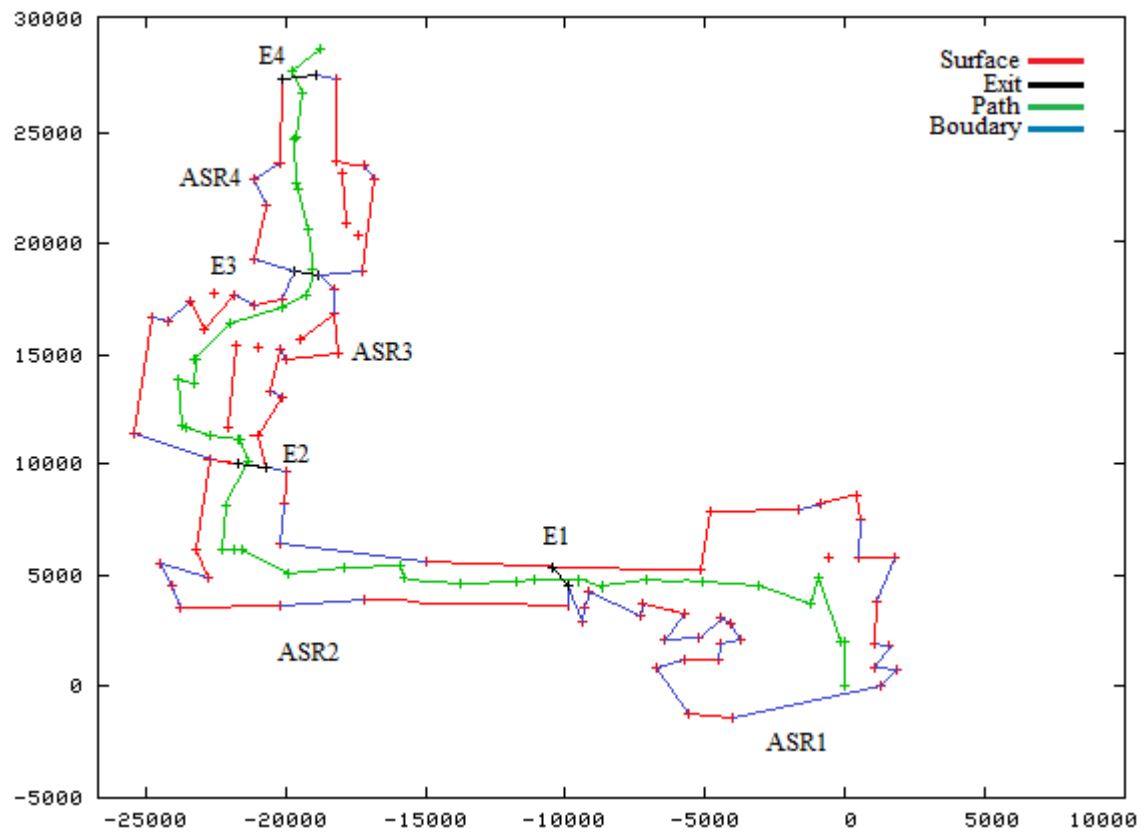


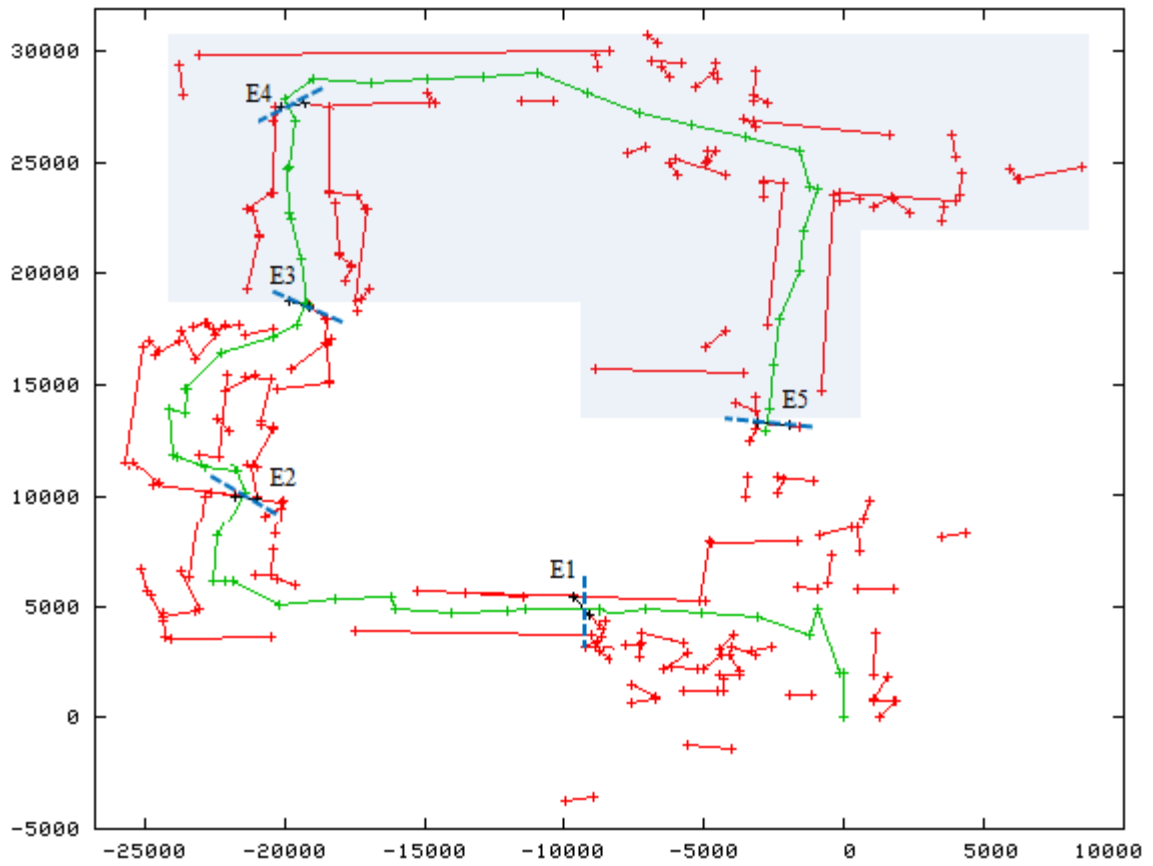
Fig. 4.10: ASR 4 surfaces extracted from the MFIS



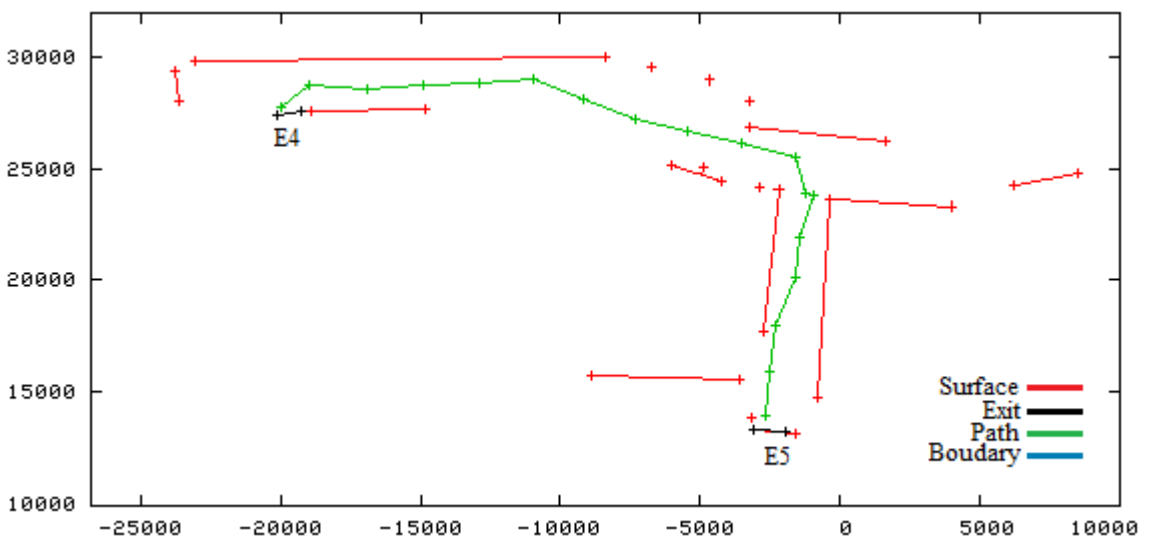
**Fig. 4.11:** Corner points and large surfaces for ASR 4



**Fig. 4.12:** Network of ASR showing ASR 1 to ASR 4



**Fig. 4.13:** ASR 5 surfaces extracted from the MFIS



**Fig. 4.14:** Corner points and large surfaces for ASR 5

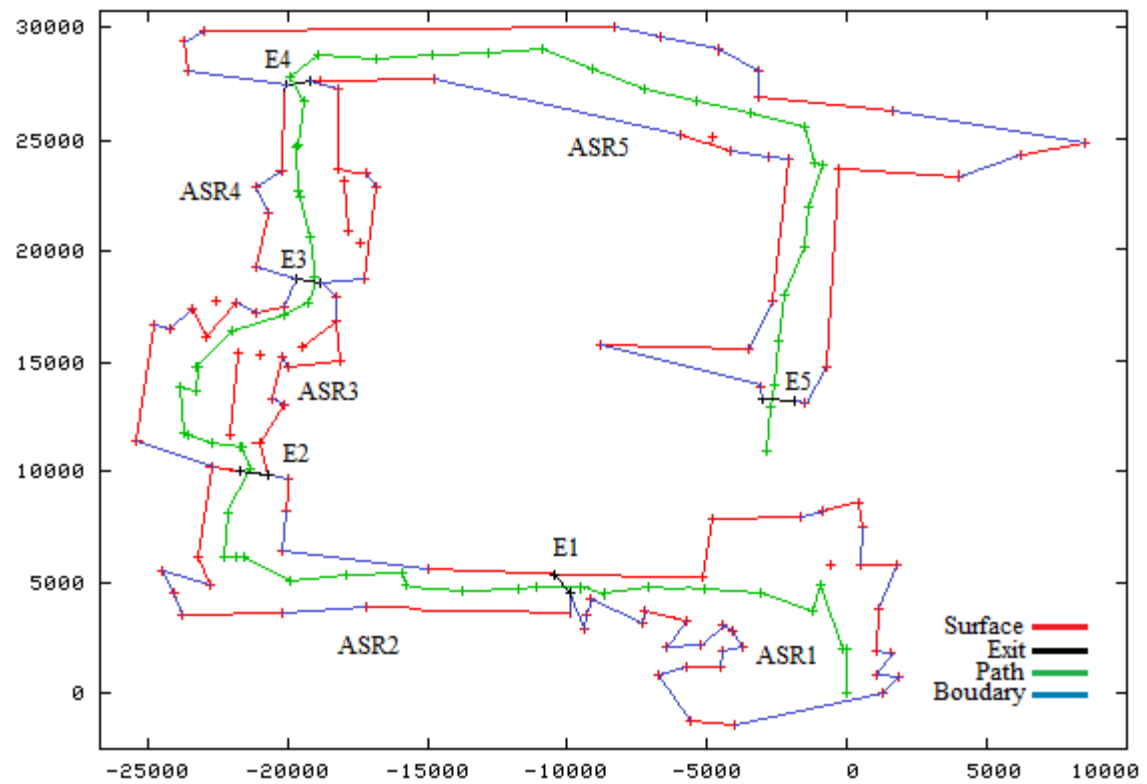


Fig. 4.15: Network of ASR showing ASR 1 to ASR 5

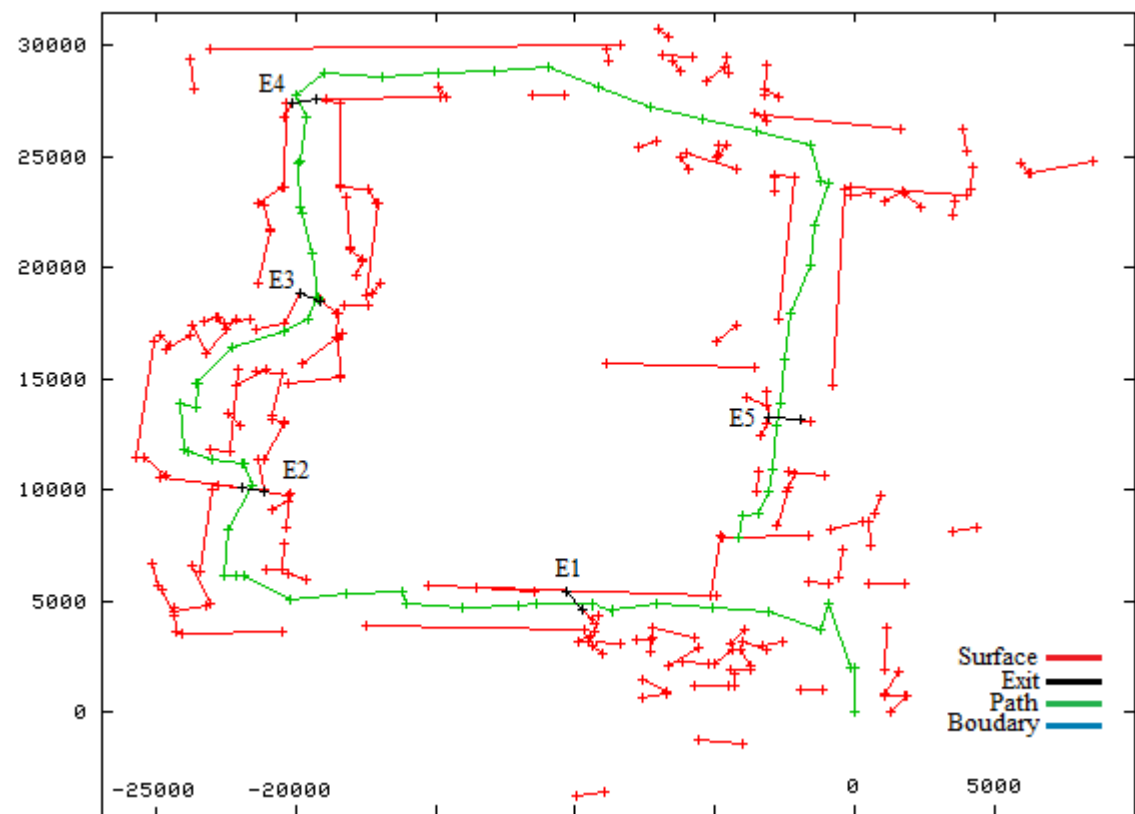
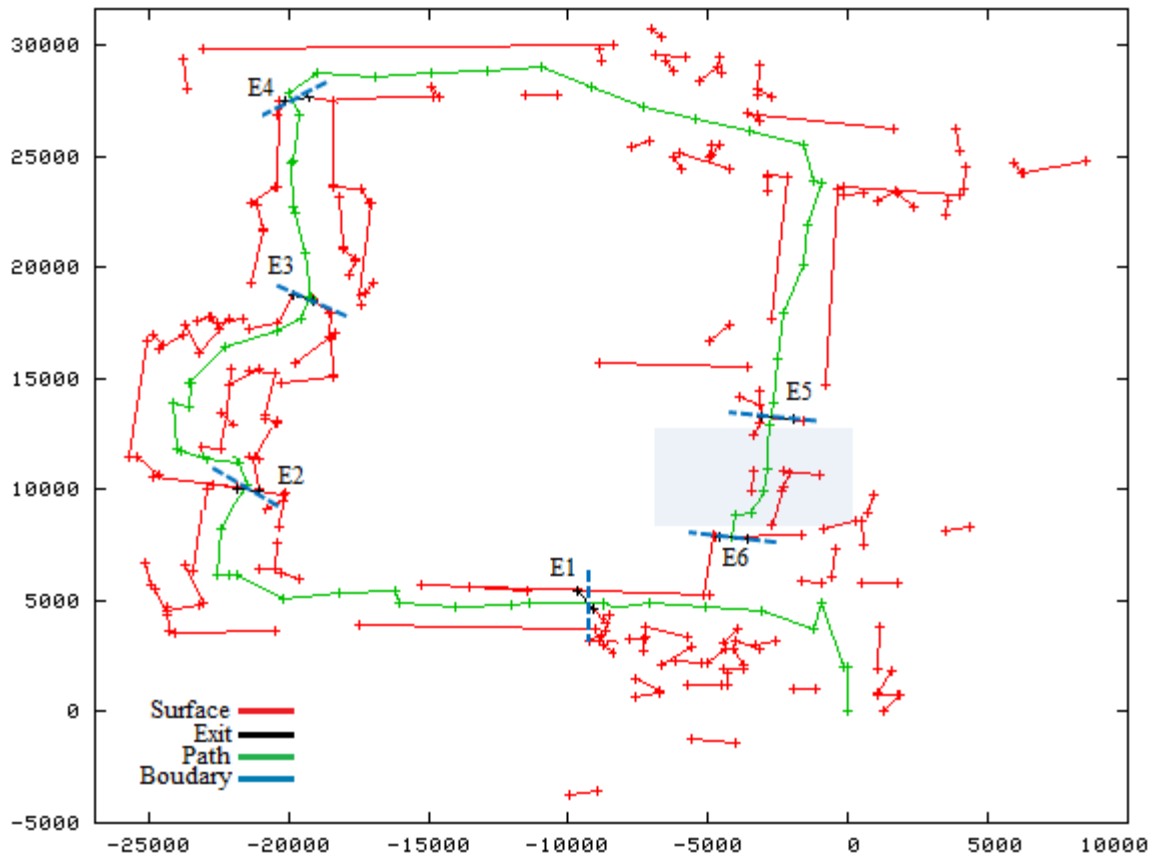
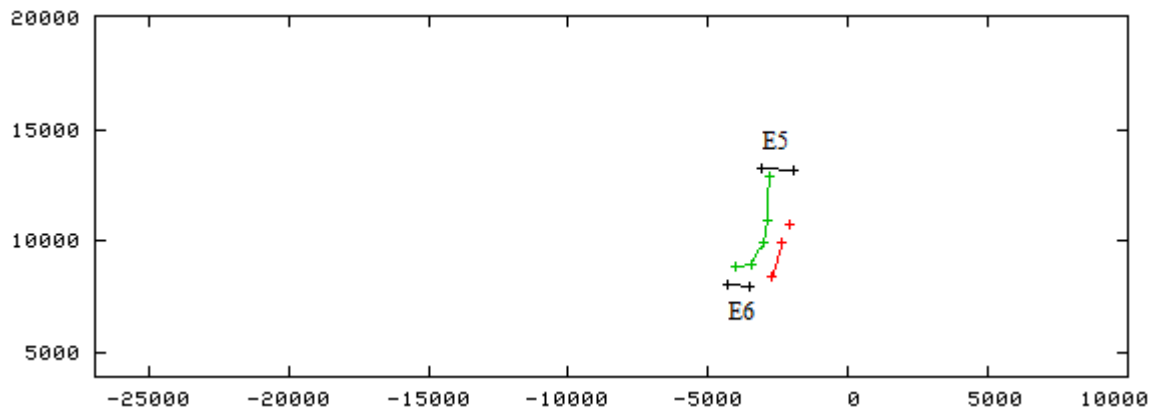


