



HOKKAIDO UNIVERSITY

Title	Novel Algorithms for Multi-Robot Path Planning, Task Coordination, Mapping and Localization in Indoor Environment
Author(s)	RAVANKAR, ABHIJEET
Degree Grantor	北海道大学
Degree Name	博士(工学)
Dissertation Number	甲第12762号
Issue Date	2017-03-23
DOI	https://doi.org/10.14943/doctoral.k12762
Doc URL	https://hdl.handle.net/2115/66434
Type	doctoral thesis
File Information	Abhijeet_Ravankar.pdf



Novel Algorithms for Multi-Robot Path Planning, Task Coordination, Mapping and Localization in Indoor Environment



Abhijeet Ravankar

Graduate School of Engineering
Hokkaido University, Japan

This dissertation is submitted for the degree of
Doctor of Philosophy

Hokkaido University, Japan

February 2017

© Copyright by Abhijeet Ravankar, February 2017

All Rights Reserved.

Dedicated to my parents

Acknowledgements

I am heartily thankful to my supervisor, Professor Yukinori Kobayashi, for accepting me as a student in his lab. His encouragement, supervision and support enabled me to develop an understanding of the subject. He is always ready for discussions and support in research. He has encouraged me to develop independent thinking in research. Moreover, he has provided me with ample opportunities to grow as a researcher.

I thank Associate Professor Takanori Emaru for his guidance and help. He made sure that research is not hindered due to the lack of equipment and has always provided with all the tools and resources whenever they were needed.

I am thankful to Assistant Professor Ankit Ravankar who helped me in experiments. He is always ready for brainstorming potential ideas. Most of the ideas presented in this thesis developed from such active and constructive brainstorming.

I express my sincere gratitude to the examination committee for giving critical and important suggestions to improve the thesis work.

I would like to thank MEXT (Ministry of Education, Culture, Sports, Science and Technology) Japan, which provided me the scholarship to support my graduate studies in Japan for my masters and PhD.

I am also thankful to the staff members of Graduate school of Engineering, Hokkaido University for helping me in my day-to-day life in Japan.

Words cannot describe the support of my family. I owe my deepest gratitude towards my parents. My brothers Aprit and Ankit have been very encouraging. I am thankful to my wife Neeta for her support and understanding. Finally, I am thankful to my son Advait for bringing a lot of joy and happiness in my life.

Abstract

Indoor mobile service robots face many different problems to perform tasks autonomously. These problems become more complex when multiple robots are deployed in the environment for different services. This thesis addresses four important problems faced by multiple autonomous mobile robots in indoor environments.

First, is the problem of smooth path generation for multiple mobile robots. Careful path planning is necessary for autonomous robots to navigate from their current position to the service location. A smooth and continuous path is desired for robot motion which avoids abrupt and sharp turns. This problem is further complicated in case of multiple robots as two or more robots may have intersecting and common paths which increases the chances of a robot colliding with another robot during its operation. Multiple robots in the environment also poses problems of deadlock which must be resolved.

The second problem which frequently occurs in multi-robots scenarios is the problem of common resource (like charging point or narrow path) sharing. Service robots for cleaning and patrolling are mostly in continuous operation and require frequent recharging. Using a large number of docking points proportional to the number of robots is costly and requires large space. A manager which can resolve conflicts and allocate the resource to the most appropriate robot by considering several factors like task priority, battery power left, distance travelled, etc. is required.

Third, is the problem of autonomous task coordination and exploration by multiple mobile service robots. The problem is related to but more abstract than the first problem of smooth path generation, in which, the goal locations are given to each mobile robot. However, introducing multiple robots also introduce the problem of programming the robots to efficiently serve the region. The areas to serve in a map may vary with time. Moreover, the number of the robots available to serve may also be dynamic, in real world situations, as some of the robots may be charging, while

some may be out of order. In case of robots used for surveillance, a robot may want other robot or robots to follow itself while chasing a suspicious person, for backup. This situation is also dynamic in terms of availability of robots, selecting the nearest robot for quick response, and selecting the same path towards a particular area as taken by the previous robot. In both the cases, explicitly programming the robots is cumbersome, and demands for a simpler scheme in which multiple robots can intelligently coordinate the tasks and explore the map. Moreover, if multiple robots are serving the same area, or navigating to the same area, they must coordinate tasks to maximize efficiency.

The fourth problem is related to the third problem and relates to virtual pheromone deposition methodologies for multi-robot task coordination. Most of these approaches assumes pheromone deposition at perfect locations in the map. In reality however, it is difficult to achieve perfect localization of the robot due to errors in encoders and sensors attached to the robot, and the dynamics of the environment in which the robot operates. In real world scenarios, there is always some uncertainty associated in estimating the pose (i.e. position and orientation) of the mobile service robot. Failing to model this uncertainty would result in service robots depositing pheromones at wrong places. A leading robot in the multi-robot system might completely fail to localize itself in the environment and be lost. Other robots trailing its pheromones will end up being in entirely wrong areas of the map. This results in a ‘blind leading the blind’ scenario which reduces the efficiency of the multi-robot system.

The final problem is a fundamental problem of Simultaneous Localization and Mapping (SLAM) in autonomous mobile robots. Feature based SLAM algorithm are popular which work by matching features like lines extracted from the sensors like camera or laser range sensors (LRS). Particularly, line detection is an important problem in computer vision, graphics, and autonomous robot navigation. However, in order to achieve efficiency in path planning, path smoothing, task distribution and other tasks, a robust SLAM algorithm is indispensable.

Thesis Outline

Solutions for each of the aforementioned described problems have been presented in the separate following Chapters:

Chapter 1: This chapter presents an introduction to indoor service robots. It presents a brief introduction to the field of robotics and how mobile service robots are entering the day-to-day lives of people. The challenges faced by multiple service robots are presented along with the motivation of the work presented in this thesis. The problems with the current techniques are also discussed which forms the base for the rest of the chapters. Most importantly, this chapter describes the interdependence of the different modules of a multi-robot system which lays the foundation of the interdependence of the work in the chapters of this thesis.

Chapter 2: This chapter deals with path planning in multi-robot systems. This chapter presents a novel smooth hypocycloidal paths (SHP) for robot motion. It is integrated with collision-free and decoupled multi-robot path planning. SHP diffuses (moves points along segments) the points of sharp turns in the global path of the map into nodes which are used to generate smooth hypocycloid curves which maintain a safe clearance from the obstacles. These nodes are also used as safe points of retreat to avoid collision with other robots. The novel contributions of this work are: (1) The proposed work is the first use of hypocycloid geometry to produce smooth and continuous paths for robot motion. A mathematical analysis of SHP generation in various scenarios is discussed. (2) The proposed work is also the first to consider the case of smooth and collision-free path generation for a load carrying robot. (3) Traditionally, path smoothing and collision avoidance have been addressed as separate problems. This work proposes an integrated and decoupled collision-free multi-robot path planning. ‘Node caching’ is proposed to improve efficiency. A decoupled approach with local communication enables dynamically changing the paths of robots without configuring the paths of all the robots. (4) A novel ‘multi-robot map update’ in case of dynamic obstacles in the map is proposed, in which, robots update other robots about the positions of dynamic obstacles in the map. A time-stamp feature ensures that all the robots have the most updated map. Comparison of SHP with other path smoothing techniques and experimental results in real environments confirm that SHP can generate smooth paths for robots and avoid collision with other robots through local communication.

Chapter 3: This chapter deals with common resource sharing in multi-robot systems. This chapter presents an intelligent docking station manager for multiple

mobile service robots. The manager maintains a modified queue based on a score which is calculated from the task priority and the current power level of the robot. The behaviour of the station manager in different scenarios when the docking station has empty docks and when it has none are discussed. The docking station can intelligently instruct a robot to forcefully relinquish a dock and give preference to the robots with higher task priorities and lower power levels. Emergency conditions for robots with power levels below threshold, and intelligent path allocation are also discussed.

Chapter 4: This chapter deals with the problem of task sharing and exploration in multi-robot systems. This chapter presents a novel bio-inspired hybrid communication framework that incorporates the repelling behaviour of anti-aphrodisiac pheromones and attractive behaviour of pheromones for efficient map exploration of multiple mobile service robots. The proposed communication framework presents a scheme for robots to efficiently serve large areas of map, while cooperating with each other through proper pheromone deposition. This eliminates the need of explicitly programming each service robot to serve particular areas of the map. The paths taken by robots are represented as nodes across which pheromones are deposited. This reduces the search space for tracking pheromones and reduces data size to be communicated between robots. A novel pheromone deposition model is presented which takes into account the uncertainty in the robot's position. This eliminates robots to deposit pheromones at wrong places when localization fails. The framework also integrates the pheromone signalling mechanism in landmark based Extended Kalman Filter (EKF) localization and allows the robots to capture areas or sub-areas of the map, to improve the localization. A scheme to resolve conflicts through local communication is presented. Experimental and simulation results are discussed with two cases of floor cleaning task, and surveillance task, performed by multiple robots. Results show that the proposed scheme enables multiple service robots to perform cooperative tasks intelligently without any explicit programming.

Chapter 5: This chapter builds upon the work in the previous chapter which used virtual pheromones for task coordination. This chapter presents a pheromone deposition algorithm which models the uncertainty of the robot's pose. Results from experiments in both simulated and real environments demonstrate that modeling the uncertainty in pheromone deposition is crucial, and that the pro-

posed algorithm can model the uncertainty well. The modeling proposed in this chapter, provides extra information to the trailing robot about the preceding robot. A ‘confident’ deposition concentrated in a small area indicates that the preceding robot was certain about its location at that place. Whereas, low and dispersed pheromones indicates that the preceding robot was not sure about its location while depositing pheromones. Two cases of behavioral difference of follower robot with and without the proposed method, and speed adaptation of trailing robots using the proposed method are presented.

Chapter 6: Virtual trailing based multi-robot task coordination and navigation as discussed in Chapters 4 and 5 are useful over large distances, particularly when the leader robot is not in the field of view (FoV) of the follower robot. However, there are cases when the leader robot is in the FoV of the trailing robot. In such scenarios, a more straightforward solution is to directly follow the robot. In these cases, such direct following can be incorporated with other mechanisms for performance efficiency in multi-robot systems. A robot must localize, do path planning, obstacle avoidance, and update the map if there are new features in the environment while navigating to its goal location. When two or more robots have the same goal location, it is a waste of computing if all the robots performs all these tasks individually. To this end, this chapter introduces a novel concept of ‘hitchhiking’ as a means of efficiency gains. The hitchhiker robot can save computation required for (a) path planning, (b) localization, (c) dynamic obstacle avoidance, and (d) map update. The system has an award mechanism in which the hitchhiker ‘owes’ to the driver robot which receives points. The admissible characteristics of the hitchhiker and the driver robots are discussed. Important points about when should hitchhiking be allowed and when not are also discussed.

Chapter 7: In all the previous chapters, it was assumed that multiple robots had a SLAM component. This chapter proposes an efficient algorithm for line detection from LRS data using a novel hopping-points Singular Value Decomposition (SVD) and Hough transform based algorithm, in which, SVD is applied on intermittent LRS points to accelerate the algorithm. A reverse-hop mechanism ensures that the end points of the line segments are accurately extracted. Line segments extracted from the proposed algorithm are used to form a map and subsequently LRS data points are matched with the line segments to localize the robot. The

proposed algorithm eliminates the drawbacks of point based matching algorithms like Iterative Closest Points (ICP) algorithm whose performance degrades with increasing number of points.

Chapter 8: This chapter discusses the results and their implications, concludes the thesis work, and points out the potential future works.

Table of Contents

List of Figures	xv
List of Tables	xxiii
1 Introduction	1
1.1 Background	1
1.1.1 Rise of Embedded Computing and Embedded Sensors	2
1.1.2 Rise of Service Robots	3
1.2 Multi-Robot Systems	4
1.3 Challenges Faced by Service Multi-Robot Systems	5
1.4 Motivation	8
1.4.1 Interdependence of Various Multi-Robot Components	9
1.5 Thesis Organization and Chapter Flow	10
2 SHP: Smooth Hypocycloidal Paths and Decoupled Multi-Robot Path Planning with Collision Avoidance	14
2.1 Introduction	15
2.2 Skeleton Map and Node Generation	19
2.3 Generating Smooth Hypocycloid Paths (SHP)	20
2.3.1 SHP for Different Angles	26
2.3.2 SHP Curvature, and Continuity	28
2.4 Determining Clearance Distance (δ)	31
2.5 Multi-Robot Path Planning on SHP	33
2.5.1 Node Caching and Policy Exchange	36
2.6 Dynamic Obstacles and Map Update	36
2.7 System Specifications	37
2.8 Motion Model	38

2.9	Results	40
2.9.1	Experiment 1: SHP traversal and comparison with D* path traversal	41
2.9.2	Experiment 2: Collision Avoidance on SHP nodes	41
2.9.3	Experiment 3: Avoiding Deadlock on SHP	43
2.9.4	Experiment 4: Dynamic Obstacle Avoidance on SHP	44
2.9.5	Experiment 5: SHP with different clearance threshold (δ)	47
2.9.6	Experiment 6: Obstacle Avoidance on SHP with Load Carrying Robot	49
2.10	Movies of Results	49
2.11	SHP Comparison With Other Path Planning and Smoothing Techniques	49
2.12	SHP as an Extension for Various Global Path Planners	54
2.13	Summary	57
3	Sharing Limited Charging Docks with Multiple Mobile Service Robots	58
3.1	Introduction	59
3.2	Docking Station Manager	60
3.2.1	Network and Communication Module	61
3.2.2	Command Executor Module	61
3.2.3	Power Estimator Module	61
3.2.4	Request Queue Manager Module	61
3.2.5	Docking Station Status Updater Module	64
3.2.6	Access Token Granting and Path Planning Module	64
3.2.7	Monitoring and Controlling GUI	67
3.3	Experimental Results	68
3.3.1	Experiment 1	69
3.3.2	Experiment 2	69
3.3.3	Experiment 3	69
3.3.4	Experiment 4 and 5	69
3.4	Summary	70
4	Multi-Robot Task Cooperation and Exploration Using a Bio-Inspired Hybrid Pheromone Signalling	72
4.1	Introduction	73
4.2	Pheromone and Anti Pheromone Representation	77

4.3	Node Representation of Path	79
4.4	Anti-Pheromone Mechanism for Map Exploration	80
4.4.1	Path Selection	80
4.4.2	Area Capturing and Area Selection	81
4.4.3	Pheromone Mechanism	82
4.5	Integration of Pheromone signalling With Landmark Based EKF Local- ization	82
4.5.1	Collision Avoidance and Resolving Conflict Amongst Robots . .	85
4.6	Results and Discussion	86
4.6.1	Anti-Pheromone Experiment	88
4.6.2	Pheromone Experiment	91
4.7	Summary	92
5	Robust Virtual Pheromone Trailing of Multi-Robots by Uncertainty Integration in Pheromone Deposition	94
5.1	Introduction	95
5.1.1	Main Idea of this Work	98
5.1.2	New Contributions of this Work	98
5.2	Related Works	99
5.3	A Brief Introduction to Robot State and Uncertainty Estimation Using Various Filters	100
5.3.1	Extended Kalman Filter	101
5.3.2	Unscented Kalman Filter	101
5.3.3	Particle Filter	102
5.4	Pheromone Deposition with Uncertainty Modeling	104
5.4.1	Pheromone Deposition	104
5.4.2	Pheromone Evaporation	107
5.5	Results	110
5.5.1	Results in Simulation Environment	112
5.5.2	Results in Real Environment	116
5.5.3	Results of Speed Adaptation of Trailing Robots using the Pro- posed Method	121
5.6	Summary and Discussion	125
6	Hitchhiking in Multi-Robot System	126
6.1	Introduction	127

6.2	Hitchhiking in Robots	128
6.2.1	Hitchhiking Mechanism	128
6.2.2	Four Steps of Hitchhiking	129
6.3	Hitchhiking Points	131
6.3.1	When To Hitchhike and When Not	131
6.4	Problem of ‘Driver Lost’ Scenario	132
6.5	Award Mechanism	134
6.6	Experimental Results	134
6.6.1	Experiment 1	134
6.6.2	Experiment 2	136
6.7	Summary	138
7	SLAM Based on a Novel Hopping-points SVD and Hough Transform Based Line Detection Algorithm	141
7.1	Introduction	142
7.2	Straight Line Detection from LRS data with Hough Transform	144
7.3	SVD Based Line Detection Algorithm	146
7.3.1	Straight Line Detection with SVD	147
7.3.2	Line Segment Detection Algorithm	149
7.3.3	Accelerating Line Detection Algorithm using Hopping-points SVD	151
7.3.4	Comparison with Hough Transform	155
7.4	Integration of Line Extraction Algorithm in SLAM Framework	156
7.4.1	Mapping from LRS Information	156
7.4.2	Self-Localization by Matching LRS Data with Estimated Global Map	157
7.5	Results and Discussion	161
7.5.1	Simulation Results	162
7.5.2	Experiments in Real Environment	165
7.6	Summary	168
8	Discussion, Conclusion, and Future Works	170
8.1	Discussion	170
8.2	Conclusion	173
8.3	Future Works	174
	References	177

List of Figures

1.1	Service Robots in Operation. (a) iRobot's Roomba Cleaning Robot [79]. (b) Robot at Southmead Hospital [116]. (c) Knightscope patrolling robot [89]. (d) Knightscope in operation [164].	3
1.2	Pepper robot[153, 166] in service for entertaining kids at at Shiroi Koibito Park [159] in Sapporo, Japan	4
1.3	Strong interdependence amongst the major components of a multi-robot system.	9
1.4	Components of multi-robot system and the chapters describing them. .	10
2.1	Two problems of multi-robot path planning: (1) the paths should be short, smooth and safe (i.e. far from obstacles), (2) robots must avoid collision with each other.	16
2.2	Node extraction from skeleton map. (a) Map with obstacles and free space. (b) Corresponding skeleton map. Nearby nodes are clustered to form a single node.	20
2.3	Geometry of hypocycloid curves shown in red. (a) Deltoid curve: $R_L = 3 \times r_s, \xi = \frac{1}{3}$. (b) Astroid curve: $R_L = 4 \times r_s, \xi = \frac{1}{4}$. (c) Voronoi paths keep a safe distance from obstacles but robot encounters sharp turns at O from L_C or L_A to L_B	20
2.4	Smooth path generation. (a) with deltoid or lower half of astroid, (b) astroid curves at cross-road map, (c) a portion of hypocycloid to generate curve at corner.	21
2.5	Smooth path generation when clearance thresholds are not same. (a) & (b) Lower astroid with $\delta_1 \neq \delta_2$. The dotted lines and points A' and B' are auxiliary.	23

2.6	SHP generation with different angles. The first segment of the curve in blue is taken to generate SHP, whereas other curve segments in black and red are ignored. The red curve segment shows the portion of the hypocycloid which fails to close on the start point as the angle does not perfectly divide 360 degrees. (a) $\theta = 15.3^\circ$. (b) $\theta = 31.4^\circ$. (c) $\theta = 64^\circ$. (d) $\theta = 78^\circ$. (e) $\theta = 111.9^\circ$. (f) $\theta = 150^\circ$	25
2.7	Applying transition curves for smoothing joints between straight segments and curve. (a) A transition curve (blue) gradually decreasing curvature from infinite (on line) to that of SHP (red). (b) Employing transition curves with SHP in environment to get smooth path in red. .	27
2.8	Determining clearance threshold δ in case of load carrying robot. (a) No load. (b) Centered load. (c) Un-centered load (robot turning right). (d) Un-centered load (robot turning left).	28
2.9	Robots used in the experiments. (a) Pioneer P3DX. (b) Kobuki Turtlebot. (c) Motion model of two wheel differential drive robots.	38
2.10	Test environment with dimensions, grip-map, SHP nodes, and PRM paths. (a) Environment (front-view). (b) Environment (side-view). (c) Dimensions of the map. (d) Corresponding grid map. (e) SHP path with nodes. (f) PRM result on grid map.	39
2.11	Experiment 1: SHP vs D* path traversal. (a) D* path from start node 'c' to goal node 's'. Intensity of the color represents proximity to the goal. (b) Dotted green line shows the traversed SHP path, and dotted blue shows the traversed D* path. D* path is very close to the obstacles, whereas SHP is not. <i>Please refer to Movie No. 1, 2, and 3 in Table 2.3.</i>	41
2.12	Experiment 2: (Case A) Collision Avoidance on SHP nodes. (a) Initial configuration of robots. (b) Collision detection. (c) Low priority Turtlebot retreats to cached safe node 'l' (Fig.2.10e). (d,e) P3DX continues towards its goal. (f,g) Turtlebot moves towards its goal when the path is clear. <i>Please refer to Movie No. 4 in Table 2.3.</i>	42
2.13	Experiment 2: (Case B) Collision Avoidance on SHP nodes. (a) Initial configuration of robots. (b) Collision detection. (c) High priority Turtlebot retreats to cached safe node 'm' (Fig.2.10e), as P3DX has no non-overlapping safe node to retreat. (d,e) P3DX continues towards its goal. (f,g) Turtlebot moves towards its goal when the path is clear. <i>Please refer to Movie No. 5 in Table 2.3.</i>	43

2.14	Experiment 3: Avoiding Deadlock on SHP. (a) Initial configuration of robots. P3DX starts early. (b) Collision detection. (c) High priority P3DX retreats to cached safe node 'l' (Fig.2.10e), as low priority Turtlebot has no non-overlapping safe node to retreat. (d,e) Turtlebot continues towards its goal. (f,g) P3DX moves towards its goal when the path is clear. <i>Please refer to Movie No. 6 in Table 2.3.</i>	44
2.15	Experiment 4: Dynamic Obstacle Avoidance on SHP. (a) Position of the two obstacles placed dynamically in the map. (b) Initial path calculated by PRM before dynamic obstacles were placed. (c) New path calculated after sensing the dynamic obstacles.	45
2.16	Experiment 4: Dynamic Obstacle Avoidance on SHP. (a) Initial configuration. (b) P3DX moves on the initially calculated short path. (c) P3DX stops sensing the dynamic obstacles. (d) P3DX calculates new path and moves. (e,f,g) Positions of the new obstacles are updated in the map. (h) P3DX transfers the coordinates of the new obstacles to Turtlebot. (i,j,k) Turtlebot calculates the path with the updated map and moves. <i>Please refer to Movie No. 7 in Table 2.3.</i>	46
2.17	New grid-map with dynamic obstacles.	46
2.18	Experiment 5: SHP with different clearance threshold (δ). (a) Test environment. (b) Dimensions of the environment. Obstacle points P and Q are not collinear. For configuration $\mathbf{C}_{\text{pio}}^{\text{H}} :: \mathbf{a} \rightsquigarrow \mathbf{b}$, P3DX takes a turn at node n_1 maintaining $\delta = 60$ cm. For configuration $\mathbf{C}_{\text{pio}}^{\text{H}} :: \mathbf{a} \rightsquigarrow \mathbf{c}$, P3DX takes a turn at node n_2 maintaining $\delta = 70$ cm. (c) Grid map of the environment. <i>Please refer to Movie No. 8 in Table 2.3.</i>	47
2.19	Experiment 6: Obstacle Avoidance on SHP with Load Carrying Robot. (a) No load. (b) Load length (centered) = 75 cm. (c) Load length (centered) = 115 cm. <i>Please refer to Movie No. 9 in Table 2.3.</i>	48
2.20	Experiment 6: Obstacle Avoidance on SHP with Load Carrying Robot. (a) Grid map of the test environment shown in Fig.2.19. (b) Map dimensions with different SHP curves for different values of δ	48
2.21	SHP result comparison with QPMI [20], PRM, and D*. (a) Simulation environment. (b) Skeleton map. (c) Paths generated by QPMI, D*, PRM, and SHP.	52
2.22	SHP as an extension for various global path planners.	54
2.23	A global path with sharp turns at B, C, and D.	55

2.24	Flowchart of turn-point detection from angular paths.	56
3.1	Modified queue. (a) For similar score, the first request in the queue is processed first. (b) Requests can be processed from the middle of the queue if the priority score is high or in emergency condition.	60
3.2	Environment of the experiment. (a) Layout of the experiment with dimensions, (b) actual grid map of the environment.	67
3.3	Path Planning, and Monitoring & Controlling GUI. (a) Figure showing non overlapping paths of requesting robot (blue) and dock relinquishing robot (green). (b) Monitoring and Controlling GUI. Docking point status is shown in red(occupied) and green(empty). Status of robots is also shown.	68
3.4	Experiments in different scenarios. Time in seconds is shown on the top-left position of each frame. (a) Experiment 1. (b) Experiment 2. (c) Experiment 3. (d) Experiment 4. (e) Experiment 5.	70
4.1	System Architecture.	74
4.2	Sub-mapping, node representation, and actual environment. (a) Map divided into sub regions to serve with nodes along passages, (b) A node (green) with four diverging vectors and anti-pheromone values, and a terminal node (red), (c) Actual environment, (d) Robots exploring the map.	76
4.3	Anti-pheromone deposition. (a) Initial condition with no anti-pheromones on nodes, (b) Robots R_1 and R_2 depositing anti-pheromones on nodes, (c) A particular node state in future with $-\infty$ values on terminal nodes. (d,e) Division into sub-regions. (d) Entire sub-region blocking, (e) Division of a sub-region further into sub-blocks.	77
4.4	Test environment. (a) Environment's grid map, (b) Service regions (green) from A to E, and docking station (orange), along with passage showing nodes. (c) Comparison of time taken by robots by anti-pheromone signalling and shortest path method with conflict resolve.	87
4.5	Final node configuration. (a) Final node configuration in anti-pheromone experiment with anti-pheromones shown in red, (b) Final node configuration in pheromone experiment with pheromones shown in blue.	88

4.6	Simulation of map exploration using the proposed method with 8 robots. Map is shown with skeleton path, signalling matrix, area capture matrix and robots marked from numbers 1 to 8. (a) At time=7sec (b) At time=19sec. Regions A and D are captured by robots R1 and R2. (c) At time=30sec. Regions A, D, and B are captured by robots R1, R2, and R3, respectively. (d) At time=49sec. The entire area has been explored and captured.	89
4.7	Improvement in landmark matching using proposed method for capture and un-captured area case with and without the proposed method. . .	90
5.1	Bob's localization failure results in its pheromone deposition at wrong places. Like Bob, the pheromone trailing Alice too will end up in the wrong place.	96
5.2	Pheromone deposition at wrong places. Robots deposit pheromones at their estimated positions which might differ from their true position. In this example, Robot's estimated position is Pos.E at coordinates $X=15, Y=10$. Robot's true position is Pos.T. at coordinates $X=25, Y=10$. Hence, the pheromones are wrongly deposited at Pos.E ($X=15, Y=10$). .	97
5.3	Bob reduces pheromone deposition and disperses them depending on its uncertainty. Trailing Alice now has some idea that Bob was lost at point P, and is in a better position to take decision.	98
5.4	Addition of Pheromone Fields in Vicinity	105
5.5	Flowchart	105
5.6	Pheromone evaporation with $P_0 = 50$, $K = 0.0057$, $P_a = 0$ in Eq.(5.19). .	106
5.7	Pheromone evaporation with $P_0 = 35$, $K = 0.0057$, $P_a = 0$ in Eq.(5.19). .	106
5.8	Pheromone evaporation with $P_0 = 35$, $K = 0.0057$, $P_a = 10$ in Eq.(5.19). .	106
5.9	Alternate view of pheromone deposition in case of Fig.5.6a. Pheromone deposition is in the error ellipse given by the eigenvalues and eigenvectors of the covariance matrix. The lighter color intensities show more pheromone deposition.	109
5.10	Simulation Environment	111
5.11	Figure of laser range sensor URG-04LX (HOKUYO AUTOMATIC CO., LTD.) and measurement range	112
5.12	Result of pheromone deposition while localizing in map	113
5.13	Estimation error of sensor angle ϕ	114

5.14	Estimation error in X -axis	114
5.15	Estimation error in Y -axis	115
5.16	Combined error representing spatial uncertainty	115
5.17	Value of pheromones deposited	116
5.18	Comparison of Pheromones and Uncertainty	116
5.19	Real Environment	117
5.20	Good localization at the start of the operation. Uncertainty in localization is represented by blue ellipses and particles in cyan.	118
5.21	Pheromones deposited at different locations shown in Fig.5.22. Blue areas show pheromones above threshold (10). Total pheromones at location L12 were 8.15.	119
5.22	Leader (P3DX) robot's uncertainty at different locations L1 to L15. Uncertainty in localization is represented by blue ellipses and particles in cyan. The uncertainty well within the bounds of the corridor until location L12. After location L12 there is bad localization and minimum number of pheromones are deposited. The robot goes to location A which is highly uncertain location due to large uncertainty of the leader robot.	119
5.23	Trajectory of the follower robot (Turtlebot). 'Leader lost' is detected at location L12, and Turtlebot goes to location B. Blindly following the pheromones using the traditional pheromone deposition schemes would have caused Turtlebot to end up at uncertain location A, which is avoided using the proposed uncertainty modeling.	120
5.24	Speed variation of the second trailing robot with respect to the pheromones deposited by the first robot (in simulation experiment). (a) Pheromones deposited by the first robot. (b) Speed of the second trailing robot.	122
5.25	Speed variation of the third trailing robot with respect to the total pheromones deposited by the first and second robots (in simulation experiment). (a) Pheromones deposited by the second robot. (b) Total pheromones deposited by the first and second robots. (c) Speed of the third trailing robot.	123

5.26	Speed variation of the trailing Turtlebot robot with respect to the pheromones deposited by Pioneer P3DX robot (in real experiment). (a) Pheromones deposited by Pioneer P3DX robot. (b) Speed of trailing Turtlebot robot.	124
6.1	QR code and camera system for coupling.	128
6.2	Hitchhiking steps.	130
6.3	Different hitchhiking scenarios. (a) When the start and goal locations of the two robots are the same. (b) When the goal locations of the two robots are nearby. (c) Hitchhiking in between.	132
6.4	Intermittent information transfer to prevent ‘driver-lost’ scenario.	133
6.5	Experiment 1: Environment with obstacles marked in red.	135
6.6	Experiment 1. (a) Coupling between the two robots. (b) Turtlebot following P3DX through visual servoing.	135
6.7	Experiment 1. (a) Initial map without obstacles. (b) Map with obstacles. Driver robot transfers the obstacle coordinates (marked in red) to the hitchhiker robot.	136
6.8	Experiment 2: Environment with obstacles marked in red.	137
6.9	Experiment 2. (a) Coupling between the two robots. (b) Turtlebot following P3DX through visual servoing.	137
6.10	Experiment 2. (a) Initial map without obstacles. (b) Map with obstacles. Driver robot transfers the obstacle (marked in red) coordinates to the hitchhiker robot.	139
6.11	Experiment 2. (a) Initial map without obstacles. (b) Map with obstacles. Driver robot transfers the obstacle coordinates to the hitchhiker robot.	140
6.12	Experiment 2. 3D Map. (a) Initial map without obstacles. (b) Map with obstacles. Driver robot transfers the obstacle coordinates to the hitchhiker robot.	140
7.1	Environment for simulation	144
7.2	Local coordinate for laser range sensor	145
7.3	Example of laser range sensor data	146
7.4	Detection of a line from points with SVD	146
7.5	Flow chart of line-segment detection	148
7.6	Explanation of hopping point SVD based line segment detection.	150
7.7	Hopping with large and appropriate hop values.	153

7.8	Proposed line segment detection with Hough transform.	154
7.9	Matching algorithm	156
7.10	Example of $\delta \mathbf{f}_{m,i}^q$ ($\gamma_f = 10^{-4}$, $\epsilon = 10^{-4}$, $\kappa = 3$)	159
7.11	Flow chart of the SLAM process integrated with proposed line segment detection. Marked portion shows the proposed line extraction accelerated by hopping.	163
7.12	Simulation result of estimation of sensor position with map matching .	165
7.13	Result of simulation of simultaneous localization and mapping	166
7.14	Estimation error in X -axis	166
7.15	Estimation error in Y -axis	167
7.16	Estimation error of sensor angle ϕ	167
7.17	Result of simulation of iteration number for estimation of sensor position	168
7.18	Result with experiment environment.	169

List of Tables

2.1	Load carrying robot on SHP with load width and actual clearance. For turn nodes, refer Fig.2.20b.	49
2.2	Execution Time for SHP Generation (for 100×100 pixel map of Fig.2.21a). Implementation in C++ on 1.80GHz Intel Core-i7-4500CPU	51
2.3	Movie list of all the experiments in this work.	53
3.1	Database of Task and Actual Power Consumed	62
3.2	Mapping Tasks and Task Priority	62
4.1	Signal Notation and Behaviour	76
4.2	Table showing sub-regions captured by robots at particular times . . .	86
4.3	Table showing final pheromone matrix in simulation of Fig.4.6	92
4.4	Table showing the final configuration of the area capture matrix in simulation of Fig.4.6	92
5.1	Laser range sensor for simulation	111
5.2	Pheromone range and corresponding robot speed	122
7.1	Laser range sensor for simulation	162
7.2	Execution time for line detection using hopping point SVD on 2280 (760 $\times$ 3) points and speed-up for various values of ψ	164

Chapter 1

Introduction

1.1 Background

Robots have come a long way since the beginning of the industrial age. Traditionally, robots were meant to be bulky, with huge arms and fixed to a certain place away from human interaction. Early industrial robots brought a revolution into the manufacturing and production industries with rapid product development and round the clock use of robots in industries. At present, multiple robots are being employed in military, manufacturing, hospitals, disaster mitigation and for entertainment. The most significant change has been brought up by the advancement of artificial intelligence and machine learning techniques in robotics. This has made possible for robots to become completely autonomous in unstructured environments by utilizing sensors. Today there are robots operating in hazardous areas in applications such as search and rescue, mapping of collapsed mines, disaster site surveying, disposing dangerous chemicals and bombs and numerous other examples. Autonomous robots have become crucial for our quest to outer space explorations. An example is the “Opportunity” rover by NASA, which is an autonomous robot exploring the unknown terrain at Mars[120]. Such is the impact of these autonomous bots that in 2005 a team led by Sebastian Thrun won the DARPA Grand Challenge, a prize competition for American autonomous vehicles, funded by the Defense Advanced Research Projects Agency, the most prominent research organization of the United States Department of Defense [31]. Following the Fukushima nuclear disaster in Japan, the 2016 DARPA Robotics Challenge focused on disaster or emergency response scenarios [192]. The robots had to complete a set of complicated tasks like: drive a utility vehicle, remove debris, open a door and enter building, climb an industrial ladder, use tools to break through a

concrete panel, locate and close a valve near a leaking pipe, and connect a fire hose to a standpipe and turn on the valve. Three teams were able to complete the task and team KAIST with their DRC-Hubo robot was declared the winner [192]. We already have driverless cars running on our streets that are estimated to take over manual drivers by year 2030.