Fig. 4.16: MFIS computed just after the robot crosses E6 and re-enters ASR1





**Fig. 4.17:** ASR 6 surfaces (shaded) extracted from the MFIS



**Fig. 4.18:** Surface and corner point for ASR 6

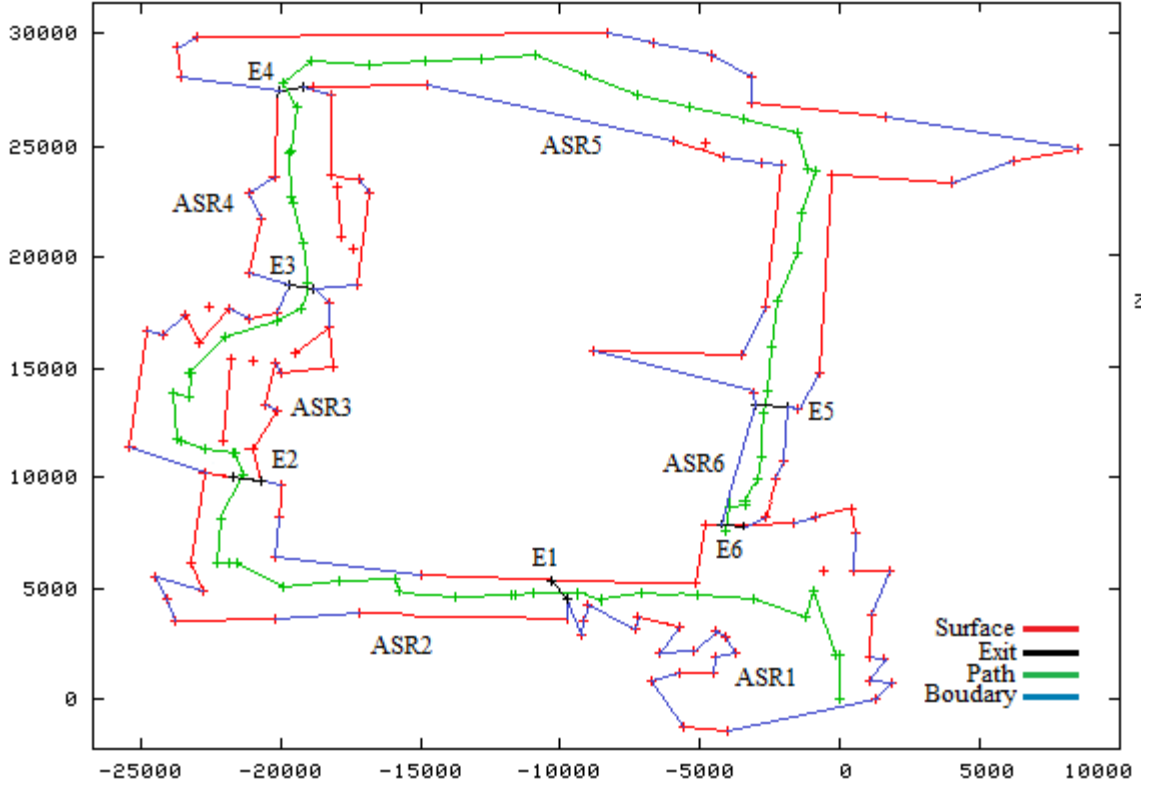
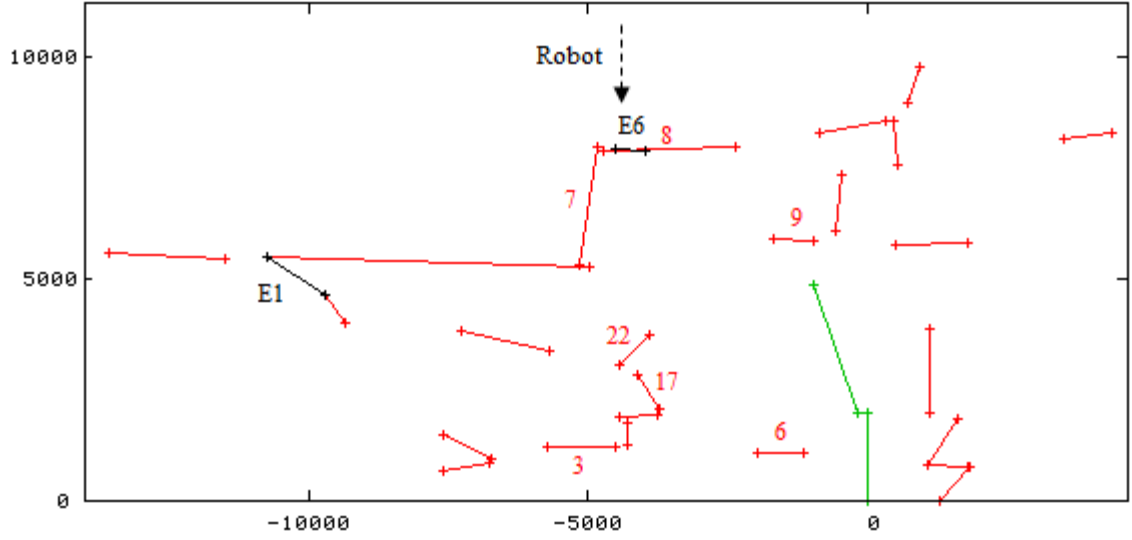


Fig. 4.19: Network of ASR showing ASR 1 to ASR 6

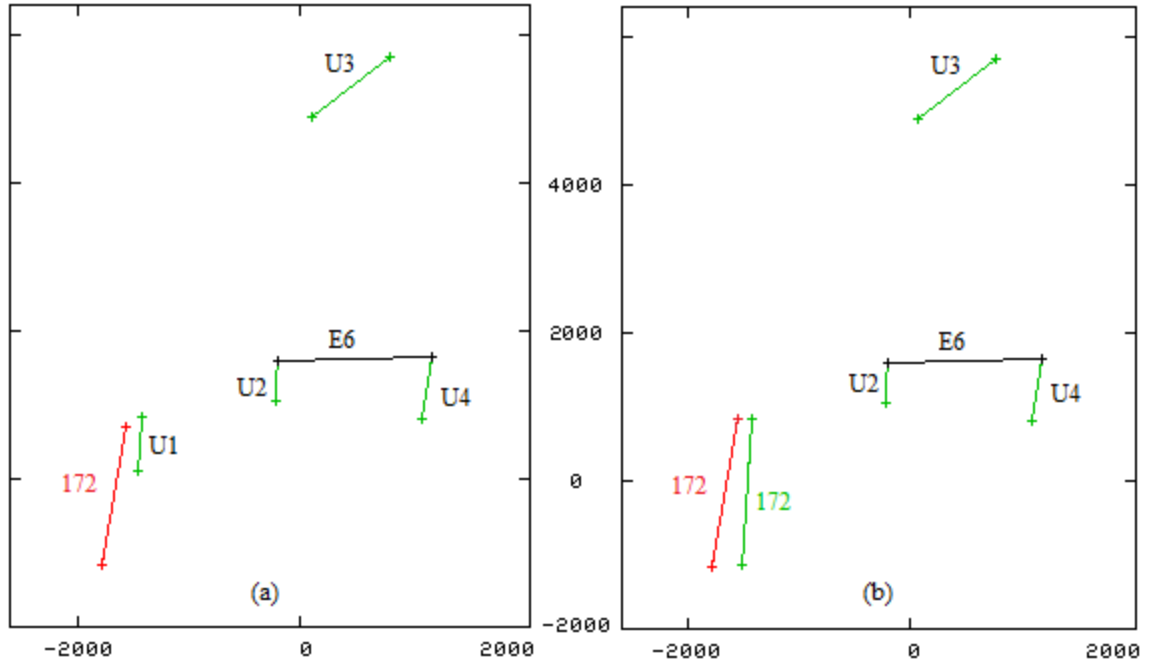
#### 4.2.2 Closing the loop

The previous section has shown how the robot computes the MFIS and extracts a number of ASRs during its exploration. At ASR6, the robot is about to re-enter a familiar environment. In the next move, the robot needs to perform loop closing and recognise that it is entering ASR1. Fig. 4.20 shows the ASR before the robot crosses the latest exit (E6) and leaves ASR6. Note that E6 is matched to surface ID8 in ASR1. It was a door that was closed earlier in order to force the robot to travel in a clockwise direction. Consequently, we allow exits to match to surfaces but we do not normalise such “landmarks”.

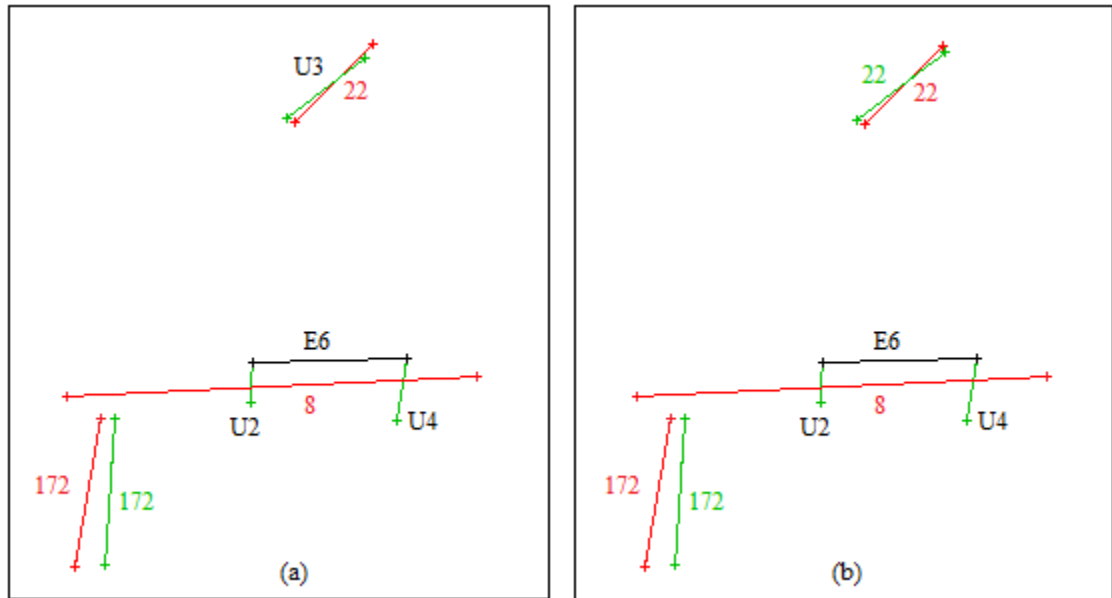


**Fig. 4.20:** MFIS surfaces computed inside ASR1. E6 has been computed on surface ID 8 and the robot will go through the exit computed and revisit ASR1

Fig. 4.21 shows the landmark computation before the robot crosses E6. The four surfaces depicted in green are the new surfaces picked up at the scene. When  $V_n$  and  $V_{n-1}$  is compared, it is found that new surface U1 matches landmark 172 from  $V_{n-1}$  (see Fig. 4.21(a)). They are normalised as depicted in Fig. 4.21(b). Using the landmark 172, the algorithm updates the MFIS with the intention to add U2, U3, U4 and the exit E6. However, the updating algorithm returns with a match between the new surface U3 and the landmark ID 22 (see Fig. 4.22(a)). U3 is then normalised (see Fig. 4.22(b)) so it is now representing the landmark ID 22 in  $V_n$ . When trying to add U2 and U4, the updates fail since the two surfaces intersect the landmark ID 8 from the MFIS. For this reason, only the exit E6 is the new addition inside the MFIS. Due to the detection of landmarks in the MFIS (landmarks 22 and 8) and these landmarks are marked as part of an ASR, the robot now believes it is about to re-enter ASR1 (since both landmarks belongs to ASR1).



**Fig. 4.21:** Comparison between  $V_n$  and  $V_{n-1}$  before crossing E6. Robot is at (0, 0). (a) U1 matches landmark 172 and (b) the normalisation process



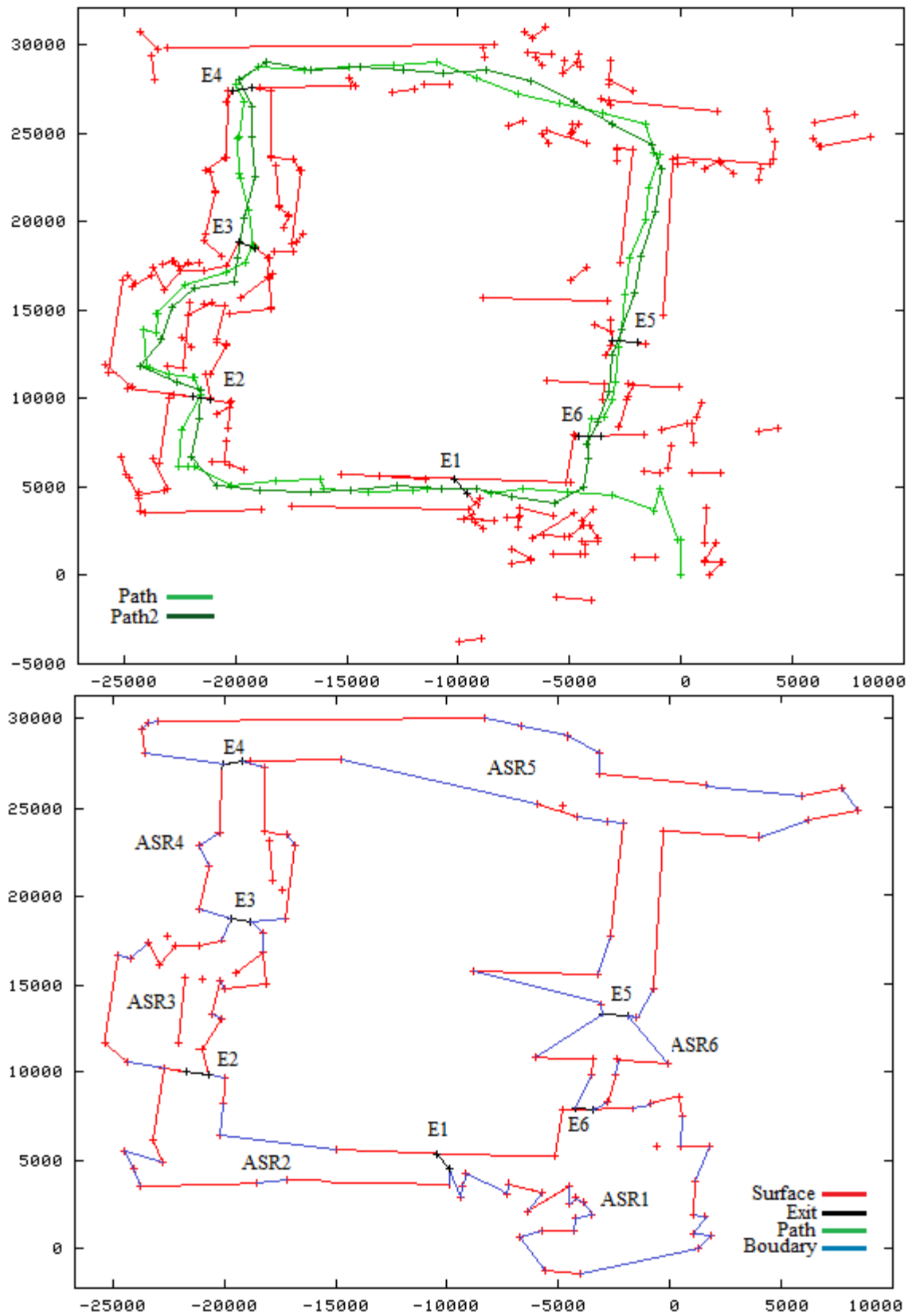
**Fig. 4.22:** Updating before the robot crosses E6. (a) Showing U3 being matched to an old landmark ID 22 and (b) the normalisation process

The results show the robot is able to close the loop by finding surfaces in the MFIS that were seen earlier.

### **4.2.3 Going around the second time**

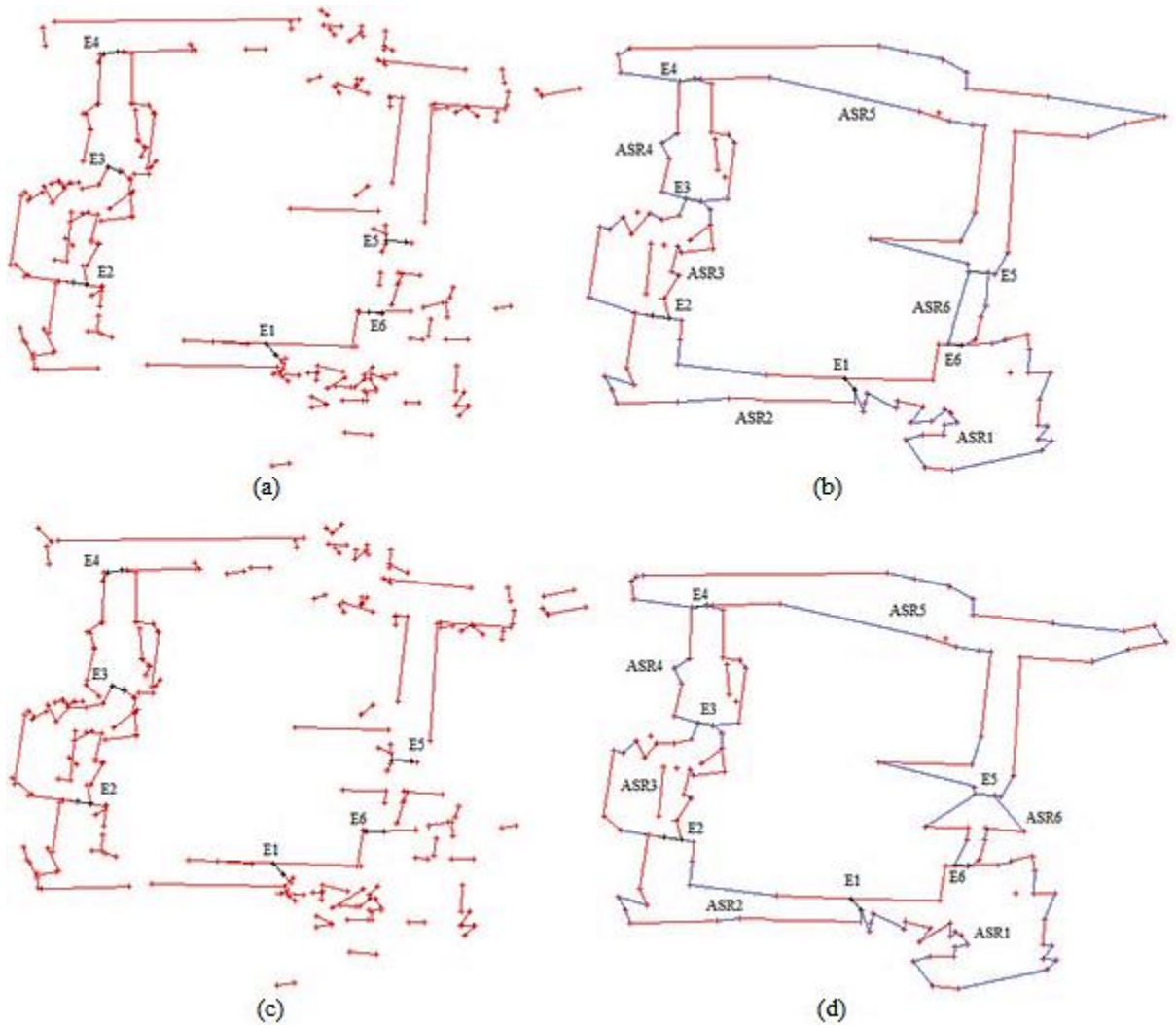
Robot is now set to continue its exploration. The goal here is to make the robot re-visits all of the ASRs computed to test if (1) the MFIS would still be stable, and (2) if the revisit changes the description of the ASRs computed, when it is traversing in the same direction again. Fig. 4.23 shows the MFIS and the network of ASRs computed after completing the second exploration.