1.1.1 Rise of Embedded Computing and Embedded Sensors

All these intelligent robots run complex algorithms and need accurate sensor data for operation. For mobile robots, a critical requirement of higher level applications (ex. navigation) is that the mobile robot has some reasonable knowledge of its current position with respect to a fixed reference frame. For example, in navigation applications that entail motion to a target position, the robot needs an accurate estimate of its current position to plan a path to the goal and to confirm success. This requires a lot of computational power and accurate sensors. In the early decades of robotics, computational power was one of the critical bottlenecks which limited the complexity of the tasks performed by the robots.

However, past decade has seen a tremendous increase in the computational power. Moore's law [118] still holds and multi-core processors are increasingly becoming more and more widespread. Amazingly, the cost of computing per GFLOP (Gigabit of Floating Point Operations) was \$1.1 trillion in 1961, it fell down to \$30000 in 1997, and today it is just around \$0.08 [26]. Moreover, the size of the processors is getting smaller. We have also seen an increase in the number of open source electronics projects. The robotics society has benefited tremendously from availability of open source electronic processing boards like Arduino[8], Raspberry-Pi [136], and Pandaboard [124]. For high-end computing tasks like 3D reconstruction and 3D mapping, robots have benefited tremendously from graphical processing units (GPU) and embedded GPU boards like nVidia's Jetson TK board [122].

We have also seen an increase of accurate and inexpensive sensors. Availability of efficient Laser range finders [183, 184], monocular cameras[55], stereo cameras [68, 173], and depth sensors[193] have opened a wide class of applications for the robots.

Another worth mentioning progress has been in the field of networking capabilities. The current devices (and robots) can not only communicate with themselves but also with far off entities via internet. This is often referred to as Internet of things (IoT) [78]. The Internet of things is the inter-networking of physical devices, robots, vehicles (also referred to as "connected devices" and "smart devices"), buildings, and

other items—embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data [77].

1.1.2 Rise of Service Robots

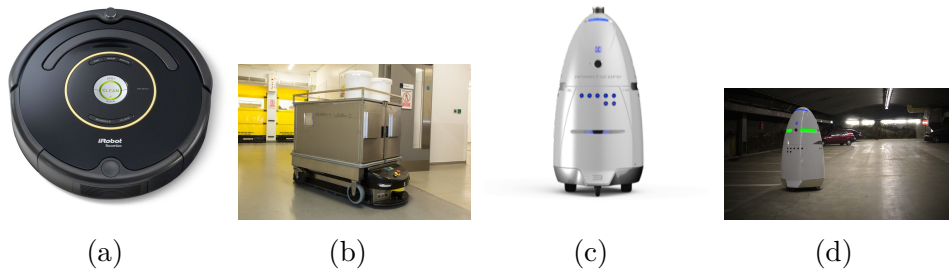


Fig. 1.1 Service Robots in Operation. (a) iRobot's Roomba Cleaning Robot [79]. (b) Robot at Southmead Hospital [116]. (c) Knightscope patrolling robot [89]. (d) Knightscope in operation [164].

The trends of progress in embedded computing [141] and sensor technology described in Subsection 1.1.1 are pushing the robots more and more into the daily lives of people. Robots are no longer constrained to factories and assembly lines, but are increasingly being used for other tasks like cleaning. In other words, we have seen an increase in the number of 'service robots' shown in Fig.1.1.

Perhaps the most common service robot used by people at home is iRobot's Roomba cleaning robot [79]. It can intelligently detect slopes and plan a path to efficiently clean the room.

Service robots have also been employed for patrolling. Knightscope patrolling robot [89] is equipped with 3D cameras, lasers, and GPS. They can send critical information about suspicious people to a tele-operator who can instruct the robot take appropriate action.

Pepper robot developed by Softbank company [166] is a humanoid service robot capable of recognising the principal human emotions and adapting its behaviour to the mood of his interlocutor. Pepper identifies person's emotions and selects the behaviour best suited to the situation [153]. Such robots are soon expected to be seen at public places like shopping malls for marketing and entertainment. For example, Pepper robot as shown in Fig.1.2 has already been employed for entertaining the kids at Shiroi Koibito Park [159] situated in Sapporo city of Japan.

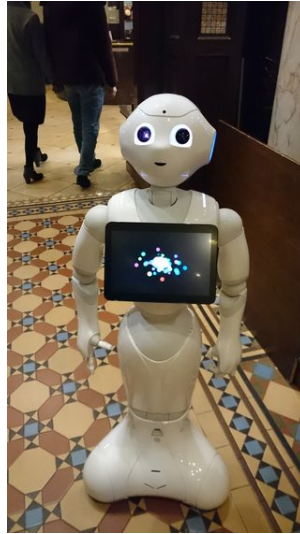


Fig. 1.2 Pepper robot[153, 166] in service for entertaining kids at at Shiroi Koibito Park [159] in Sapporo, Japan

1.2 Multi-Robot Systems

Compared to a single robot, multi-robot systems offer several advantages. The field of cooperation and coordination of multi-robot systems has been object of considerable research efforts in the last years. The basic idea is that multi-robot systems can perform tasks more efficiently than a single robot, or can accomplish tasks not executable by a single one. Although many, some of the main advantages of multi-robot systems are briefly summarized below:

1. **Wide Area Coverage:** Multiple robots can cover larger areas than a single robot. This is particularly important considering the battery life of service robots. A single robot can serve an area for a certain time and must recharge before commencing the other tasks. Such a problem does not exists in a multi-robot system.
2. **Task Distribution:** Tasks can easily be distributed in a multi-robot system. Particularly, tasks suitable to certain robots can be allocated accordingly. This eliminates the problems in ‘one robot does it all’ scenario.
3. **Task Parallelism:** Tasks can be executed in parallel in a multi-robot system. Tasks may be of same or different nature.

4. **Continuous Operation**: Since most of the service robots are battery powered, a continuous operation is often not possible in case of a single robot which must recharge before starting any task. However, in case of multi-robot system, certain robots can be in operation whereas others recharging. The robots can later exchange roles, and a continuous operation is possible. This is particularly useful for tasks like patrolling which demands continuous operation.
5. **Task Efficiency**: Certain tasks explicitly require multiple robots. For example, in disaster scenarios, multiple robots are often employed to cooperatively perform the rescue operations.
6. **Fault Tolerance**: Unlike single robot systems, failure of a single robot does not stop the task execution in a multi-robot system.
7. **Cooperative Task Execution**: In many scenarios, a task is simply not possible or extremely inefficient to be executed by a single robot. As an example, in a heterogeneous multi-robot system, a UAV (Unmanned Ariel Vehicle) can provide a bird's eye view of the area to UGV (Unmanned Ground Vehicle) and mark areas of interest for the UGV to explore. Such a task would be very difficult for a single UGV.
8. **Inherently Multi-Entity Tasks**: Some tasks like robo-soccer and other sports inherently require multi-robot coordination.

1.3 Challenges Faced by Service Multi-Robot Systems

Although there are several potential advantages of using multiple service robots, a multi-robot systems faces several challenges. The complexity of these problems is proportional to the number of service robots employed in the system. Some of the major problems discussed below are addressed in this thesis:

1. **Muti-Robot Path Planning**: A mobile robots must plan its path to the goal without colliding with the obstacles in the map. This becomes more challenging when multiple robots are employed in the same environment and it is desired that the planner used should have the following important characteristics:

- (a) **Safe Path Generation**: The paths generated by mobile robots must be safe from the obstacles. The planner must consider factors like the width of the robot itself which may vary if it is carrying some load.
 - (b) **Collision-Free**: Path planning should be done so that the robots do not collide with each other. If the paths of the two robots collide with each other, the robots must be able to detect such collision and take appropriate actions to avoid collision.
 - (c) **Smooth Path Generation**: Most path planners generate a path which has sharp and angular turns. Such paths are not good for robots to traverse as the robots cannot make these sharp turns abruptly without stopping. Instead, smooth paths are desired by the robots on which the robots can move with a constant velocity. The collision avoidance scheme should be integrated with smooth path generation for multiple robots.
 - (d) **Task Priority Consideration**: While avoiding collision with other robots, the planner must consider several factors like the task priorities of the robots. Situations such as ‘deadlock’ in which neither of the robots can move should be avoided by considering such factors.
 - (e) **Energy Efficiency**: The total service time of the robots is directly dependent on the available battery life. Hence, collision avoidance scheme must try to conserve battery life of robots whenever possible.
 - (f) **Real-time**: The algorithms must be computationally efficient enough so that entire path planning is performed in real time by the robots.
2. **Common Resource Sharing**: The second problem which frequently occurs in multi-robots scenarios is the problem of common resource (like charging point or narrow path) sharing. Service robots for cleaning and patrolling are mostly in continuous operation and require frequent recharging. Using a large number of docking points proportional to the number of robots is costly and requires large space. A manager which can resolve conflicts and allocate the resource to the most appropriate robot by considering several factors like task priority, battery power left, distance travelled, etc. is required.
3. **Autonomous Task Coordination**: Third, is the problem of autonomous map exploration and autonomous task distribution by multiple mobile service robots. The problem is related to but more abstract than the first problem of smooth

path generation, in which, the goal locations are given to each mobile robot. However, introducing multiple robots also introduce the *problem of explicitly programming the robots to go to certain areas to serve*. The areas to serve in a map may vary with time. Moreover, the number of the robots available to serve may also be dynamic, in real world situations, as some of the robots may be charging, while some may be out of order. In case of robots used for surveillance, a robot may want other robot or robots to follow itself while chasing a suspicious person, for backup. This situation is also dynamic in terms of availability of robots, selecting the nearest robot for quick response, and selecting the same path towards a particular area as taken by the previous robot. In both the cases, *explicitly programming the robots is cumbersome, and demands for a simpler scheme in which multiple robots can intelligently explore the map and distribute tasks*. Moreover, if multiple robots are serving the same area, or navigating to the same area, they must coordinate tasks to maximize efficiency.

4. **Modelling Uncertainties of Multi-Robots:** Most multi-robot systems (particularly those based on virtual pheromone trailing) assume a ‘perfect-world’ scenario, in which, particularly the positional attributes of the robots are assumed to be perfect. In reality however, it is difficult to achieve perfect localization of the robot due to errors in encoders and sensors attached to the robot, and the dynamics of the environment in which the robot operates. In real world scenarios, there is always some uncertainty associated in estimating the pose (i.e. position and orientation) of the mobile service robot. Failing to model this uncertainty would result in service robots in wrong places which has adverse effects on the multi-robot system. For example, if we consider a ‘leader-follower’ multi-robot system, the leader robot might completely fail to localize itself in the environment and be lost. Other robots trailing its pheromones will end up being in entirely wrong areas of the map. This results in a ‘blind leading the blind’ scenario which reduces the efficiency of the multi-robot system.
5. **Simultaneous Localization and Mapping:** The final problem is a fundamental problem of Simultaneous Localization and Mapping (SLAM) in autonomous mobile robots. This problem is closely and implicitly related to all the above aforementioned problems which assumed an integrated SLAM module in the robot. An inefficient and slow SLAM algorithm would adversely affect path planning, collision avoidance, and task coordination. On the other hand, a

fast, accurate, and real-time SLAM algorithm can leverage the performance of aforementioned tasks. A robust and real-time SLAM algorithm is indispensable for multi-robot systems.

1.4 Motivation

The work presented in this theses addresses the specific aforementioned problems of multi-robot systems. The goal is to develop algorithms to reach a high level of autonomy in multi-robot system to perform certain tasks easily and efficiently.

Concretely, the goal is to reach an autonomy in which an instructor must be able to give a very high level command like, *“Eight robots, clean all the rooms of the hospital”*. Notice that, the instruction given is at a very high level of abstraction. The number of robots may be specified. If it is not specified, for example in the instruction: *“Go and clean the garage floor”*, then the system must choose the most appropriate number of robots suitable to do the given task.

The given instruction must be decomposed into a number of components for the problem to be solved. The system must first identify where are the rooms located, or where is the garage. This requires a mapping component in the system. The robots must know their own current locations in the map which requires a localization component. They must generate a path to go to specific areas for which a path-planning module is required. Path planning also requires the location of obstacles which requires a map. Robots must have a task coordination module to devise a strategy to capture areas in the building so that they can work without interruption from other robots. Moreover, only the absolutely necessary number of robots must serve an area. If the robots don't have sufficient battery power, then resources like the available charging docks must efficiently be shared.

These components directly corresponds to the problems mentioned in Section 1.3. At a high level, each of the problems mentioned in Section 1.3 appears to be an isolated problem which must be solved on its own. However, actual implementation of the multi-robot system requires a seamless integration of these modules.

The work in this thesis presents algorithms developed for each of the components and their integration in the multi-robot system to reach the autonomy desired. The next section describes the interdependence of the various components of the multi-robot system.

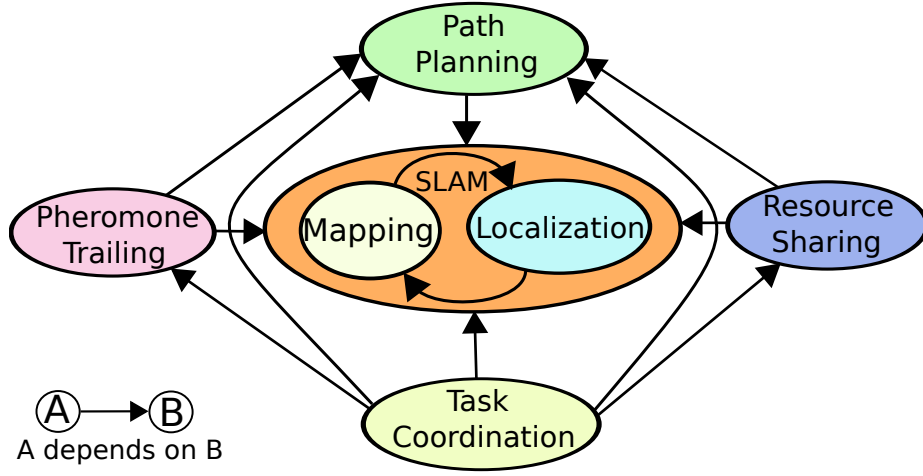


Fig. 1.3 Strong interdependence amongst the major components of a multi-robot system.

1.4.1 Interdependence of Various Multi-Robot Components

Most of the major components of a multi-robot system demonstrate a strong dependency on each other. With reference to the work presented in the current thesis, the major components of the multi-robot system and their interdependence is shown in Fig.1.3. Take the example of path planning, for example. In order to generate a path a robot must first be given map and the goal location must be specified. Hence, mapping is essential. The current location must also be known. Hence, localization is needed. The robot must generate necessary control sequences to avoid obstacles. If there are new obstacles, the map must be updated for future reference, and hence SLAM is required. Take another example of multi-robot navigation based on pheromone trailing. Pheromone trailing is the process in which robots follow a leader robot by sensing the virtual pheromones left in the map by the leader map to perform a collaborative task. Since pheromones are deposited at some location in the map, localization of the robot, mapping, and consequently SLAM becomes indispensable. Moreover, a path must be generated for the robots to move for which a path planning module is required. Similarly, in order to share a resource like a charging point, or a common path, SLAM is necessary as the robots must estimate their pose (i.e. orientation and pose) with respect to the shared resource. They need a path towards the common resource in order to access it. In similar fashion, it is obvious that task coordination too depends on SLAM, path planning, pheromone trailing (if implemented), and resource sharing modules.

It is interesting to see in Fig.1.3 that SLAM forms the core module and all the modules depend on it. In fact, for the successful execution of a task in a multi-robot system, it is highly desired that all the modules work to their fullest potential. Failure of any of the modules will have adverse effects on the task execution. For example, if the SLAM module is not robust, the robots will fail to traverse the path generated. If localization is not proper, robots may access incorrect resources. An inaccurate map, or pheromone deposition at wrong places will send robots to wrong location and task coordination will adversely be affected.

This strong interdependency between modules also implicitly suggests that each module assumes the availability of other modules for functionality. This can be elaborated by picking any module in Fig.1.3, say path planning. It assumes that we have a mapping module which provides the map, a localization module which provides the current location, and SLAM module to update map and track position with respect to the modules. This assumption has an advantage that each module can focus explicitly on executing the task at hand in the most efficient manner. Similarly, other modules have the same assumption and focuses on their own task which decentralizes the multi-robot system and improves overall efficiency.

1.5 Thesis Organization and Chapter Flow

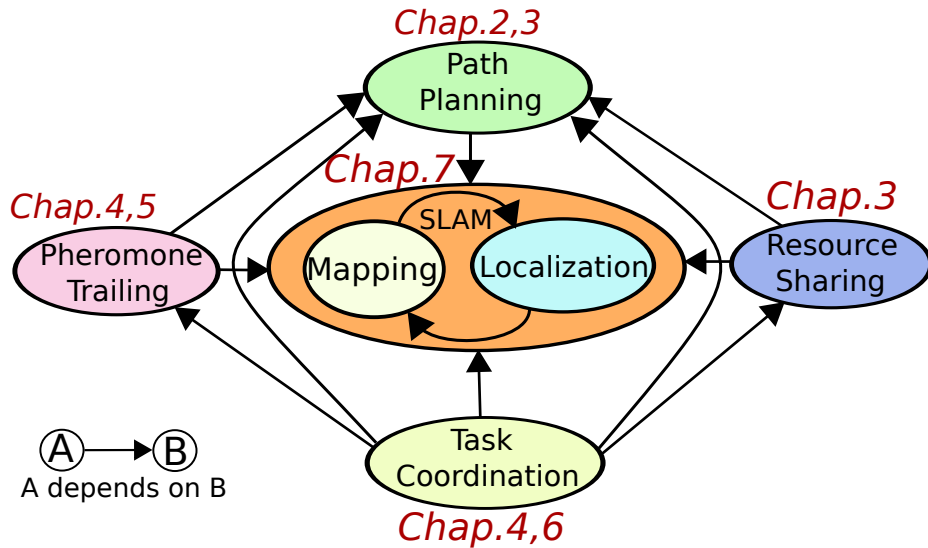


Fig. 1.4 Components of multi-robot system and the chapters describing them.

The novel algorithms corresponding to the various components shown in Fig.1.3 are discussed in separate chapters of this thesis. The novel ideas, new contributions,

and related works for each component are also discussed in corresponding chapters. Graphically, this thesis is organized as shown in Fig.1.4. Briefly, the thesis is organized in the following manner:

Chapter 2: This chapter deals with path planning in multi-robot systems. This chapter presents a novel smooth hypocycloidal paths (SHP) for robot motion. It is integrated with collision-free and decoupled multi-robot path planning. SHP diffuses (moves points along segments) the points of sharp turns in the global path of the map into nodes which are used to generate smooth hypocycloid curves which maintain a safe clearance from the obstacles. These nodes are also used as safe points of retreat to avoid collision with other robots. The novel contributions of this work are: (1) The proposed work is the first use of hypocycloid geometry to produce smooth and continuous paths for robot motion. A mathematical analysis of SHP generation in various scenarios is discussed. (2) The proposed work is also the first to consider the case of smooth and collision-free path generation for a load carrying robot. (3) Traditionally, path smoothing and collision avoidance have been addressed as separate problems. This work proposes an integrated and decoupled collision-free multi-robot path planning. ‘Node caching’ is proposed to improve efficiency. A decoupled approach with local communication enables dynamically changing the paths of robots without configuring the paths of all the robots. (4) A novel ‘multi-robot map update’ in case of dynamic obstacles in the map is proposed, in which, robots update other robots about the positions of dynamic obstacles in the map. A time-stamp feature ensures that all the robots have the most updated map. Comparison of SHP with other path smoothing techniques and experimental results in real environments confirm that SHP can generate smooth paths for robots and avoid collision with other robots through local communication.

Chapter 3: This chapter deals with common resource sharing in multi-robot systems. This chapter presents an intelligent docking station manager for multiple mobile service robots. The manager maintains a modified queue based on a score which is calculated from the task priority and the current power level of the robot. The behaviour of the station manager in different scenarios when the docking station has empty docks and when it has none are discussed. The docking station can intelligently instruct a robot to forcefully relinquish a dock and give preference to the robots with higher task priorities and lower power

levels. Emergency conditions for robots with power levels below threshold, and intelligent path allocation are also discussed.

Chapter 4: This chapter deals with the problem of task sharing and exploration in multi-robot systems. This chapter presents a novel bio-inspired hybrid communication framework that incorporates the repelling behaviour of anti-aphrodisiac pheromones and attractive behaviour of pheromones for efficient map exploration of multiple mobile service robots. The proposed communication framework presents a scheme for robots to efficiently serve large areas of map, while cooperating with each other through proper pheromone deposition. This eliminates the need of explicitly programming each service robot to serve particular areas of the map. The paths taken by robots are represented as nodes across which pheromones are deposited. This reduces the search space for tracking pheromones and reduces data size to be communicated between robots. A novel pheromone deposition model is presented which takes into account the uncertainty in the robot's position. This eliminates robots to deposit pheromones at wrong places when localization fails. The framework also integrates the pheromone signalling mechanism in landmark based Extended Kalman Filter (EKF) localization and allows the robots to capture areas or sub-areas of the map, to improve the localization. A scheme to resolve conflicts through local communication is presented. Experimental and simulation results are discussed with two cases of floor cleaning task, and surveillance task, performed by multiple robots. Results show that the proposed scheme enables multiple service robots to perform cooperative tasks intelligently without any explicit programming.

Chapter 5: This chapter builds upon the work in the previous chapter which used virtual pheromones for task coordination. This chapter presents a pheromone deposition algorithm which models the uncertainty of the robot's pose. Results from experiments in both simulated and real environments demonstrate that modeling the uncertainty in pheromone deposition is crucial, and that the proposed algorithm can model the uncertainty well. The modeling proposed in this chapter, provides extra information to the trailing robot about the preceding robot. A 'confident' deposition concentrated in a small area indicates that the preceding robot was certain about its location at that place. Whereas, low and dispersed pheromones indicates that the preceding robot was not sure about its location while depositing pheromones. Two cases of behavioral difference of

follower robot with and without the proposed method, and speed adaptation of trailing robots using the proposed method are presented.

Chapter 6: Virtual trailing based multi-robot task coordination and navigation as discussed in Chapters 4 and 5 are useful over large distances, particularly when the leader robot is not in the field of view (FoV) of the follower robot. However, there are cases when the leader robot is in the FoV of the trailing robot. In such scenarios, a more straightforward solution is to directly follow the robot. In these cases, such direct following can be incorporated with other mechanisms for performance efficiency in multi-robot systems. A robot must localize, do path planning, obstacle avoidance, and update the map if there are new features in the environment while navigating to its goal location. When two or more robots have the same goal location, it is a waste of computing if all the robots performs all these tasks individually. To this end, this chapter introduces a novel concept of ‘hitchhiking’ as a means of efficiency gains. The hitchhiker robot can save computation required for (a) path planning, (b) localization, (c) dynamic obstacle avoidance, and (d) map update. The system has an award mechanism in which the hitchhiker ‘owes’ to the driver robot which receives points. The admissible characteristics of the hitchhiker and the driver robots are discussed. Important points about when should hitchhiking be allowed and when not are also discussed.

Chapter 7: In all the previous chapters, it was assumed that multiple robots had a SLAM component. This chapter proposes an efficient algorithm for line detection from LRS data using a novel hopping-points Singular Value Decomposition (SVD) and Hough transform based algorithm, in which, SVD is applied on intermittent LRS points to accelerate the algorithm. A reverse-hop mechanism ensures that the end points of the line segments are accurately extracted. Line segments extracted from the proposed algorithm are used to form a map and subsequently LRS data points are matched with the line segments to localize the robot. The proposed algorithm eliminates the drawbacks of point based matching algorithms like Iterative Closest Points (ICP) algorithm whose performance degrades with increasing number of points.

Chapter 8: This chapter discusses the results and their implications, concludes the thesis work, and points out the potential future works.

Chapter 2

SHP: Smooth Hypocycloidal Paths and Decoupled Multi-Robot Path Planning with Collision Avoidance

Overview

This chapter deals with path planning in multi-robot systems. As discussed in the previous chapter, the paths must be smooth, far from the obstacles, and collision must be avoided on these paths. As it is evident from Fig.1.3 which shows the interdependence of various modules of a multi-robot system, path planning depends on a localization and mapping module (discussed in Chapter 7). This chapter focuses mainly on path planning, and availability of a SLAM module is assumed.

Chapter Abstract

Generating smooth and continuous paths which avoid sharp turns for robots with collision avoidance is an important problem in autonomous robot navigation. This chapter presents a novel smooth hypocycloidal paths (SHP) for robot motion. It is integrated with collision-free and decoupled multi-robot path planning. SHP diffuses (moves points along segments) the points of sharp turns in the global path of the map into nodes which are used to generate smooth hypocycloid curves which maintain a safe clearance from the obstacles. These nodes are also used as safe points of retreat to avoid collision with other robots. The novel contributions of this work are: (1) The proposed work is the first use of hypocycloid geometry to produce smooth and continuous paths for robot motion. A mathematical analysis of SHP generation in

various scenarios is discussed. (2) The proposed work is also the first to consider the case of smooth and collision-free path generation for a load carrying robot. (3) Traditionally, path smoothing and collision avoidance have been addressed as separate problems. This work proposes an integrated and decoupled collision-free multi-robot path planning. ‘Node caching’ is proposed to improve efficiency. A decoupled approach with local communication enables dynamically changing the paths of robots without configuring the paths of all the robots. (4) A novel ‘multi-robot map update’ in case of dynamic obstacles in the map is proposed, in which, robots update other robots about the positions of dynamic obstacles in the map. A time-stamp feature ensures that all the robots have the most updated map. Comparison of SHP with other path smoothing techniques and experimental results in real environments confirm that SHP can generate smooth paths for robots and avoid collision with other robots through local communication.

2.1 Introduction

Mobile robots with autonomous navigation capabilities have been used in various operations like industry automation, floor cleaning, and managing warehouse. In such operations, multiple mobile robots are generally deployed where they need to autonomously move between various locations. A smooth and continuous path is desired for robot motion which avoids abrupt and sharp turns. This problem is further complicated in case of multiple robots as two or more robots may have intersecting and common paths which increases the chances of a robot colliding with another robot during its operation. Hence, the path taken by each robot needs to be carefully planned. This is explained in Fig.2.1, where two problems need to be addressed: (1) The path of the robots must be smooth and sufficiently far from obstacles. Robot R3 in Fig.2.1 is dangerously close to obstacles, and path of R4 has sharp turns of robot. (2) Robots must avoid colliding with other robots which may have the same path. In Fig.2.1, R1 and R2 are in a deadlock position and there is a collision possibility of R4 and R5. There is a large plethora of research literature on path planning for autonomous robots. Back in 1960, Pollock and Wiebenson[132] first reviewed algorithms to find a minimum cost path from any given graph. Another influential prior work by Peter Hart et. al.[66], proposed in 1968 incorporates heuristic information in graph searching to find minimum cost paths. Thomas et. al proposed an algorithm[102] for collision-free paths in an environment with polyhedral obstacles, in 1979. Since then, a number of path planning algorithms have been proposed, and a detailed summary of those algorithms

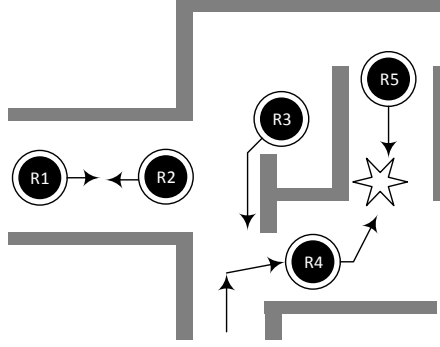


Fig. 2.1 Two problems of multi-robot path planning: (1) the paths should be short, smooth and safe (i.e. far from obstacles), (2) robots must avoid collision with each other.

can be found in [36], [95], and [93]. Among these, A^* algorithm [66], D^* algorithm [171, 172], probabilistic roadmap planner (PRM) [85], rapidly exploring random tree (RRT) [94, 96], and potential fields [75] algorithms are the most widely used techniques today.

In general, path planning is done in the two steps of global planning, and local planning. First, it is necessary to find the overall path from the start to the goal location, which avoids the static obstacles of the map. This is done in the global planning. At this stage, it is neither necessary to consider the dynamic obstacles of the environment, nor worry about the fine control sequences which needs to be sent to the robot to take the turns. Uniform-cost search Dijkstra's algorithm[38] has widely been used for global planning. However, it is computationally expensive with a time complexity of $\mathcal{O}(N^2)$, where N is the number of nodes. A number of ways to optimize the algorithm[160] and variants[58] have been proposed. The A^* algorithm [66] is an extension of Dijkstra's algorithm with better time performance with a good initial heuristic. D^* (or Dynamic A^*) finds an optimal path in real-time by incrementally updating paths to the robot's state as new information is discovered, and is more efficient than A^* algorithm. Rapidly-exploring Random Tree (RRT) algorithm [94, 96] efficiently searches high dimensional spaces by building space-filling trees randomly. RRT algorithm plans a path with a biased growth towards the goal with high probability. A review of many variations and implementations of RRT planners has been undertaken by S. LaValle et. al in [97].

The next step is that of local planning which focuses on generating a smooth curves on the global path. The sharp and angular turns on the paths are sometimes infeasible and difficult for robot control, and needs to be smoothed out, so that, the robot can

move continuously and avoid abrupt motion. Path smoothing using parametric curves has been proposed as early as 1957 by L. Dublin in [40, 41]. Generally known as Dublin curves, they employ lines and circles, but suffer from the problem of discontinuities. Since then, path smoothing using various types of curves have been proposed. Worth mentioning among these are proposals which make use of Clothoids [54, 99], quintic G^2 splines [130], and intrinsic splines [35]. Other approaches includes curves with closed-form expressions like B-spline [90], quintic polynomials [178], and bezier curve [24]. However, these methods suffer from problems of complicated curvature functions with many parameters, and improved methods have been proposed [199]. There is a different but old class of path smoothing algorithms which employ interpolation [111]. Interpolation technique was proposed as early as 1779, by E. Waring in [188]. Interpolation techniques are generally plagued by computational costs and Runge's phenomenon [45] which is a classic illustration of polynomial interpolation non-convergence. Recently, a very inspiring work ([73] and [20]) has been proposed by S.R. Chang and U.Y. Huh, in which they propose a QPMI (Quadratic Polynomial and Membership Interpolation) algorithm, which avoids Runge's phenomenon [45] and the weakness of spline interpolation, by creating a G^2 continuous path using just the quadratic polynomials and membership functions. In [20], they propose a collision free continuous G^2 path using interpolation. However, their methods required explicit collision detection checks, and reprogramming of smooth paths which can be expensive in case of crowded environments. Work in [48] is a good resource which summarizes various path smoothing techniques.

The aforementioned algorithms discussed the state of the art regarding path planning for a single robot. The prior art related to multi-robot collision free path planning are briefly discussed. Classifying broadly, multi-robot planning algorithms may be either coupled, in which the paths of all the robots are calculated concurrently like in [177], or decoupled [149], in which the path of each robot is individually calculated and later the velocities are tuned to avoid collision, as in [62]. Many prior works address smooth path generation and multi-robot collision free path planning, as separate problems. For example, work by Yang et. al. [198] proposes to use polygons around the crossing points of skeleton maps. However, their method produces a path which has many sharp turns undesired for robot motion. However, smooth path generation and multi-robot collision avoidance are integrated problems and must be addressed together. In other words, it is not good to only have smooth paths for a robot to traverse from start to

goal location, but generate abrupt and sharp turn motions for collision avoidance with another robot, in a dynamic scenario.

This chapter proposes SHP (Smooth Hypocycloidal Paths) to produce smooth paths for robot motion using the geometry of hypocycloids [100, 191]. SHP falls into the later category of local planning to generate smooth paths. The smooth paths avoid sharp and angular turns and also keep the robot at a safer distance from the environmental obstacles. Diffusion is defined as moving a point along the segment(s). SHP works by diffusing the intersecting points of skeleton maps[156] generated by thinning [7] or Voronoi diagrams [14] into nodes, and generating continuous hypocycloidal curves using the diffused nodes. Well known path planning techniques such as A^* algorithm [66], D^* algorithm [171, 172], are used to generate the global motion policy, and later SHP smooths out the sharp turns. A decoupled multi-robot scheme on SHP is also proposed in which the diffused nodes are used as retreat points for the robot to avoid collision with other robots.

The novel contributions of this work are:

- a) The proposed SHP is the first use of hypocycloid geometry to generate smooth continuous paths for robot motion. A mathematical derivation of SHP is presented in Section 2.3.
- b) To the best of our survey, the proposed work is the first to also consider the case of smooth path traversal maintaining a safe clearance distance from the obstacles in case of load carrying robot (in both centered, and un-centered load). It is important, as the collision-free smooth path traversed by a robot, when it is carrying and not carrying a load will be different. Section 2.4 explains the calculation of clearance threshold distance in both the cases, mathematically.
- c) A decoupled collision-free multi-robot path planning integrated with SHP is proposed (Section 2.5). It is advantageous as the robot still traverses smooth paths while also avoiding collision with other robots. Intelligent communication between robots is important in a decoupled multi-robot path planning scheme. This work proposes novel and practical idea of ‘*node caching*’ for better efficiency (Section 2.5.1).
- d) An integrated dynamic obstacle avoidance on SHP, with ‘map update’ is proposed. Robot finds a new SHP path if a dynamic obstacle is encountered, and shares the coordinates of the dynamic obstacle with other robots. Only the coordinates are transferred for efficiency. This is explained in Section 2.6.

Experiments (Section 7.5) in real environments with robots were performed to test the proposed techniques. The performance of the proposed SHP has been compared (Section 2.11) with standard path planning techniques like D*[171, 172], PRM[85], and the recently proposed QPMI[73] algorithm based G^2 collision-free smooth path[20] algorithm. Results show that SHP gives a better performance without explicit collision detection and safe clearance from the obstacles even when it is carrying some load. Readers are strongly encouraged to also refer all the experimental videos which have been provided online and the URLs have been summarized in Table 2.3, while reading this work.

2.2 Skeleton Map and Node Generation

First, skeleton maps and nodes or points of intersection in skeleton maps which needs to be generated from a map constructed by a robot using exteroceptive sensors is briefly discussed. A grid map is generated using the sensors like laser range sensors, or RGBD sensors attached to the robot. For noisy environments, mapping techniques proposed in [151, 152] can be used. Later, mathematical morphological operations [81, 140, 145] of dilation followed by erosion [150] is applied to remove noise which may appear in the map as small dots from laser range sensor. To remove larger blobs, dilation can be done multiple (three to four) times followed by a single erode. A Voronoi map generated from the binary map has path [14] which keeps the robot at a safe distance from obstacles but has sharp turns as shown in Fig.2.3c in which robot encounters sharp turns at O from L_C or L_A to L_B . Similarly, a skeleton path [156] generated, for example, by thinning technique [7], in case of Fig 2.4a will resemble the shape of English alphabet ‘T’ with cross-point shown at O . However, most of the thinning algorithms does not generate a sharp point with perpendicular turns like the point ‘o’ shown in Fig 2.4a, and instead generate three points close to each other. To generate a single point, a modification of the thinning algorithm proposed by T. Zhang et.al [204] is used. After generating the skeleton map by thinning, turn points are detected on the path. Later, the points which are close to each other are clustered [138] to generate a single point called a node. This is shown in Fig.2.2b which is the skeleton map of Fig.2.2a. The enlarged portion in Fig.2.2b shows the three points of turn on the pathway in red, and a single cluster in blue color. Thinning algorithms suffers from the problem of generating too many undesired sub-branches, which can be removed by pruning algorithms[11].

These nodes along the Voronoi or skeleton paths will be used to generate SHP which is discussed in the next section.

2.3 Generating Smooth Hypocycloid Paths (SHP)



Fig. 2.2 Node extraction from skeleton map. (a) Map with obstacles and free space. (b) Corresponding skeleton map. Nearby nodes are clustered to form a single node.

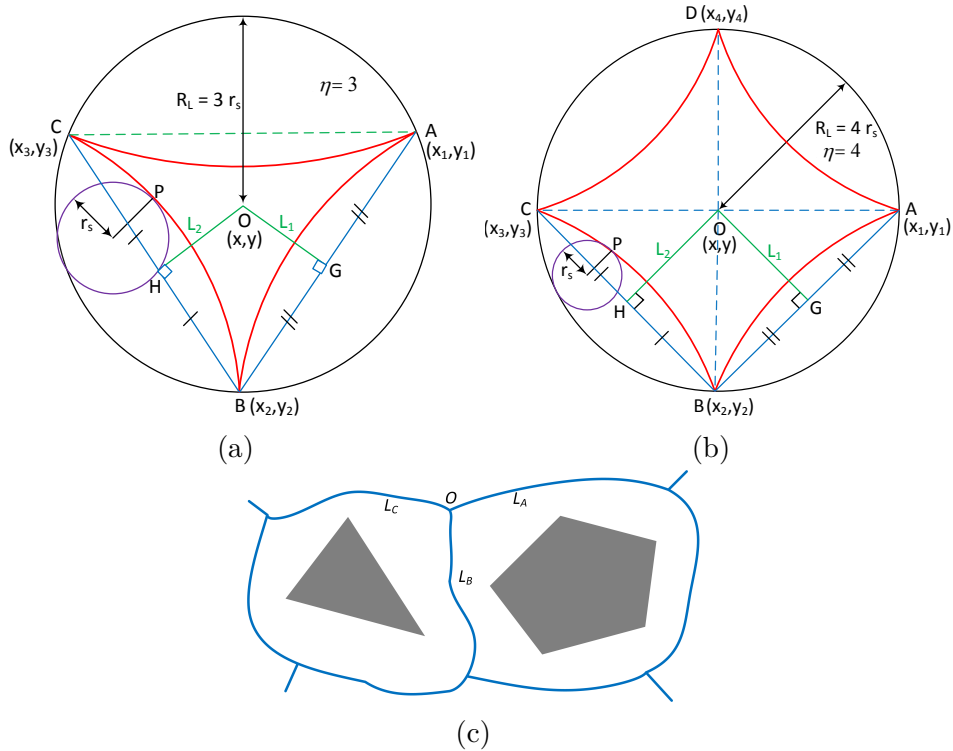


Fig. 2.3 Geometry of hypocycloid curves shown in red. (a) Deltoid curve: $R_L = 3 \times r_s, \xi = \frac{1}{3}$. (b) Astroid curve: $R_L = 4 \times r_s, \xi = \frac{1}{4}$. (c) Voronoi paths keep a safe distance from obstacles but robot encounters sharp turns at O from L_C or L_A to L_B

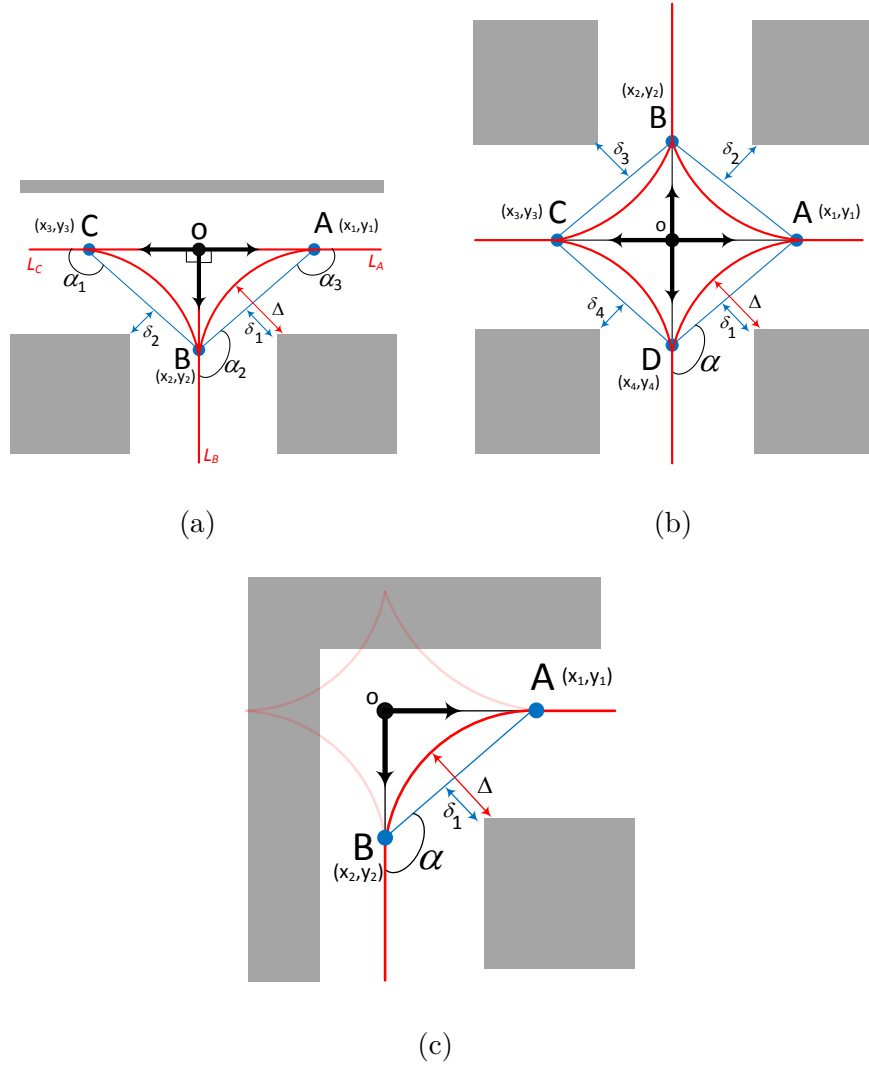


Fig. 2.4 Smooth path generation. (a) with deltoid or lower half of astroid, (b) astroid curves at cross-road map, (c) a portion of hypocycloid to generate curve at corner.

A hypocycloid is a geometrical curve which is produced by a fixed point P which lies on the circumference of a small circle of radius r_s rolling inside a larger circle of radius $R_L > r_s$ [100, 191]. In the case of the astroid curve as shown in Fig.2.3b, the rolling circle has a radius of r_s and the large circle has a radius of R_L , and the curve is defined

in general by,

$$\begin{aligned}
 x(\theta) &= (R_L - r_s)\cos(\theta) + r_s \cdot \cos\left(\frac{R_L - r_s}{r_s}\theta\right) \\
 y(\theta) &= (R_L - r_s)\sin(\theta) - r_s \cdot \sin\left(\frac{R_L - r_s}{r_s}\theta\right) \\
 \text{or, } x(\theta) &= R_L \cdot \xi\{(\eta - 1)\cos(\theta) + \cos((\eta - 1)\theta)\} \\
 y(\theta) &= R_L \cdot \xi\{(\eta - 1)\sin(\theta) - \sin((\eta - 1)\theta)\}
 \end{aligned} \tag{2.1}$$

where, η is the number of cusps and ξ is the ratio of the radius of the rolling circle to the radius of the large circle, i.e. $\xi = \frac{r_s}{R_L}$. A cusp is defined as the sharp corner where the curve is not differentiable. Fig.2.3a shows a 3-cusped hypocycloid called as a deltoid or tricuspoid. A 4-cusped hypocycloid called as an astroid is shown in Fig.2.3b. The values of angle θ to generate different hypocycloids is explained later in Section 2.3.1.

In general, to obtain n cusps in a hypocycloid, the radius of smaller circle is set to $r_s = \frac{R_L}{n}$, as n rotations of the smaller circle brings it back to the original position, generating n cusps while traversal [100]. For a deltoid, $R_L = 3 \times r_s$, $\xi = \frac{1}{3}$ and for astroid, $R_L = 4 \times r_s$, and $\xi = \frac{1}{4}$.

As shown in Fig 2.4a, a robot moving from location L_C to location L_B will encounter a very sharp turn of nearly 90 degrees at point o , along path CoB . The same is true for path AoB . To avoid sharp turn, the node o along open path directions of $\vec{oA}$, $\vec{oB}$ and $\vec{oC}$ is diffused. This diffusion is continued until the magnitude of the vectors (i.e. $|\vec{oA}|$, $|\vec{oB}|$, and $|\vec{oC}|$) are such that the line segment $\overline{BC}$ and $\overline{AC}$ are far from the surrounding obstacle by a obstacle-clearance-distance δ . The value of clearance distance δ is pre-calculated and will depend upon the geometry and dimension of the robot, and the obstacle. Now, a robot moving from location L_C to location L_B can avoid making a sharp turn at point o , and instead take a right-turn at point C , traverse the path CB shown as a blue line in Fig 2.4a, and take a right-turn again at point B and continue straight to location L_B . However, the robot still has to make a considerably sharp turn (shown by $\angle\alpha_i$) at points A , B and C , which is not desired. The smoothness of the path is further improved by applying hypocycloidal geometry and generate curves $\widehat{AB}$ and $\widehat{BC}$ shown in red in Fig.2.4a. The mathematical derivation of generating hypocycloidal curves is presented.

An explanation using the previous case of Fig.2.4a is provided. Let (x_1, y_1) , (x_2, y_2) and (x_3, y_3) be the coordinates of points A , B and C , respectively. If d_c represents

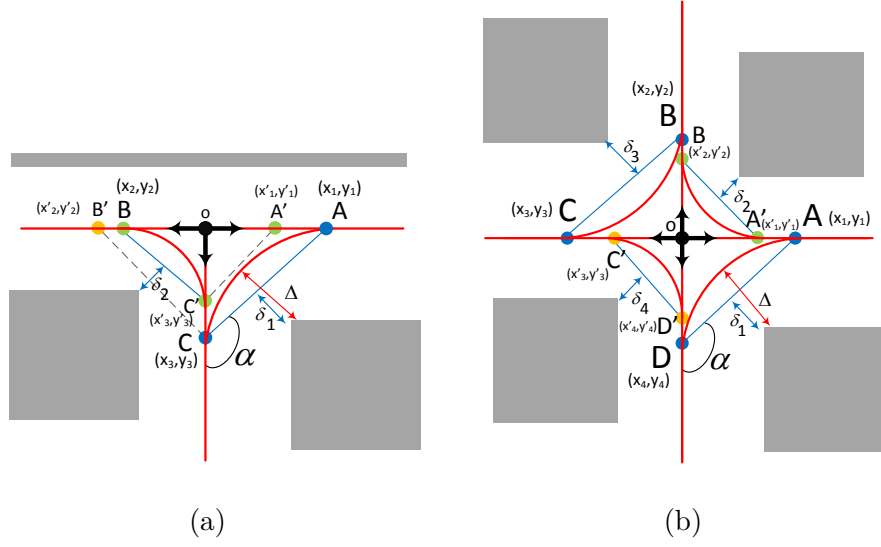


Fig. 2.5 Smooth path generation when clearance thresholds are not same. (a) & (b) Lower astroid with $\delta_1 \neq \delta_2$. The dotted lines and points A' and B' are auxiliary.

the distance between the left obstacle and line segment $\overline{BC}$, and right obstacle and $\overline{AB}$, diffusion is performed incrementally in very small steps of d_c . The coordinates of points A , B , and C will be known once the diffusion of point o stops when the condition $d_c = \delta$ is satisfied. Although the value of δ may be different for different obstacles ($\delta_i \neq \delta_j, i \neq j$), for the sake of simplicity, it is assumed that the clearance distance is the same. The case of different clearance distances is discussed later.

A hypocycloid (astroid) curve is generated with points A , B and C . Points A , B and C in Fig.2.4a corresponds to respective points in Fig.2.3b. In Fig.2.3b, the midpoints of line segments AB and BC are $G(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$ and $H(\frac{x_2+x_3}{2}, \frac{y_2+y_3}{2})$, respectively. The slope of line $\overline{AB}$ (m_1) is $\frac{y_2-y_1}{x_2-x_1}$ and the slope of line $\overline{BC}$ (m_2) is $\frac{y_3-y_2}{x_3-x_2}$. Hence, the slope of lines L_1 and L_2 perpendicular to $\overline{AB}$ and $\overline{BC}$ are $\frac{-1}{m_1}$ and $\frac{-1}{m_2}$, respectively. Hence, the equation of line L_1 is,

$$y - \frac{y_1 + y_2}{2} = -\frac{1}{m_1} \left(x - \frac{x_1 + x_2}{2} \right)$$

$$\text{or, } y_{1p} = -\frac{1}{m_1} \left(x - \frac{x_1 + x_2}{2} \right) + \frac{y_1 + y_2}{2} \quad (2.2)$$

and L_2 is,

$$\begin{aligned} y - \frac{y2 + y3}{2} &= -\frac{1}{m_2} \left(x - \frac{x2 + x3}{2} \right) \\ \text{or, } y_{2p} &= -\frac{1}{m_2} \left(x - \frac{x2 + x3}{2} \right) + \frac{y2 + y3}{2}. \end{aligned} \quad (2.3)$$

Lines L_1 and L_2 intersect at the center of the circle O ($\because y_{1p} = y_{2p}$). From Eq.(2.2) and Eq.(2.3),

$$\begin{aligned} -\frac{1}{m_1} \left(x - \frac{x1 + x2}{2} \right) + \frac{y1 + y2}{2} &= \\ -\frac{1}{m_2} \left(x - \frac{x2 + x3}{2} \right) + \frac{y2 + y3}{2} \\ \text{or, } -\frac{x}{m_1} + \frac{x1 + x2}{2m_1} + \frac{y1 + y2}{2} &= \\ -\frac{x}{m_2} + \frac{x2 + x3}{2m_2} + \frac{y2 + y3}{2} \end{aligned} \quad (2.4)$$

Solving Eq.(2.4), the center of the circle O (O_x, O_y) and radius R_L are obtained as,

$$\begin{aligned} O_x &= -\frac{m_1 m_2 (y3 - y1) + m_1 (x2 + x3) - m_2 (x1 + x2)}{2(m_1 - m_2)} \\ O_y &= \\ &= -\frac{1}{m_1} \left(-\frac{m_1 m_2 (y3 - y1) + m_1 (x2 + x3) - m_2 (x1 + x2)}{2(m_1 - m_2)} \right. \\ &\quad \left. - \frac{x1 + x2}{2} \right) + \frac{y1 + y2}{2} \\ R_L &= \sqrt{(x_i - O_x)^2 + (y_i - O_y)^2}, \quad i \in 1, 2 \text{ or } 3. \end{aligned} \quad (2.5)$$

Once the radius of the larger circle R_L is found, value of the radius of smaller rolling circle r_s can be found. In case of astroid, $r_s = \frac{R_L}{4}$. Curves $\widehat{AB}$ and $\widehat{BC}$ shown in red in Fig.2.4a can now be found easily from Eq.(2.1).

In the case of Fig.2.4b too, smooth path can be obtained from the geometry of an astroid. The steps for curve generation are similar to those described above. As shown in Fig.2.4b, any traversal with a turn across the point o would encounter a sharp turn. Even in case of the paths across the blue lines representing line segments $\overline{AB}$, $\overline{BC}$, $\overline{CD}$, or $\overline{DA}$, the robot has to endeavour sharp turns. However, smooth path is generated along the astroid-curves $\widehat{AB}$, $\widehat{BC}$, $\widehat{CD}$, and $\widehat{DA}$ shown in red.

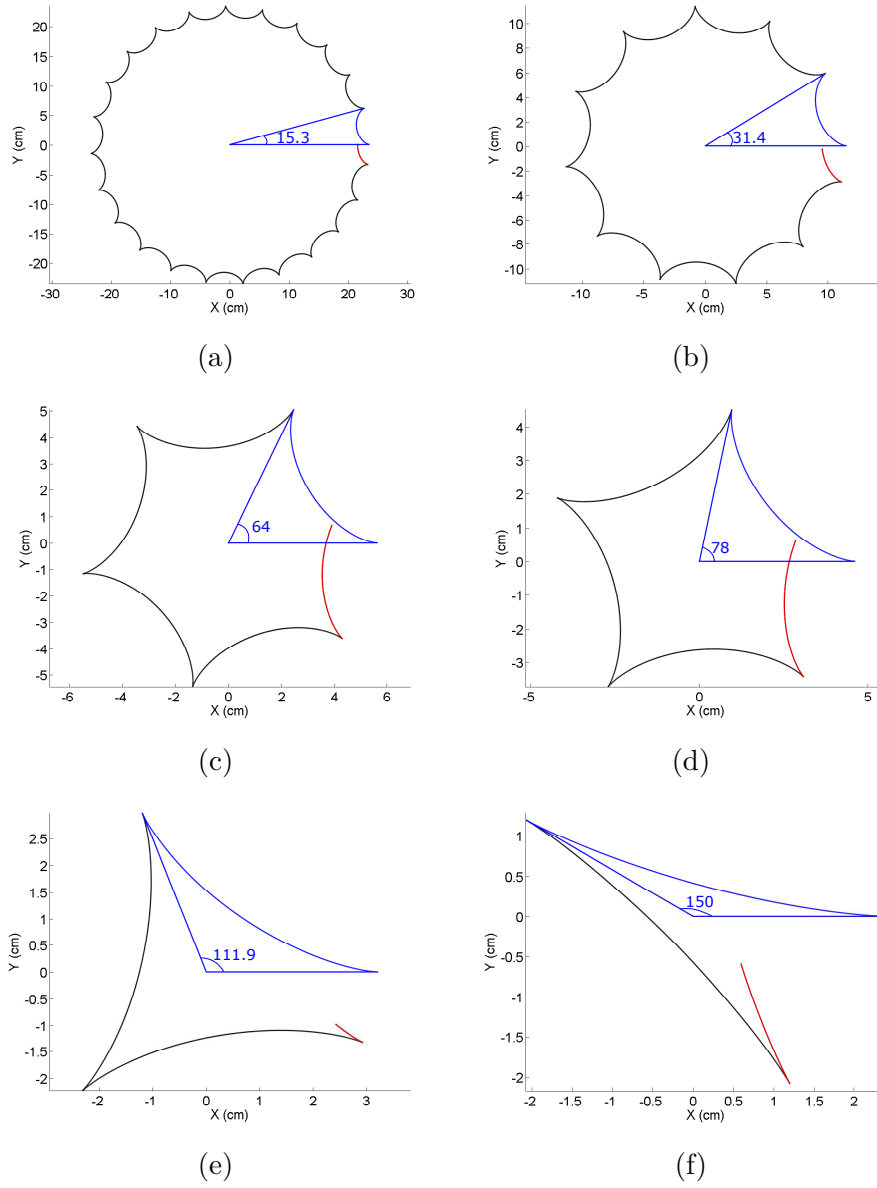


Fig. 2.6 SHP generation with different angles. The first segment of the curve in blue is taken to generate SHP, whereas other curve segments in black and red are ignored. The red curve segment shows the portion of the hypocycloid which fails to close on the start point as the angle does not perfectly divide 360 degrees. (a) $\theta = 15.3^\circ$. (b) $\theta = 31.4^\circ$. (c) $\theta = 64^\circ$. (d) $\theta = 78^\circ$. (e) $\theta = 111.9^\circ$. (f) $\theta = 150^\circ$.

For the corner section of the map, a section of astroid or deltoid can be used for smooth path generation. For example, in case of Fig.2.4c, which represents a corner section, only a section of astroid across curve $\widehat{AB}$ is used (and other curves omitted). Notice

that, in case of Fig.2.4a, the curves $\widehat{AB}$ and $\widehat{BC}$ were also generated from an astroid by using the two lower curves and omitting the upper curves.

The case when the clearance thresholds for different obstacles are not equal, i.e. $\delta_i \neq \delta_j, i \neq j$, is now considered. Although a maximum unique value of clearance threshold δ_{max} is desired, the value may vary in case of parts of the map which are too narrow and δ_{max} clearance is not possible, and the actual clearance value is smaller than δ_{max} . Such a case is described in Fig.2.5a and Fig.2.5b. A simple and straightforward way could be to get the minimum of $\delta_i, i \in \{1, 2, \dots, n\}$, and generate curves with that δ_{min} . However, it would be inefficient as some paths would have lesser clearance where more clearance was possible. In Fig.2.5a, clearance $\delta_1 \neq \delta_2$ and the obstacle on the left side is a bit upper than the one on the right side. In this case, each curve is individually formed, taking individual clearance threshold in consideration. For the curve $\widehat{BC'}$, point A' is assumed and curve $\widehat{BC'}$ is generated using only δ_2 clearance, and points A', B and C' . Similarly, point B' is assumed and curve $\widehat{AC}$ is generated with δ_1 clearance, with points A, B' , and C . Even though the line $\overline{B'C}$ is closer to the obstacle and clearance δ_2 is not obeyed, it is permitted as the curve $\widehat{B'C}$ is auxiliary and will not be a part of the actual path. The final and actual path generated has curves $\widehat{AC}, \widehat{BC'}$ and line $\overline{AB}$. Similarly, in Fig.2.5b, individual curves are generated for each clearance δ_i , individually.

In all the cases, it can geometrically be shown that the new obstacle clearance $\Delta_i > \delta_i$, in Fig.2.4a, 2.4b, 2.4c, 2.5a, and 2.5b. Hence, the path is not only smooth but also safe and avoids collision with environmental obstacles. Each of the points (A, B, C, D) formed after diffusion is referred as a 'node'.

2.3.1 SHP for Different Angles

Skeleton maps can generate paths in which segments make different angles which are not necessarily 90 degrees. SHP can generate smooth paths for all the angles. For any angle θ , the total number of cusps are given by $\frac{360}{\theta}$ (θ in degrees). If $\frac{360}{\theta}$ is a whole number, then the hypocycloid curve closes on itself, as in case of astroid ($\theta = 90$ deg). If $\frac{360}{\theta}$ is a fraction, the curve does not close on itself. However, irrespective of whether the hypocycloid closes on itself or not, the first hypocycloid curve segment generated is used, and the rest of the curve segments are discarded. SHP curves for different angles are shown in Fig.2.6. In all the cases of Fig.2.6, the first blue curve segment is taken to generate SHP, whereas other segments are ignored. In all the cases, the red curve is

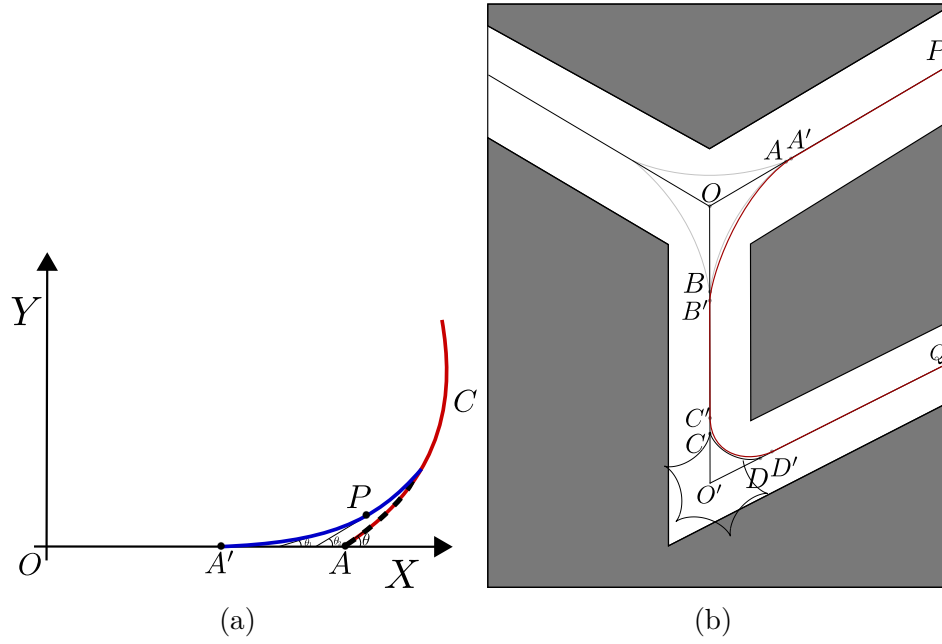


Fig. 2.7 Applying transition curves for smoothing joints between straight segments and curve. (a) A transition curve (blue) gradually decreasing curvature from infinite (on line) to that of SHP (red). (b) Employing transition curves with SHP in environment to get smooth path in red.

the last curve segment which fails to close on the starting point of the hypocycloid. It is also discarded.

First, the angle between the two line segments (θ_{max}) is calculated from any two points (x_1, y_1) and (x_3, y_3) on respective line segments, and the intersection point or the node (x_2, y_2) . These three points form a triangle. From cosine rule, the angle θ_{max} is calculated as,

$$\theta_{max} = \cos^{-1} \left(\frac{a^2 + c^2 - b^2}{2ac} \right),$$

where,

$$\begin{aligned} a &= \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2}, \\ b &= \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2}, \\ c &= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \end{aligned} \tag{2.6}$$

The first segment of the SHP curve is generated using the parametric equation given by Eq.(2.1), by setting the parameter η (number of cusps) to $\frac{360}{\theta_{max}}$ (ex. For astroid, $\theta_{max} = 90^\circ$, hence $\eta = 4$. For $\theta_{max} = 131.94^\circ$, $\eta = 2.728$). The value of θ is then varied

from 0 to $\frac{\theta_{max}\pi}{180}$ in small increments of θ (say 0.01, depending upon the density of the points desired on SHP with smaller increments producing denser points). Since the value of the angle is started θ from zero, the coordinate system employed needs to be transformed to the local coordinates accordingly.

2.3.2 SHP Curvature, and Continuity

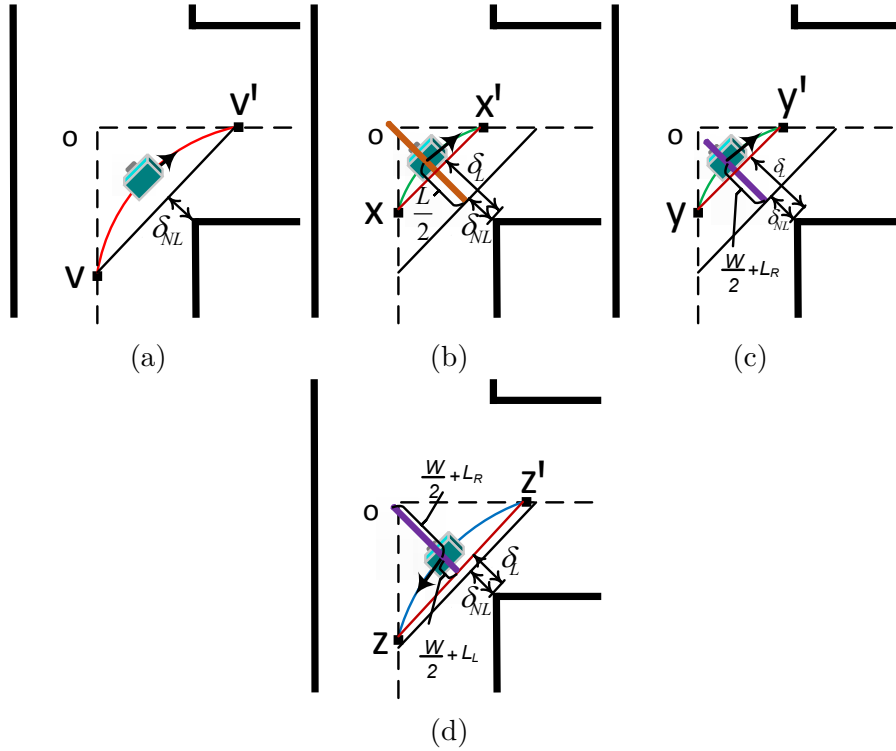


Fig. 2.8 Determining clearance threshold δ in case of load carrying robot. (a) No load. (b) Centered load. (c) Un-centered load (robot turning right). (d) Un-centered load (robot turning left).