The results of the MFIS after the second looping shows that without alteration to the physical environment or change of direction, the robot perceives the environment in a fairly similar fashion to its earlier exploration. Therefore, it does not discover different routes, exits or ASRs. One important aspect observed is the fact that the robot did not have to compute a new MFIS ‘on top’ of the old one, instead the mapping algorithm is able to recognise familiar surfaces (i.e. the landmarks) from each ASR and reuse them to localise and guide its journey.



**Fig. 4.23:** The MFIS (top) and the network of ASRs (bottom) after looping the environment twice

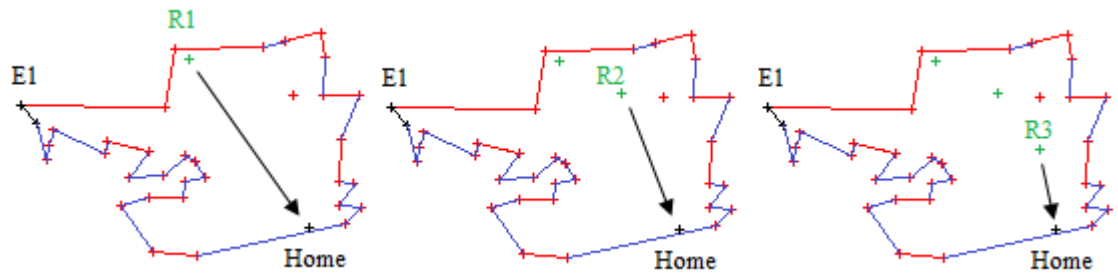
Since the robot chooses gaps and exits based on what it perceives at a particular moment, the robot's trajectory between the first and the second loop differs a little. However there are some viewpoints in the second loop where the robot is able to add new surfaces or extend old surfaces which have been initially computed in the first loop. Some of these updates enhance the description to a local space. The following figure compares the MFIS and ASRs built in the first round of exploration with the MFIS and ASRs computed in the second (see Fig. 4.24).



**Fig. 4.24:** (a) and (b) depict the MFIS and the network of ASRs built in the first round. (c) and (d) are the MFIS and ASRs computed in the second round. Changes are apparent to ASR6, ASR5 and ASR1

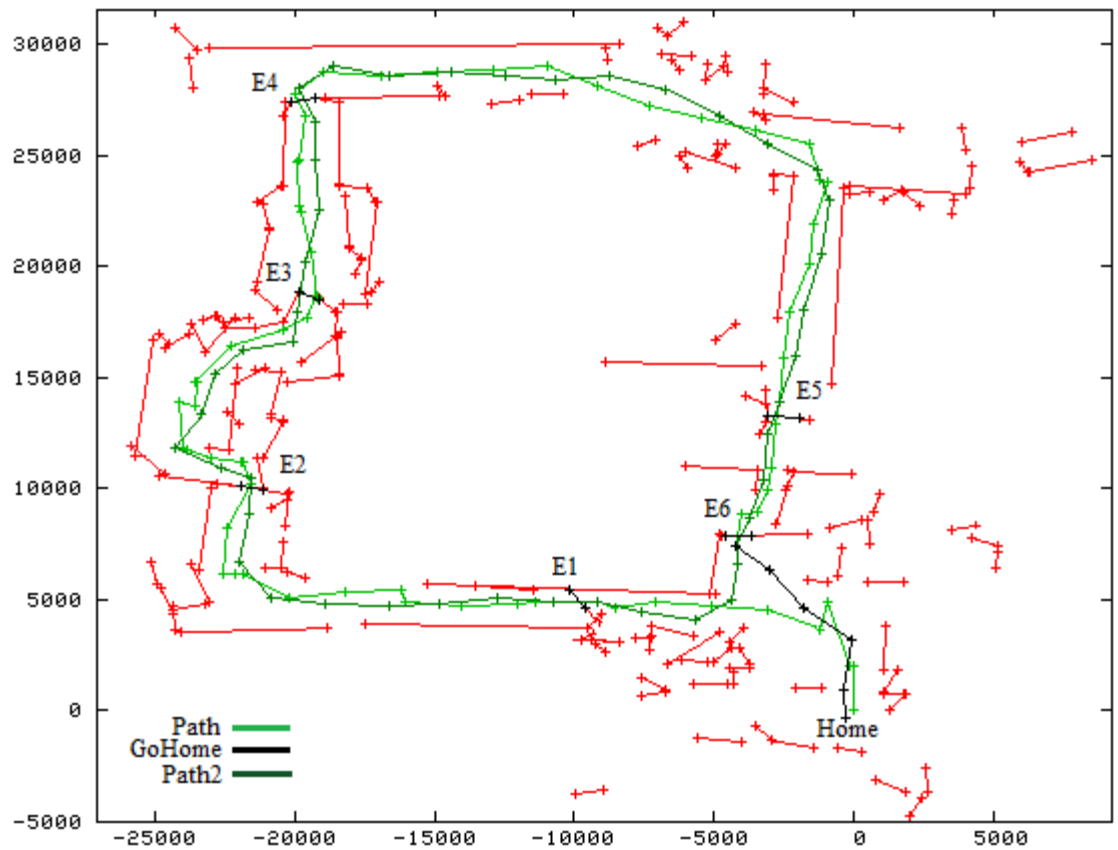
#### 4.2.4 Going Home

The robot is then instructed to go home at this point. Home is where the robot started its journey and in the map home is (0, 0). The strategy to go-home for the robot is simple; from wherever it is, calculates the direction to home and use it like a beacon to guide its movements (see Fig. 4.25). The process of calculating direction to home is repeated at every step until the robot thinks it has reached home (usually within a meter). Fig. 4.26 shows the robot's path going home. With its new perspective on its home environment, new information is entered into the MFIS and ASR1. Fig. 4.27 shows the final ASRs computed for this experiment.

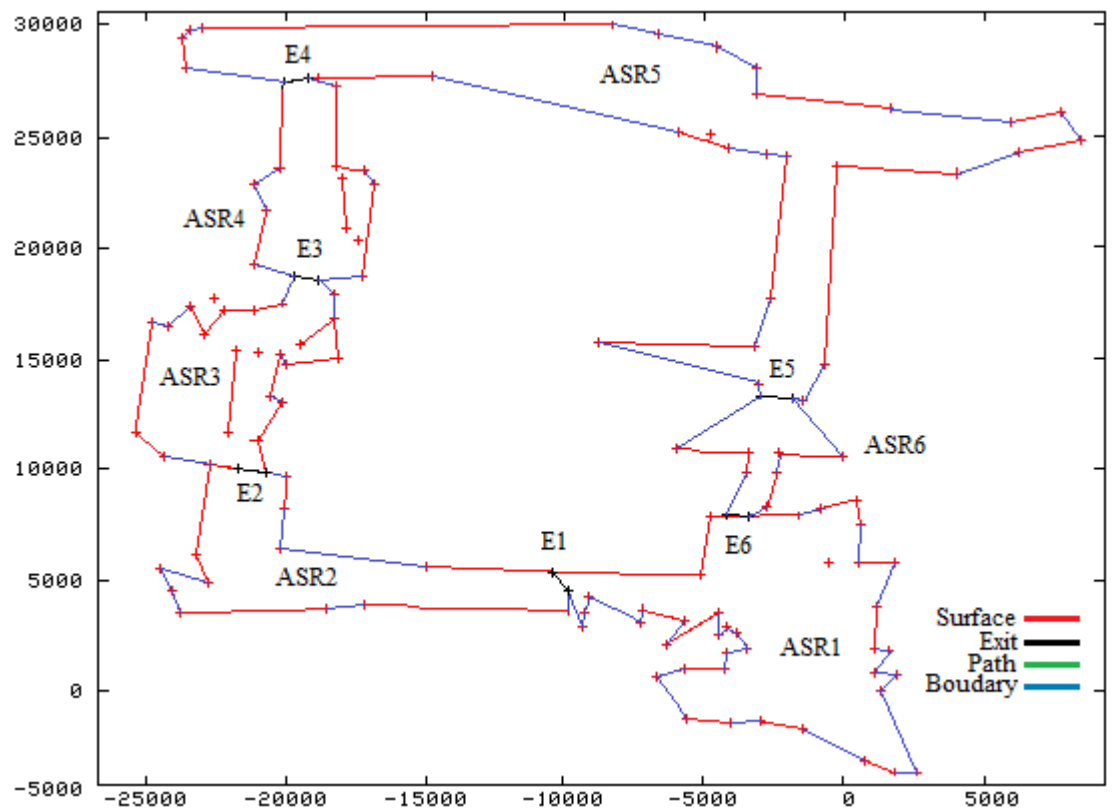


**Fig. 4.25:** Example of using direction to navigate home. R1, R2 and R3 are three steps in the navigation. Direction are recalculated at each step until the robot reach closer to home





**Fig. 4.26:** Black path lines indicates the robot's trajectory to home after crossing E6



**Fig. 4.27:** The network of ASRs after looping twice and the robot is instructed to go home

### **4.3 Experiment 2 (Going anti-clockwise)**

The second experiment replicates the goals of the first experiment however this time the direction in which the robot traverses is reversed. It would be interesting to see how many ASRs the robot computes this way and if the description of the MFIS and individual ASRs differs much from the first experiment.

#### **4.3.1 Computing the MFIS and ASRs**

Fig. 4.28 shows the result of the robot computing the MFIS from where it started its journey (home) out and around the environment in an anti-clockwise manner. There is not much difference between the overall shape and size of the MFIS computed in this experiment as compared to Experiment 1. The robot is able to find all six exits even though they are in different order due to the exploration. The most noted change would be the large wall on the left of the computed exit E2. It was not present in the previous experiment as the robot was not able to see them from the opposite direction. Fig. 4.29 shows the network of ASR computed after the robot crosses the exit E6 in this experiment. The number of ASRs computed is the same as in the previous experiment and there are only minor variations in the ASR description.

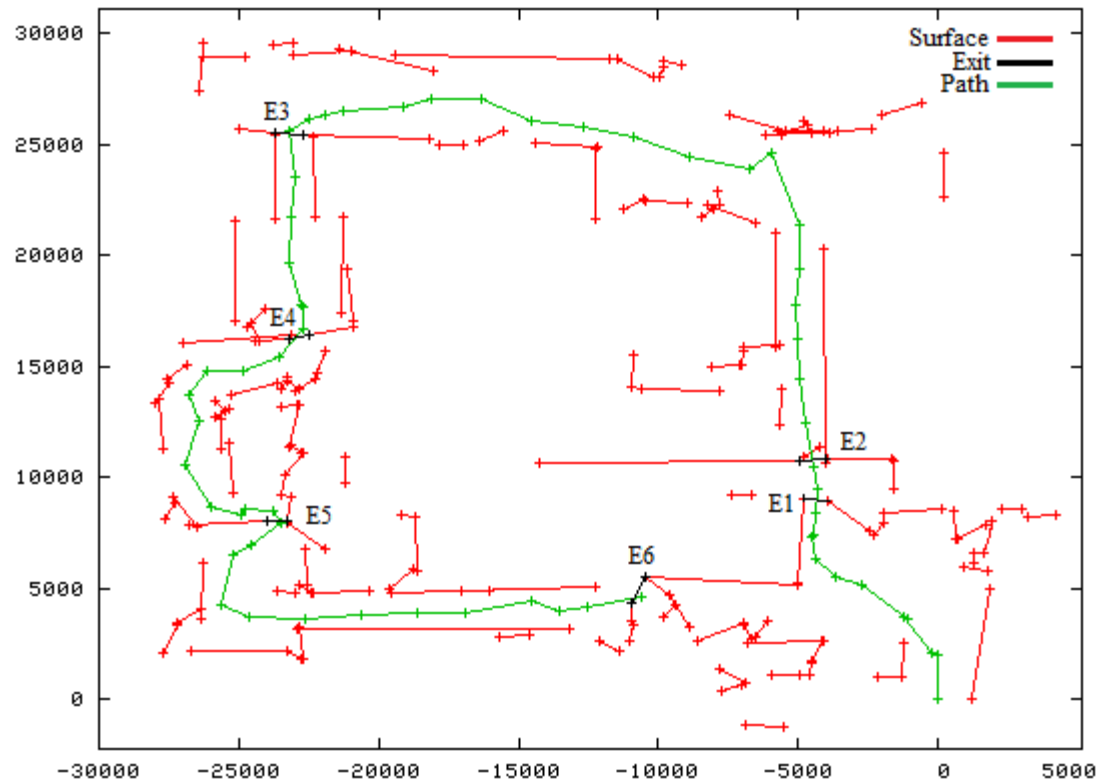


Fig. 4.28: The MFIS computed from home until the robot crosses the exit E6

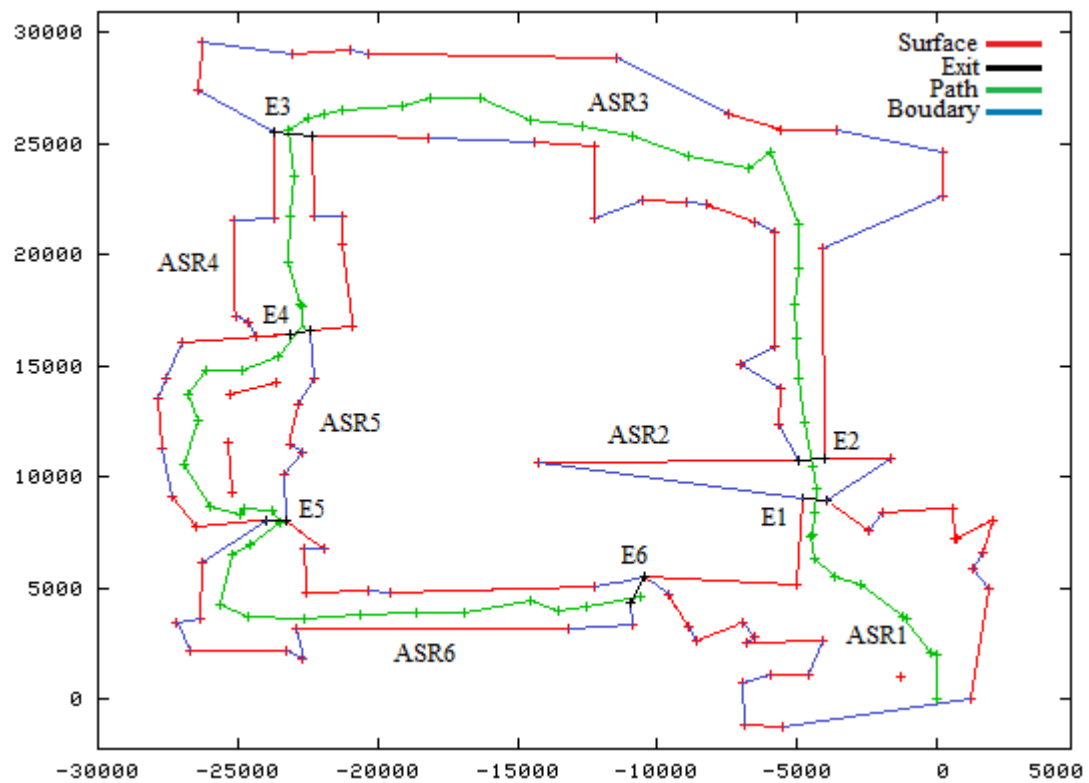
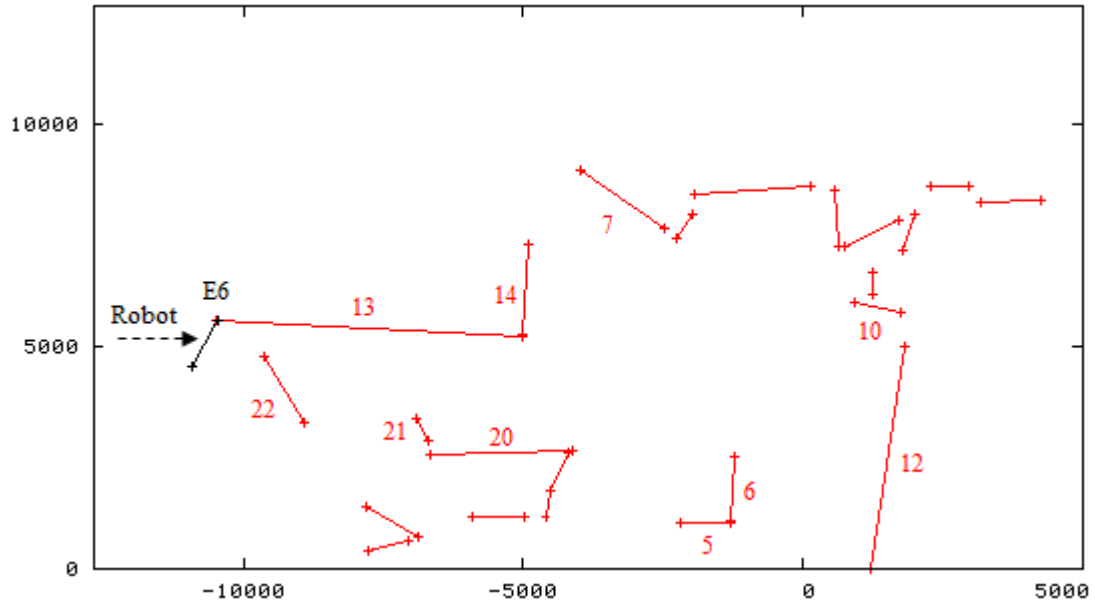


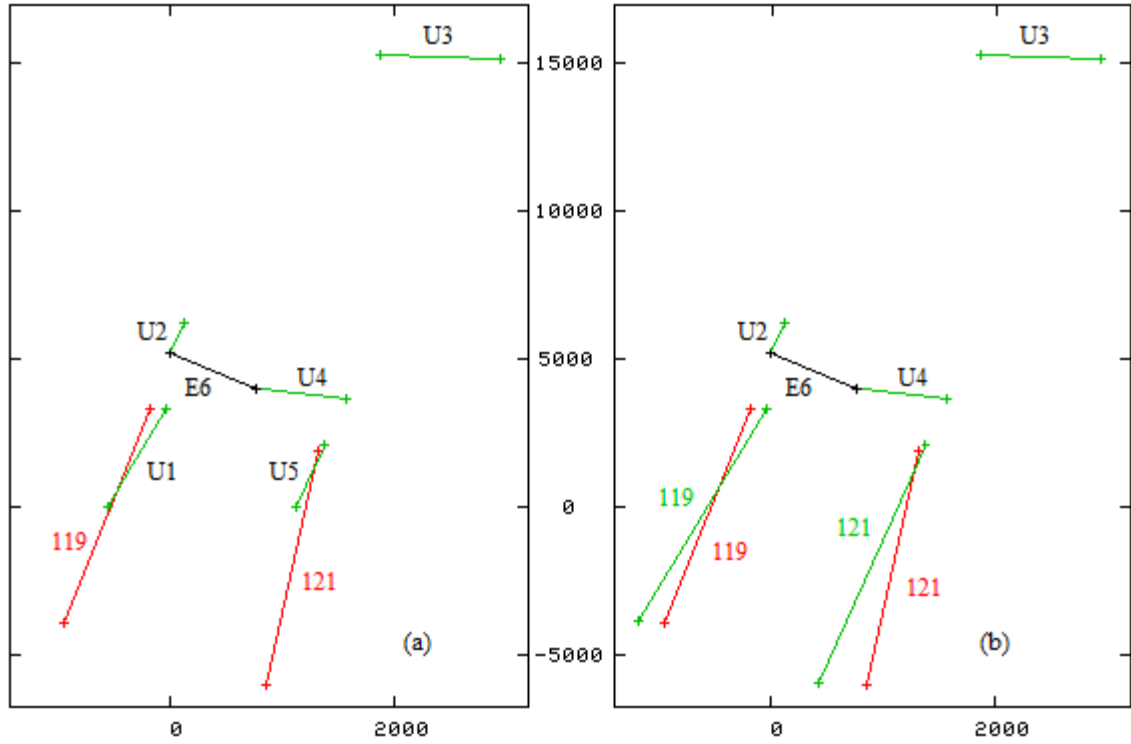
Fig. 4.29: The network of ASR depicting ASR1 to ASR6 in the second experiment

### 4.3.2 Closing the loop

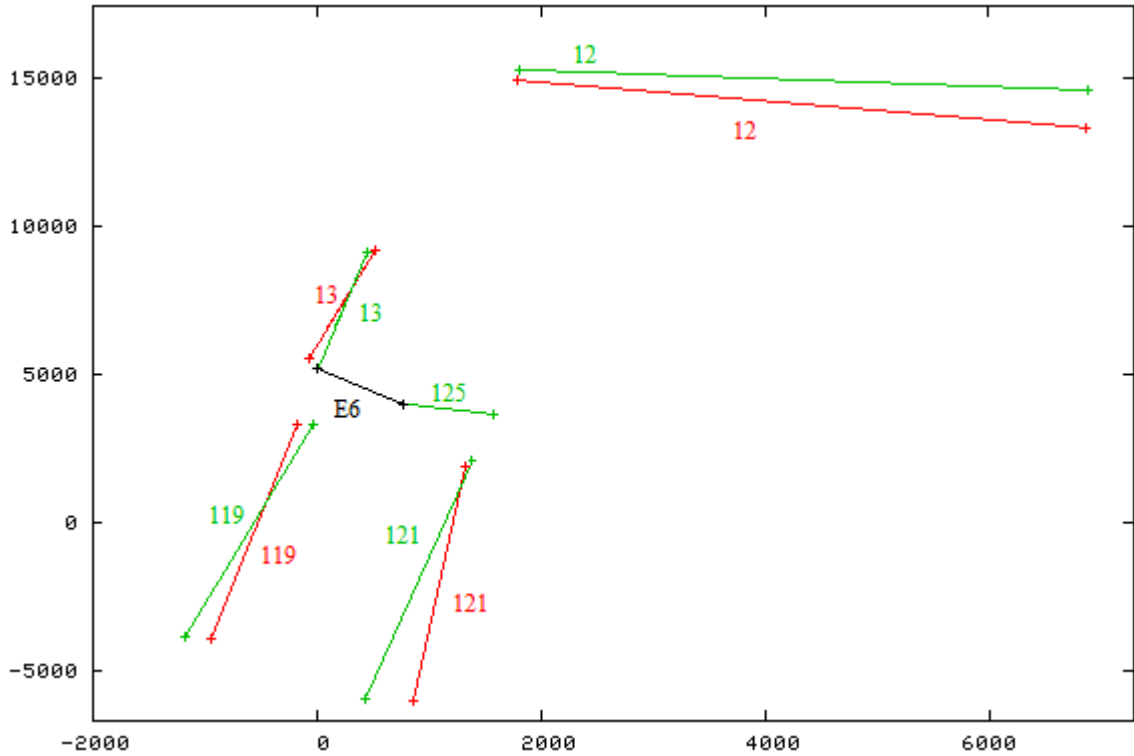


**Fig. 4.30:** Some of the earlier MFIS surfaces computed inside ASR1

Fig. 4.30 shows the part of the MFIS at the time before loop closing. Fig. 4.31 compares the robot's current view ( $V_n$ ) and its remembered view ( $V_{n-1}$ ), at the point where the robot is just about to cross an exit (E6). Two of the  $V_n$  surfaces, U1 and U5 were a match to the landmarks ID 119 and 121 respectively (see Fig. 4.31(a)). Results of the normalisation process can be seen in Fig. 4.31(b). The next step is to update the MFIS with new surfaces found (U2, U3 and U4). When compared with the MFIS, the position where U2 and U3 are supposed to be projected is very similar to where existing landmarks ID13 and ID12 are located inside the MFIS. Since they pass the threshold for a match, they are normalised and both U2 and U3 inherited the ID13 and 12 respectively (shown in Fig. 4.32). The landmark 13 and 12 are now available to guide the robot's next move.

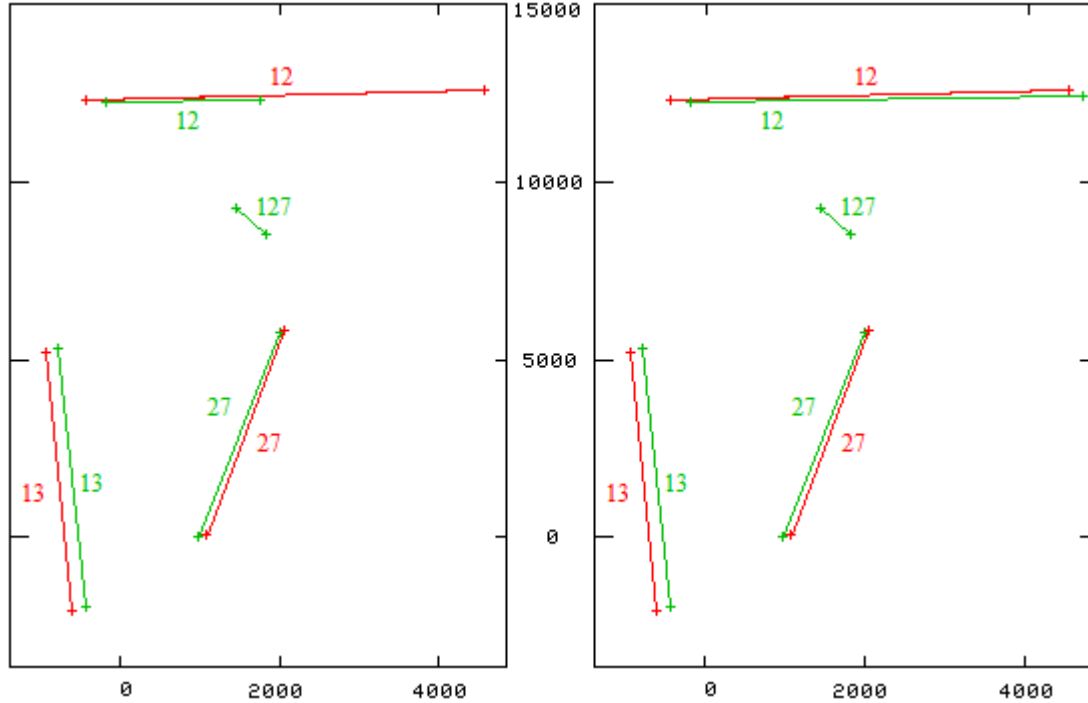


**Fig. 4.31:** Comparison between  $V_n$  and  $V_{n-1}$  before crossing E6. Robot is at (0, 0). (a) Denotes U1 and U5 matching the landmarks 119 and 121 respectively. Results of the normalisation is shown in (b)



**Fig. 4.32:** Updating the MFIS before crossing the exit E6

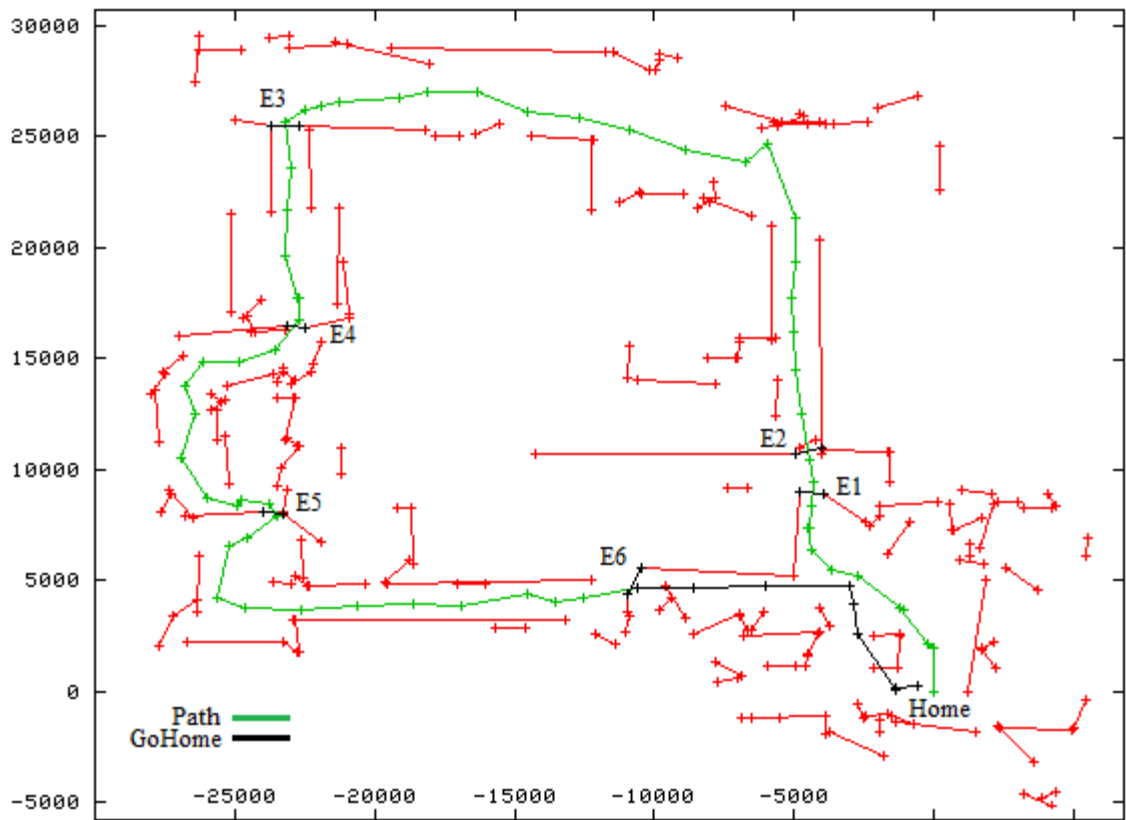
To confirm it is returning to a previously visited local space, the robot's next move is observed. This time, the landmarks 13 and 12 became the references to update the MFIS. As shown in Fig. 4.33, another landmark from ASR1 reappears during the update (landmark ID 27).



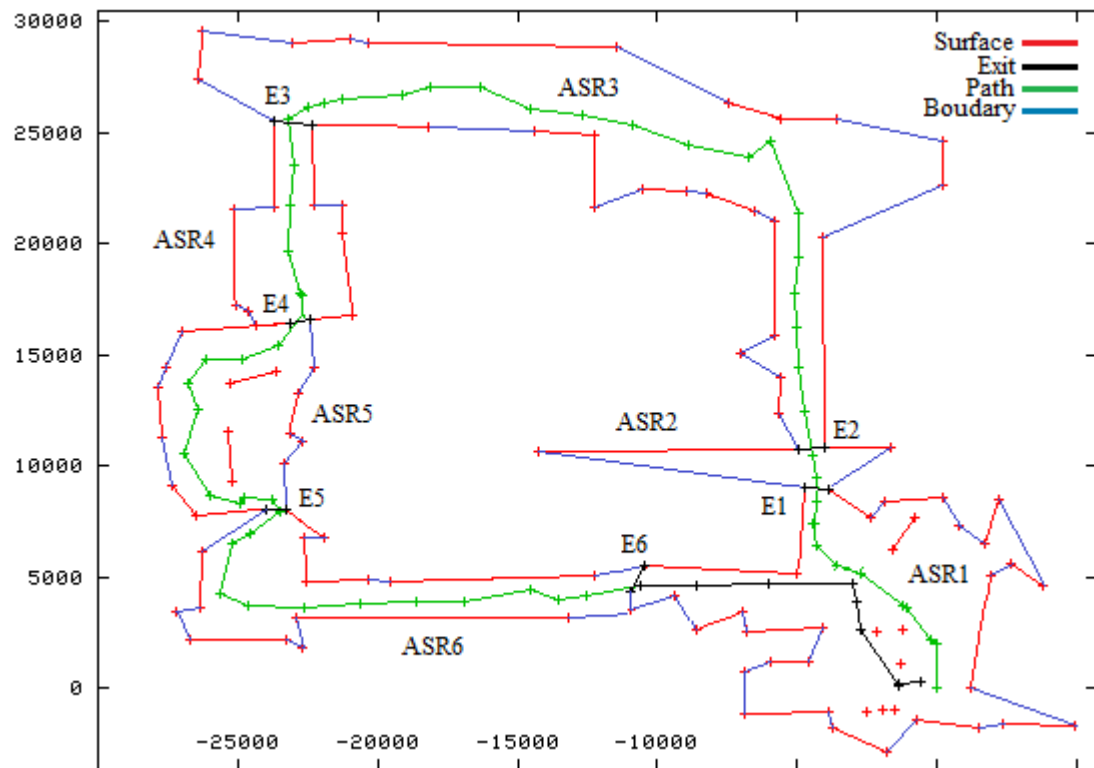
**Fig. 4.33:** Updating after crossing the exit E6 where three current surfaces matching the centroids 12, 13 and 27

### 4.3.3 Going Home

In Experiment 1, the robot is instructed to go home after going around the environment twice. This time, the robot is instructed to go home directly after it crosses the exit E6 for the first time. Fig. 4.34 and Fig. 4.35 show the MFIS computed as the robot perform the strategy to go home.



**Fig. 4.34:** The MFIS when the robot reaches home. Path lines in black denotes the steps towards home



**Fig. 4.35:** Changes to ASR1 after the go home activity

#### **4.3.4 Going around the second time**

The robot is then instructed to continue exploring the environment. Fig. 4.36 is the result of the MFIS after the robot revisits all the ASR computed in the same manner (no changes of direction). Once the robot closes the loop again after crossing E6, the command to go home is passed to the robot. The robot stopped its journey once it reaches home the second time. Fig. 4.37 is the results of the ASRs computation. Note that in ASR5 and ASR1, there are some surfaces and corner points inside the boundary. These are some surfaces and corner points which were not used but left as it is in the ASR representation.



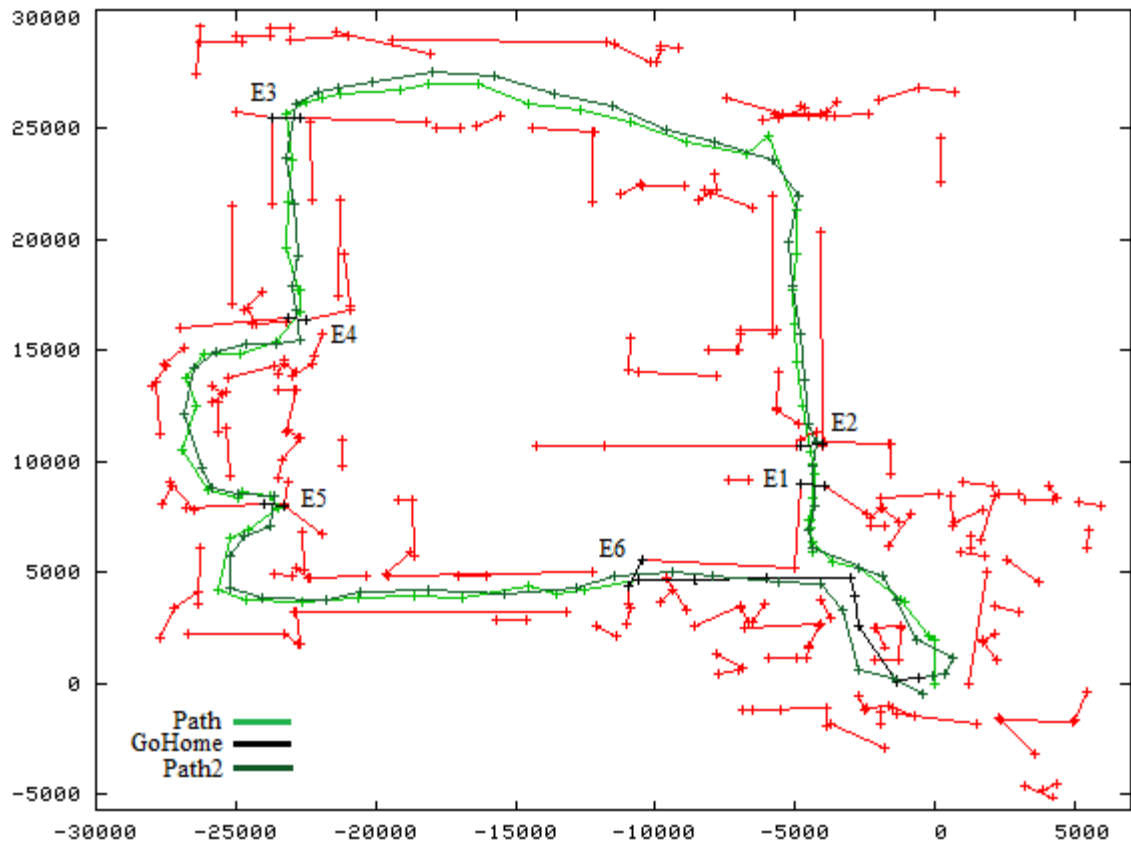


Fig. 4.36: MFIS after traversing the environment the second time. The robot ended the journey at home