SHP curvature is generated using parametric equations described in Eq.(2.1) for a given clearance distance. If ϕ is the angle from the center of the larger circle to that of the smaller circle, then curvature $c(\phi)$ is defined as [191],

$$c(\phi) = \frac{2R_L - r_s}{4r_s(R_L - r_s)} \operatorname{cosec}\left(\frac{a\phi}{2r_s}\right). \quad (2.7)$$

Arc length ($s(\phi)$) can then be found by integrating over the curve γ the differential displacement dl as,

$$\begin{aligned} s(\phi) &= \int_{\gamma} |dl| \\ &= \frac{8(R_L - r_s)r_s}{R_L} \sin^2\left(\frac{R_L\phi}{4r_s}\right), \end{aligned} \quad (2.8)$$

and the tangential angle as,

$$\phi_{\theta}(\phi) = \left(1 - \frac{R_L}{2r_s}\right)\phi. \quad (2.9)$$

From Eq.(2.7), the curvature of a hypocycloid gradually decreases to a minimum value, and then gradually increases again to meet a point on the straight segment which makes SHP ideal for curve generation between two points on the segments of skeleton paths. Compare this to a clothoid [189] whose curvature is equal to its arc length which, in turn, is equal to the parameter θ . This property makes clothoids a good choice for global smoothing of curves [54, 99]. However, as θ increases, clothoids degenerate quickly and are difficult to fit between points on the skeleton paths. Moreover, unlike hypocycloids, clothoids are generally defined by Fresnel integral[190] which are transcendental functions and unlike algebraic functions does not satisfy a polynomial equation. In case of robots with an on-board computer, these are computationally expensive to compute compared to hypocycloid which can be computed fast. This could also be a critical factor in real time manoeuvre of robot to avoid obstacles while traversing a newly computed smooth path.

The continuity of the curves is often discussed geometrically. A $G0$ continuous path connects all the points between the start and the goal location such that there is no gap. A $G1$ continuous path also preserves the tangency between the points. In other words, the path matches first differential values at each point in the path. A $G2$ continuous path preserves the second differential values at each point in the path. In general, a curve has C^n continuity if it's n^{th} derivative $\frac{d^n s}{dt^n}$ is also continuous. In terms of robot motion, the $G1$ continuous motion preserves velocity, whereas $G2$ continuous path preserves acceleration.

Many previously proposed path smoothing techniques [20, 21, 43, 73] try to smooth out the global path, for example, by interpolating global points (calculated by PRM or other techniques) from start to goal location. This approach preserves the geometric

continuities. However, most of these points also lie on the straight passages, and the path un-necessarily ends up in a wavy form, coming close to one of the walls of the corridors. The global path smoothing techniques proposed in these works are quite interesting. However, most of these works assume robot to be a point entity, and issues such as topological admissibility are completely ignored. Particularly in case of narrow passages, they generate paths which are dangerously close to walls and undesired for robot motion. This is discussed in detail in Section 2.11.

In this chapter, SHP is introduced as a local path smoother, and not a global one. In other words, in all the aforementioned examples, only the turns in the skeleton path were smoothed out. Whereas, the already existing straight line segments of the skeleton map were untouched. This is desirable because, in a straight corridor, a robot is expected to move on a straight path keeping equal distance from walls on both the sides.

SHP preserves $G1$ continuity, and SHP segments are tangential to the straight segments. However, $G2$ continuity is not guaranteed. This is not a problem within the bounds of the robot travelling at low speed (as also experimentally verified in results in Section 7.5). In case of robot travelling with high speed with acceleration on SHP $G2$ continuity is desired, and the transition between the straight line and the curved section, although sufficient for most practical cases of indoor robots moving at low speeds, can be further smoothed out. This is a well studied problem and various solutions are available. In fact, smoothing out a straight section of track to a curved section while maintaining curvature continuity using transition curves [72] has extensively been studied for building railway tracks and highway roads. The transition curves are briefly discussed, as their solutions in parametric forms can easily be obtained and computed.

Transition curves [72] connect a straight segment of the path at one end to the curve at the other end. Hence, radius of curvature changes from zero on the straight segment to a finite value of the curve, at a uniform rate. It eliminates the kink generated by directly connecting the straight and the curve section. Transition curves have the following important properties desirable for robot motion: (a) They are tangential to the straight line of the path, i.e. curvature at start is zero. (b) It joins the circular curve tangentially. (c) It's curvature increases at the same rate. Transition curves have rigorously been studied for $G2$ continuity in many works [2, 3, 63], and can also provide smooth $G2$ transition between two straight lines.

Figure 2.7a shows a curved path in red meeting the straight segment at point A. To generate transition curve, point A is shifted back to A' which is the starting point of

the transition curve shown in blue. Although different types of curves can be used to generate transition curves, Euler's curve are the most common[72]. Taking into account linear increase of curvature from zero on the straight segment to the curvature of the hypocycloid, rate of change of angle at any point (P) on the transition curve of length L (shown in blue in Fig.2.7a) is given by,

$$\frac{d\theta}{ds} = \frac{s}{R_L \cdot L}, \quad (2.10)$$

where s is the length of the transition curve from point A' to P in Fig.2.7a. Integrating Eq.(2.10), we obtain,

$$\theta = \frac{s^2}{2R_L \cdot L}. \quad (2.11)$$

During the initial condition (on straight segment) when $s = 0$, θ is also 0. Apart from polar coordinates, the equation of transition curve in parametric form is given as[72, 170],

$$\begin{aligned} x &= s - \frac{s^5}{40R_L^2L^2}, \\ y &= \frac{s^3}{6R_L} - \frac{s^7}{336R_L^3L^3}. \end{aligned} \quad (2.12)$$

Fig.2.7b shows SHP result with transition curves from point P to Q. In Fig.2.7b, $\angle AOB = 120^\circ$, and hence a deltoid curve is used. Similarly, $\angle CO'D = 64^\circ$ and a 6 cusps hypocycloid is used. Initial points of contact A, B, C, and D, have been shifted to A', B', C', and O', respectively. The amount of shift is determined experimentally and is larger for smaller values of θ_{max} , and vice versa. In order to minimize this shifting, the SHP curve itself gets slightly shifted downwards[72], but is a good approximation of the original curve. Moreover, it is not computationally expensive to generate transition curves which can easily be computed from the parametric equation (Eq.(2.12)).

2.4 Determining Clearance Distance (δ)

Many path planning techniques ignore the practical case of a mobile robot carrying load. However, it is important, and both the cases to calculate the value of the threshold clearance distance δ when the robot is not carrying any load, and when the robot is carrying some load are discussed. Since the sensors mounted on the robot are prone to

errors, the robot motion is uncertain, and hence the clearance threshold is increased by an additional distance δ_{buf} . Clearance threshold δ_{NL} in case of no load is given as,

$$\delta_{NL} = \frac{W}{2} + \delta_{buf}, \quad (2.13)$$

where W is the width of the robot. Fig.2.8a shows the case of robot which carries no load. The clearance threshold (δ_L) in case of a load carrying robot is given as,

$$\delta_L = \frac{W_{eff}}{2} + \delta_{buf}, \quad (2.14)$$

where, the effective width (W_{eff}) of the robot needs to be calculated for the following three cases:

1. Small Load: It implies the case when the load (of length L) is small enough to be within the frame of the robot, i.e. $L < W$. This case is considered to be similar to robot not carrying any load and $\delta_L = \delta_{NL}$.
2. Centered Load: It implies the case when the load is centered on the robot. In other words, a load of length L has dimensions $\frac{L}{2}$ on either side of the center point of the robot. Fig.2.8b depicts such a situation. The effective width of the robot is equal to half of the load length (i.e. $W_{eff} = \frac{L}{2}$).
3. Un-centered Load: It implies the case when the load is not centered on the robot. In other words, a load of length L has on the dimensions greater than $\frac{L}{2}$ on any of the sides from the center point of the robot. This situation is depicted in Fig.2.8c and Fig.2.8d. Notice that, the effective width of the robot would change according to the direction of motion of the robot and the position of the obstacle, and is the total load length hanging out from the center of the robot frame. In case of Fig.2.8c, load length is more on the right side of the robot, and the obstacle is on the right side too. Hence, the effective robot width $W_{eff} = |\vec{L}_{dir}| + \frac{W}{2}$, where $|\vec{L}_{dir}| = |\vec{L}_R|$ represents the load length on the right side of robot frame. In case of Fig.2.8d, the robot obstacle is on the left hand side of the robot, and the load length on the left side is smaller than the right side (i.e. $|\vec{L}_L| < |\vec{L}_R|$). The effective width of the robot is also therefore smaller from the point of view of the obstacle in this case.

In a nutshell, the effective width is given by,

$$W_{eff} = \begin{cases} \frac{W}{2} & W \geq L & \text{Centered Load} \\ \frac{L}{2} & L > W & \text{Centered Load} \\ \frac{W}{2} + |\vec{L}_{dir}| & L > W & \text{Un-centered Load} \end{cases} \quad (2.15)$$

The difference of δ_L and δ_{NL} is given as,

$$\begin{aligned} \delta_L - \delta_{NL} &= \frac{W_{eff}}{2} + \delta_{buf} - \left(\frac{W}{2} + \delta_{buf} \right) \\ &= \frac{W_{eff}}{2} - \frac{W}{2} \\ &= |\vec{L}_{dir}| + \frac{W}{2} - \frac{W}{2} = |\vec{L}_{dir}|. \end{aligned} \quad (2.16)$$

In order to create SHP with load carrying robot, first δ_L is calculated and the nodes are diffused until the perpendicular distance between the line joining the diffused nodes and the obstacle is less than δ_L . Once the diffusion stops, hypocycloidal paths are calculated as explained in Section 2.3. Notice that, in case of Fig. 2.8c, the robot turns at a point ‘y’ generating a smaller curve, to maintain δ_{NL} clearance. Whereas, in case of Fig. 2.8d, the same robot takes a turn at z’, as the load length is smaller on the left side. In both the cases, it can be seen that δ_{NL} clearance is observed. In all cases, centrifugal forces are ignored for small velocity of robots.

2.5 Multi-Robot Path Planning on SHP

A decentralized multi-robot path planning is proposed. Each robot R_i , $i \in \{1, 2, \dots, n\}$ has a communication unit and is able to send and receive messages from other robots encountered on the way. Since the hypocycloidal paths need to be computed only once for the entire environment, it is assumed that the entire map of the surrounding is known to the robots beforehand along with the smoothed hypocycloidal paths and nodes. Each robot R_i knows only about its own start S_i and goal G_i configuration, and has no knowledge about the configuration of other robots. Moreover, each robot has a computational unit to calculate paths, and sensors to localize itself in the map, and detect the presence of other robots in vicinity. Task priority (Φ_i) may be assigned to robot R_i . If not, then priorities are assumed to be equal.

Listing 2.1 A sample motion policy

```

1: {"policy": {           // policy file
2:   "robot-id": "03",    // id of robot
3:   "start": "<10,30>",   // start coordinates
4:   "goal": "<90,82>",    // goal coordinates
5:   "dim": "<R:30,H:10>", // robot dimensions
6:   "actions": [         // actions
7:     {"GS": "s": "<10,30>", "g": "<10,50>", "node": "Q5"},
8:     {"TR": "s": "<10,50>", "g": "<15,55>", "node": "Q7"},
9:     {"GS": "s": "<15,55>", "g": "<70,55>", "node": "Q8"},
10:    ... ,
11:    ... ,
12:    {"GS": "s": "<80,82>", "g": "<90,82>", "node": "Q9"}
13:    {"ST": "CC"}]       // stop, communicate
14:  }
15: "cache": "Q6,Q7,Q8" } // recently cached nodes

```

The entire scheme is decentralized and once the start and goal configurations of robots have been set, each robot R_i calculates the path from S_i to G_i . Given a map with obstacles and free path, there are various algorithms to calculate the global path from S_i to G_i , such as A^* algorithm [66], D^* algorithm [171, 172], shortest path algorithm, probabilistic roadmap planner (PRM) [85], or rapidly exploring random tree (RRT) [94, 96]. Each algorithm has its own merits in terms of computational/space/time complexity, accuracy etc. and a complete description of these algorithms can be found in various literature sources [36, 95]. Any of these path planning algorithms can be employed by robot R_i . It is also possible for different robots to use different algorithms for path planning. Each path planning algorithm generates a path from S_i and G_i and this path is used to generate a motion ‘*policy*’ of the robot R_i . A *policy* is defined as the set of actions which a robot must take sequentially from S_i in order to reach the goal G_i . For example, actions of a *policy* may comprise of : go-straight (*GS*), turn-left (*TL*), turn-right (*TR*), go-back (*GB*), retreat (*RT*), stop (*ST*), and communicate (*CC*). Complicated actions may also be carried out based on robot’s dynamics and programming. A minimal JSON listing of policy (P_i) of robot (R_i) is given as an example in Listing 1.

Many algorithms like A^* or D^* algorithm optimize mainly in terms of the cost it takes to go from start to goal, and generate paths which are sometimes impractically close to the obstacles in the environment. Hence, the paths generated by these algorithms

are used to generate only the motion *policy* of the robot. The actual path traversed by the robot is the smooth curved path comprising of nodes.

Once the shortest path (or, safest path, depending upon the algorithm) has been computed and motion policy determined, a robot can start its motion. Other robots may or may not start motion simultaneously. Two robots with different start and goal configurations may have intersecting paths. In such a case, the nodes which are formed during the generation of hypocycloidal curves after the diffusion step are used by the robot as points where the robot can rest momentarily in the map to avoid collision with other robot on intersecting path. When a robot R_i with motion policy encounters another robot R_j with policy P_j on the path, they exchange a dictionary which comprises of: (1) start location (2), (b) goal location (G), (3) task-priority (Φ), and (4) dimensions of robot(D), as key-value pairs.

$$P_i = \{\{S : val\}, \{G : val\}, \{\Phi : val\}, \{D : val\}\}$$

With this minimum information, the two robots can calculate the path policies of each other and calculate the common intersecting path ($P_i \cap P_j$) and common nodes with respect to the current position. The robot with lower priority will retreat to one of the nearby nodes which does not lie in the path of the robot with higher task priority.

Once the robot with lower priority has moved out of the intersecting path, the other robot will resume its motion. If the priorities are equal, or if there is no priority assigned, other schemes can be employed to prioritize a robot. For example, the ratio of the length of path already traversed by the robot and the length of the path still to be traversed can be calculated, and the robot with higher ratio can be prioritized, or vice versa. Centers of deltoid or astroid can also be used as retreat points for the robots provided that their area (A_{hypo}) is safely lesser than the area of the robot.

Area of the hypocycloid is given by,

$$A_{hypo} = \int_0^{2\pi} Y(\theta) \left(\frac{\partial}{\partial \theta} X(\theta) \right) d\theta.$$

$\therefore$ Area of deltoid, $A_{del} =$

$$\begin{aligned} &= \int_0^{2\pi} R_L \frac{2(\sin)(\theta) - (\sin)(2\theta)}{3} \left(\frac{\partial}{\partial \theta} \frac{R_L(2(\cos)(\theta) + (\cos)(2\theta))}{3} \right) d\theta \\ &= \frac{2}{9} \pi R_L^2. \end{aligned}$$

Similarly, area of astroid, $A_{ast} = \frac{3}{8} \pi R_L^2$ (as, $\xi = \frac{1}{4}$)

In both cases, R_L is defined by Equation 2.5.

It may happen that the lower priority robot has no nearby node to retreat. This leads to a situation called as a deadlock in which both the robots are stuck and neither retreats. In this case, the robot with higher priority retreats to the nearest non-colliding safe node to avoid deadlock condition.

2.5.1 Node Caching and Policy Exchange

Calculation of the nearest safe node is avoided by ‘caching’ the nodes. Each robot *caches* the location of the nearest unoccupied node locations as it traverses the path. Hence, when the robot encounters a collision situation with other robot, the *cached* safe node location is immediately available, and a lot of computation is saved. Moreover, instead of calculating the overlapping paths ($P_i \cap P_j$) between the two robots, the entire motion policies of the two robots themselves may also be exchanged, which would save computation, but at the cost of more data communication.

2.6 Dynamic Obstacles and Map Update

Path planning is generally done taking the static obstacles of the map into consideration. However, there can also be dynamic obstacles in the map, which must be avoided by the robot. Avoidance of a mobile robot and humans has been proposed in various works like [64, 82]. Instead, in this work, dynamic obstacles implies the obstacles which are not the part of static obstacles, but which were placed later or displaced in the environment, for example, a chair, or a box. Unlike moving people and pet animals whose positions keep on changing in the environment, there is a high probability of other robot to encounter the same dynamic obstacle at the same position. A robot can detect dynamic obstacle in the map and calculate its coordinates in the global map with the sensors mounted on it. JSON format was used to save the dynamic obstacle’s coordinates with a Unix time-stamp as follows:

Listing 2.2 Dynamic Obstacle Map Update

```

1: {"dynamic-obstacle": {           // Obstacle meta-data
2:   "robot-id": "01",              // Id of robot
3:   "time-stamp": "1440233037"     // Unix time-stamp
4:   "obstacles": [                 // Coordinates
5:     {"obj1": {"x": "786", "y": "99", "l": "20", "w": "73"}},
6:     {"obj2": {"x": "529", "y": "23", "l": "32", "w": "52"}}
7:   ]
}
```

This JSON file is transferred to another robot, which parses the file using a JSON parser, and updates the coordinates of the obstacles in its map. This scheme is advantageous because of two main reasons:

- a) Other robots are informed only about the new obstacles coordinates, and they can update their maps instead of having to build the map by themselves. Only bare minimum information is communicated instead of the entire map. The other robot can plan its path by taking into account the position of the new obstacles. This is efficient not only in terms of the time saved to reach the destination, but also in terms of computation and battery power.
- b) The time-stamp ensures that the latest information about the obstacles is used to update the map, as time-stamps can be compared easily.

Notice that, the presented JSON file for dynamic obstacle map update has only geometric coordinates of the two dynamic obstacles, as only laser range sensor was used. In case of other sensors like visual sensors, other attributes of the dynamic obstacle, like color, could also be easily saved and communicated to other robots. Using time-stamps require that the clocks of the different robots are synchronized beforehand.

2.7 System Specifications

Two types of robots were used in the experiments. (1) Pioneer P3DX [131] equipped with UHG08LX laser range sensor [183], and (2) Kobuki [182] which is equipped with Microsoft Kinect sensor. Both the robots are differential drive kind of robots. Scanning area of UHG08LX sensor is 270 degrees with 0.36 degrees/pitch. The detectable range varies from 0.02 to 8 m. The Kinect sensor consists of an RGB camera, 3D depth sensor, a multi-array microphone, and a motorized tilt. In this work only the depth sensors are used to provide the input data for the Turtlebot. Its range is specified to be between 0.7 and 5 meters. The field of view is 57 degrees horizontal, 43 degrees vertical, with a tilt range of ± 27 degrees. Both the sensors are known to have a good performance, and are well suited for indoor environments.

Each robot was equipped with a system running on 64 bit Ubuntu Linux 14.04 operating system. Both the robots were run on ROS [135] (Robot Operating System), which has capability to perform inter-node communication between the robots. The grid map of the environment was also created from depth data using ROS. To parse JSON files, the open source JSMN parser [202] was used due to its small code footprint and no dependencies.

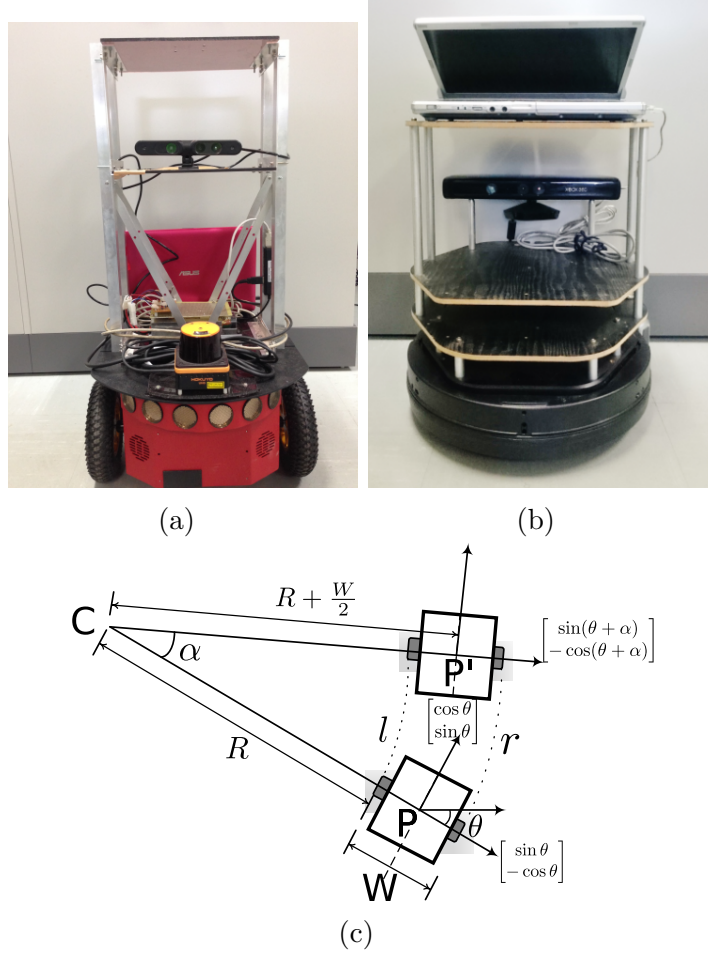


Fig. 2.9 Robots used in the experiments. (a) Pioneer P3DX. (b) Kobuki Turtlebot. (c) Motion model of two wheel differential drive robots.

2.8 Motion Model

Both Pioneer and Turtlebot used in the experiments are two wheeled differential drive robots. Hence, the motion for a two wheeled differential drive robot without slippage on surfaces with good traction is modelled. The distance between the left and the right wheel is W , and the robot state at position P , is given as $[x, y, \theta]$. From Fig.2.9c, turning angle α is calculated as,

$$\begin{aligned}
 r &= \alpha \cdot (R + W), \\
 l &= \alpha \cdot R \\
 \therefore \alpha &= \frac{r - l}{W}
 \end{aligned} \tag{2.17}$$

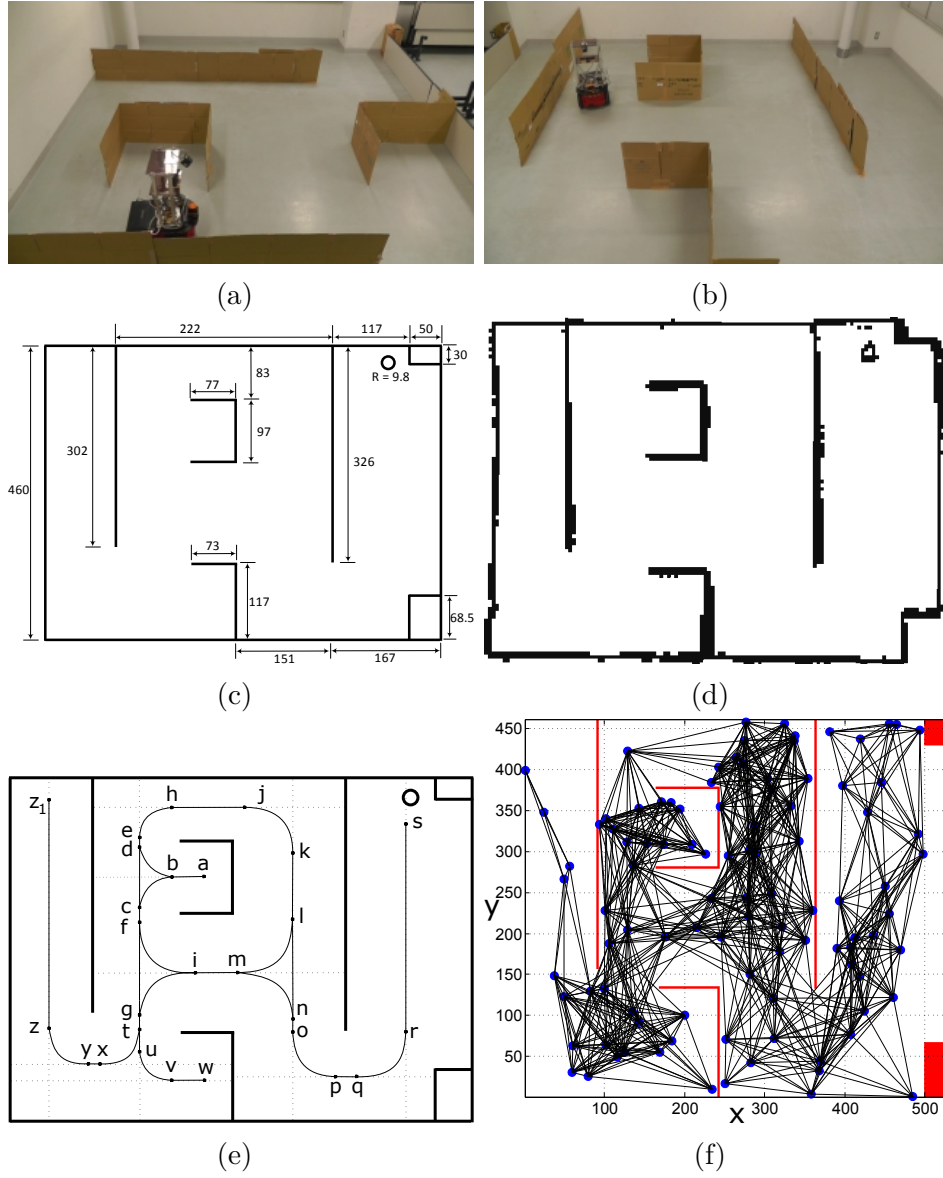


Fig. 2.10 Test environment with dimensions, grid-map, SHP nodes, and PRM paths. (a) Environment (front-view). (b) Environment (side-view). (c) Dimensions of the map. (d) Corresponding grid map. (e) SHP path with nodes. (f) PRM result on grid map.

and the radius of turn R as,

$$R = \frac{l}{\alpha}, \alpha \neq 0. \quad (2.18)$$

The coordinates of the center of rotation (C , in Fig.2.9c), are calculated as,

$$\begin{bmatrix} Cx \\ Cy \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \left(R + \frac{W}{2}\right) \cdot \begin{bmatrix} \sin\theta \\ -\cos\theta \end{bmatrix} \quad (2.19)$$

The new heading θ' is,

$$\theta' = (\theta + \alpha) \bmod 2\pi, \quad (2.20)$$

from which the coordinates of the new position P' are calculated as,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} Cx \\ Cy \end{bmatrix} - \left(R + \frac{W}{2}\right) \cdot \begin{bmatrix} \sin\theta' \\ -\cos\theta' \end{bmatrix}, \quad \alpha \neq 0 \implies r \neq l. \quad (2.21)$$

If $r = l$, i.e. if the robot motion is straight, the state parameters are given as,

$$\theta' = \theta, \quad (2.22)$$

and,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + l \cdot \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \quad (l = r). \quad (2.23)$$

2.9 Results

This section presents the results of smooth path generation, collision avoidance, deadlock avoidance, smooth path traversal in case of load carrying robot, dynamic obstacle avoidance with map update, and SHP with different δ .

For the sake of conciseness and readability, the type of robot, its priority, start and goal nodes are represented as: $\mathbf{C}_{\text{tur}}^{\text{H}} :: \mathbf{S} \rightsquigarrow \mathbf{G}$, where $\mathbf{S}$ and $\mathbf{G}$ are the start and goal node locations, respectively. Subscript ‘tur’ denotes a Turtlebot, and ‘pio’ denotes Pioneer. H and L represents high and low priority, respectively.

Figure 2.10a shows the environment used for the experiments. Figure 2.10b shows the side view of the structured indoor environment with regular non slippery surface. The dimensions of the map are shown in Fig. 2.10c. The robot was moved in the environment and grid map shown in Fig. 2.10d was constructed beforehand. A skeleton map is constructed from the grid map and the nodes are diffused as explained in Section 2.3 to generate hypocycloidal paths with the nodes (marked a,b,c,...) are shown in Fig. 2.10e.

The probabilistic roadmap of the environment is shown in Fig.2.10f. Both the robots had the map and node information beforehand.

2.9.1 Experiment 1: SHP traversal and comparison with D* path traversal

Pioneer is initialized with configuration $C_{pio}^H :: c \rightsquigarrow s$. The nodes ‘c’ and ‘s’ are shown in Fig.2.10e. The D* algorithm calculates a path from start to goal which is very close to the walls of the environment with sharp turns, as shown in Fig.2.11a. Figure 2.11 shows the smooth hypocycloidal path traversed by the robot in green. For comparison, it also shows the D* path traversed by the robot. It can be seen that SHP path is not only smooth but also maintains a safe distance from the obstacles.

2.9.2 Experiment 2: Collision Avoidance on SHP nodes

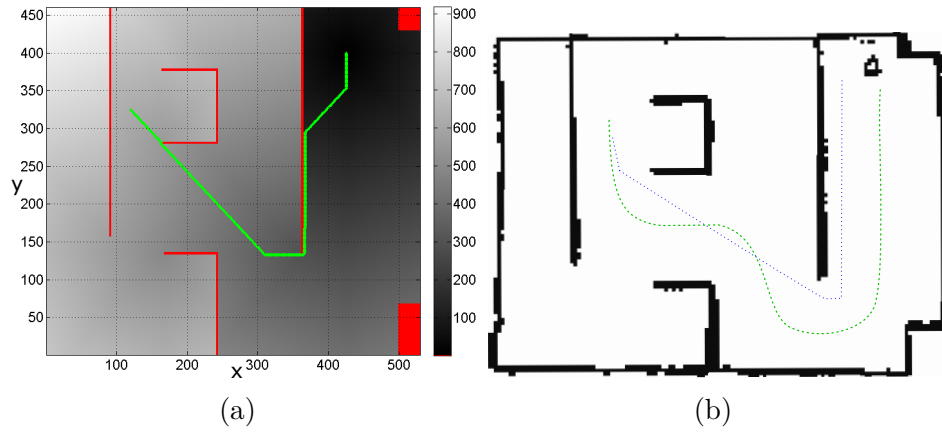


Fig. 2.11 Experiment 1: SHP vs D* path traversal. (a) D* path from start node ‘c’ to goal node ‘s’. Intensity of the color represents proximity to the goal. (b) Dotted green line shows the traversed SHP path, and dotted blue shows the traversed D* path. D* path is very close to the obstacles, whereas SHP is not. *Please refer to Movie No. 1, 2, and 3 in Table 2.3.*

Case-A

In case-A, the start and goal locations of the each robot are different. Pioneer is initialized with configuration $C_{pio}^H :: s \rightsquigarrow w$, whereas, Turtlebot with configuration $C_{tur}^L :: k \rightsquigarrow s$ (Fig.2.12a). Turtlebot and Pioneer stop after sensing each other (Fig.2.12b) on the common path, and the low priority Turtlebot retreats at the nearest cached node ‘l’ (Fig.2.12c). Once Pioneer crosses over, it instructs the Turtlebot from node

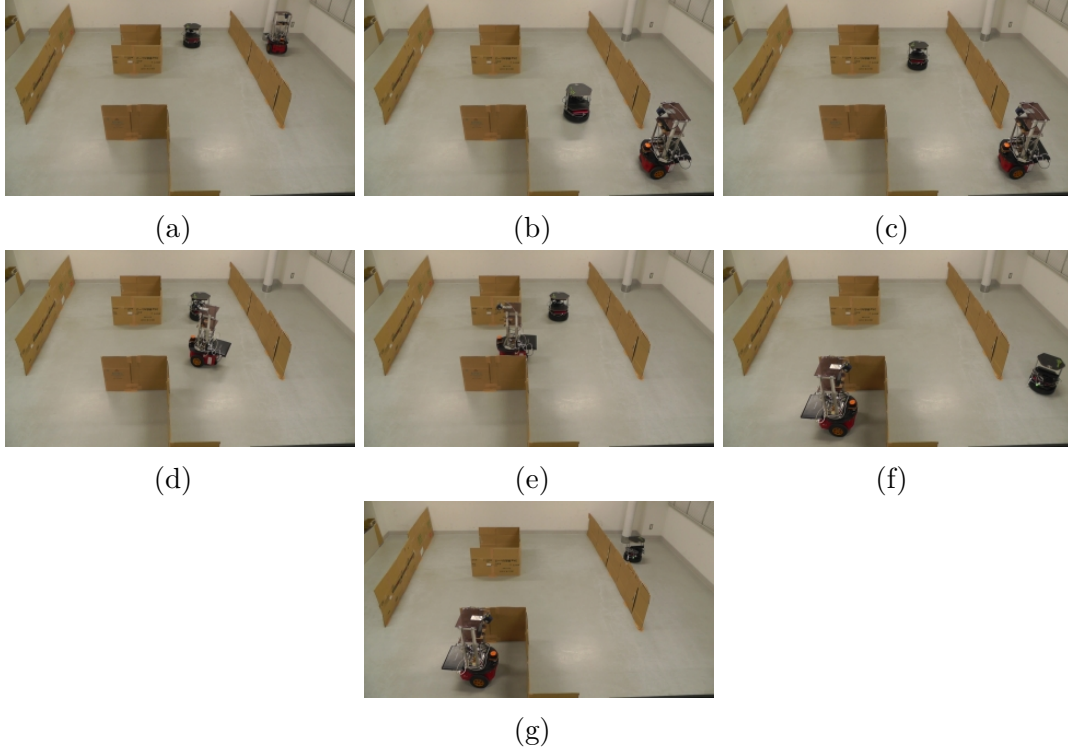


Fig. 2.12 Experiment 2: (Case A) Collision Avoidance on SHP nodes. (a) Initial configuration of robots. (b) Collision detection. (c) Low priority Turtlebot retreats to cached safe node ‘l’ (Fig.2.10e). (d,e) P3DX continues towards its goal. (f,g) Turtlebot moves towards its goal when the path is clear. *Please refer to Movie No. 4 in Table 2.3.*

‘m’ to go ahead on the cleared path (Fig.2.12e). Finally, both the robots stop at their respective goal locations (Fig.2.12g).

Case-B

In case-B, the start location of each robot is set as the goal location of the other robot, i.e. $C_{tur}^H :: k \rightsquigarrow s$, and $C_{pio}^L :: s \rightsquigarrow k$ (Fig.2.13a). Similar to case-A, both the robots stop after sensing the presence of other robot in the way (Fig.2.13b). Although Turtlebot has higher priority, it still retreats to node ‘m’ to avoid a deadlock (Fig.2.13c), as there is no safe node for Pioneer to retreat which is not in the way of Turtlebot. Once Pioneer crosses over, it instructs the Turtlebot from node ‘l’ to go ahead on the cleared path (Fig.2.13d). Finally, both the robots stop at their respective goal locations (Fig.2.13g).

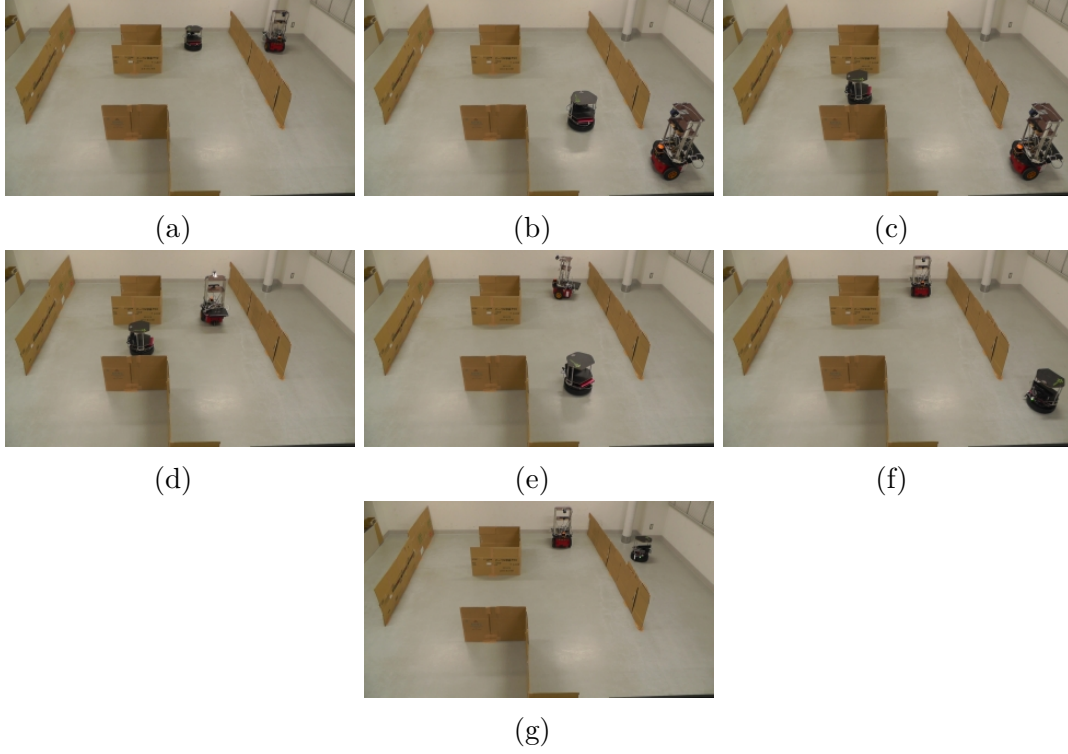


Fig. 2.13 Experiment 2: (Case B) Collision Avoidance on SHP nodes. (a) Initial configuration of robots. (b) Collision detection. (c) High priority Turtlebot retreats to cached safe node ‘m’ (Fig.2.10e), as P3DX has no non-overlapping safe node to retreat. (d,e) P3DX continues towards its goal. (f,g) Turtlebot moves towards its goal when the path is clear. *Please refer to Movie No. 5 in Table 2.3.*

2.9.3 Experiment 3: Avoiding Deadlock on SHP

A deadlock situation can occur when a robot with low priority has no place to retreat in order to clear the path for a robot with high priority. The initial configurations were set as: $\mathcal{C}_{\text{pio}}^H :: a \rightsquigarrow s$, and $\mathcal{C}_{\text{tur}}^L :: s \rightsquigarrow w$. In order to achieve a condition of deadlock, Turtlebot was programmed to start 5 seconds after the other robot has started. The deadlock situation is shown in Fig.2.14b where there is no safe node for the Turtlebot to retreat. Hence, Pioneer retreats to Turtlebot’s non-colliding node ‘1’, and proceeds to its goal once the path has been cleared. The robots at the goals are shown in Fig.2.14g. Notice that a similar situation of deadlock was also reached in case-B of experiment 2. In that case too, the robot with high priority retreated to avoid a deadlock.

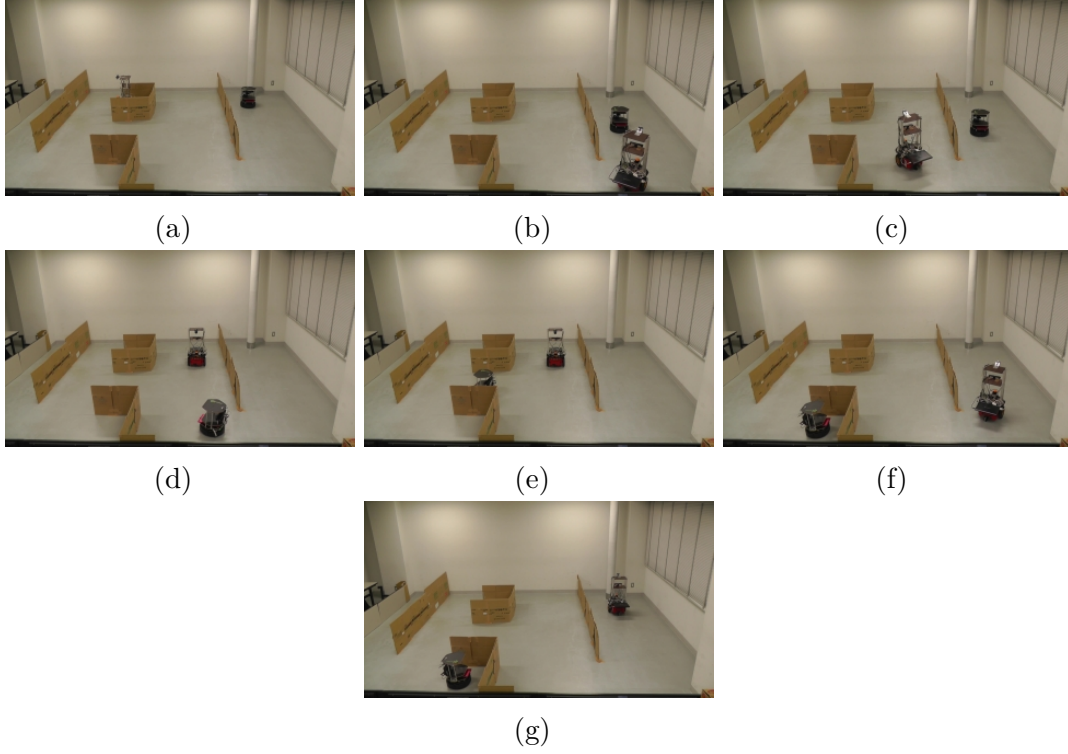


Fig. 2.14 Experiment 3: Avoiding Deadlock on SHP. (a) Initial configuration of robots. P3DX starts early. (b) Collision detection. (c) High priority P3DX retreats to cached safe node 'l' (Fig.2.10e), as low priority Turtlebot has no non-overlapping safe node to retreat. (d,e) Turtlebot continues towards its goal. (f,g) P3DX moves towards its goal when the path is clear. *Please refer to Movie No. 6 in Table 2.3.*

2.9.4 Experiment 4: Dynamic Obstacle Avoidance on SHP

Robot paths can easily be planned with static obstacles of the map. However, obstacles can also be dynamic and path planning may have to be done in real time. To test dynamic obstacle avoidance, Pioneer was initialized with $\mathcal{C}_{\text{pio}}^{\text{H}} :: \mathbf{w} \rightsquigarrow \mathbf{s}$ and the return to the original position, i.e. $\mathcal{C}_{\text{pio}}^{\text{H}} :: \mathbf{s} \rightsquigarrow \mathbf{w}$ (Fig.2.16a), and there were no dynamic obstacles in the environment. The motion policy generated by the robot at this point is shown in the PRM map of Fig.2.15b in green, based on which the SHP path's goal nodes are: $\mathbf{v}, \mathbf{u}, \mathbf{t}, \mathbf{g}, \mathbf{l}, \mathbf{m}, \mathbf{n}, \mathbf{o}, \mathbf{p}, \mathbf{q}, \mathbf{r}$ and $\mathbf{s}$ as shown in Fig.2.10e. As soon as the robot starts motion, two obstacles X1 and X2 are manually placed in the environment shown in Fig.2.15a. The robot stops after sensing the obstacle (Fig.2.16c) and a new motion policy is computed from the PRM map which is shown in Fig.2.15c, and the corresponding SHP from Fig.2.10e. The robot takes a detour across the new path (Fig.2.16d - 2.16h).

The position of the obstacles in the map is shown in Fig.2.15a. The dimensions of the two obstacles (**width** $\times$ **breadth** $\times$ **height**) were:

- Obstacle X1 : $32 \times 21 \times 73$ cm, and
- Obstacle X2 : $69 \times 27 \times 34$ cm.

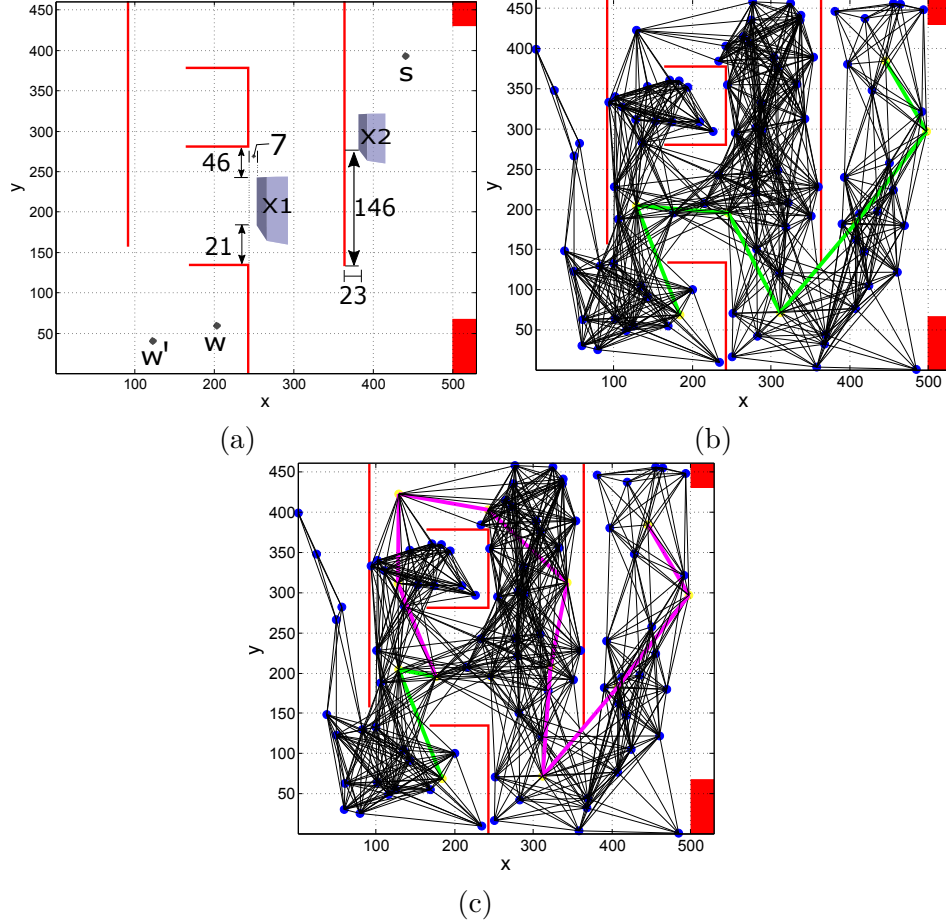


Fig. 2.15 Experiment 4: Dynamic Obstacle Avoidance on SHP. (a) Position of the two obstacles placed dynamically in the map. (b) Initial path calculated by PRM before dynamic obstacles were placed. (c) New path calculated after sensing the dynamic obstacles.

Pioneer updates the map with the new obstacle locations, and once it returns to its start node w , it updates the Turtlebot stationed at node w' (Fig.2.15a) about the positions of the obstacles in the map. Later, when the Turtlebot is initialized with $C_{tur}^H :: w' \rightsquigarrow s$, it calculates the obstacle free path towards the goal s shown in Fig.2.16i-2.16l using the updated map with new obstacles.

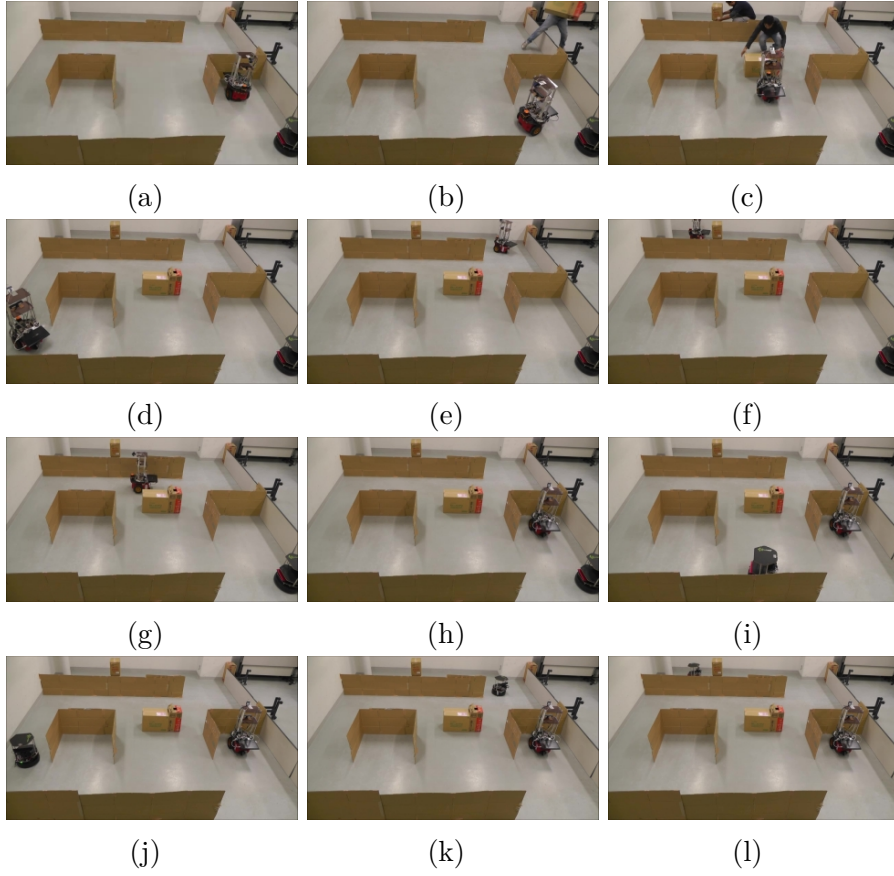


Fig. 2.16 Experiment 4: Dynamic Obstacle Avoidance on SHP. (a) Initial configuration. (b) P3DX moves on the initially calculated short path. (c) P3DX stops sensing the dynamic obstacles. (d) P3DX calculates new path and moves. (e,f,g) Positions of the new obstacles are updated in the map. (h) P3DX transfers the coordinates of the new obstacles to Turtlebot. (i,j,k) Turtlebot calculates the path with the updated map and moves. *Please refer to Movie No. 7 in Table 2.3.*

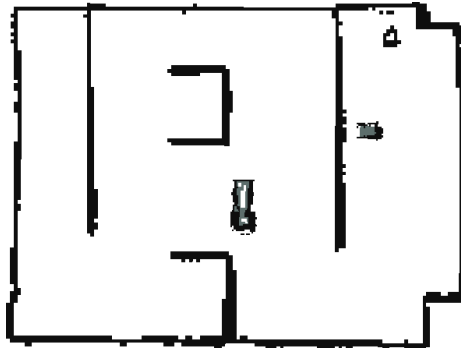


Fig. 2.17 New grid-map with dynamic obstacles.

2.9.5 Experiment 5: SHP with different clearance threshold (δ)

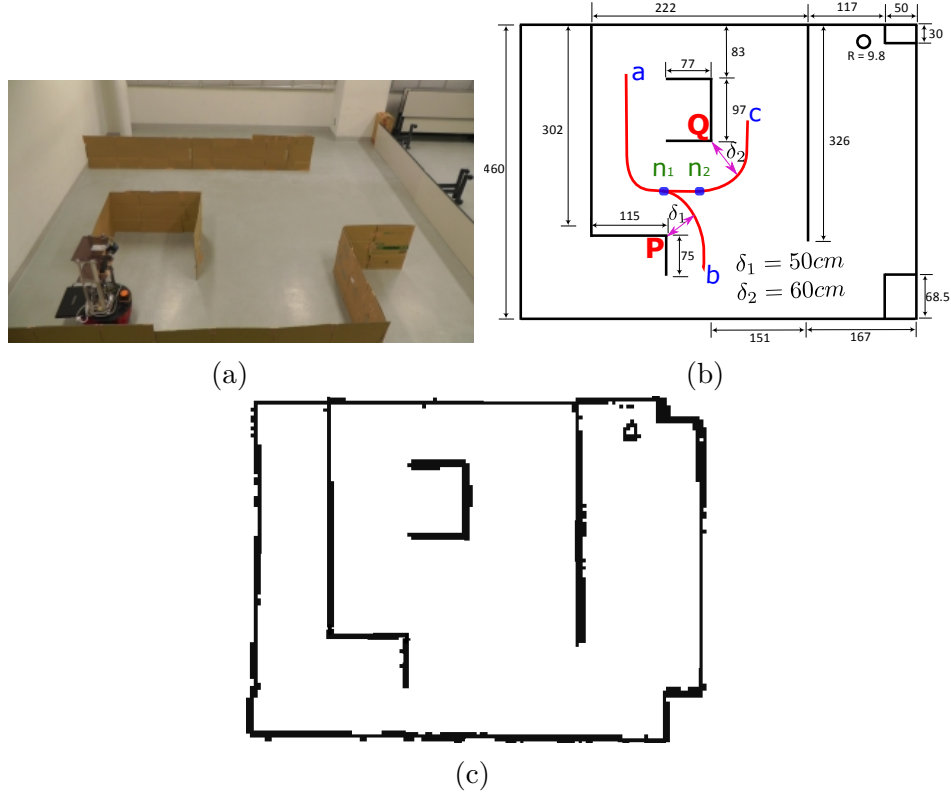


Fig. 2.18 Experiment 5: SHP with different clearance threshold (δ). (a) Test environment. (b) Dimensions of the environment. Obstacle points P and Q are not collinear. For configuration $C_{\text{pio}}^{\text{H}} :: a \rightsquigarrow b$, P3DX takes a turn at node n_1 maintaining $\delta = 60\text{ cm}$. For configuration $C_{\text{pio}}^{\text{H}} :: a \rightsquigarrow c$, P3DX takes a turn at node n_2 maintaining $\delta = 70\text{ cm}$. (c) Grid map of the environment. *Please refer to Movie No. 8 in Table 2.3.*

SHP generation with different thresholds ($\delta_i \neq \delta_j, i \neq j$) was earlier shown in Fig.2.5a and explained in Section 2.3. A different and modified environment shown in Fig.2.18a was used to test SHP with different clearance thresholds. The dimensions of the new environment are shown in Fig.2.18b, and the corresponding gridmap is shown in Fig.2.18c which was also made beforehand. The same robot was first initialized with configuration $C_{\text{pio}}^{\text{H}} :: a \rightsquigarrow b$, and then with $C_{\text{pio}}^{\text{H}} :: a \rightsquigarrow c$. In both the cases, as the corner points P and Q are not collinear and δ are different, robot takes turns at different points of SHP and maintains the clearance distance.

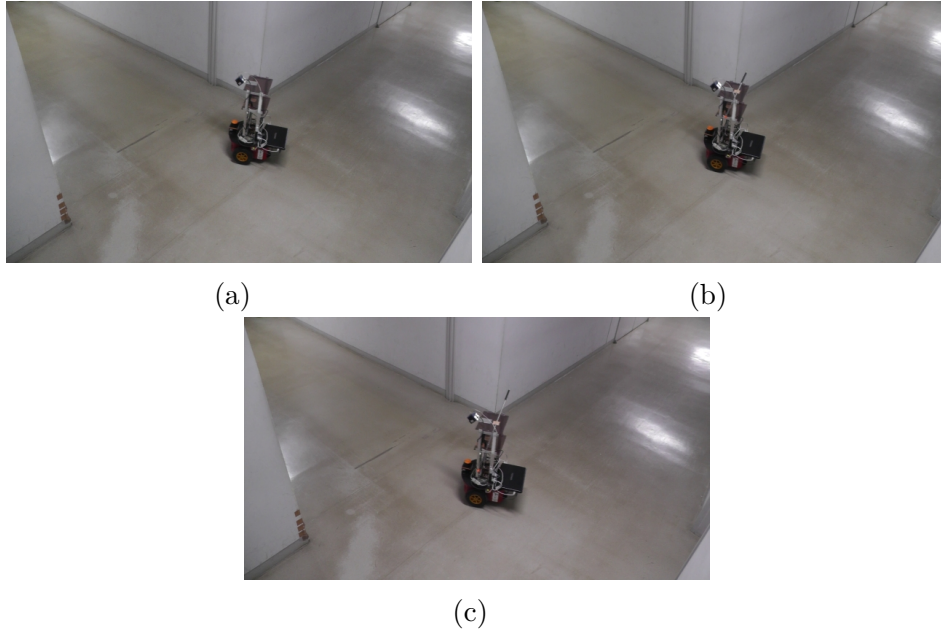


Fig. 2.19 Experiment 6: Obstacle Avoidance on SHP with Load Carrying Robot. (a) No load. (b) Load length (centered) = 75 cm. (c) Load length (centered) = 115 cm. *Please refer to Movie No. 9 in Table 2.3.*

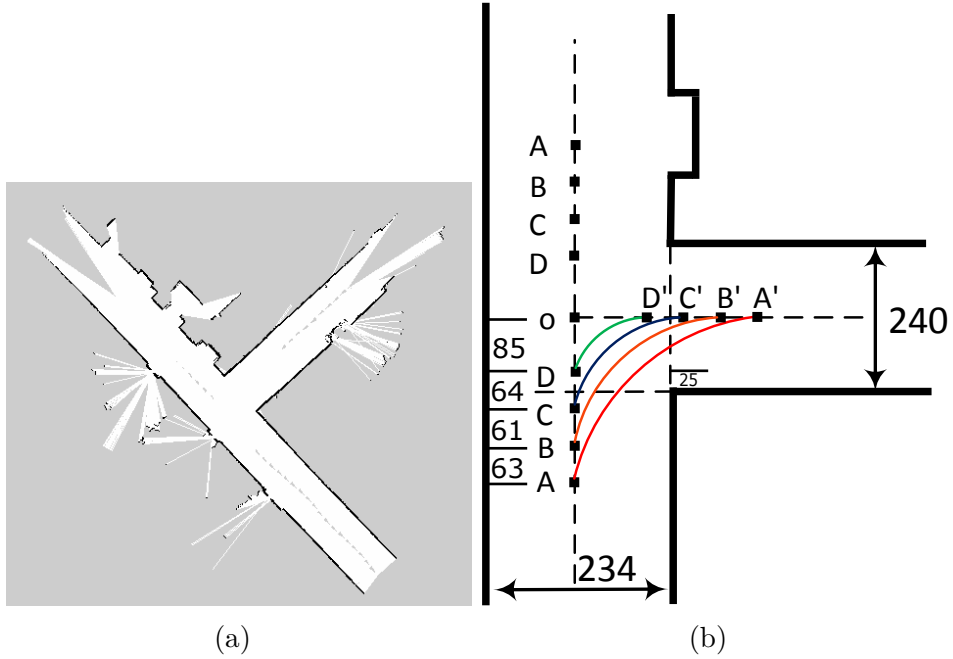


Fig. 2.20 Experiment 6: Obstacle Avoidance on SHP with Load Carrying Robot. (a) Grid map of the test environment shown in Fig.2.19. (b) Map dimensions with different SHP curves for different values of δ .

2.9.6 Experiment 6: Obstacle Avoidance on SHP with Load Carrying Robot

The practical case of robot carrying a load is also considered. A stick is fixed on the robot to simulate a load. Three experiments were performed with varying stick lengths of: (a) 0 cm, i.e. no stick, (b) 75 cm (medium load), and (c) 115 cm (longest load). The clearance distance δ was set to 60 cm. Fig.2.19 shows the passage environment used for the experiment. Fig.2.20a shows the grid map of the environment. In the first experiment with no load (stick), the robot followed SHP across the curve $\widehat{BB'}$. When the load is medium size, the robot follows SHP curve $\widehat{CC'}$, and for the longest load it follows the curve $\widehat{DD'}$. The positions of the nodes are shown in Fig.2.20b. The top view of the robot in the three scenarios is shown in Fig.2.19a, Fig.2.19b, and Fig.2.19c. In all the three cases, the clearance of the robot was greater than clearance distance $\delta = 60\text{cm}$, as summarized in Table 2.1. In all the experiments, the size of the load (stick) mounted on the robot was manually set .

2.10 Movies of Results

The respective movies of all the experimental results that are presented in this work can be downloaded from the internet. The details of the movies have been summarized in Table 2.3.

2.11 SHP Comparison With Other Path Planning and Smoothing Techniques

SHP was compared with the collision-free QPMI (Quadratic Polynomial and Membership Interpolation) [20], along with D* [172] path planning, and PRM [85]. A smooth and collision-free path is proposed by S.R. Chang and U.Y. Huh in [20] which

Table 2.1 Load carrying robot on SHP with load width and actual clearance. For turn nodes, refer Fig.2.20b.

S.No	Load Width	Turn Node	Clearance
1	No Load	'A'	'67 cm'
2	75 cm	'C'	'61 cm'
3	115 cm	'D'	'65 cm'

uses interpolation [111]. They first use PRM algorithm to obtain linear waypoints on which QPMI[73] algorithm is applied. However, the smooth path obtained from QPMI algorithm may have points which collide with the obstacles, and hence they require a further collision check. If collisions are detected, they create a collision free smooth path by creating a perpendicular line between the colliding point and the linear piecewise path joining the two PRM waypoints. The details of the algorithm can be found in [20, 73].

In order to compare the proposed SHP with the collision-free QPMI, the simulation environment (shown in Fig.2.21a) in the works of S.R. Chang and U.Y. Huh[20] was replicated using the same toolkit [25] as used in their work. The intermediate Skeleton map to create SHP is shown in Fig.2.21b for comparison and explanation purpose. Similar to the work in [20], the start location was fixed at coordinates (5,5), and the goal location at (95,95). Fig.2.21c shows the results of paths generated by D*, PRM, QPMI[20], and SHP are shown in dotted green, red, black, and blue colors, respectively. The path generated by D* algorithm is too close to the obstacles. Even if the path is shifted by a distance equal to the width of the robot and some buffer, the path will still be angular. In the case of PRM path, there are points (P_1 , P_2 , and P_3) which are close to the obstacles. Moreover, the direct PRM path is not smooth. QPMI generates a smooth path. The circles on the QPMI path are the waypoints generated by the PRM algorithm which are used to generate a smooth path using interpolation. There are points (Q_1 , Q_2 , and Q_3) in the QPMI path where free space is available but still the robot comes close to the obstacles. On the other hand, SHP path is not only smooth and continuous, but also maintains a better clearance from the obstacles. Originally, the point Q_3 is shifted right in the QPMI algorithm after collision detection (for details, refer [20], page 9). However, considering the dimensions of the robot itself, it is still close. If the straight passage between points A and B of the map is considered, the robot should maintain an equal distance between the two walls, to avoid coming close to either side of the walls. A wavy motion in the straight passage will obviously bring the robot close to one side of the wall at some points, and is therefore not desired. QPMI algorithm ends up generating undesired wavy paths in straight passages as it uses all the waypoints generated by the PRM algorithm which are further smoothed using interpolation. On the other hand, SHP uses skeleton maps (as shown in Fig.2.21b) which already generates straight pathways (passage AB shown in Fig.2.21b) in corridors, and optimizes the paths at corners for smoothing using node diffusion. QPMI algorithm requires an explicit and computationally expensive collision

detection and trajectory changing steps which are not required in SHP. Moreover, SHP is very fast and the SHP generation for the map shown in Fig.4.4c was completed in 48 milliseconds without using any parallel programming. The time required for various sub-modules is summarized in Table 2.2. A parallel implementation would further reduce the computation time.

The diffused nodes in SHP can also act as points where a robot must retreat to avoid collision with other robots in case of multi-robot collision free path planning, which are not considered in QPMI, D*, or PRM. Moreover, neither of D*, PRM, or QPMI algorithms consider the practical case of obstacle avoidance with load carrying robot, which is addressed in SHP. The major advantage of the proposed scheme is that it eliminates the need for a centralized controller for multiple robots. However, it is expensive in terms of waiting time as the robot has to wait on the safe nodes upon retreat. The overall time required to complete the tasks by multiple robots may also be larger than centralized methods. However, since the decision to retreat and detour is taken in real time by local communication, it eliminates the need to recalculate the path of all the robots in case of dynamically changing the goal location (or tasks) of multiple robots. A limitation of the proposed SHP in its pure form is that for robots with high velocities, further smoothing of transition from the straight line segment to the SHP curve for G^2 continuity may be required to be computed, for which using transition curves was proposed.