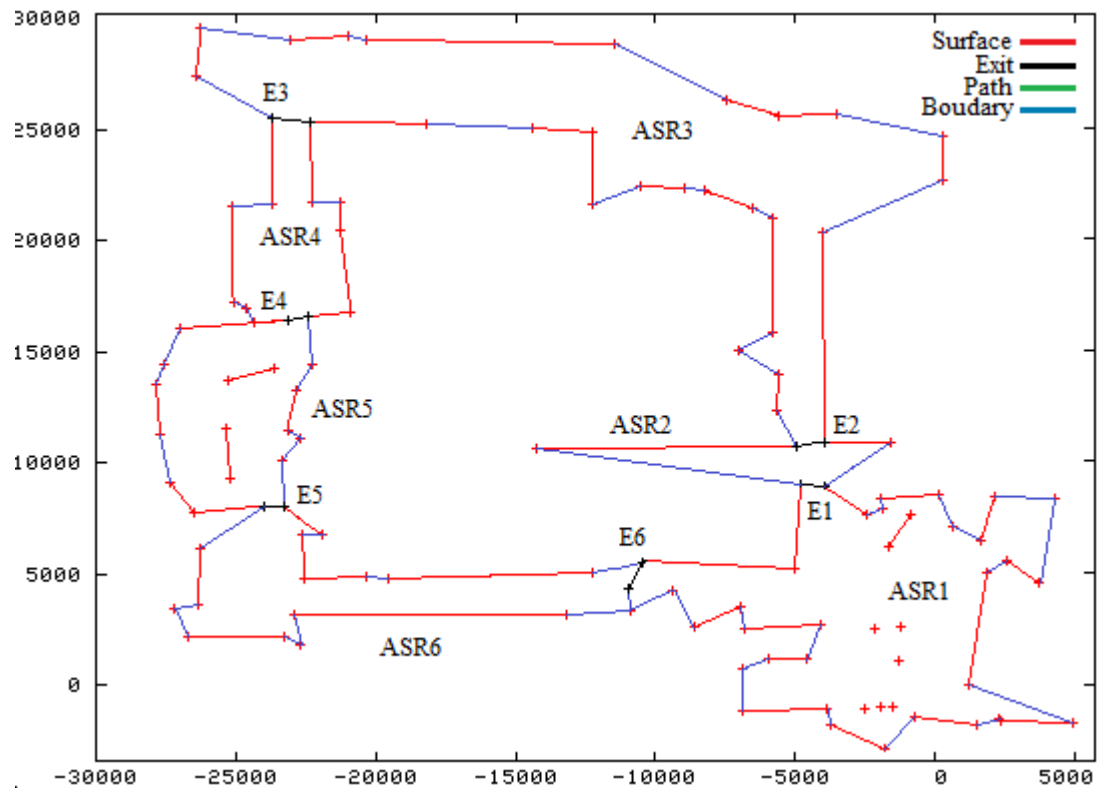
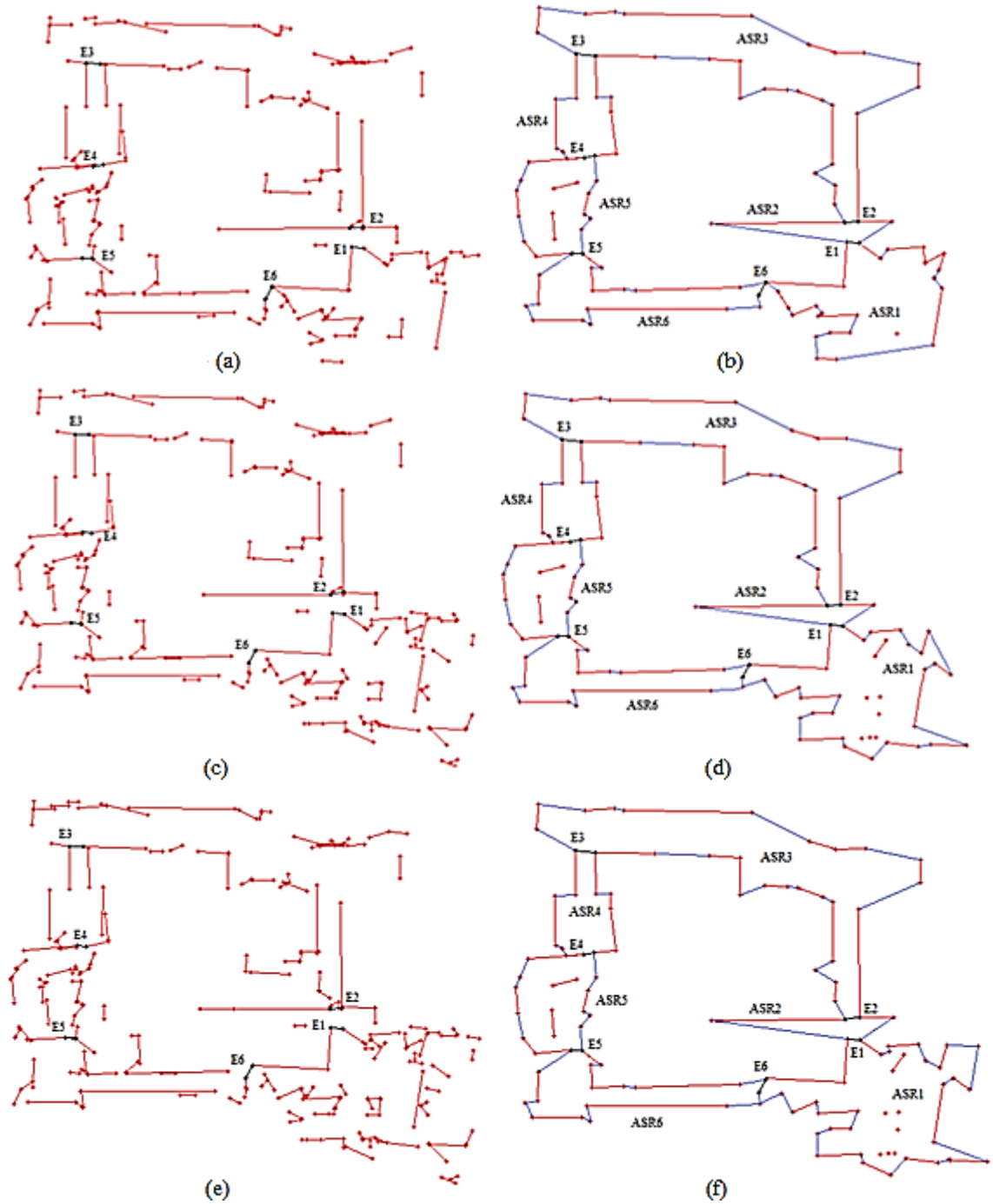


Fig. 4.37: The network of ASR after the second looping

Fig. 4.38 depicts the MFIS and network of ASRs for the three main tasks assigned to the robot in this experiment.



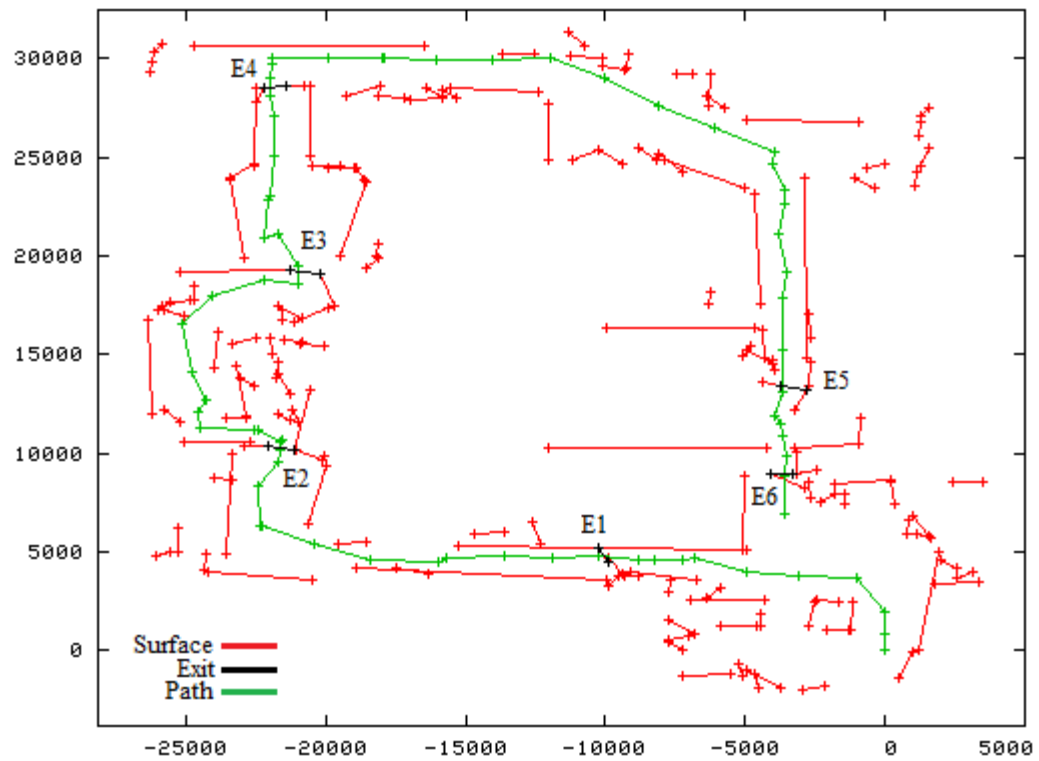
**Fig. 4.38:** Three sets of MFIS and its corresponding network of ASRs. (a) and (b) for going about in the first round, (c) and (d) for going home, and (e) and (f) indicate the representations when the robot is instructed to go around again

## **4.4 Experiment 3**

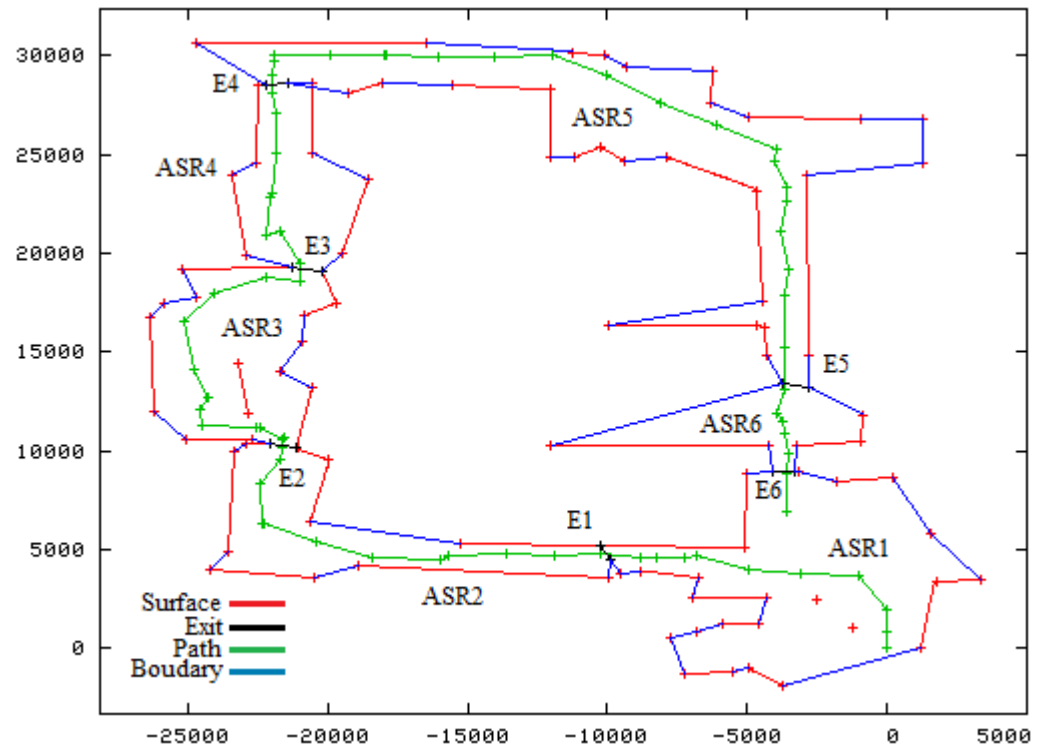
The first two experiments showed that there are not many changes to the description of the environment when the robot is looping in the same direction. The goal in the third experiment is then to make the robot loop in one direction then immediately go around again in the opposite direction. Then after computing its network of ASRs, the robot is asked to perform a number of go-to tasks to see if it is able to use the network to optimise its navigation.

### **4.4.1 Computing the MFIS and ASR**

The robot is first asked to move in a clockwise manner. The following figures are the results starting with the computed MFIS (Fig. 4.39) then the network of ASRs (Fig. 4.40). The robot is asked to stop once it closes the loop. The robot is then instructed to go home from where it currently is in the environment. See Fig. 4.41 for the MFIS computation and Fig. 4.42 for the network of ASRs established. After reaching home, the robot is instructed to loop the environment again, this time in the opposite direction. Results from the test can be found in Fig. 4.43 and 4.44.