Table 2.2 Execution Time for SHP Generation (for 100×100 pixel map of Fig.2.21a). Implementation in C++ on 1.80GHz Intel Core-i7-4500CPU

Function	Time
Gridmap Binarize	3.0 ms
Gridmap Erode	3.2 ms
Gridmap Dilate	3.2 ms
Skeletonization (Algorithm of [204])	15.4 ms
Node Clustering	13.6 ms
SHP Curve Generation on Nodes	9.5 ms
TOTAL	47.9 ms

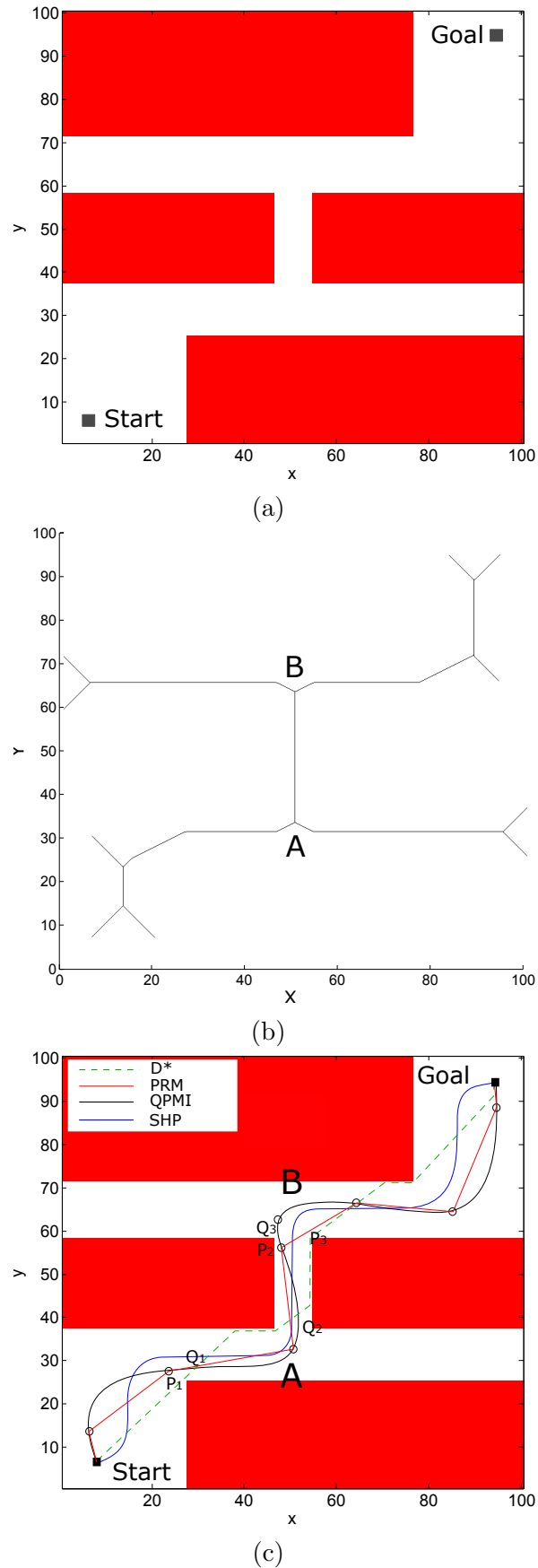


Fig. 2.21 SHP result comparison with QPMI [20], PRM, and D*. (a) Simulation environment. (b) Skeleton map. (c) Paths generated by QPMI, D*, PRM, and SHP.

Table 2.3 Movie list of all the experiments in this work.

Movie No.	Exp	Explanation	Download URL
1, 2, and 3	Exp 1	SHP traversal (front view) SHP traversal (side-view) D* traversal	https://youtu.be/bvVEwbr4FWU
4 and 5	Exp 2	Collision Avoid (Case A) Collision Avoid (Case B)	https://youtu.be/hVb6JUDLq7I
6	Exp 3	Deadlock avoid- ance	https://youtu.be/IdJys3LbL4c
7	Exp 4	Dynamic obsta- cle avoidance	https://youtu.be/RYnLK3TA19Y
8	Exp 5	SHP with differ- ent δ	https://youtu.be/QDjWeO5SuEo
9	Exp 6	SHP with load carrying	https://youtu.be/Hn2jJ50Zu4w

This study considered favourable surfaces with good wheel traction for the indoor wheeled robots used in the experiments. Moreover, ROS was used to control the wheeled robots. However, other types of surfaces needs to be considered, particularly for outdoor robots. On slippery surfaces, wheel encoders will give a false data which would adversely affect robot's localization in the map and ultimately the path planning. An estimation of slip can help prevent errors in localization, as discussed in [200]. Path planning considering wheel slip is discussed in [80]. For rough surfaces, the control system of the robot needs to be robust enough to cover the taxonomy of soil, rubble, grass, and water. Path planning for rough surfaces in discussed in [168, 203]. A particularly interesting case with regards to the presented work is to consider wheel slip when robot carries a load on slippery surfaces, as also discussed in [201]. SHP evaluation on different types of surfaces is considered as a potential future work.

2.12 SHP as an Extension for Various Global Path Planners

SHP based path smoothing is not limited only to skeleton paths. In fact, paths made from any of the global path planners like A*, D*, PRM, or RRT planners can be smoothed using SHP. In this section, SHP is presented as an ‘extension’.

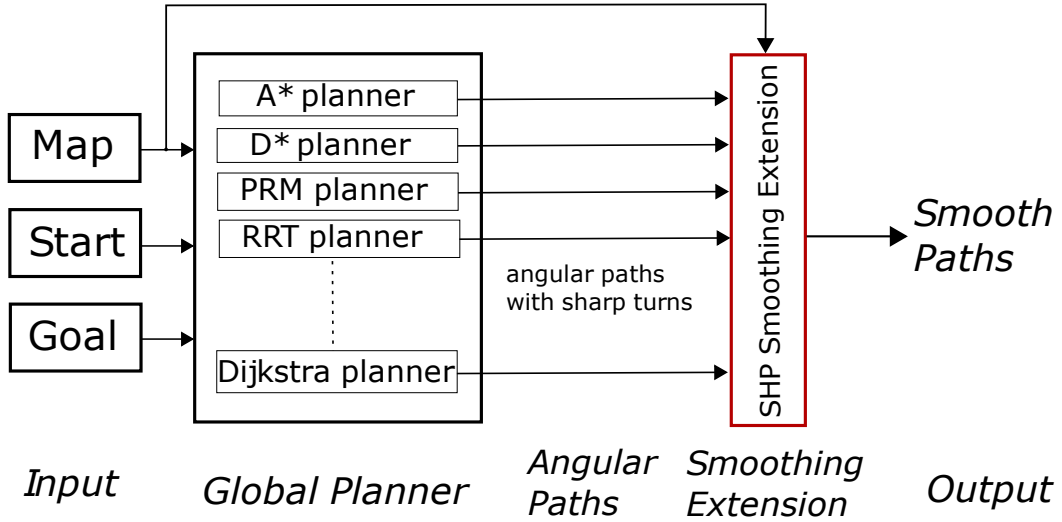


Fig. 2.22 SHP as an extension for various global path planners.

As shown in Fig.2.22, the input to the proposed extension are the map with the start and goal locations. The map can be represented in any form, but should indicate the obstacles and free areas in some form or the other. The start and goal locations are provided as spatial coordinates. This information is given to any of the global planners which generates a path which has angular turns. This path and the map is given as an input to the proposed extension and it smooths out the angular portions of the path generating a smooth path for the robot.

The node generation process is same as that explained in Section 2.3. In case of global paths generated by other planners, the end points of the paths needs to be detected first.

Figure 2.23 shows a typical global path generated by PRM, or D* path planners. These algorithms generate a sequence of n points $P = \{P_1, P_2, \dots, P_n\}$, where $P_i = (x_i, y_i)$ represents the spatial coordinate in the two dimensional XY-plane. It can be seen that a robot traversing this path from A to E encounters sharp turns at points B, C, and D. Since the proposed extension smooths out the turns, it is necessary to detect these turns beforehand. The entire path from A to E consists of several line segments

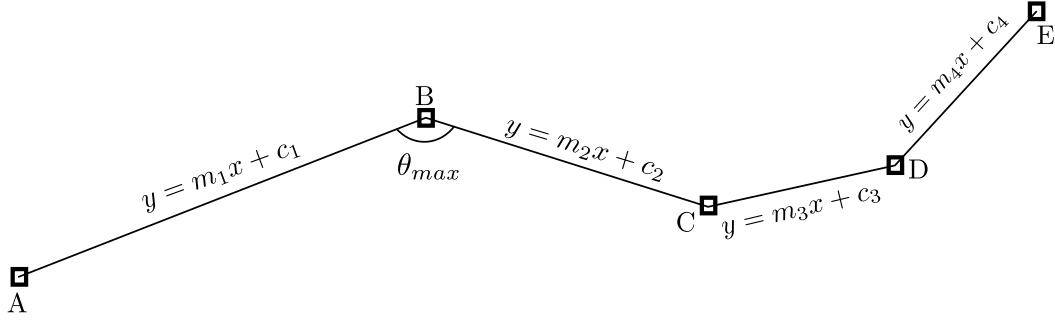


Fig. 2.23 A global path with sharp turns at B, C, and D.

$\overline{AB}$, $\overline{BC}$, $\overline{CD}$, and $\overline{DE}$. Each of the four line segment is represented by,

$$y = m_i \cdot x + c_i, \quad i \in \{1, \dots, 4\}, \quad (2.24)$$

where m_i and c_i are the slopes and the intercepts of the four lines. Depending upon the accuracy of the global planner employed, if the section of m points P_i between the points A and B ($m < n$) is considered, they may or may not accurately lie on the same line $\overline{AB}$. This may arise due to rounding-off errors, or the precision scale (for ex. single precision or double precision floating point) employed. However, considering a correction range $\pm \lambda_{th}$, the m points will lie on the straight line $\overline{AB}$ within this correction range. The equation of line $\overline{AB}$ can be obtained by considering a seed point (starting point) A ($P_i = (x_i, y_i)$), and the next point ($P_2 = (x_{i+1}, y_{i+1})$). Equation of $\overline{AB}$ is,

$$y = y_i + \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right) (x - x_i) \pm \lambda_{th}. \quad (2.25)$$

Next, a check is performed if the subsequent point P_{i+2} also lies on the same line given by Eq.(2.25). If P_{i+2} lies on the same line, i is incremented and the next point (P_{i+3}) is checked if it lies on the same line. If a point P_q ($q < n$) does not lie on the same line given by Eq.(2.25), it indicates the start of a new line segment. It also means that the last point on the previous line segment is P_{q-1} ($q < n$). The seed point is set to P_q , and the next point is set to the point P_{q+1} . This process is continued for all n points, and the turning points are determined. The points of turns are the ‘nodes’. This process is described in the form of a flowchart in Fig.2.24.

Other techniques like Hough Transform can also be used to detect the corners. Once the turn points are found out, the angles are calculated using the inverse cosine law

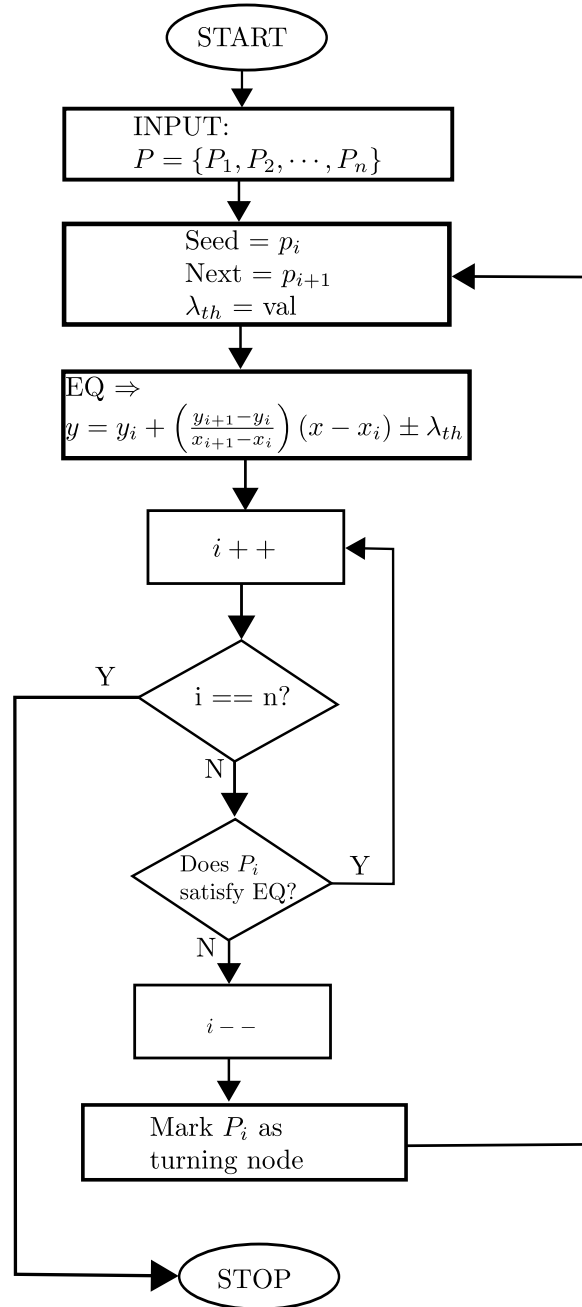


Fig. 2.24 Flowchart of turn-point detection from angular paths.

and the smooth paths are generated as explained in Section 2.3. Thus, any global angular path can be smoothed out using SHP.

2.13 Summary

This chapter presented smooth hypocycloidal paths (SHP) for robot motion. SHP paths are at a safe distance from the environmental obstacles and avoid sharp turns, unlike the paths generated by D* algorithm or PRM algorithm. A mathematical description of node diffusion with SHP generation was provided under different kinds of obstacles, in both the cases when the robot is carrying load, and when it is not. The work introduced a novel motion policy in parsable JSON format for robots which could be exchanged with other robots for collision avoidance in a multi-robot scenario. In the proposed decoupled scheme to avoid multi-robot collision, robots communicate locally and decide to crossover or retreat to cached safe nodes to avoid collision. It was shown that the nodes of the curved paths can be used as points for the robot to retreat in order to avoid collision. The cases of robot retreat with different priorities was discussed. In case of dynamic obstacle, a decoupled approach is used to update other robots with the coordinates of the new and dynamic obstacles in the map. The proposed techniques were tested in real environments with multiple robots, and compared with other path planning techniques. Experimental results show that the proposed SHP technique can generate smooth paths desired for robot motion with collision avoidance of multiple robots with different start and goal configurations. SHP was proposed as an extension and paths made from any of the global path planners like A*, D*, PRM, or RRT planners can be smoothed using SHP. Robot motion on SHP in case of moving entities like people is considered as future work. An extension of SHP for creating smooth 3D trajectories, and SHP evaluation on different types of surfaces is also under work.

Chapter 3

Sharing Limited Charging Docks with Multiple Mobile Service Robots

Overview

This chapter deals with the problem of common resource sharing in multi-robot systems. As discussed in the Chapter 1, conflicts must be resolved in a multi-robot system when multiple robots are trying to access a common resource. The resource could be a charging point, a narrow path, or any other resource. Relevant parameters must be considered when resolving the conflict and the most appropriate robot must be given access to the resource. As it is evident from Fig.1.3 which shows the interdependence of various modules of a multi-robot system, resource sharing depends on SLAM (Chapter 7) and path planning module (Chapter 2). The path planning module was discussed in previous chapter and SLAM module are assumed to be available in this chapter.

Chapter Abstract

Robot docking stations are of utmost important for service robots like security robots and automated guided vehicles in warehouses which are in continuous operation, and hence require frequent recharging. While a lot of research has been done on designing the actual hardware of the docking station itself, a practical situation of intelligently managing a limited number of docking stations in case of multiple robots has largely been ignored. A docking station manager for multiple mobile service robots is proposed, which intelligently allots a mobile robot to the most appropriate docking station, based

on the robot's task priority, location awareness of the robot and the docking station, power left in robot, and request order. It is demonstrated, through experiments in real environment, that the proposed manager can function intelligently and resolve conflicts when the docking stations are fully occupied. The proposed station manager is experimentally tested in different conditions of varying task priorities, power levels, and emergency conditions of the robots, and found to be robust to intelligently allocate the most appropriate robot with the appropriate charging dock.

3.1 Introduction

Robots have successfully been applied for task automation in industries and public places like shopping malls, hospitals, and warehouses. Service robots used in tasks like surveillance and patrolling often needs to be operational for long hours and an effective power management is very important. Most of the robots are equipped with one or multiple rechargeable batteries, and can be operational for a particular duration. Mobile service robots needs to be recharged if the battery power levels are low. Manually recharging the service robots is a very tedious task, especially, in case of multiple robots. Hence, robot docking stations are important for multiple service robots to ensure continuous working. Modern robots are equipped with sensors which enables them to navigate towards the station where they can dock and recharge for future tasks.

In case of multiple service robots, multiple docking points needs to be made available. With more and more tasks being automated and performed by robots, it can be foreseen that the number of service robots will increase in the future. In that case, a large number of docking points needs to be made available ideally, which will give rise to two main problems: (1) purchasing multiple docking points will be expensive, and most importantly, (2) multiple docking points will consume a lot of space. However, all the robots in service need not have access to the docking station for recharging at the same time, and can share a limited number of docking points, intelligently, based on certain criterion. This demands for a docking station manager which can keep a track of the docking station's status and can intelligently provide access of the most appropriate docking points to the most appropriate robot(s), and ensuring that all the services are not affected adversely. Such a docking station manager is proposed which can intelligently grant the robots access to the docking points. A priority based scheme is used for access to docks which are based on the task priority of the robot, current power levels, and robot location. The proposed docking manager can handle situations

when the docking station is full, and emergency situation when the robot's battery is about to die. The manager also instructs the robot to follow non-overlapping paths to avoid collision in case multiple robots are accessing or leaving the docking station. The locations of the docks may be at in proximity to each other, or can even be distributed at different locations.

Most of the work which has been done regarding robot docking has mostly been related to the hardware design of a single docking point itself and how robot can precisely dock into the station, whereas, the problem of efficiently managing a limited number of docking stations in case of multiple mobile robots, has largely been ignored. A multisensor based control of security robot has been proposed in [105] which includes motion planning. Vision based [105, 109, 115, 162], sound based [104], infrared sensor based [186], wifi based [167], and RFID based [88] design of docking stations have been proposed. An interesting approach of battery exchange has been proposed in [197] which can be a quick way to overcome the long recharging problem. Other approaches include market based resource sharing [76] and mobile charging docks [27] have also been proposed. The work presented in this chapter is mainly focused on efficiently managing a limited number of docking station points in presence of a large number of mobile robots.

3.2 Docking Station Manager

The components of the proposed docking station manager with their working are discussed in this section.

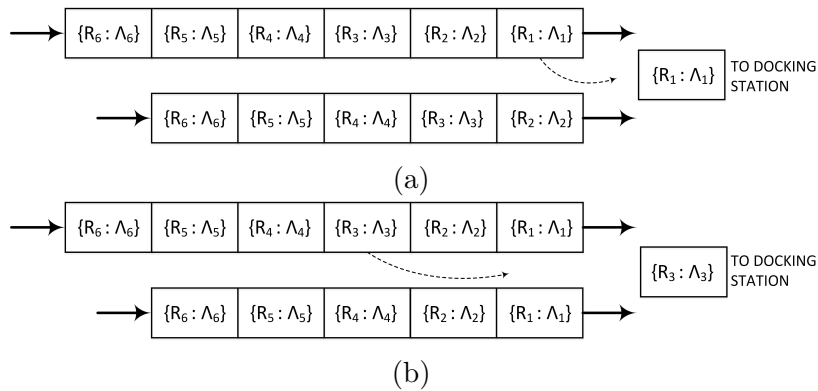


Fig. 3.1 Modified queue. (a) For similar score, the first request in the queue is processed first. (b) Requests can be processed from the middle of the queue if the priority score is high or in emergency condition.

3.2.1 Network and Communication Module

The network module handles and is capable of communicating messages with all the robots. In our implementation, all the robots and the manager server were on the same network, but it is not necessary as long as they can communicate with each other. ROS [135] was used for inter-node communication in the implementation.

3.2.2 Command Executor Module

This module generates specific command for the robot to execute. A typical example of a command in our implementation is to instruct the robot to follow a particular path, leave the docking station, stay in the queue, or go to a particular docking station.

3.2.3 Power Estimator Module

A power estimator module maintains a database which keeps a track of the power required to do a particular task. Each robot sends its current battery status before the start of a task, and once again, when the task is finished. Thus, the power (P_i) required to perform the i^{th} task is calculated as,

$$P_i = P_{\text{before } i^{\text{th}} \text{ task}} - P_{\text{after } i^{\text{th}} \text{ task}}. \quad (3.1)$$

The database maintained is shown in Table 3.1. The actual power (P_i) required to complete a particular task T_i can be different for different runs, and need not be the same as shown in Table 3.1. Effective power ($P_{E(i)}$) required to perform the i^{th} task as,

$$P_{E(x)} = \frac{1}{N} \sum_{i=1}^N P_{T(x_i)}. \quad (3.2)$$

If a robot is instructed to perform multiple (say, ' m ') tasks, the total effective power required to do all the tasks is the sum of power required to do each task which is given by,

$$P_{total} = \sum_{i=1}^m P_{T(i)}. \quad (3.3)$$

3.2.4 Request Queue Manager Module

A modified queue is maintained for the incoming requests from the robots to dock and charge. The queue is a 'modified' queue in the sense that it differs from the traditional 'first-in first-out' queue, but maintains a 'score' which determines which robot needs to be allotted with which docking station. Each task is given a unique ID (T_i) and

Table 3.1 Database of Task and Actual Power Consumed

Task ID (T_i)	Power Consumed (P_i)
T_1	P_1
T_2	P_2
T_1	P_3
T_3	P_4
$\vdots$	$\vdots$
T_1	P_5

Table 3.2 Mapping Tasks and Task Priority

Task	Task ID (T_i)	Task Priority (T_p)
Surveillance	T_1	10
Cleaning	T_2	3
Delivery	T_3	8
Patrolling	T_4	9
$\vdots$	$\vdots$	$\vdots$
Others	T_5	5

task priority (T_p). Tasks related to security like surveillance and patrolling are given high priorities, whereas tasks related to cleaning, etc. are given lower priorities, as summarized in Table 3.2.

Request from the robot comes in the form of a key-value pair of the robot-id (R_i), and a corresponding vector (Λ_i) which comprises of the task priority (T_{pi}), current power level (P_{li}), and current x and y coordinates (X_{ri}, Y_{ri}). Hence, a request message is a dictionary key-value pair of $\{R_i : \Lambda_i\}$, where,

$$\Lambda_i = \{T_{pi}, P_{li}, X_{ri}, Y_{ri}\}. \quad (3.4)$$

In case the robot is docked and charging, Λ_i contains the priority of the next task. A score is calculated for all robot requests ($\{R_i : \Lambda_i\}$). A score determines which robot needs to be given preference to have access to the docking station. Obviously, robots which have high task priorities needs to be prioritized, and the same applies for robots

with low remaining power too. A score is calculated as,

$$\begin{aligned} \text{Priority Score, } S_i &= \left(\frac{1}{\text{Power Left}} \right) \times \text{Task Priority} \\ &= \left(W_P \cdot \frac{1}{P_{li}} \right) \times W_T \cdot T_{pi} \end{aligned} \quad (3.5)$$

where, W_P and W_T are the weighing coefficients for the power preference, and the task-priority preference, respectively. The coefficients increase or decrease the effect of the power left and the task priorities on final priority score. For small robots with limited power and fast discharging, higher W_P value, and in conditions where task priorities must definitely be obeyed, a higher W_T value will affect the priority score, as desired.

However, a robot may be in emergency condition where the battery power is just about to go off. Such ‘emergency’ situations needs to be handled separately as a service robot whose battery is completely discharged may stop in the middle of the passage, which may be an obstruction for other robots, or may even permanently block the way if the passage is narrow. Such a situation can be avoided by a service robot by requesting access to the docking station for recharging when the battery power level is below a certain threshold (P_{TH}). Hence, in such ‘emergency’ cases, the robot request for access to docking station is given highest priority and hence the score is set to a maximum possible value. The priority score is calculated as,

$$\text{Score, } S_i = \begin{cases} \infty, & \text{if } P_{li} \leq P_{TH} \\ \left(W_P \cdot \frac{1}{P_{li}} \right) \times W_T \cdot T_{pi}, & \text{otherwise} \end{cases} \quad (3.6)$$

A score is calculated for all the request messages in the queue from $\{R_i : \Lambda_i\}$, and the messages are sorted according to the score in the queue. The message with the highest score is prioritized. Hence, unlike a traditional first-in first-out queue, requests can be processed from any position of the queue. If two requests have the same score, then, the earlier request is processed first, like in a normal queue. Fig.3.1a shows the first message ($\{R_1 : \Lambda_1\}$) to enter the queue is processed first as all the messages have equal scores. Fig.3.1b shows that a message request $\{R_3 : \Lambda_3\}$ with highest score is prioritized even if it was in the middle of the queue. In both the figures, the state of the queue after the request has been processed is also shown.

3.2.5 Docking Station Status Updater Module

This module keeps a track of the status of the docking station, which includes information such as, whether the docking station has an empty slot, current charge level of the docked robots and next task priorities of the docked robots. This information is maintained in a docking manager matrix (DMM) form which is discussed later. The matrix is updated every time there is a change.

$$\begin{array}{c}
 \left[\begin{array}{cccc}
 & & & \text{Docked Robot Spec} \\
 & & & \overbrace{\hspace{1.5cm}} \\
 D_1 & X_{d1} & Y_{d1} & 1 & Rid_1 & T_{p1} & P_{l1} & X_{r1} & Y_{r1} \\
 D_2 & X_{d2} & Y_{d2} & 1 & Rid_2 & T_{p2} & P_{l2} & X_{r2} & Y_{r2} \\
 D_3 & X_{d3} & Y_{d3} & 0 & - & - & - & - & - \\
 D_4 & X_{d4} & Y_{d4} & 1 & Rid_4 & T_{p4} & P_{l4} & X_{r4} & Y_{r4} \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 D_n & X_n & Y_n & 1 & Rid_n & T_{pn} & P_{ln} & X_{rn} & Y_{rn} \\
 \underbrace{\hspace{1.5cm}} & & & & \underbrace{\hspace{1.5cm}} \\
 \text{Docking Station Spec} & & & & \Lambda_n
 \end{array} \right]
 \end{array}$$

The first four columns of the DMM show the docking station's state i.e. docking station ID (D_i), coordinates of D_i (X_{di}, Y_{di}), and a boolean flag to represent if D_i is occupied (1) or not (0). The next five columns show the docked robot's specifications. The fifth column contains the docked robot ID (R_{id}) or ID of the robot which has been granted access to D_i . The last four columns comprise of the value for the key R_{id} i.e. Λ_i .

3.2.6 Access Token Granting and Path Planning Module

Depending upon the state of DMM, two situations can arise when a robot request is being processed:

1. An Empty Dock Exits: An empty dock D_i is checked by checking the flag in the fourth column of DM matrix (i.e. the value of $DMM[i][4]$). In case it does, and a single empty dock is found, the robot is granted a token with access privileges to that dock. A shortest path is calculated from the dock location X_{di}, Y_{di} and robot's current location X_{ri}, Y_{ri} using any of the path planning algorithms like A-star[66], or D-star algorithm [171, 172]. In case when multiple empty docks

Algorithm 1 Pseudo-code of Dock Allotting Algorithm

```

1: Q :: Queue
2: Q ← {}
3: # -----
4: # Calculate Score and append it with
   Rid
5: for each new request Rid do
6:   if Pl ≤ 10% then
7:     S ← ∞
8:   else
9:     S ← (WP ·  $\frac{1}{P_{ln}}$ ) × WT · Tpn
10:  end if
11:  Append Rid with Score in Q
12: end for
13: # -----
14: # Sort the Queue by Score
15: i ← 0
16: while i < len(Q): do
17:   j ← i+1
18:   while j < len(Q): do
19:     if Q[i] > Q[j]:
20:       Q[i], Q[j] ← Q[j], Q[i] # swap
21:       j ← j+1
22:   end while
23:   i ← i+1
24: end while
25: # -----
26: # Check if there is DS is empty
27: DSEFlag ← False
28: i ← 0
29: while i < n do
30:   # 4th col is empty/occupied flag
31:   if DMM[i][4] == 0: then
32:     Path ← Shortest_Path(Rid,
       DMM[i])
33:     Token ← DMM[i]
34:     DMM[i][4] ← 1 # mark as oc-
       cupied
35:     DMM[i][5] ← Λid # append
       robot info
36:     DSEFlag ← True
37:     break
38:   end if
39:   i ← i+1
40: end while
41: if DSEFlag == True then
42:   # -----
43:   # If empty space is found
44:   Assign Path and Token to the
     Robot
45: else
46:   # -----
47:   # If empty space is not found
48:   Rxid ← find_appropriate_robot(DMM)
49:   Forcefully exit robot Rxid alongside
     non-colliding path
50: end if
51: Function find_appropriate_robot(DMM)
52: Rxid ← Null
53: # 5th col is Tp for next task
54: Rxid ← DMM[xid][4]
55: Txid ← DMM[xid][5]
56: Pxid ← DMM[xid][6]
57: # -----
58: # Exit the appropriate Rxid
59: L0: Return Rxid where Rxid == Null
60: L1: Return Rxid where Txid == min &
     Pxid == max
61: L2: Return Rxid where Txid ==
     medium & Pxid == adequate
62: L3: Return Rxid where Tpxid == high
     & Pxid == max
63: L4: Return Rxid where Tpxid == high
     & Pxid == adequate

```

are found, the robot is assigned the nearest dock among all. In both the cases, DMM is updated with the R_{id} and Λ_i , and occupancy flag is set to 1.

2. Docking Station is Full: If the docking station is full and a request exists in the queue, access is granted depending upon the score of the request. In case of ‘emergency’ conditions in which the requested robot’s battery power is about to die, an empty dock has to be created by forcefully removing a robot already in the dock. DMM matrix is checked at certain levels to ascertain that only the appropriate robot is forcefully exited. Level 1 checks and removes a robot whose task priority T_i is minimum and power level P_i is maximum. If no robot is found in level 1 check, level 2 checks for robot with medium task priority and adequate power to do the task. For level 3, a robot with high priority and high power level is checked. Finally, level 4 checks a robot with high task priority and adequate power to do the task. The robot is instructed to relinquish the token for the dock which is given to the new robot. For requests in queue with high priority and minimum power level, a robot in station is forcefully exited only if there exists robot with low task priority and high power level. More levels can be added depending upon the specific requirements of the robots and the nature of services. In case everything fails and a request could not be processed, the requesting robot is denied access and has to wait in the queue which ensures that the most appropriate robot is given access to dock first.