**Fig. 4.39:** The MFIS computed once the robot closes the loop after crossing E6



**Fig. 4.40:** Network of ASRs after the robot crosses E6 and into ASR1

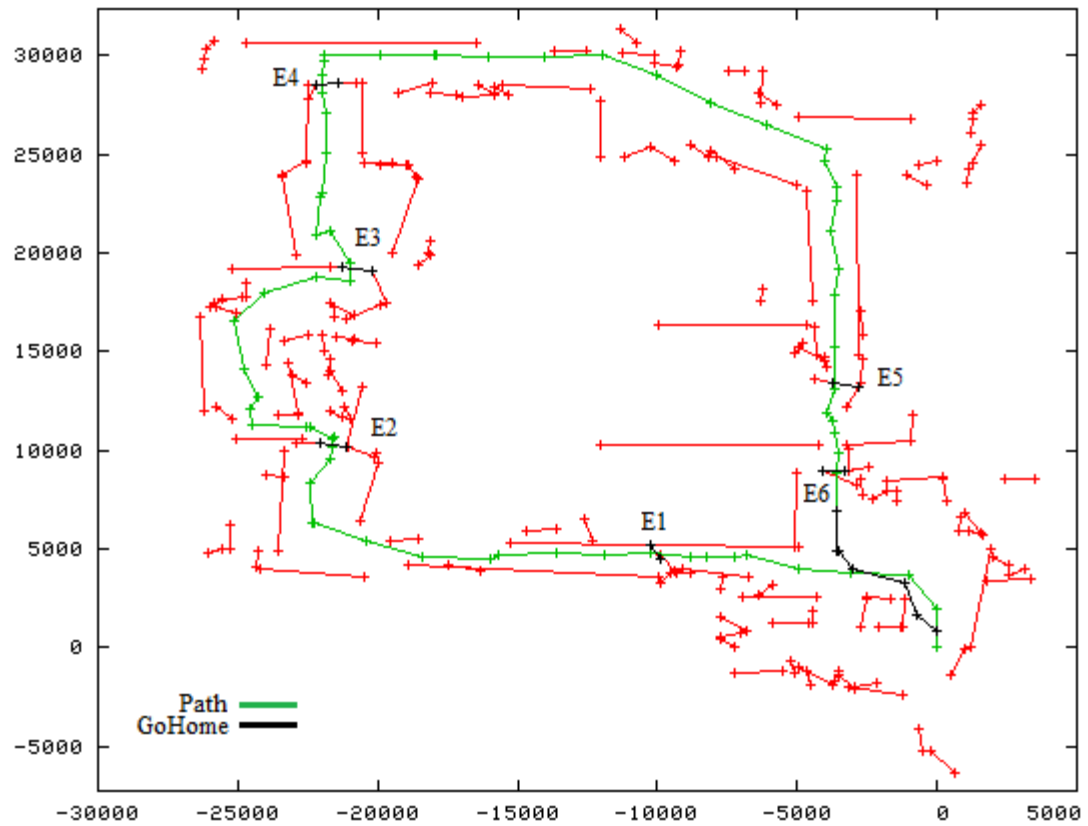


Fig. 4.41: The MFIS as the robot goes home after entering ASR1 from ASR6

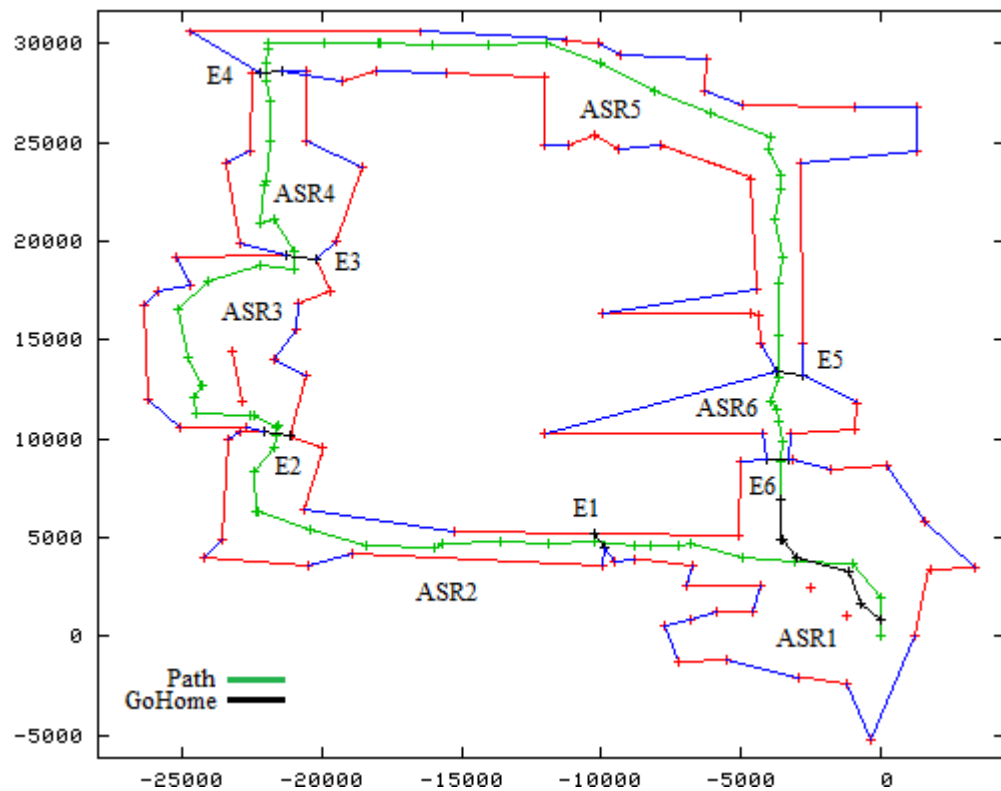
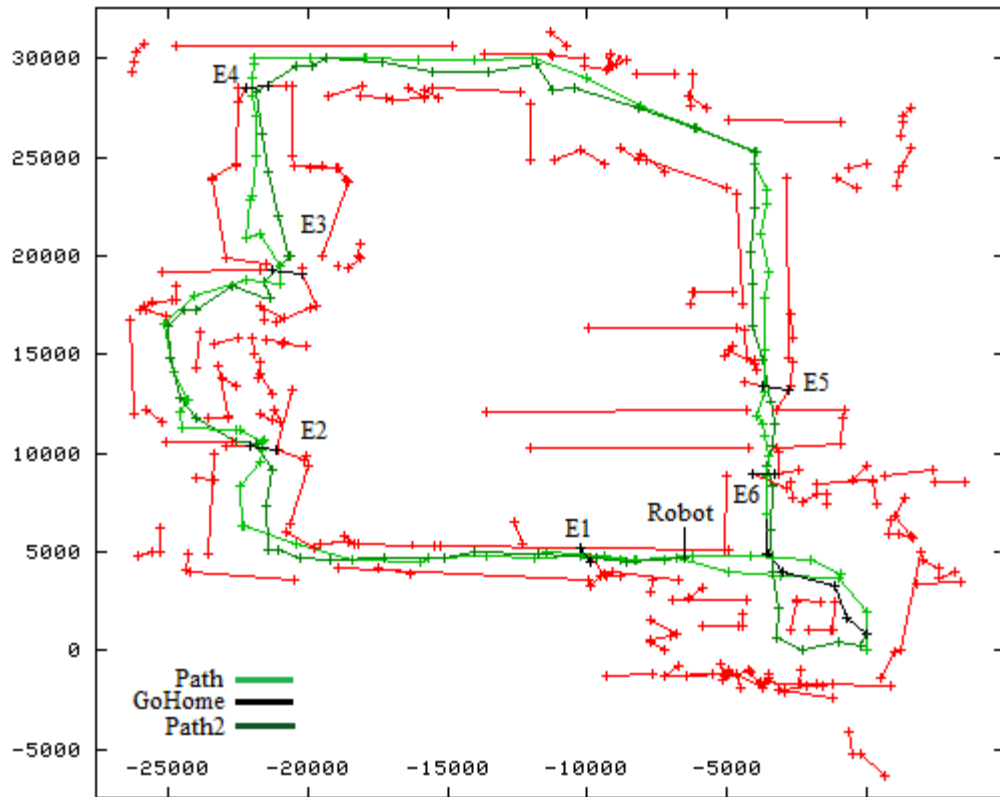
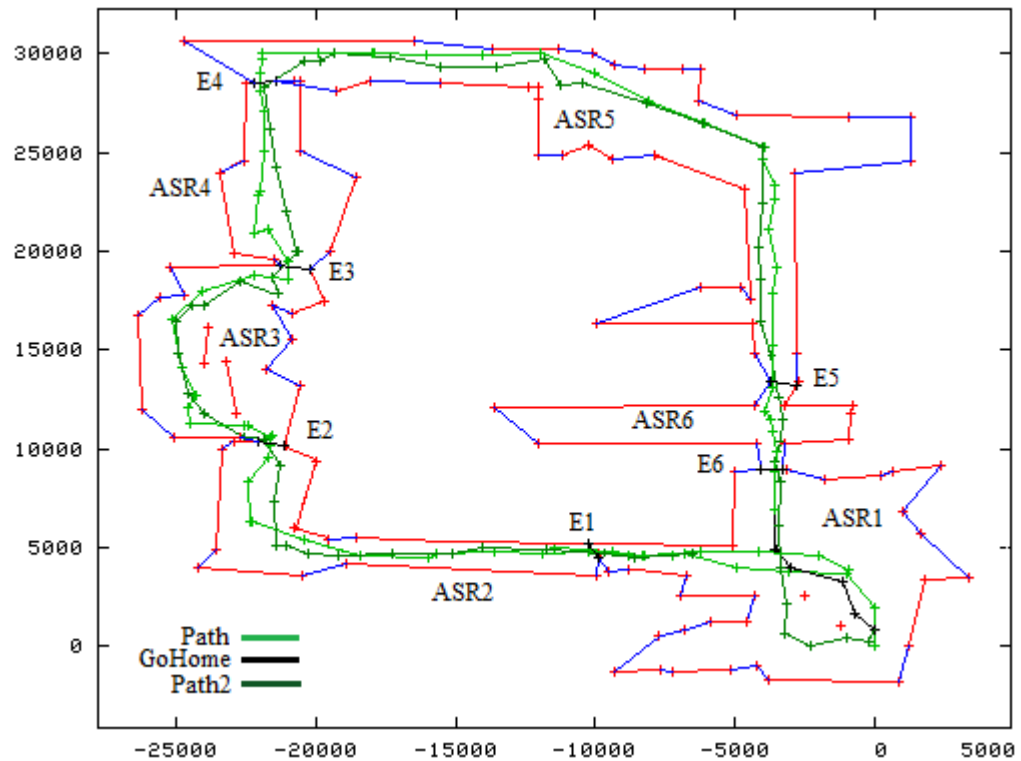


Fig. 4.42: ASR1 is expanded due to the robot seeing surfaces behind the robot when it first started



**Fig. 4.43:** MFIS after completing the second loop in the opposite direction. 'Robot' indicated where the robot is after completing the second loop

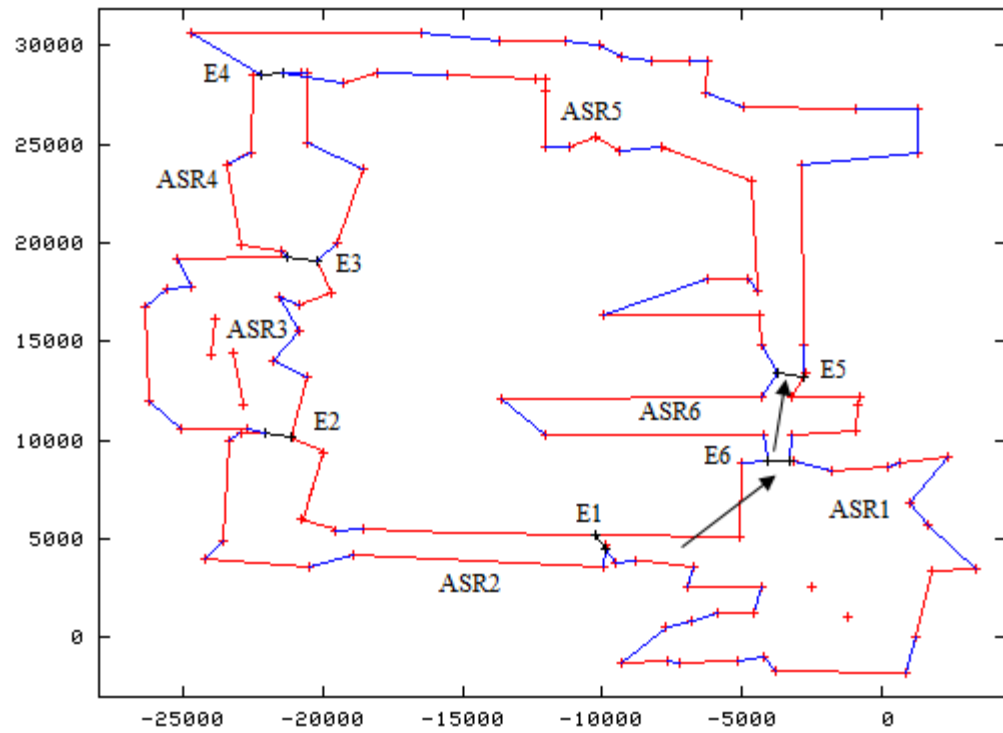


**Fig. 4.44:** Network of ASRs after the robot closes the second loop and re-enters ASR1 via E1

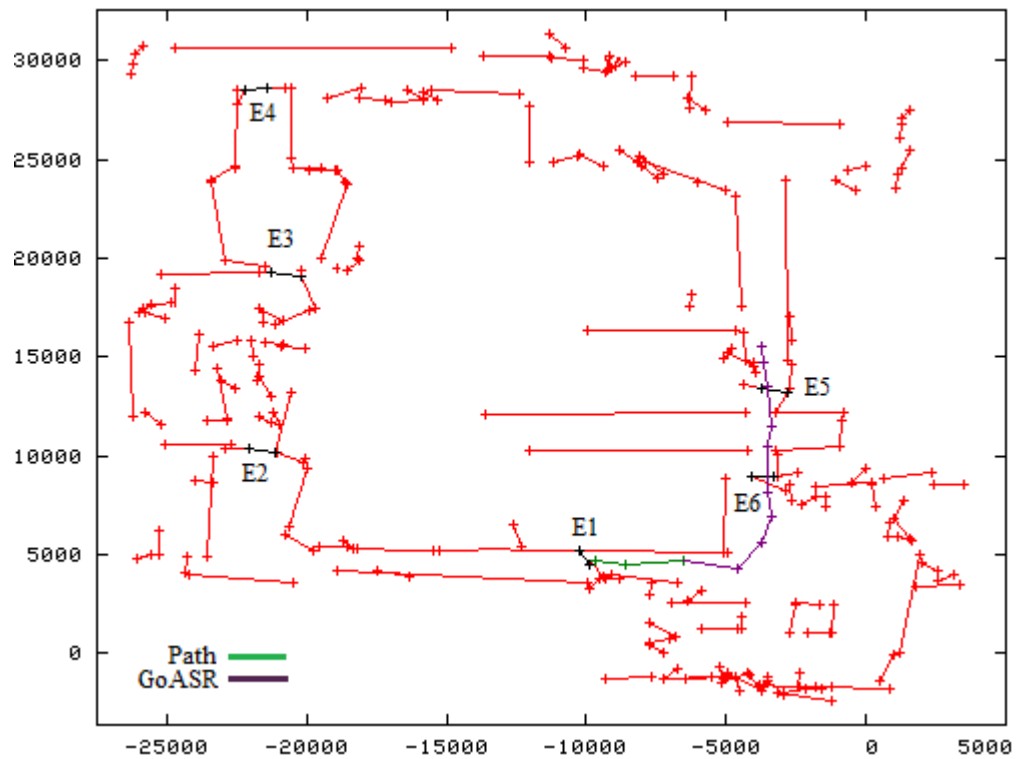
#### **4.4.2 Go To ASR**

Instead of going home, the robot is instructed to navigate from where it is (ASR1) to ASR5. Since the robot is already in ASR1, it calculated the shortest path into ASR5 is via E5. And the shortest route to cross E5 from where it is now is via E6. Using similar strategy like going home, E6 becomes the beacon followed by E5 (Fig. 4.45 and Fig. 4.46).

After crossing E5 and re-entering ASR5, the robot is instructed to navigate to ASR3. Since the robot has crossed E5, calculating the optimal route returns E4 and E3 as the exits to cross to enter ASR3. However a physical barrier is put up so the robot could not use the familiar route inside ASR5. The options left are to either turn back and go to ASR3 via E6-E1-E2 or cross a new found exit nearby (Fig. 4.47 – the blockage is not shown). When comparing the two options, go back to ASR6, or choose the new exit, the robot weighs its options by measuring the possible distances to the goal with either move. The robot chose the new exit because it appears to be a shorter route (Fig. 4.48 and Fig. 4.49).

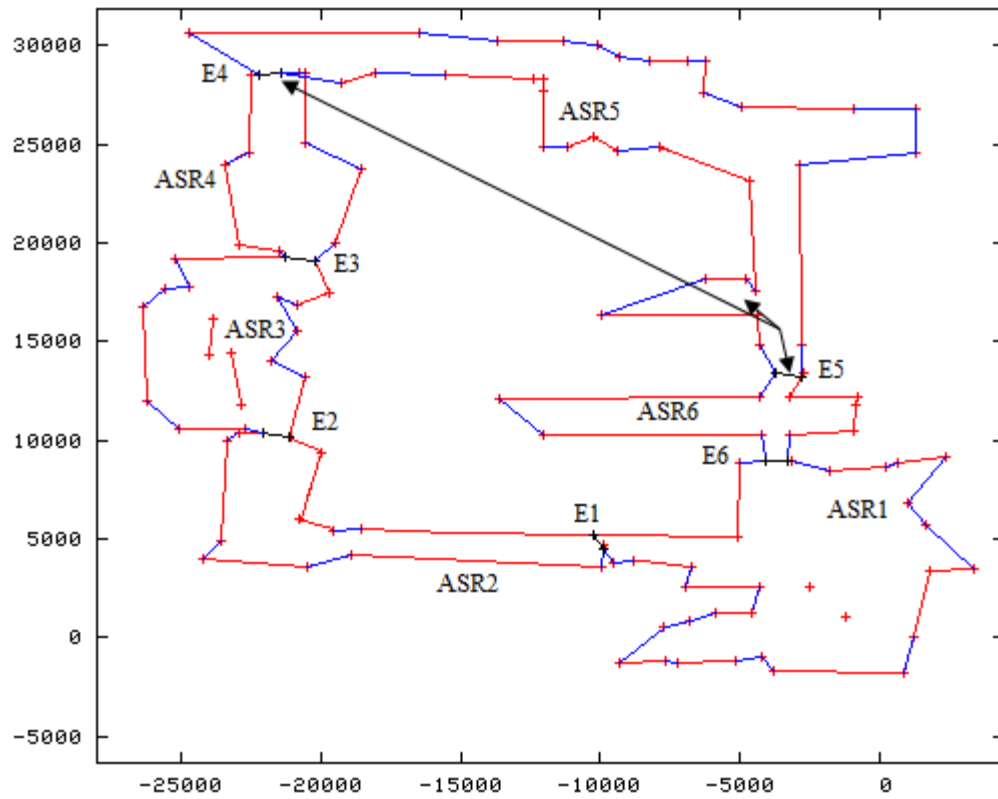


**Fig. 4.45:** Strategy to navigate towards E6 and E5

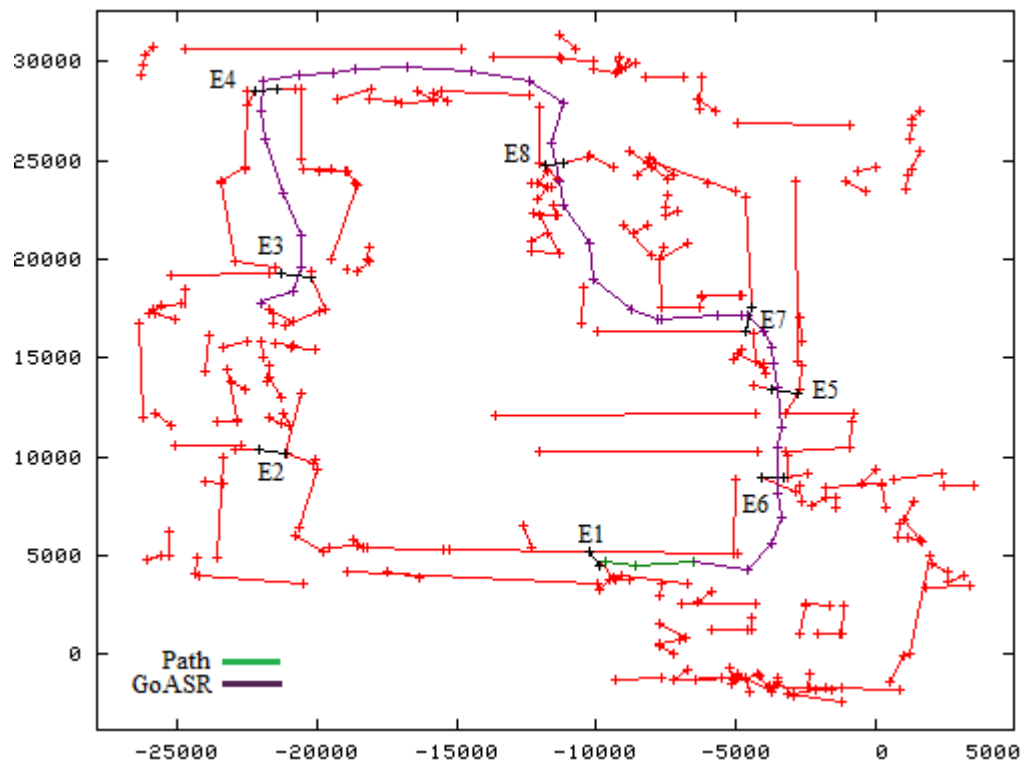


**Fig. 4.46:** MFIS as the robot travels from ASR1 to ASR5 via E6 and E5 (in that order)

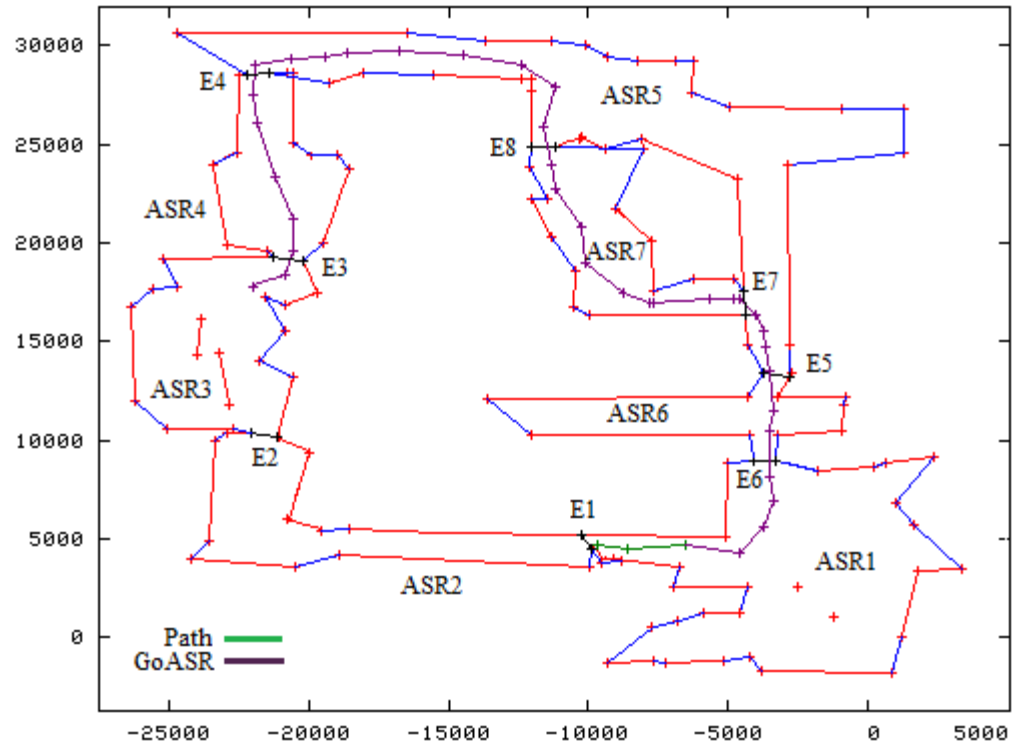




**Fig. 4.47:** Choosing new exit to go to ASR3



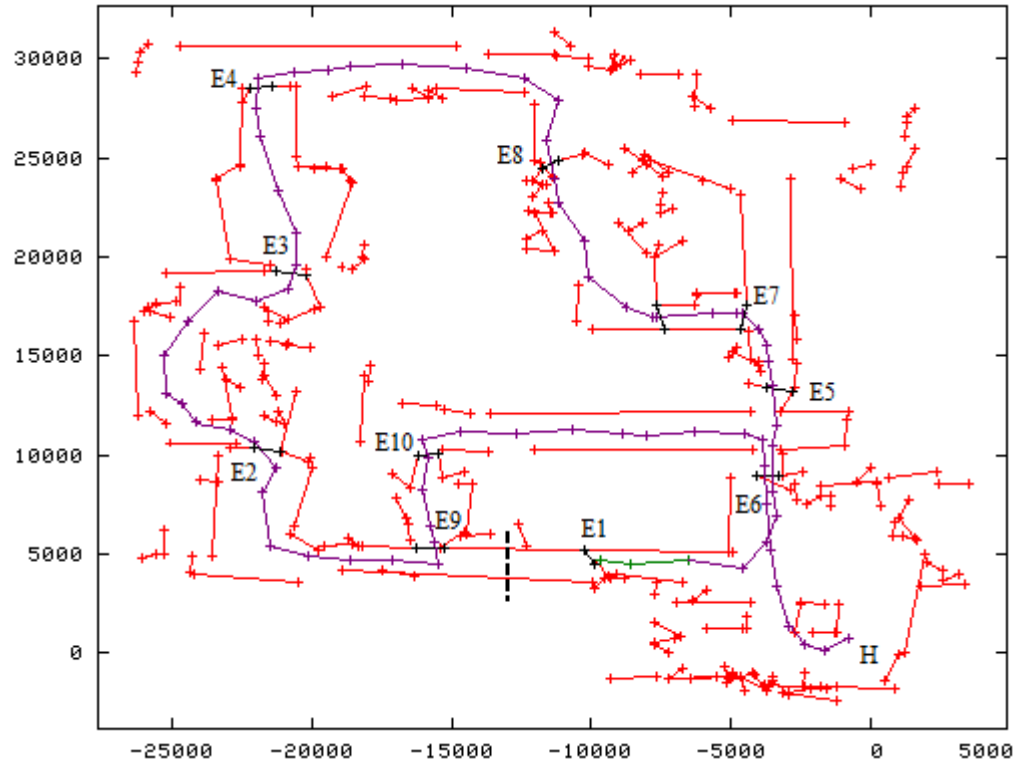
**Fig. 4.48:** The MFIS updated with E7 and E8



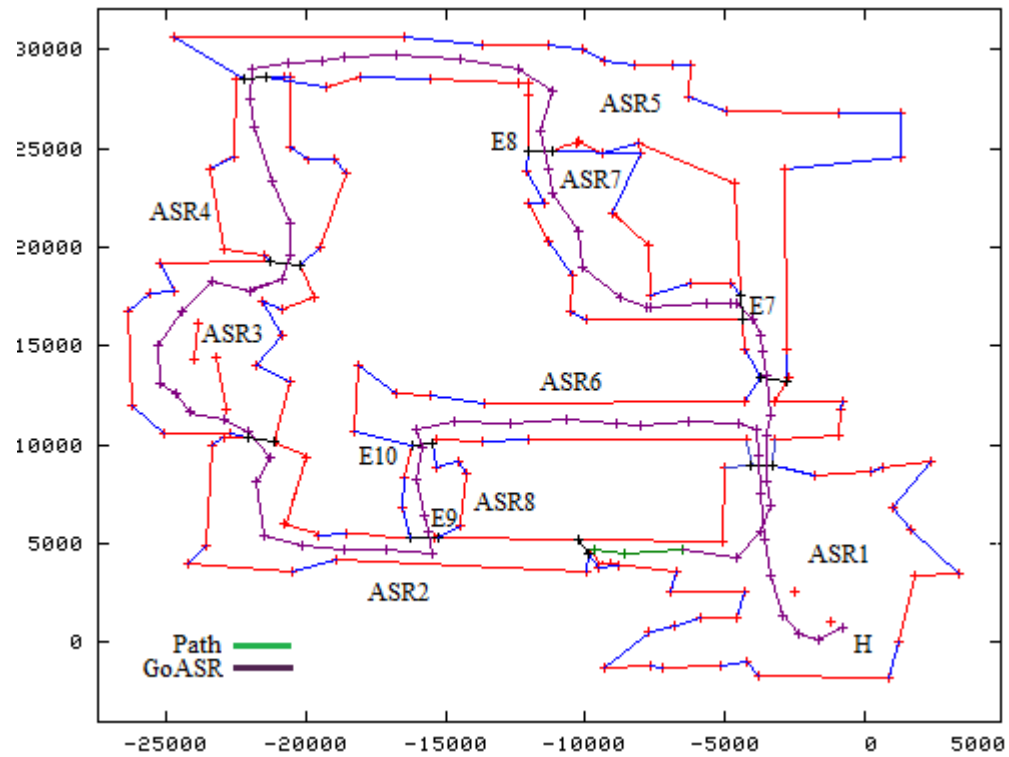
**Fig. 4.49:** New ASR7 and its location inside the network of ASR

#### 4.4.3 Home finding when known route is blocked

From here (ASR3) the robot is instructed to go home. So the optimal route is to continue via E2 then E1. However another physical block is placed so the robot crosses E2 but not able to cross E1. The robot again chooses to cross a new exit instead of going backward (E3-E4-E5-E6-Home) (Fig. 4.50 and Fig 4.51).



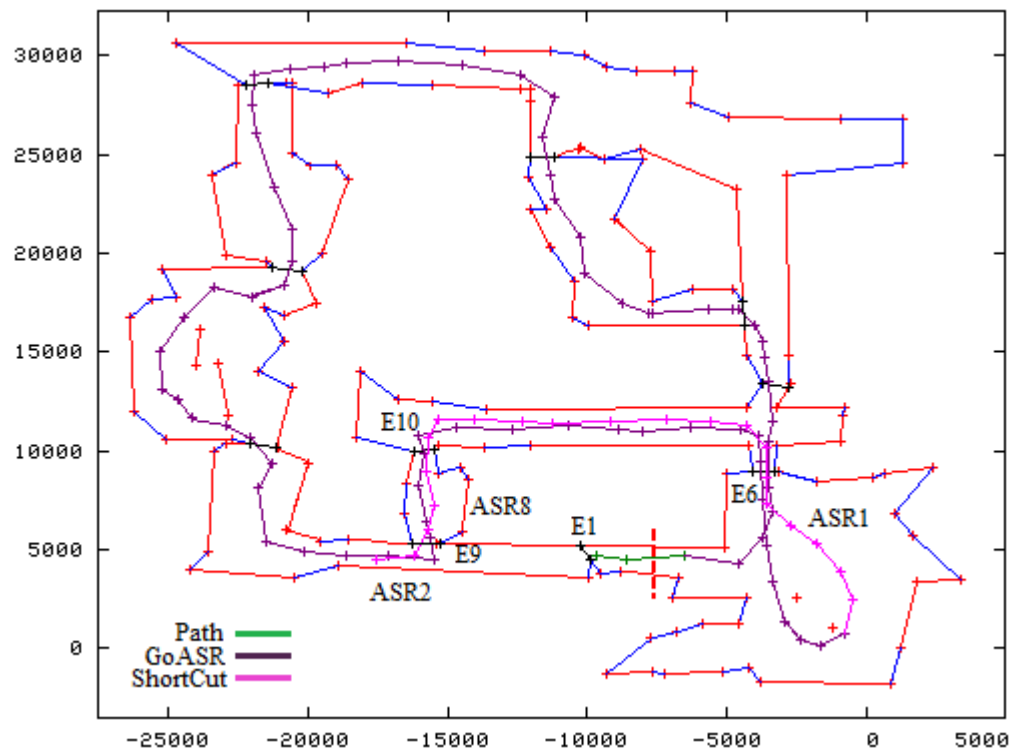
**Fig. 4.50:** Locations of new found exits E9 and E10 in the MFIS. Dotted line is where the physical barrier is put up so the robot cannot choose to re-enter ASR1 via E1



**Fig. 4.51:** Network of ASR depicting the new ASR8

#### 4.4.4 Novel short-cutting

To demonstrate the robot deciding to use a short-cut, the robot is instructed to go to ASR2 from home. However a barrier is put up to block the way out via E1. The robot is forced to exit ASR1 via E6 and hence enter into ASR6. Given the connection between ASR6 to ASR8 and ASR8 to ASR2, the robot decided to utilise this information to achieve the task. Fig. 4.52 shows the results.



**Fig. 4.52:** Utilizing new connection on the network (ASR8) to reach ASR2 from home when known exit E1 is inaccessible. Dotted line is where the physical barrier is put up

## **4.5 Conclusion**

This chapter presented three experiments which show the robot's performance in building then use an imprecise map to perform cognitive spatial tasks like home finding, navigating to different ASRs, finding alternative routes when known route is blocked and short cut discovery. The next chapter concludes the thesis.

# 5

---

## Conclusion & Future Work

---

This thesis begins with a question: If a cognitive map is an inexact map, can a robot compute one too? The answer is yes and this constitutes the most significant contribution of this work: the discovery of a mapping algorithm that does not need to correct errors due to sensors in order to compute a global map.

The success of the algorithm is attributed to the use of an intermediate buffer for identifying landmarks and to update the MFIS using information in the MFIS itself. The former meant that any errors generated would be the result of the last move rather than an accumulation of errors since the start of the journey. The latter meant that the MFIS captures only the relative positions of new surfaces to the nearest landmarks used in locating them. Thus, the absolute displacement errors that one gets in successive views

which hamper robotic researchers' efforts in computing a map are simply ignored in the computation. Loop closing is done by association rather than by correction.

While it is pleasing and motivating given the roboticists' claim that without error correction, the map computed will be distorted, it is even more exciting that the mapping process might shed light on how human cognitive mapping process work. From this cognitive standpoint, what is important is that we now have an algorithm that computes an inexact map by attending to recognizable surfaces (referred to as landmark surfaces) in successive views rather than being dependent on continuous tracking of one's position and orientation in the environment. Furthermore, it does not require continuous updating of the map as long as there are some overlapping surfaces between views. Given that humans are good at object recognition and unlikely to continuously update their maps; I argue that this algorithm is more cognitively plausible.

Developing this approach also reveals an interesting gap in Yeap's theory. The network of ASRs computed from the MFIS functions as a network of places rather than a network of ASRs. If so, this raises the question of where and when ASRs are computed. Does it really exist? From a cognitive standpoint, the current algorithm has two serious limitations – it computes too detailed an MFIS and that the MFIS covers the entire test environment. One's cognitive map is not detailed and yet the MFIS computed is updated with every new surface in view. Can we compute an incomplete MFIS as well? My tests of the algorithm currently focus on showing that a stable inexact map is maintained as the robot performs some spatial tasks in its environment. Worse, the

MFIS computed is as large as the environment – this is not correct since the MFIS is meant to be for one’s immediate surroundings only.

The two questions raised above provide motivations for future work. Further research needs to be done to understand the nature of such an algorithm. In particular, how do we create dynamic MFIS where information in it is deleted as the MFIS grows in size? How do we compute inexact *and* incomplete MFIS? The algorithm needs to be evaluated against psychological theories of human spatial mapping and in particular the work of Wang & Spelke (2002) and McNamara (2003).

From a practical standpoint, this new approach could be used as a basis for robot mapping. This could lead to a new paradigm for robot mapping. However, we need to develop a more robust algorithm and this implies performing more tests in different environments and comparing its performance with those using the SLAM approach.

Finally, the approach developed here is not restricted to a robot equipped with laser and odometer sensors. It should also work well using vision, if not better, since it is easier to recognise landmarks using vision. This would also be an important future project – cognitive mapping using a robot with vision.



## REFERENCES

- Atkinson, R. C., & Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. *The psychology of learning and motivation*, 2, 89-195.
- Ahn, S., Chung, W. K., & Oh, S. R. (2007, October). Construction of hybrid visual map for indoor SLAM. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on* (pp. 1695-1701). IEEE.
- Andersen, R. A., Snyder, L. H., Bradley, D. C., & Xing, J. (1997). Multimodal representation of space in the posterior parietal cortex and its use in planning movements. *Annual review of neuroscience*, 20(1), 303-330.
- Angeli, A., Doncieux, S., Meyer, J. A., & Filliat, D. (2008, September). Incremental vision-based topological slam. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* (pp. 1031-1036). IEEE.
- Angeli, A., Doncieux, S., Meyer, J. A., & Filliat, D. (2008b, May). Real-time visual loop-closure detection. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* (pp. 1842-1847). IEEE.
- Arras, K. O. (2003). *Feature-based robot navigation in known and unknown environments* (Doctoral dissertation, Swiss Federal Institute of Technology Lausanne).
- Arras, K. O., & Siegwart, R. Y. (1998). Feature extraction and scene interpretation for map-based navigation and map building. In *Intelligent Systems & Advanced Manufacturing* (pp. 42-53). International Society for Optics and Photonics.
- Aulinas, J., Petillot, Y. R., Salvi, J., & Lladó, X. (2008, October). The SLAM problem: a survey. In *CCIA* (pp. 363-371).
- Baddeley, A. (2003). Working memory: Looking back and looking forward. *Nature Reviews Neuroscience*, 4(10), 829-839.
- Bailey, T., & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part II. *Robotics & Automation Magazine, IEEE*, 13(3), 108-117.
- Benhamou, S. (1998). Place navigation in mammals: a configuration-based model. *Animal Cognition*, 1(1), 55-63.
- Borges, G. A., & Aldon, M. J. (2000). A split-and-merge segmentation algorithm for line extraction in 2d range images. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on* (Vol. 1, pp. 441-444). IEEE.

- Burgess, N., & O'Keefe, J. (2003). Hippocampus: spatial models. *The Handbook of Brain Theory and Neural Networks*, M. Arbib, ed., The MIT Press, Cambridge, MA, 539-543.
- Burgess, N., Recce, M., & O'Keefe, J. (1994). A model of hippocampal function. *Neural networks*, 7(6), 1065-1081.
- Carroll, S. M., Press, W. H., & Turner, E. L. (1992). The cosmological constant. *Annual Review of Astronomy and Astrophysics*, 30(1), 499-542.
- Castellanos, J. A., & Tardós, J. D. (1996). Laser-based segmentation and localization for a mobile robot. *Robotics and manufacturing: recent trends in research and applications*, 6, 101-109.
- Castellanos, J. A., & Tardos, J. D. (2000). *Mobile robot localization and map building: A multisensor fusion approach*. Kluwer academic publishers.
- Chatila, R., & Laumond, J. (1985, March). Position referencing and consistent world modeling for mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on* (Vol. 2, pp. 138-145). IEEE.
- Cheng, K., & Newcombe, N. S. (2005). Is there a geometric module for spatial orientation? Squaring theory and evidence. *Psychonomic bulletin & review*, 12(1), 1-23.
- Chong, K. S., & Kleeman, L. (1999). Feature-based mapping in real, large scale environments using an ultrasonic array. *The International Journal of Robotics Research*, 18(1), 3-19.
- Choset, H., & Nagatani, K. (2001). Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *Robotics and Automation, IEEE Transactions on*, 17(2), 125-137.
- Chown, E., & Boots, B. (2008). Learning cognitive maps: Finding useful structure in an uncertain world. In *Robotics and Cognitive Approaches to Spatial Mapping* (pp. 215-236). Springer Berlin Heidelberg.
- Chown, E., Kaplan, S., & Kortenkamp, D. (1995). Prototypes, location, and associative networks (PLAN): Towards a unified theory of cognitive mapping. *Cognitive Science*, 19(1), 1-51.
- Cummins, M., & Newman, P. (2007, April). Probabilistic appearance based navigation and loop closing. In *Robotics and automation, 2007 IEEE international conference on* (pp. 2042-2048). IEEE.
- Cummins, M., & Newman, P. (2008, May). Accelerated appearance-only SLAM.

- In *Robotics and automation, 2008. ICRA 2008. IEEE international conference on* (pp. 1828-1833). IEEE.
- Cummins, M., & Newman, P. (2011). Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 30(9), 1100-1123.
- Cuperlier, N., Quoy, M., & Gaussier, P. (2006, January). Navigation and planning in an unknown environment using vision and a cognitive map. In *European Robotics Symposium 2006* (pp. 129-142). Springer Berlin Heidelberg.
- Cuperlier, N., Quoy, M., Giovannangeli, C., Gaussier, P., & Laroque, P. (2006). Transition cells for navigation and planning in an unknown environment. In *From Animals to Animats 9* (pp. 286-297). Springer Berlin Heidelberg.
- Diosi, A., Taylor, G., & Kleeman, L. (2005, April). Interactive SLAM using laser and advanced sonar. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on* (pp. 1103-1108). IEEE.
- Dissanayake, M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *Robotics and Automation, IEEE Transactions on*, 17(3), 229-241.
- Doucet, A., De Freitas, N., Murphy, K., & Russell, S. (2000, June). Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence* (pp. 176-183). Morgan Kaufmann Publishers Inc.
- Downs, R. M., & Stea, D. (1973). Cognitive maps and spatial behavior: Process and products. *Image and Environment*. Chicago: Aldine, 8-26.
- Durrant-Whyte, H. F. (1987). *Integration, coordination and control of multi-sensor robot systems*. Kluwer Academic Publishers.
- Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: part I. *Robotics & Automation Magazine, IEEE*, 13(2), 99-110.
- Elfes, A. (1987). Sonar-based real-world mapping and navigation. *Robotics and Automation, IEEE Journal of*, 3(3), 249-265.
- Engelson, S. P., & McDermott, D. V. (1992, May). Error correction in mobile robot map learning. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on* (pp. 2555-2560). IEEE.
- Fabrizi, E., & Saffiotti, A. (2000). Extracting topology-based maps from gridmaps. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International*

- Conference on* (Vol. 3, pp. 2972-2978). IEEE.
- Filliat, D. (2008, September). Interactive learning of visual topological navigation. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* (pp. 248-254). IEEE.
- Filliat, D., & Meyer, J. A. (2003). Map-based navigation in mobile robots: I. a review of localization strategies. *Cognitive Systems Research*, 4(4), 243-282.
- Franz, M. O., Scholkopf, B., Mallot, H. A. & Bulthoff (1998) Learning view graphs for robot navigation, *Autonomous Robots*, 5:111-125
- Friedman, A. (2005). Examining egocentric and allocentric frames of reference in virtual space systems.
- Gallistel, C. R. (1989). Animal cognition: The representation of space, time and number. *Annual review of psychology*, 40(1), 155-189.
- Gallistel, C. R. (1990). *The organization of learning*. The MIT Press.
- Gerstner, W., & Kistler, W. M. (2002). Mathematical formulations of Hebbian learning. *Biological cybernetics*, 87(5-6), 404-415.
- Gibson, J. J. (1966). *The senses considered as perceptual systems*. Boston: Houghton Mifflin.
- Giovannangeli, C., Gaussier, P., & Banquet, J. P. (2006). Robustness of visual place cells in dynamic indoor and outdoor environment. *International Journal of Advanced Robotic Systems*, 3(2), 115-124.
- Golledge, R. G., & Garling, T. (2003). Cognitive maps and urban travel. In D. A. Hensher, K. J. Buttom, K. E. Haynes & P. R. Stopher (Eds), *Handbook of Transport Geography and Spatial Systems*. Oxford: Elsevier.
- Grush, R. (2000). Self, World and Space: The Meaning and Mechanisms of Ego-and Allocentric Spatial Representation. *Brain and Mind*, 1(1), 59-92.
- Guivant, J., Nebot, E., & Baiker, S. (2000). Autonomous navigation and map building using laser range sensors in outdoor applications. *Journal of robotic systems*, 17(10), 565-583.
- Habib, M. K., & Yuta, S. I. (1993). Map representation of a large in-door environment with path planning and navigation abilities for an autonomous mobile robot with its implementation on a real robot. *Automation in construction*, 2(2), 155-179.
- Hafner, V. V. (2005). Cognitive maps in rats and robots. *Adaptive Behaviour*, 13(2), 87-96.