In case the docking station is full and a robot is forcefully exited, a check is performed to ensure that the paths of the requesting robot and the dock relinquishing robot do not overlap to ensure collision avoidance, particularly when the common path is too narrow for both the robots to cross over. If $H_{a \rightarrow di}$ represents the path from the requesting robot location (say a) to the allotted dock (di), this is accomplished by assigning the shortest path to the requesting robot (as it has low power level). The path ($H_{di \rightarrow b}$) of relinquishing robot from the dock (di) to its task location (say b) is selected in such a way that it is the path with minimum possible overlap, i.e. entirely different if possible, or even relatively longer (as it has recently been recharged and can afford to take the longer route). A pseudo-code for the above explained process is given in Algorithm 1.

$$\begin{aligned}
 H_{a \rightarrow di} &= \min_{\forall i} \{H_{i \rightarrow di}\} \\
 H_{di \rightarrow b} &= \min_{\forall j} \{H_{j \rightarrow b} \cap H_{a \rightarrow di}\}
 \end{aligned}
 \tag{3.7}$$

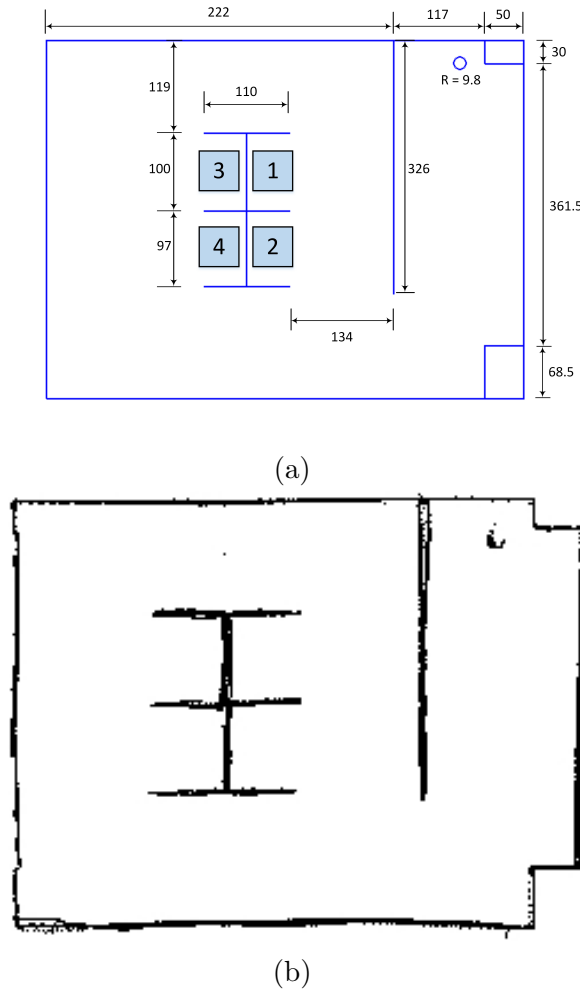
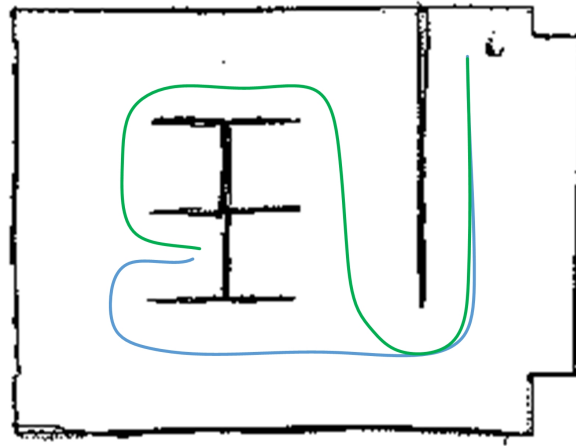


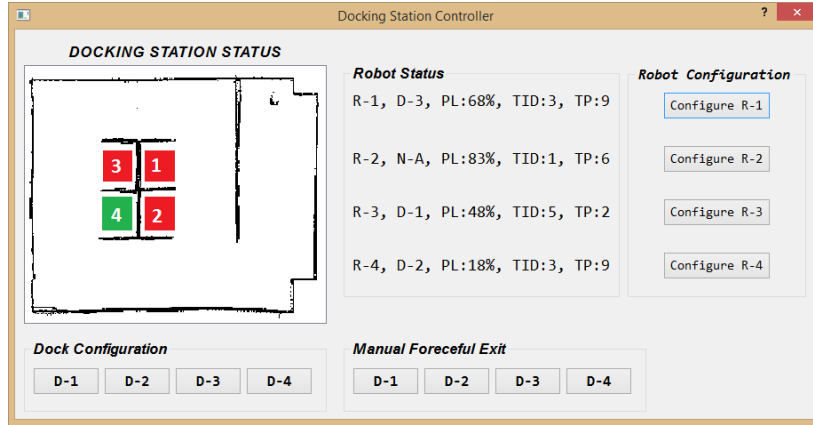
Fig. 3.2 Environment of the experiment. (a) Layout of the experiment with dimensions, (b) actual grid map of the environment.

3.2.7 Monitoring and Controlling GUI

A status monitoring and controlling GUI (Fig.3.3b) to monitor and control the status of the docking station, and the robots was developed. It is useful to check not only the empty docking points, but also the status of robots (task priority, power levels, etc.). The same GUI can be used to assign new tasks to robots, set new priorities, enable or disable a particular docking point, and forcefully (and manually) exit robot at docking point, etc. It is integrated with the command executor module, and is very useful to dynamically change the task configuration of robots. The GUI is a wrapper for ROS nodes [135] for robots. The GUI for Windows operating system, and GUI for other operating systems can easily be developed.



(a)



(b)

Fig. 3.3 Path Planning, and Monitoring & Controlling GUI. (a) Figure showing non overlapping paths of requesting robot (blue) and dock relinquishing robot (green). (b) Monitoring and Controlling GUI. Docking point status is shown in red(occupied) and green(empty). Status of robots is also shown.

3.3 Experimental Results

Fig.4.2c shows the docking station environment used in the experiment. Fig.3.2a shows the layout of the environment showing four charging docks with open space and passage for the robots to wait. Fig.3.2b shows the grid map of the environment which was constructed from laser range sensor and all the robots had information about the map. Two types of robots were used in the experiments: (1) Pioneer P3-DX [131] equipped with URG 04LX laser range sensor [184], and (2) Kobuki [182] which is equipped with Microsoft Kinect sensor. Each robot is run on ROS [135] (Robot Operating System) which has capability to perform inter-node communication between the robots and the

central server. The central server comprised of 64 bit Ubuntu 14.04 operating system. The grid map of the environment was constructed from laser data using ROS. Five experiments under different conditions were performed which are listed below.

3.3.1 Experiment 1

In experiment 1 (Fig.3.4a) docks D_1 , D_2 and D_3 (Fig.3.2a) are preoccupied, whereas dock D_4 is empty. Requesting robot is therefore assigned the empty dock D_4 . A shortest path is traversed from the starting location of the requesting robot to D_4 as shown in Fig.3.4a. The time in seconds is also shown on the frames in Fig.3.4a.

3.3.2 Experiment 2

Experiment 2 (Fig.3.4b) is similar to experiment 1 with the difference being the configuration of docks in which D_2 , D_3 and D_4 are preoccupied and D_1 is empty. Hence, the robot is assigned the empty dock D_1 and robot traverses the shortest path as shown in Fig.3.4b.

3.3.3 Experiment 3

Experiment 3 (Fig.3.4c) incorporates the situation in which the docking station is full and a robot requests access to the station. As shown in Fig.3.4c, docks D_1 to D_4 are all full. (a box was used to compensate for the lack of robot at dock D_1). The flag in DMM matrix corresponding to docks D_1 , D_2 and D_3 was set to 1 with highest task priority (T_P) and lowest power levels (P_l). Turtlebot in D_4 was set with minimum T_P and maximum P_l . The request message for charging was configured with a Λ with highest T_P and low P_l . As per the algorithm, the low priority Turtlebot was instructed to forcefully exit D_4 which was allotted to the requesting robot. Notice that the path taken by the dock relinquishing Turtlebot is longer and non overlapping with that of the requesting robot. This is shown in Fig.3.3a, in which the shorter blue path represent the requesting robot's path, whereas, the longer green path is the path taken by the dock relinquishing robot. In the absence of this path planning, both the robots would have taken the same shortest overlapping path with greater chances of collision.

3.3.4 Experiment 4 and 5

Experiment 4 (Fig.3.4d) and 5 (Fig.3.4e) deals with simulation of multiple robots in queue. In experiment 4 (Fig.3.4d), DMM is configured similar to that in experiment 3, but the two robots are maintained in a queue with the second robot in an emergency state. Hence, it is given priority and is given access to D_4 . Overlapping path avoidance

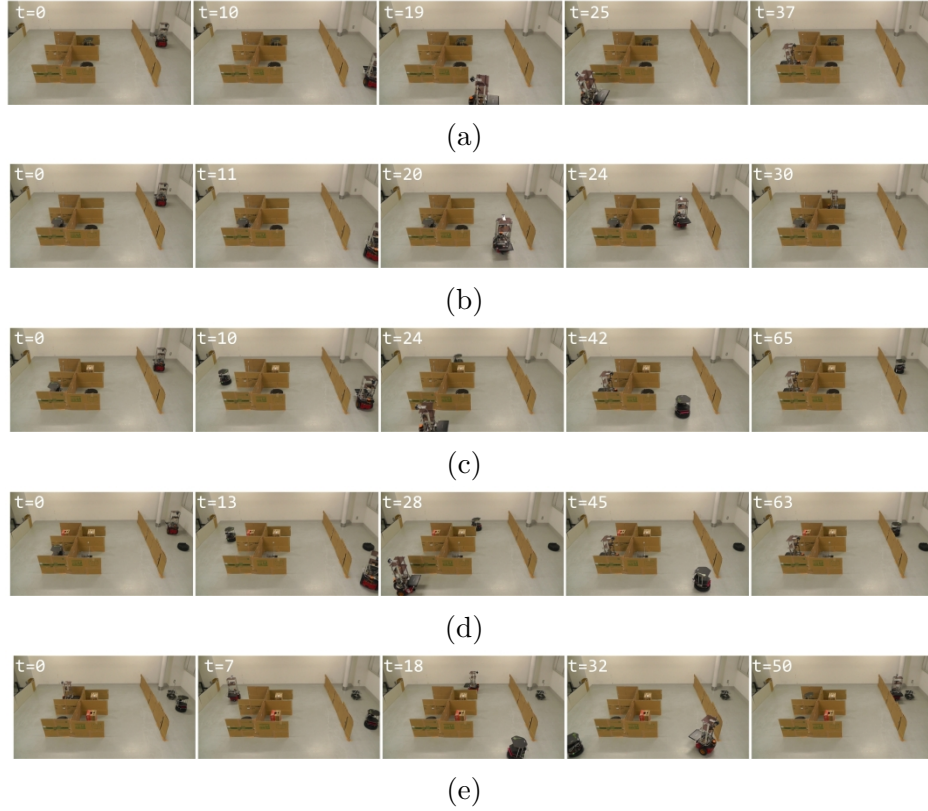


Fig. 3.4 Experiments in different scenarios. Time in seconds is shown on the top-left position of each frame. (a) Experiment 1. (b) Experiment 2. (c) Experiment 3. (d) Experiment 4. (e) Experiment 5.

can be observed in this experiment too. In experiment 5 (Fig.3.4e), the two waiting robots have the same Λ configuration (i.e. the same task priority and power levels), and hence the same priority score. Here, the first robot's request is processed and the first robot is assigned D_3 which was set to the lowest task priority and highest power level.

3.4 Summary

This work proposed an intelligent docking station manager for multiple mobile service robots. The manager maintains a modified queue based on a score which is calculated from the task priority and the current power level of the robot. The behaviour of the station manager in different scenarios when the docking station has empty docks and when it has none were discussed. The docking station can intelligently instruct a robot to forcefully relinquish a dock and give preference to the robots with higher task priorities and lower power levels. Emergency conditions for robots with power

levels below threshold were also discussed. The station manager was tested in real environment under different conditions and found to intelligently and robustly handle the management of a limited number of charging docks and multiple robots. The monitoring and controlling GUI gives a complete picture about useful information like the status of the docking points, tasks assigned to robots, and their current power levels. It can be used to configure the robot's tasks and docking points.

The location of docks used in the experiment were close to each other, but they can also be at different locations, as the docking manager matrix has the information about each dock's location. For similar reasons, in case the docks are shifted to some new location, the new locations can be updated in the matrix, and the manager will still continue to manage the docks properly.

In this work, similar types of robots and charging docks were assumed. However, in practical scenarios, there may be different types of robots employed in service, with different docking characteristics. An inclusion of different docking characteristics in the proposed framework is considered as future work. The dynamic features of the environment, like moving people, etc., are also considered for better robot navigation to and from the docking point, as future work.

Chapter 4

Multi-Robot Task Cooperation and Exploration Using a Bio-Inspired Hybrid Pheromone Signalling

Overview

This chapter deals with the problem of task sharing and exploration in multi-robot systems. The work in this chapter is inspired from biology and uses virtual pheromones mechanism for automatic task coordination and exploration. As briefly explained in Chapter 1, explicitly programming and instructing each robot to do tasks is cumbersome and not scalable. The areas to serve in a map may vary with time. Moreover, the number of the robots available to serve may also be dynamic, in real world situations, as some of the robots may be charging, while some may be out of order. Moreover, if a robot is serving (ex. cleaning) an area, other robots must not cause hindrance in that area, unless explicitly required. As evident from Fig.1.3, this work is interdependent on, and assumes the availability of SLAM module (chapter 7), path planning module (Chapter 2), and resource sharing (Chapter 3).

Chapter Abstract

This chapter presents a novel bio-inspired hybrid communication framework that incorporates the repelling behaviour of anti-aphrodisiac pheromones and attractive behaviour of pheromones for efficient map exploration of multiple mobile service robots. The proposed communication framework presents a scheme for robots to efficiently serve large areas of map, while cooperating with each other through proper

pheromone deposition. This eliminates the need of explicitly programming each service robot to serve particular areas of the map. The paths taken by robots are represented as nodes across which pheromones are deposited. This reduces the search space for tracking pheromones and reduces data size to be communicated between robots. A novel pheromone deposition model is presented which takes into account the uncertainty in the robot's position. This eliminates robots to deposit pheromones at wrong places when localization fails. The framework also integrates the pheromone signalling mechanism in landmark based Extended Kalman Filter (EKF) localization and allows the robots to capture areas or sub-areas of the map, to improve the localization. A scheme to resolve conflicts through local communication is presented. Two cases of floor cleaning task, and surveillance task, performed by multiple robots are discussed through experimental and simulation results. Results show that the proposed scheme enables multiple service robots to perform cooperative tasks intelligently without any explicit programming.

4.1 Introduction

Mobile robots are increasingly being employed for tasks like floor cleaning, and surveillance in shopping malls, hospitals, and universities. To serve such large areas, multiple robots are often used for efficiency. However, introducing multiple robots also introduce the problem of programming the robots to efficiently serve the region. For example, if multiple robots are employed for floor cleaning in shopping malls or industries, each robot must explicitly be programmed to serve a specific area. On the other hand, multiple robots serving the same small area will reduce the efficiency. The areas to serve in a map may vary with time. Moreover, the number of the robots available to serve may also be dynamic, in real world situations, as some of the robots may be charging, while some may be out of order. In case of robots used for surveillance, a robot may want other robot or robots to follow itself while chasing a suspicious person, for backup. This situation is also dynamic in terms of availability of robots, selecting the nearest robot for quick response, and selecting the same path towards a particular area as taken by the previous robot. In both the cases, explicitly programming the robots is cumbersome, and demands for a simpler scheme to accomplish the task.

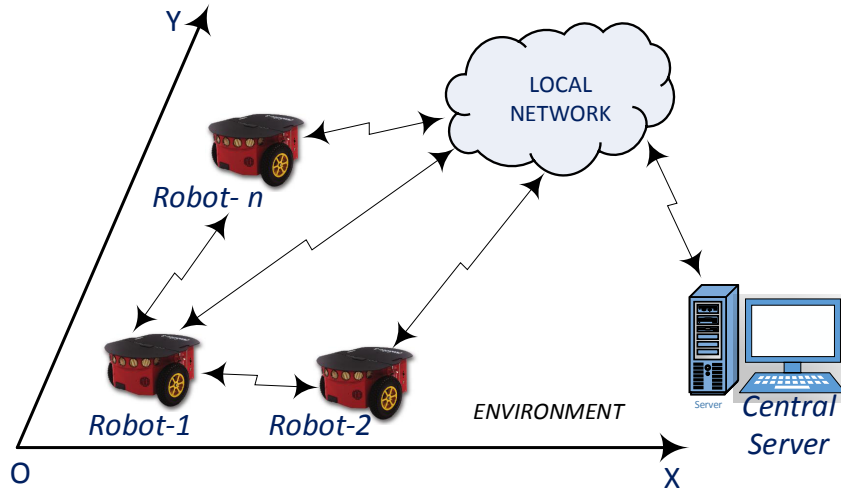


Fig. 4.1 System Architecture.

The proposed work is inspired by biology where insects deposit particular chemicals called as ‘pheromones’ [84] to signal other insects of the same species to either attract or go away from a particular resource. In this chapter, chemicals which attract other insects are called as ‘pheromones’, and this signalling mechanism can be found in honeybees, ants, wasps, and termites [112]. Ants use pheromones to attract the population to food source, and bees to attract the population to an empty hive [181]. There are other type of pheromones called anti-aphrodisiac pheromones with opposite behaviour, i.e. they turn away other insects from a resource. They are used to raise alarm and claiming territory. The anti-aphrodisiac pheromones are denoted as ‘anti-pheromones’ in this chapter.

Previous works on pheromone signalling in robotics has been mainly concentrated to mimic the swarm behaviour using attractive pheromones [125, 144] with application in process control [49], communication [56, 57], and to mimic swarm behaviour [133]. Tasks like exploration and surveillance requires that multiple robots disperse themselves in the region to cover maximum possible area. A multi-agent exploration algorithm has been proposed in [51] in which a coverage algorithm has been proposed with a pheromone barriers. The larger the pheromone value, stronger is the barrier for other robots. Similar dispersive behaviours which employs repellent virtual pheromones has been proposed in [126, 127] to survey a disaster site. Other works which uses repelling behaviour of pheromones includes [161] which proposes solutions for multi-agent rescue mission, and [18, 39] for robot surveillance. An extensive review of research related to pheromone signalling and swarm robotics can be found in [117, 179, 205].

The proposed work incorporates the advantages of both the ‘repelling anti-pheromone’ signalling mechanism and the ‘attracting pheromone’ mechanism which is presented in a hybrid framework. The novel contributions of the proposed work are:

- **Integration of Pheromone Signalling with EKF Localization:** An integrated algorithm is presented in Section 4.5 which allows efficient ‘area capture’ and ‘sub-region capture’ by robots so that they can work efficiently without interruption from other robots (Section 4.4.2). Service robots like cleaning robots or patrolling robots are expected to work in terms of areas of the map like rooms, corridors, and halls, instead of direct spatial coordinates like in [51, 53]. Hence, the proposed ‘area capture’ and ‘sub-region capture’ translates to efficient task distribution in which robots can work without interruption. Once an area has been captured by a robot, it is also efficient in terms of reducing the localization cost as explained in Section 4.5.
- **Modelling Robot’s Uncertainty in Pheromone Deposition:** Localization is very important for mobile robots to ascertain their position in the region. Many previous works like [53, 126] assume a perfect robot localization which is generally not true as the exteroceptive sensors attached on the robots are prone to errors, and this uncertainty needs to be modelled to estimate the state of the robot [143]. The proposed work takes into account the uncertainty in robot’s position for pheromone deposition (Section 4.2). No or less pheromones are deposited if the uncertainty in localization is high. On the other hand, pheromones are deposited with confidence if the localization is good.
- **Node Representation for Reduced Search Space and Efficiency:** In previous works like [51] or [53], pheromones are deposited anywhere in the map. Therefore, the search area to find the total pheromones is nearly quadratic ($W \times H$, where W and H are the width and breadth of the map, respectively). However, node representation (described in Section 4.3) drastically reduces the search space as pheromones are deposited only across the nodes of the map, reducing the search space from quadratic ($W \times H$) to linear $2n \times m_{dir}$, where m_{dir} is the maximum number of diverging directions across a node. This is also efficient energy-wise as less data needs to be transmitted with node representation. Work by [69] reports that among sensing, computing, and communication, a considerable amount of the energy is consumed during communication than computation. Moreover,

conflicts (Section 4.5.1) are resolved and obstacles are avoided using only local communication.

Table 4.1 Signal Notation and Behaviour

Signal	Value	Force
Pheromone	+ve number	Attractive
Anti-Pheromone	-ve number	Repulsive
None	zero	Empty

The proposed framework was tested in real environment with real robots. The results are presented and discussed in Section 7.5. In order to test the robustness of the method in complex environment and large robots, a simulation software was also developed. Both anti-pheromone signalling (Sec 4.6.1) and pheromone signalling (Sec 4.6.2) experiments in real and simulated environment are described in detail. Finally, Section 8.2 concludes the chapter.

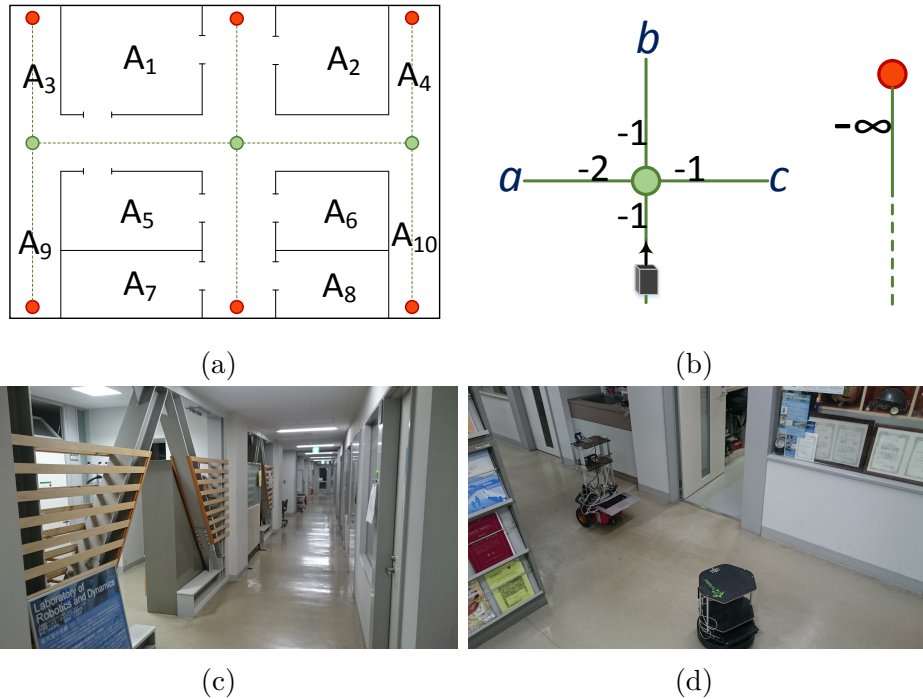


Fig. 4.2 Sub-mapping, node representation, and actual environment. (a) Map divided into sub regions to serve with nodes along passages, (b) A node (green) with four diverging vectors and anti-pheromone values, and a terminal node (red), (c) Actual environment, (d) Robots exploring the map.

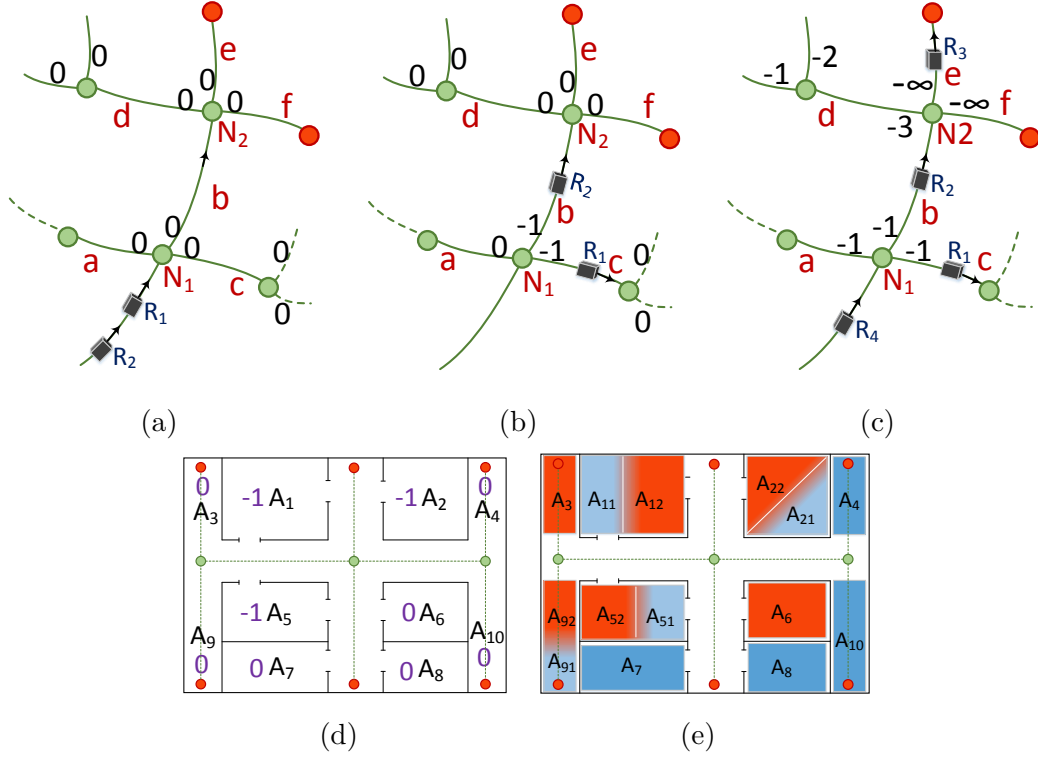


Fig. 4.3 Anti-pheromone deposition. (a) Initial condition with no anti-pheromones on nodes, (b) Robots R_1 and R_2 depositing anti-pheromones on nodes, (c) A particular node state in future with $-\infty$ values on terminal nodes. (d,e) Division into sub-regions. (d) Entire sub-region blocking, (e) Division of a sub-region further into sub-blocks.

4.2 Pheromone and Anti Pheromone Representation

The system architecture (Fig.4.1) comprises of multiple robots ($R_1, R_2, \dots, R_n$) which are connected to the central server, to each other robot, via. a local network. The proposed system uses numerical values to represent chemicals as virtual pheromones. The entire map of the environment is represented in a $(W_e \times H_e)$ grid, where W_e and H_e represents the width and height of the environment, respectively. A 2D signalling matrix (M_s) of equal dimensions ($W_e \times H_e$) which overlaps with the grid map is used as a signalling matrix where virtual pheromones can be deposited by the robot on any place $[(x, y), 0 \leq x < W_e, 0 \leq y < H_e]$ of the map. Practically, this is achieved by setting the value of the appropriate row and column of the matrix ($M_s[x][y]$) with the signalling value. The 2D signalling matrix is stored in the central server. The pheromones and anti-pheromones are represented using the notations shown in Table

4.1. Pheromones have positive values in the signalling matrix ($M_s[x][y]$) represents a pheromone (P_h) deposited at that location. The strength of the attractive force (F_a) is directly proportional to the pheromone signal value at a location (x, y) .

$$F_a \propto f_p(P_h, M_s, x, y) \quad (4.1)$$

Anti-Pheromones (A_p) have negative values in the signalling matrix ($M_s[x][y]$) at a location. The strength of the repulsive force (F_r) is directly proportional to the value of the anti-pheromone signal value (A_p).

$$F_r \propto f_a(A_p, M_s, x, y) \quad (4.2)$$

A zero value in signalling matrix represents that the corresponding area has not yet been visited by any of the robots.

The sensor range of the laser range finder used in the experiments is denoted by z_{rng} . Pheromones can be deposited on the top of already existing pheromones at any position q ($\xi_q(t)$). A modified pheromone model described in [123] was used. Unlike [123] which uses a fixed number of pheromone deposition by a robot, a probabilistic model was used in which the amount of pheromone deposition depends upon the uncertainty of robot's state while localizing itself. The amount of pheromones or anti-pheromones deposited by the i^{th} robot R_i at location q and time t is given by:

$$\Delta_q^i(t) = (1 - \xi(t - 1))\Psi_q^i(t) \quad (4.3)$$

$$\Psi_q^i(t) = \begin{cases} \varrho_t e^{\frac{(q_{ix} - q_x)^2 + (q_{iy} - q_y)^2}{\lambda^2}} & \text{if } q_x \in z_{rng} , \\ 0 & \text{otherwise} \end{cases}$$

where (q_{ix}, q_{iy}) is the coordinate of the i^{th} robot. ϱ is a non-constant probabilistic factor which controls the amount of pheromone deposition at a location based on how good has the localization of the robot been achieved, and $0 \leq \varrho_t \leq 1$. An integrated pheromone signalling in Extended Kalman Filter (EKF) landmark based localization is explained in Section 4.5. The algorithm is given in Algorithm 1 which estimates the state μ_t (i.e. position x, y and orientation θ) and the uncertainty associated with this state estimation Σ_t , at time t for a robot. The factor ϱ_t depends on this uncertainty

as,

$$\varrho_t \propto \frac{1}{\sum_t}. \quad (4.4)$$

In other words, if the localization is good, more pheromones are deposited. However, if the robot fails to localize itself, less or no pheromones are deposited. λ specifies the dispersion of the pheromones or anti-pheromones. In the presented work, the value of dispersion factor λ is chosen to affect only a very limited area around the robot's current position. Concretely, if the entire area is divided into micro grids of size $\Delta_x \times \Delta_y$, factor λ is chosen so that maximum number of pheromones are deposited in the current grid, and only a tiny fraction in the nearby micro-grids which are directly touching the current grid. The total pheromones are given by,

$$\Delta_q = \Delta_{(qx,qy)} = \sum_{i=-1}^1 \sum_{j=-1}^1 \Delta_{(qx+j,qy+i)}(t). \quad (4.5)$$

Pheromones are not fixed and evaporate with time. The rate of evaporation of pheromones is given by ρ , and the total amount of pheromones evaporated at position q and time t is given by the function,

$$\Phi(t) = \rho \xi_q(t) \quad (4.6)$$

Considering the evaporation of pheromones, the total pheromones on position q at time t is,

$$\begin{aligned} \xi_q(t) &= (\xi_q(t-1) - \Phi_q(t-1)) \\ &+ \sum_{i=-1}^1 \sum_{j=-1}^1 \Delta_{(qx+j,qy+i)}(t) \end{aligned} \quad (4.7)$$

The pheromones are only deposited on the side of 'nodes' which are described in the next section. This is unlike the traditional methods of pheromone deposition in which the pheromones can be deposited anywhere in the map.

4.3 Node Representation of Path

The spatial environment is shown in Fig.4.2a, and is divided into several sub-regions represented by letters ($A_1, A_2, A_3 \dots$). The pathways to different sections of the map are represented in the form of a graph with nodes.

An SHP node [147, 148] is shown in Fig.4.2b. It comprises of one or more vectors (links) diverging from the center towards the passages of the map. Fig.4.2b shows four such vectors diverging from the node. A robot deposits anti-pheromones on the diverging vectors while it traverses them. There are two types of nodes: (a) Terminal Nodes: which represent the terminal pathways of the map shown in red color in Fig.4.2b. Robots deposit a negative infinity value on the terminal nodes when it traverses it. (b) Intermediate Nodes: which are not terminal.

4.4 Anti-Pheromone Mechanism for Map Exploration

4.4.1 Path Selection

Initially, the signalling matrix is set to zero, representing no pheromones. The initial position of the robots could be anywhere in the map, and they are not assumed to start from the same location. Fig.4.3a shows the initial state of the nodes of the map with two robots R_1 and R_2 . When robot R_1 encounters node N_1 which has 3 diverging vectors, it calculates from the signalling matrix (M_s) that none of the three diverging paths have been traversed by any of the other robots. Path selection of R_1 is according to the following objective function:

$$\begin{aligned} \text{path} &= \max_{P_h} (g(N_i)) \\ &= \max_{P_h} (\{ \vec{N}_i a : 0 \}, \{ \vec{N}_i b : 0 \}, \{ \vec{N}_i c : 0 \}) \end{aligned} \quad (4.8)$$

The function $g(N_i)$ takes a node value N_i and returns a dictionary containing pairs of the vector path and pheromone values deposited on the diverging vectors of that node, as key-value pairs ($\{key : value\}$). Then, a path representing the maximum pheromone value is chosen. In case of N_1 , since there are no pheromones deposited by any of the previous robots, all the three diverging paths have pheromone count of zero. In case of same pheromone count, a random path is selected out of the paths returned by the function g . Hence, robot R_1 takes a random path $\vec{N}_1 c$, and deposits an anti-pheromone by decreasing the current value over $\vec{N}_1 c$ by 1.

For robot R_2 which initially followed robot R_1 , the condition is different at node N_1 . Since, path $\vec{N}_1 c$ is already taken, a random path $\vec{N}_1 b$ is selected between $\vec{N}_1 b$ and $\vec{N}_1 a$, as both of them have the same maximum pheromone count. R_2 deposits anti-pheromone and proceeds as shown in Fig.4.3b. If a robot R_i takes a path and

it encounters a dead-end, it deposits an extremely large value of anti-pheromones ($-\infty$) on that side of the node. The path selection objective function makes sure that paths with larger anti-pheromone values are not prioritized. This mechanism checks that paths leading to terminal points of the map ($\vec{N_2e}$, $\vec{N_2f}$) which have already been selected by other robots as shown in Fig.4.3c, are not selected by other robots.

Similarly, other robots keep on exploring the map by taking the maximum of the anti-pheromone values deposited across the nodes. Since the anti-pheromones have negative values, the overall result is that the robots take the paths which have not been visited, or which have been least visited. Notice that, this is opposite to that of pheromone trailing mechanism in ants and other biological species, in which, ants keeps depositing pheromones and other ants follow them, ultimately optimizing their path towards the food source. However, anti-pheromone mechanism makes sure that robots take as much of diversified paths as possible, and don't just follow each other whenever possible. This is desired for tasks like cleaning and patrolling where robots must explore diverse areas of the map.

4.4.2 Area Capturing and Area Selection

This section discusses area capturing and area selection by robots.