- Hafner, V. V. (2008). Robots as Tools for Modelling Navigation Skills—A Neural Cognitive Map Approach. In *Robotics and cognitive approaches to spatial mapping* (pp. 315-324). Springer Berlin Heidelberg.
- Hirtle, S. C., & Jonides, J. (1985). Evidence of hierarchies in cognitive maps. *Memory & Cognition*, 13(3), 208-217.
- Holmes, M. C., & Sholl, M. J. (2005). Allocentric coding of object-to-object relations in overlearned and novel environments. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 31, 1069–1087.
- Iyengar, S. S., & Elfes, A. (1991). Autonomous Mobile Robots: *Perception*. IEEE Computer Society Press.
- Jefferies, M. E. (1999). *Cognitive Maps: Understanding How Local Environments are Computed* (PhD Thesis, University of Otago).
- Jefferies, M. E., & Yeap, W. K. (Eds.). (2008). *Robotics and cognitive approaches to spatial mapping* (Vol. 38). Springer.
- Jefferies, M. E., Baker, J., & Weng, W. (2008). Robot cognitive mapping—A role for a global metric map in a cognitive mapping process. In *Robotics and cognitive approaches to spatial mapping* (pp. 265-279). Springer Berlin Heidelberg.
- Jefferies, M. E., Cree, M., Mayo, M., & Baker, J. T. (2004). Using 2D and 3D landmarks to solve the correspondence problem in cognitive robot mapping. In *Spatial Cognition IV. Reasoning, Action, Interaction* (pp. 434-454). Springer Berlin Heidelberg.
- Jefferies, M. E., Yeap, W. K., Smith, L., & Ferguson, D. (2001). Building a map for robot navigation using a theory of cognitive maps. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Application* (pp. 348-353).
- Jensfelt, P., & Christensen, H. (1998). Laser based position acquisition and tracking in an indoor environment. In *International Symposium on Robotics and Automation- ISRA* (Vol. 98).
- Jensfelt, P., Kragic, D., Folkesson, J., & Bjorkman, M. (2006, May). A framework for vision based bearing only 3D SLAM. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on* (pp. 1944-1950). IEEE.
- Kaplan, S. (1973). Cognitive maps in perception and thought. *Image and environment: Cognitive mapping and spatial behavior*, 63-78.
- Kempler, R., Gerstner, W., & Van Hemmen, J. L. (1999). Hebbian learning and spiking neurons. *Physical Review E*, 59(4), 4498.

- Klatzky, R. L. (1998, January). Allocentric and egocentric spatial representations: Definitions, distinctions, and interconnections. In *Spatial cognition* (pp. 1-17). Springer Berlin Heidelberg.
- Kong, F., Chen, Y., Xie, J., Zhang, G., & Zhou, Z. (2006, June). Mobile robot localization based on extended kalman filter. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on* (Vol. 2, pp. 9242-9246). IEEE.
- Kortenkamp, D. (1992). Applying computational theories of cognitive mapping to mobile robots. *Ann Arbor, 1001*, 48109.
- Kortenkamp, D. M. (1993). *Cognitive maps for mobile robots: A representation for mapping and navigation* (PhD Thesis, University of Michigan).
- Kortenkamp, D., & Weymouth, T. (1994, October). Topological mapping for mobile robots using a combination of sonar and vision sensing. In *AAAI* (Vol. 94, pp. 979-984).
- Kortenkamp, D., Nourbakhsh, I., & Hinkle, D. (1997). The 1996 AAAI mobile robot competition and exhibition. *AI magazine*, 18(1), 25.
- Kretzschmar, H., Stachniss, C., & Grisetti, G. (2011, September). Efficient information-theoretic graph pruning for graph-based SLAM with laser range finders. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on* (pp. 865-871). IEEE.
- Krishnan, A. K., Krishna, M., & Achar, S. (2010, May). Image based exploration for indoor environments using local features. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1* (pp. 1499-1500). International Foundation for Autonomous Agents and Multiagent Systems.
- Kuipers, B. (1978). Modeling spatial knowledge. *Cognitive science*, 2(2), 129-153.
- Kuipers, B. (1983). The cognitive map: Could it have been any other way? In *Spatial orientation* (pp. 345-359). Springer US.
- Kuipers, B. J. (2000) The spatial semantic hierarchy. *Artificial Intelligence*, 19:191-233
- Kuipers, B. (2001, February). The skeleton in the cognitive map: A computational hypothesis. In *Proceedings of the Third International Symposium* (pp. 10-11).
- Kuipers, B. (2008). An intellectual history of the Spatial Semantic Hierarchy. In *Robotics and cognitive approaches to spatial mapping* (pp. 243-264). Springer Berlin Heidelberg.

- Kuipers, B., & Byun, Y. T. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and autonomous systems*, 8(1), 47-63.
- Kuipers, B. J., & Levitt, T. S. (1988). Navigation and mapping in large scale space. *AI magazine*, 9(2), 25.
- Kuipers, B., Modayil, J., Beeson, P., MacMahon, M., & Savelli, F. (2004, April). Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on* (Vol. 5, pp. 4845-4851). IEEE.
- Kuipers, B., Tecuci, D. G., & Stankiewicz, B. J. (2003). The Skeleton In The Cognitive Map A Computational and Empirical Exploration. *Environment and Behavior*, 35(1), 81-106.
- Lee, W. Y. (1996). *Spatial semantic hierarchy for a physical mobile robot* (PhD Thesis, University of Texas).
- Leonard, J. J., & Durrant-Whyte, H. F. (1992). *Directed sonar sensing for mobile robot navigation* (Vol. 448). Dordrecht: Kluwer Academic Publishers.
- Leonard, J. J., & Durrant-Whyte, H. F. (1991, November). Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems' 91. Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on* (pp. 1442-1447). IEEE
- Levenick, J. R. (1985). *Knowledge representation and intelligent systems: from semantic networks to cognitive maps* (Unpublished doctoral dissertation, The University of Michigan), Ann Harbor.
- Levenick, J. R. (1991). NAPS: A connectionist implementation of cognitive maps. *Connection Science*, 3(2), 107-126.
- Levitt, T. S., & Lawton, D. T. (1990). Qualitative navigation for mobile robots. *Artificial intelligence*, 44(3), 305-360.
- Lisien, B., Morales, D., Silver, D., Kantor, G., Rekleitis, I., & Choset, H. (2003, October). Hierarchical simultaneous localization and mapping. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on* (Vol. 1, pp. 448-453). IEEE.
- Loomis, J. M., & Beall, A. C. (1998). Visually controlled locomotion: Its dependence on optic flow, three-dimensional space perception, and cognition. *Ecological Psychology*, 10(3-4), 271-285.

- Lynch, K. (1960). *The image of the city* (Vol. 11). MIT Press.
- Maddern, W., Glover, A., Milford, M., & Wyeth, G. (2009). Augmenting RatSLAM using FAB-MAP-based visual data association. In *Proceedings of Australasian Conference on Robotics and Automation 2009*. Australian Robotics and Automation Association Inc.
- Maddern, W., Milford, M., & Wyeth, G. (2012). CAT-SLAM: probabilistic localisation and mapping using a continuous appearance-based trajectory. *The International Journal of Robotics Research*, 31(4), 429-451.
- Majumder, S., Durrant-Whyte, H., Thrun, S., & de Battista, M. (2000). An approximate Bayesian method for simultaneous localisation and mapping. *IEEE Trans. Automatic Control*, 45(3), 477-482.
- Marr, D., & Vision, A. (1982). A computational investigation into the human representation and processing of visual information. *WH San Francisco: Freeman and Company*.
- Martinelli, A., Svensson, A., Tomatis, N., & Siegwart, R. (2004, July). SLAM based on quantities invariant of the robot's configuration. In *IFAC Symposium on Intelligent Autonomous Vehicles*.
- Mataric, M. J. (1991) Navigating with a rat brain: A neurobiologically-inspired model for robot spatial representation. In *Proceedings of the first international conference on simulation of adaptive behaviour on From Animals to Animats*, Meyer, J. A & Wilson, S. W (Eds), MIT Press, Cambridge, MA
- McNamara, T. P., Hardy, J. K., & Hirtle, S. C. (1989). Subjective hierarchies in spatial memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15(2), 211.
- McNamara, T. P., Rump, B., & Werner, S. (2003). Egocentric and geocentric frames of reference in memory of large-scale space. *Psychonomic Bulletin & Review*, 10(3), 589-595.
- McNamara, T. P. (2003). How are the locations of objects in the environment represented in memory? In *Spatial cognition III* (pp. 174-191). Springer Berlin Heidelberg.
- Milford, M. J., & Wyeth, G. F. (2008). Mapping a suburb with a single camera using a biologically inspired SLAM system. *Robotics, IEEE Transactions on*, 24(5), 1038-1053.
- Milford, M. J., Wyeth, G. F., & Prasser, D. (2004, April). RatSLAM: a hippocampal model for simultaneous localization and mapping. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on* (Vol. 1, pp.



403-408). IEEE.

- Milford, M., Schulz, R., Prasser, D., Wyeth, G., & Wiles, J. (2007). Learning spatial concepts from RatSLAM representations. *Robotics and Autonomous Systems*, 55(5), 403-410.
- Milford, M., Wyeth, G., & Prasser, D. (2004). Simultaneous localisation and mapping from natural landmarks using RatSLAM. In *2004 Australasian Conference on Robotics and Automation*. Australian Robotics and Automation Association Inc.
- Modayil, J. (2013, June). Two Perspectives on Learning Rich Representations from Robot Experience. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*.
- Moravec, H. P., & Elfes, A. (1985, March). High resolution maps from wide angle sonar. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on* (Vol. 2, pp. 116-121). IEEE.
- Moravec, H. P. (1988). Sensor fusion in certainty grids for mobile robots. *AI magazine*, 9(2), 61.
- Mou, W., & McNamara, T. P. (2002). Intrinsic frames of reference in spatial memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 28(1), 162.
- Mou, W., McNamara, T. P., Valiquette, C. M., & Rump, B. (2004). Allocentric and egocentric updating of spatial memories. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 30(1), 142.
- Murphy, K. P. (1999, November). Bayesian Map Learning in Dynamic Environments. In *NIPS* (pp. 1015-1021).
- Nadel, L. (1991). The hippocampus and space revisited. *Hippocampus*, 1(3), 221-229.
- Newman, P., Cole, D., & Ho, K. (2006, May). Outdoor SLAM using visual appearance and laser ranging. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on* (pp. 1180-1187). IEEE.
- Newman, P., Leonard, J., Tardos, J. D., & Neira, J. (2002). Explore and return: Experimental validation of real-time concurrent mapping and localization. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on* (Vol. 2, pp. 1802-1809). IEEE.
- Nilsson, N. J. (1969). *A mobile automaton: An application of artificial intelligence techniques*. SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER.

- Nüchter, A., Lingemann, K., Hertzberg, J., & Surmann, H. (2007). 6D SLAM—3D mapping outdoor environments. *Journal of Field Robotics*, 24(8-9), 699-722.
- O'Keefe, J. (1990). A computational theory of the hippocampal cognitive map. *Progress in brain research*, 83, 301-312.
- O'Keefe, J. (1991). An allocentric spatial model for the hippocampal cognitive map. *Hippocampus*, 1(3), 230-235.
- O'keefe, J., & Nadel, L. (1978). *The hippocampus as a cognitive map* (Vol. 3, pp. 483-484). Oxford: Clarendon Press.
- Pfister, S. T., Roumeliotis, S. I., & Burdick, J. W. (2003, September). Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on* (Vol. 1, pp. 1304-1311). IEEE.
- Piaget, J., & Inhelder, B. (1969). *The psychology of the child*. Basic Books.
- Pierce, D., & Kuipers, B. (1994, October). Learning to explore and build maps. In *AAAI* (Vol. 94, pp. 1264-1271).
- Raaijmakers, J. G., & Shiffrin, R. M. (1981). Search of associative memory. *Psychological review*, 88(2), 93.
- Redhead, E. S., & Hamilton, D. A. (2007). Interaction between locale and taxon strategies in human spatial learning. *Learning and Motivation*, 38(3), 262-283.
- Reitman, J. S., & Rueter, H. H. (1980). Organization revealed by recall orders and confirmed by pauses. *Cognitive Psychology*, 12(4), 554-581.
- Remolina, E. (2001). *A logical account of causal and topological maps* (Doctoral dissertation, The University of Texas at Austin).
- Remolina, E., & Kuipers, B. (2004). Towards a general theory of topological maps. *Artificial Intelligence*, 152(1), 47-104.
- Ruggiero, G., Iachini, T., Ruotolo, F., & Senese, V.P. (2011). Spatial memory: the role of egocentric and allocentric frames of reference. In J.B. Thomas (Ed.), *Spatial Memory: Visuospatial Processes, Cognitive Performance and Developmental Effects* (Chapter 2, pp. 51-75). NY: Nova Science Publishers.
- Saleiro, M., Rodrigues, J. M. F., & du Buf, J. M. H. (2010). *Cognitive robotics: a new approach to simultaneous localisation and mapping* (Unpublished short paper, University of Algarve)

- Santosh, D., Achar, S., & Jawahar, C. V. (2008, May). Autonomous image-based exploration for mobile robot navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* (pp. 2717-2722). IEEE.
- Schmidt, T., & Lee, E. Y. (2006). Spatial memory organized by environmental geometry. *Spatial Cognition and Computation*, 6(4), 347-369.
- Se, S., Lowe, D., & Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The international Journal of robotics Research*, 21(8), 735-758.
- Shevelev, I., Kamenkovich, V. M. & Sharaev, G. A. (2003). The roles of lines and corners of geometric figures in recognition. *Acta Neurobiol Exp* 2003, 63: 361-368
- Sholl, M. J. (2001). The role of a self-reference system in spatial navigation. In D. Montello (Ed.), *Spatial Information Theory: Foundations of geographical information science* (pp. 217-232). Berlin, Germany: Springer-Verlag.
- Siegel, A. W., & White, S. H. (1975). The development of spatial representations of large-scale environments. *Advances in child development and behavior*, 10, 9-55.
- Sim, R., & Little, J. J. (2006, October). Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on* (pp. 2082-2089). IEEE.
- Taylor, R. M., & Probert, P. J. (1996, April). Range finding and feature extraction by segmentation of images for mobile robot navigation. In *Robotics and Automation, 1996. Proceedings, 1996 IEEE International Conference on* (Vol. 1, pp. 95-100). IEEE.
- Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1), 21-71.
- Thrun, S. (2000). Probabilistic algorithms in robotics. *Ai Magazine*, 21(4), 93.
- Thrun, S. (2002). Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1-35.
- Thrun, S. (2003). Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15(2), 111-127.
- Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. MIT press.
- Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological review*, 55(4), 189.

- Tolman, E. C. (1949). There is more than one kind of learning. *Psychological review*, 56(3), 144.
- Tomatis, N., Nourbakhsh, I., & Siegwart, R. (2003). Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Robotics and Autonomous systems*, 44(1), 3-14.
- Tomatis, N., Nourbakhsh, I., Arras, K., & Siegwart, R. (2001). A hybrid approach for robust and precise mobile robot navigation with compact environment modeling. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on* (Vol. 2, pp. 1111-1116). IEEE.
- Touretzky, D. S., & Redish, A. D. (1996). Theory of rodent navigation based on interacting representations of space. *Hippocampus*, 6(3), 247-270.
- Vandorpe, J., Van Brussel, H., & Xu, H. (1996). Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2D range finder. In *Robotics and Automation, 1996. Proceedings, 1996 IEEE International Conference on* (Vol. 1, pp. 901-908). IEEE.
- Wang, R. F., & Spelke, E. S. (2000). Updating egocentric representations in human navigation. *Cognition*, 77(3), 215-250.
- Wang, R. F., & Spelke, E. S. (2002). Human spatial representation: Insights from animals. *Trends in cognitive sciences*, 6(9), 376-382.
- Wendel, A., & Bischof, H. (2013). Visual Localization for Micro Aerial Vehicles in Urban Outdoor Environments. In *Advanced Topics in Computer Vision* (pp. 181-214). Springer London.
- Werner, S., Krieg-Brückner, B., Mallot, H. A., Schweizer, K., & Freksa, C. (1997). Spatial Cognition: The Role of Landmark, Route, and Survey Knowledge in Human and Robot Navigation1. In *Informatik'97 Informatik als Innovationsmotor* (pp. 41-50). Springer Berlin Heidelberg.
- Wong, C. K. (2008). *Cognitive Inspired Mapping by an Autonomous Mobile Robot* (PhD Thesis, Auckland University of Technology).
- Yeap, W. K. (1988). Towards a computational theory of cognitive maps. *Artificial Intelligence* 34(3):297-360
- Yeap, W. K. (2007). From Spatial Perception to Cognitive Mapping: How is the Flow of Information Controlled? In *AAAI Spring Symposium: Control Mechanisms for Spatial Knowledge Processing in Cognitive/ Intelligent Systems* (pp. 59-61)
- Yeap, W. K., & Jefferies, M. E. (1999). Computing a representation of the local

- environment. *Artificial Intelligence*, 107(2), 265-301.
- Yeap, W. K., Jefferies, M. E., & Naylor, P. S. (1991, August). An MFIS for computing a raw cognitive map. In *IJCAI* (pp. 373-380).
- Zhang, L., & Ghosh, B. K. (2000). Line segment based map building and localization using 2D laser rangefinder. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on* (Vol. 3, pp. 2538-2543). IEEE.
- Zhao, H., Chiba, M., Shibasaki, R., Shao, X., Cui, J., & Zha, H. (2008, May). SLAM in a dynamic large outdoor environment using a laser scanner. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* (pp. 1455-1462). IEEE.
- Zipser, D. (1985). A computational model of hippocampal place fields. *Behavioral neuroscience*, 99(5), 1006.