Sub-region selection and capture

At the start of the job ($M_s \equiv 0$), each robot R_i moves to its nearest possible region in the map. The path from the current location of the robot to the nearest region is calculated by A^* algorithm [66] or D^* algorithm [172]. Multiple robots starting from the same location of the map approach the same nearest area. However, this is controlled via. the node mechanism explained in Section 4.4.1. The first robot to approach the nearest area would deposit an anti-pheromone over the node, which would force other robots to explore different areas of the map. Once a robot enters the nearest sub-block, it deposits an anti-pheromone on the sub-block, which is achieved by altering the values of the signalling matrix to (-1) which represents the sub-region. This further prevents obstruction by other robots to enter the same sub-block. The sub-area is, in other words, 'captured' by the robot, and robot can perform its task (say cleaning) inside the sub-block. After finishing the job in the sub-block, the robot moves to the next nearest sub-block with the maximum anti-pheromone value, which is given by the following objective function,

$$\text{target} = \max_{P_h} \left(h(\text{map}, \text{current position}) \right) \quad (4.9)$$

The objective function ensures that the next target of the robot is the nearest sub-area of the map which has not yet been captured by other robots. Such a scheme is depicted in Fig.4.3d.

Region blocking and capture

If the sub-area of the maps are large enough, they are further divided into sub-blocks. In this case, the robot does not ‘capture’ the entire sub-region but only a particular block of it. This is achieved in the same way by decreasing the values of the signalling matrix representing that particular block. However, sub-blocking would be inefficient if the area is too small. Sub-blocking is performed by taking the dimensions and capabilities of the robots, and dimensions of the sub-region. A blocked approach is shown in Fig.4.3e. The areas in red are the sub-block of the sub-regions which have been captured and anti-pheromones deposited, whereas the blue areas represents the ‘un-captured’ sub-regions and sub-blocks which are yet to be explored.

4.4.3 Pheromone Mechanism

There are scenarios where a robot would like to ‘attract’ other robots in a region for collaborative task completion. In such situations, the robot must relinquish the ‘captured’ sub-region. This is achieved by depositing a pheromone with a large positive value for the respective region of the 2D signalling matrix. Regions with positive values are more attractive to the robots according to equation 4.9. For example, a surveillance robot chasing a target person would like other robots to follow it for backup. In that case, the surveillance robot would keep on depositing positive pheromones across all the traversed nodes. A positive pheromone value would attract other robots to follow the same path, according to Eq.(4.8).

Thus, a robot traverses nodes and visits regions of map, both of which have pheromones or anti-pheromones (or none) deposited. The net force for a robot to travel from one point to other via nodes to explore the map is given by,

$$\max_{P_h} \left(g(N_i) \cdot h(\text{map, current position}) \right). \quad (4.10)$$

4.5 Integration of Pheromone signalling With Landmark Based EKF Localization

Service robots must accurately localize themselves in the map which is an integral part of Simultaneous Localization and Mapping (SLAM) [180]. Extended Kalman

Algorithm 2 Landmark based EKF localization with integration of pheromone signalling

```

1: procedure EKF_PHEROMONE_LOCALIZATION( $\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, m$ )
2:    $\theta \leftarrow \mu_{t-1, \theta}$ 
3:    $G_t \leftarrow \begin{bmatrix} 1 & 0 & -\frac{v_t}{\omega_t} \cos \theta + \frac{v_t}{\omega_t} \cos(\theta + \omega_t \Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t} \sin \theta + \frac{v_t}{\omega_t} \sin(\theta + \omega_t \Delta t) \\ 0 & 0 & 1 \end{bmatrix}$  ▷ Jacobian of motion-function w.r.t state
4:    $V_t \leftarrow \begin{bmatrix} \frac{-\sin \theta + \sin(\theta + \omega_t \Delta t)}{\omega_t} & \frac{v_t (\sin \theta - \sin(\theta + \omega_t \Delta t))}{\omega_t^2} + \frac{v_t (\cos(\theta + \omega_t \Delta t) \Delta t)}{\omega_t} \\ \frac{\cos \theta - \cos(\theta + \omega_t \Delta t)}{\omega_t} & -\frac{v_t (\cos \theta - \cos(\theta + \omega_t \Delta t))}{\omega_t^2} + \frac{v_t (\sin(\theta + \omega_t \Delta t) \Delta t)}{\omega_t} \\ 0 & \Delta t \end{bmatrix}$  ▷ Jacobian of motion w.r.t control
5:    $M_t \leftarrow \begin{bmatrix} \alpha_1 v_t^2 + \alpha_2 \omega_t^2 & 0 \\ 0 & \alpha_3 v_t^2 + \alpha_4 \omega_t^2 \end{bmatrix}$  ▷ Covariance matrix of noise in control space
6:    $\bar{\mu}_t \leftarrow \mu_{t-1} + \begin{bmatrix} -\frac{v_t}{\omega_t} \sin \theta + \frac{v_t}{\omega_t} \sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t} \cos \theta - \frac{v_t}{\omega_t} \cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{bmatrix}$  ▷ Predicted belief
7:    $\bar{\Sigma}_t \leftarrow G_t \Sigma_{t-1} G_t^T + V_t M_t V_t^T$  ▷ Predicted covariance
8:    $Q_t \leftarrow \begin{bmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{bmatrix}$  ▷ Covariance matrix of measurement noise
9:   for observation  $z_t^i \leftarrow (r_t^i \phi_t^i s_t^i)^T$  do ▷ For every observation
10:    if Area captured by robot then ▷ If this robot has captured an area
11:       $L_{set} \leftarrow l(A_c, m)$  ▷  $L_{set}$  contains only landmarks of robot's captured area  $A_c$  of map  $m$ 
12:    else
13:       $L_{set} \leftarrow l(A_i, m) - l(A_c, m)$  ▷  $L_{set}$  contains only landmarks of free areas of map  $m$ 
14:    end if
15:    for all landmarks  $k$  in set  $L_{set}$  do
16:       $q \leftarrow (m_{k,x} - \bar{\mu}_{t,x})^2 + (m_{k,y} - \bar{\mu}_{t,y})^2$ 
17:       $\hat{z}_t^k \leftarrow \begin{bmatrix} \text{atan2}(m_{k,y} - \bar{\mu}_{t,y}, m_{k,x} - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta} \\ \frac{\sqrt{q}}{m_{k,s}} \end{bmatrix}$  maximum likelihood  
check for only  
appropriate landmarks  
and not all ▷ Predicted Measurement
18:       $H_t^k \leftarrow \begin{bmatrix} -\frac{m_{k,x} - \bar{\mu}_{t,x}}{\sqrt{q}} & -\frac{m_{k,y} - \bar{\mu}_{t,y}}{\sqrt{q}} & 0 \\ \frac{m_{k,y} - \bar{\mu}_{t,y}}{q} & -\frac{m_{k,x} - \bar{\mu}_{t,x}}{q} & -1 \\ 0 & 0 & 0 \end{bmatrix}$  ▷ Jacobian of measurement-function w.r.t state
19:       $S_t^k \leftarrow H_t^k \bar{\Sigma}_t [H_t^k]^T + Q_t$  ▷ Uncertainty corresponding to  $\hat{z}_t$ 
20:    end for
21:     $j(i) \leftarrow \text{argmax} \frac{1}{\sqrt{\det(2\pi S_t^k)}} e^{-\frac{1}{2}(z_t^i - \hat{z}_t^k)^T [S_t^k]^{-1} (z_t^i - \hat{z}_t^k)}$ 
22:     $K_t^i \leftarrow \bar{\Sigma}_t [H_t^{j(i)}]^T [S_t^{j(i)}]^{-1}$  ▷ Kalman gain
23:     $\bar{\mu}_t \leftarrow \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^{j(i)})$  ▷ Correct belief
24:     $\bar{\Sigma}_t \leftarrow (I - K_t^i H_t^{j(i)}) \bar{\Sigma}_t$  ▷ Correct covariance
25:  end for
26:   $\mu_t \leftarrow \bar{\mu}_t$  ▷ Updated belief
27:   $\Sigma_t \leftarrow \bar{\Sigma}_t$  ▷ Updated covariance
28: end procedure

```

Filter (EKF) [180] has been a de-facto standard for robot localization technique. EKF is fundamentally based on Bayes filter, in which, the state of the robot (x_t) and the environment are expressed through conditional probability distributions. Bayes filter applies two successive rules of prediction and update to determine the system state. With control u_t at time t , the predicted belief $\overline{bel}(x_t)$ is calculated just before, and corrected belief $bel(x_t)$ is calculated just after the sensor observation (z_t), respectively, as,

$$\begin{aligned} (Predict) \quad \overline{bel}(x_t) &= P(x_t | z_{1:t-1}, u_{1:t}) \\ (Update) \quad bel(x_t) &= P(x_t | z_{1:t}, u_{1:t}). \end{aligned} \quad (4.11)$$

EKF handles the linearity assumption by using Jacobians and Taylor expansion, and assumes a Gaussian noise distribution. As shown in Algorithm 1, G_t and V_t are Jacobians of motion function with respect to state and control, respectively, where ω_t is the angular velocity of the robot. Control noise and measurement noise is expressed as covariance matrices M_t and Q_t , respectively. Similarly, the predicted and corrected robot state, and the covariances associated with them are expressed as $\bar{\mu}_t$, μ_t , $\bar{\Sigma}_t$, and Σ_t , respectively. In traditional EKF, whenever a robot encounters a landmark, it compares it with all the registered landmarks for localization [180]. However, the proposed algorithm integrates the pheromone mechanism and the maximum likelihood check for an encountered landmark is done only with appropriate landmarks and not all. This is shown between lines 9 to 20 of Algorithm 1. An area A_i of a map has a set of landmarks represented by LM_i given by a function $l(A_i)$. The total area (A) and the total number of landmarks in the map (LM) are therefore,

$$A = \int_i A_i, \quad LM = \int_i l(A_i) \quad (4.12)$$

A robot R_i which captured the area A_i with a set of landmarks LM_i , checks only the landmarks give by $l(A_i)$, and not the entire set of landmarks. The robot can omit comparing the measured landmarks against registered landmarks that belong to other areas which have been captured or not, improving EKF landmark matching by a factor of η_1 given by Eq.(4.13).

Similarly, robots which have not yet captured any region may omit comparing the measured landmark against the set of all landmarks which belong to regions which have been captured by other robots. If A_c represents the set of all areas which have

already been captured, improving EKF landmark matching by a factor of η_2 ,

$$\eta_1 = \frac{\int_i l(A_i)}{l(A_i)}, \quad \eta_2 = \frac{\int_i l(A_i)}{\int_i l(A_i) - \int_c l(A_c)}. \quad (4.13)$$

4.5.1 Collision Avoidance and Resolving Conflict Amongst Robots

Region blocking and capture explained in Section 4.4.2, allows a large area to be divided into sub-regions. Different robots can serve each sub-region without interfering with the work of other robots which have other sub-regions to work on. However, if the region is too small, and only a single robot is sufficient enough to serve it, there might arise a situation in which two robots try to capture the same region, and conflict needs to be resolved in such situations. In the proposed work, the conflict is resolved by local communication explained below.

Listing 4.1 JSON Message To Resolve Conflict

```

1: {"robot-param": { // parameters
2: "robot-id": "01", // Id of robot
3: "time-stamp": "1440233786" // Unix timestamp
4: "battery": "79", // % of battery left
5: "capture-count": "2" // total areas served
6: "task": [ // Task
7: {"task-id": "8"}, // Task-ID
8: {"task-priority": "1"}], // Task Priority
9: "sensor-spec": [ // Sensor specifications
10: {"LRF-range": "4m"}, // LRF Range
11: {"rgb-res": "vga"}] // Camera resolution
12: }}
```

As explained in the system architecture, the robots can communicate with the server and also amongst themselves. At the start of the task, each robot (R_i) is given a task-id (T_i), and priority (P_i). The robots also keeps a track of the available battery power (B_i), total areas captured (C_i), and sensor specifications. When two or more robots try to capture an area, they exchange messages in JSON format which comprises of these parameters. A sample JSON message is shown in Listing 1.

The robot with the highest priority gets access to the area. If the priorities are equal, the battery power is considered, and the robot with sufficient power to serve the area gets prioritized. This is followed by checking C_i which is an indication of the total

number of areas already served by the robot. The robot with lower C_i value gets prioritized as it has served lesser areas. After that, robot specifications are checked, and the robot with better specifications get prioritized. In the worst case, if all the values are same, robot with smaller id is prioritized. Notice that, which parameter is prioritized is not a study of this chapter and other conflict resolving schemes can also be applied in the proposed framework as robots can communicate with each other.

Table 4.2 Table showing sub-regions captured by robots at particular times

#Robots	Two Robots					Three Robots					Four Robots				
Area	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
T_{w1}	-	R_1	R_2	-	-	R_3	R_1	R_2	-	-	R_3	R_1	R_2	R_4	-
T_{w2}	R_1	-	-	R_2	-	R_3	-	-	R_1	R_2	-	-	-	-	R_4
T_{w3}	R_1	-	-	-	R_2	-	-	-	-	-	-	-	-	-	-

Robots rely on attached sensors for local obstacle avoidance. Notice that in many previous works related to virtual pheromones, once a robot has avoided static obstacles like chairs, or boxes, other robots can follow the pheromone trail left out by the previous robot. Thus, robots can save computation related to obstacle avoidance at the cost of frequent communication with the server to keep a track of pheromones. However, in the proposed scheme, since the pheromones are deposited only on the nodes, each robot needs to perform obstacle avoidance using local sensors. However, it is efficient in terms of communication.

4.6 Results and Discussion

This section discusses the results of the experiments performed using the proposed mechanism. Experiments were performed in real environment with two robots, and simulation with large number of robots to test the robustness of the proposed scheme. Fig.4.4a shows the map of the environment (Fig.4.2c) of the passage around our laboratory used in the experiment. Fig.4.4b shows the actual 2D grid map of the environment constructed using Lidar sensor. The approximate length of the environment is about 25 m and width is about 5m The environment has been overlaid with green and yellow blocks to mark out the target regions for the robot to serve. The green regions have been labelled A, B, C, D, and E. They are the regions where service robots must perform cleaning or patrolling. The yellow region marks the docking point and starting location of all the mobile robots. Fig.4.4b also shows the nodes along the passage which

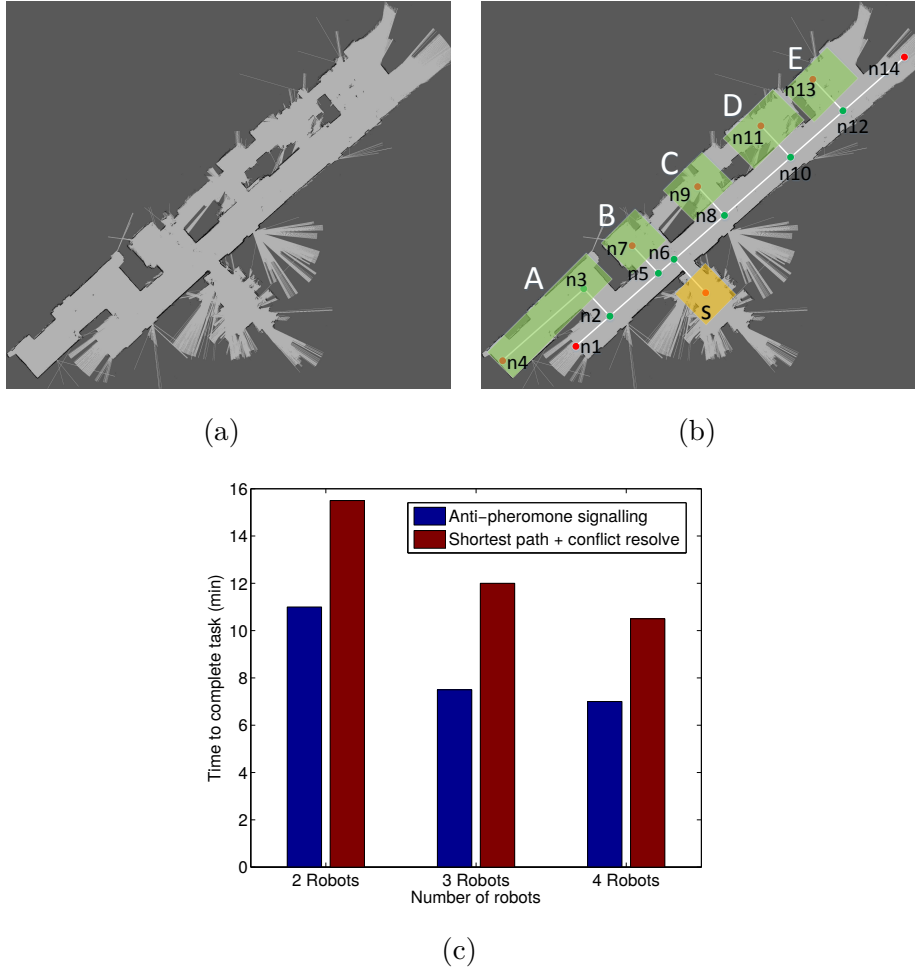
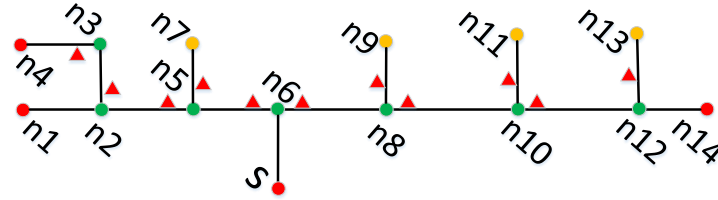


Fig. 4.4 Test environment. (a) Environment's grid map, (b) Service regions (green) from A to E, and docking station (orange), along with passage showing nodes. (c) Comparison of time taken by robots by anti-pheromone signalling and shortest path method with conflict resolve.

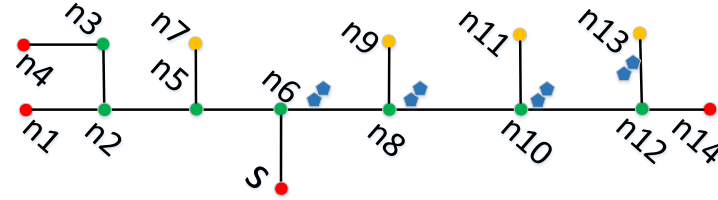
have been marked as $n_1, n_2, \dots, n_{14}$. The terminal nodes are shown in red, whereas the non-terminal ones are shown in green, and the links in white.

Two experiments were performed to test the proposed signalling mechanisms of anti-pheromones, and pheromones, in the environment shown in Fig.4.4a. The robots (shown in Fig.4.2d) used in the experiment were: (1) Pioneer P3-DX equipped with URG 04LX laser range sensor, and (2) Kobuki's Turtlebot which is equipped with Microsoft Kinect sensor. The robots were programmed and controlled using the Robot Operating System (ROS), and could perform inter-node communication between the robots and the server. The central server comprised of 64 bit Ubuntu 14.04 operating system. The map of the environment (Fig.4.4b) was pre-built and was made available

to all the robots. Apart from tests in real environment with two real robots, Simulated tests with three, and four robots, were also performed to test the robustness of the proposed method. The service regions in ascending order of distance separation from the docking station are: B, C, A, D, E. In other words, service region B is nearest to the docking station, and region E is farthest from the docking station, in that order. All the service regions have approximately the same area, except area A which is twice the size of B. Since, Pioneer 3DX and Turtlebot used in the experiment are not actual cleaning robots, they were programmed to manoeuvre the captured region for a specific work time (T_w).



(a)



(b)

Fig. 4.5 Final node configuration. (a) Final node configuration in anti-phermone experiment with anti-phermones shown in red, (b) Final node configuration in phermone experiment with phermones shown in blue.

4.6.1 Anti-Phermone Experiment

In case of two robots (R_1 and R_2), they start from the docking station and R_1 proceeds towards the nearest region B, through nodes n_6 and n_5 depositing anti-phermones on them. R_2 proceeds and encounters node n_6 , where it finds an anti-phermone deposited on the left vector, and takes a right turn towards region C, depositing anti-phermones. Both the robots capture the regions by depositing anti-phermones. After T_w time, R_1

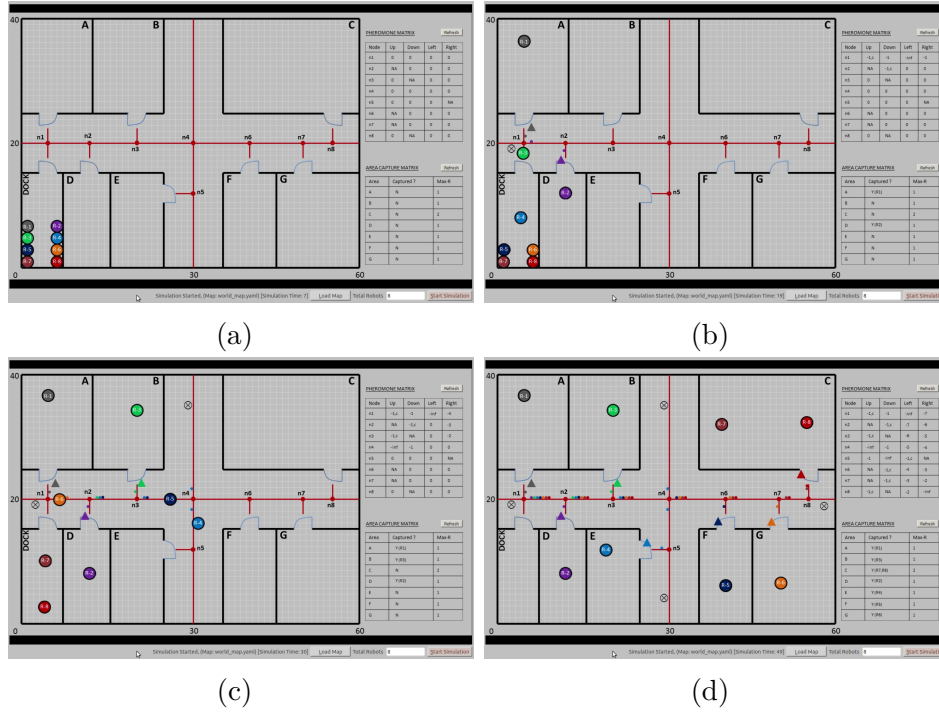


Fig. 4.6 Simulation of map exploration using the proposed method with 8 robots. Map is shown with skeleton path, signalling matrix, area capture matrix and robots marked from numbers 1 to 8. (a) At time=7sec (b) At time=19sec. Regions A and D are captured by robots R1 and R2. (c) At time=30sec. Regions A, D, and B are captured by robots R1, R2, and R3, respectively. (d) At time=49sec. The entire area has been explored and captured.

finishes the task and moves towards the nearest region A which has no pheromones. Similarly, R_2 moves towards D. R_1 takes twice the time to serve area A. Also, region E is nearest to R_2 and hence R_2 moves to serve region E. The movements of the robots in specific times is given in Table 4.4. Table 4.4 also shows the area captured by robots in case of three, and four robots. Fig.4.5a shows the final status of the deposition of anti-pheromones on the nodes.

Fig.4.4c shows the time taken by robots to serve the areas shown in Fig.4.4b with varying number of robots using anti-pheromone signalling and shortest path with conflict resolve method. In simulation, the robots were programmed to serve the nearest area through a path which was calculated by A* algorithm [66]. It can be seen that anti-pheromone signalling mechanism takes much less time to complete the task and outperforms the conventional shortest path method. In the conventional shortest paths method, the robots are governed only by the shortest distance mechanism, and multiple robots end up in the same region. Thereafter, conflict needs to be resolved as

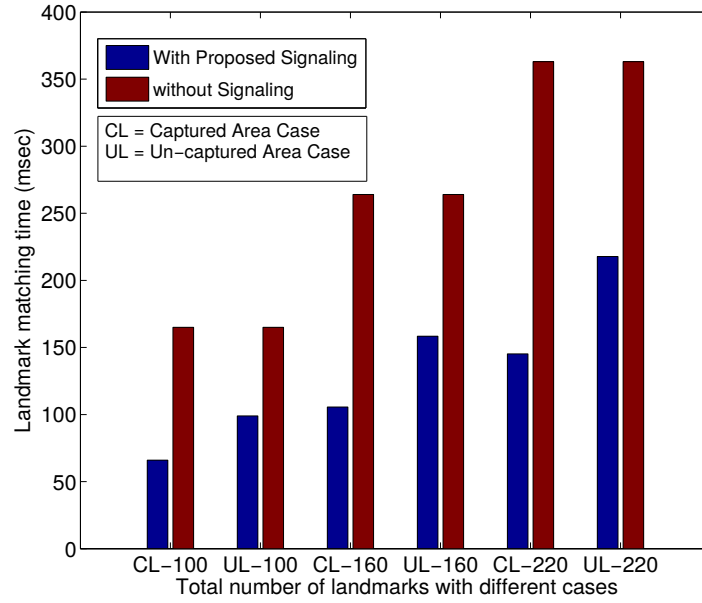


Fig. 4.7 Improvement in landmark matching using proposed method for capture and un-captured area case with and without the proposed method.

to which robot will serve (clean) the area and a lot of time is spent in resolving this conflict. Whereas, the anti-pheromone signalling mechanism prevents multiple robots in the same regions (by region capture) and directs the robots to the least traversed paths. It can be seen from Fig.4.4c that in case of four robots, time taken to complete the task in conventional shortest path method is almost similar to the time required by two robots with anti-pheromone signalling mechanism. This is because, without region capture, multiple robots end up in the same regions and more time is spent in resolving the conflicts between the robots. Fig.4.7 shows the improvement in landmark matching using the signalling mechanism with varying number of landmarks for area captured and un-captured case. Fig.4.7 reflects data with 40 percent of total landmarks captured (hence, 60 percent un-captured), and 1.65ms required for matching single landmark.

A simulation framework was build which can load complex maps, and test the proposed method with large number of robots. A snapshot of the simulation is shown in Fig.4.6a, which shows the map of the simulation environment, signalling matrix, area capture matrix, skeleton path in red, and eight cleaning robots (marked R1 to R8). None of the robots is explicitly programmed to explore a particular area. Instead, they explore

the environment using the proposed anti-pheromone signalling on nodes. There are eight regions in the loaded map which is same as the number of robots. The skeleton map was generated using the technique of skeletonization proposed in [198]. One robot is sufficient to clean each of the regions except region C which requires two robots. Sub-region division is enabled by setting the value of maximum robots in the region (Max-R) to 2, as shown in Fig.4.6a. Initially, none of the regions are captured as shown in Fig.4.6a. Figure 4.6b shows the situation at 19 seconds of simulation, at which regions A and D have been captured by robots R1 and R2 as shown in the area capture matrix. The captured marks have been indicated by a triangle ($\blacktriangle$). The corresponding pheromone values in the signalling matrix are indicated by the letter 'c' which is a large value of anti-pheromone, to capture the region. Similarly, '-inf' indicates an infinite value in the signalling matrix, which is an indication of dead-ends in the map. In the loaded map, only four directions across a node (up, down, left, and right) have been considered but can easily be extended. The colors of the anti-pheromones have no significance and are only shown to clearly indicate the respective robot which deposited it. It is assumed that robots can localize well in the environment. Consider Fig.4.6c which shows the state of the robots at time 30 seconds. It can be seen that the robot R6 takes a right turn at node n1 by travelling towards a value of lower anti-pheromones (node n1 has $-\infty$ on left, and $-c$ in the up direction as region is captured). Similarly, robot R5 moves towards node n6. Figure 4.6d shows the final configuration of the simulation in which all the robots have captured the areas correctly, and started cleaning. Table 4.3 and Table 4.4 respectively shows the final configuration of pheromone matrix, and area capture matrix in simulation.

4.6.2 Pheromone Experiment

To test the (+ve) pheromone mechanism, R_1 was instructed to go to region E depositing pheromones across nodes. Robot R_2 was not given any instructions, but just started after R_1 . According to the pheromone mechanism explained in Section 4.4.3, R_2 successfully followed R_1 sensing pheromones on nodes, without any explicit instructions. This confirmed pheromone signalling for tasks like robot surveillance where a robot may need other robots to follow it for backup. Fig.4.5b shows the final configuration of nodes with pheromones.

Table 4.3 Table showing final pheromone matrix in simulation of Fig.4.6

Node	Up	Down	Left	Right
n1	-1,c	-1	$-\infty$	-7
n2	NA	-1,c	-7	-6
n3	-1,c	NA	-6	-5
n4	$-\infty$	-1	-5	-4
n5	-1	$-\infty$	-1,c	NA
n6	NA	-1,c	-4	-3
n7	NA	-1,c	-3	-2
n8	-1,c	NA	-2	$-\infty$

Table 4.4 Table showing the final configuration of the area capture matrix in simulation of Fig.4.6

Area	Captured (Y/N)	Max Robots
A	Y (R1)	1
B	Y (R3)	1
C	Y (R7,R8)	2
D	Y (R2)	1
E	Y (R4)	1
F	Y (R5)	1
G	Y (R6)	1

4.7 Summary

This chapter proposed a hybrid pheromone and anti-pheromone signalling mechanism for map exploration by multiple service robots. The hybrid mechanism involves anti-pheromones which repel robots across diverse portions of map, whereas pheromones attract robots to particular regions. A novel pheromone deposition scheme which takes the uncertainty of the robot's localization is proposed. Unlike many previously proposed techniques which uses a constant model for pheromone deposition, in this work, the amount of pheromone deposition depends on how well is the robot able to estimate its position from the sensors in the map. This ensures that pheromones are not deposited on unwanted locations if localization fails. The chapter proposed a node representation of the map where robots deposit pheromones. This reduces the search space for signalling matrix and is communication efficient. The proposed signalling mechanism is integrated with EKF localization and an algorithm is presented. This integration allows robots to capture areas of sub-areas of the map which enables the robot to work without interference from other robots. The chapter also discussed how localization itself improves with the integrated pheromone signalling. Robots rely on local communication to resolve conflicts. The proposed method is presented in both simulated and real environment with variable number of robots. Results with both anti-pheromone and pheromone signalling show that the proposed mechanism can be effectively used to explore the map using multiple robots, for tasks which require diverse map exploration, or collaborative performance, without any explicit programming. In

future, it is planned to extend the work by incorporating fuzziness of work done by robots, into the system.

Chapter 5

Robust Virtual Pheromone Trailing of Multi-Robots by Uncertainty Integration in Pheromone Deposition

Overview

This chapter builds upon the work in the previous chapter which used virtual pheromones for task coordination. Most of the works in literature related to virtual pheromone based multi-robot systems assumes a ‘perfect-world’ scenario, in which, particularly the positional attributes of the robots are assumed to be perfect. In reality however, it is difficult to achieve perfect localization of the robot due to errors in encoders and sensors attached to the robot, and the dynamics of the environment in which the robot operates. In real world scenarios, there is always some uncertainty associated in estimating the pose (i.e. position and orientation) of the mobile service robot. Failing to model this uncertainty would result in service robots in wrong places which has adverse effects on the multi-robot system. For example, if we consider a ‘leader-follower’ multi-robot system, the leader robot might completely fail to localize itself in the environment and be lost. Other robots trailing its pheromones will end up being in entirely wrong areas of the map. This results in a ‘blind leading the blind’ scenario which reduces the efficiency of the multi-robot system. In the previous chapter, virtual pheromone deposition was only across certain nodes and used EKF based modelling. The work in this chapter builds upon the work in previous chapter

and generalizes the framework with improvements and discusses practical applications and benefits. Like the previous chapter, this work is interdependent on, and assumes the availability of SLAM module (Chapter 7), path planning module (Chapter 2), and resource sharing (Chapter 3).

Chapter Abstract

Virtual pheromone trailing has successfully been demonstrated for navigation of multiple robots to achieve a collective goal. Many previous works use a pheromone deposition scheme which assumes perfect localization of the robot, in which, robots precisely know their location in the map. Therefore, pheromones are always assumed to be deposited at the desired place. However, it is difficult to achieve perfect localization of the robot due to errors in encoders and sensors attached to the robot, and the dynamics of the environment in which the robot operates. In real world scenarios, there is always some uncertainty associated in estimating the pose (i.e. position and orientation) of the mobile service robot. Failing to model this uncertainty would result in service robots depositing pheromones at wrong places. A leading robot in the multi-robot system might completely fail to localize itself in the environment and be lost. Other robots trailing its pheromones will end up being in entirely wrong areas of the map. This results in a ‘blind leading the blind’ scenario which reduces the efficiency of the multi-robot system. A pheromone deposition algorithm is proposed which models the uncertainty of the robot’s pose. It is demonstrated through experiments in both simulated and real environments that modeling the uncertainty in pheromone deposition is crucial, and that the proposed algorithm can model the uncertainty well.

5.1 Introduction

To set the scene for this chapter, consider the following example:

There is a multi-robot system employing pheromone trailing mechanism. Bob is the leader robot and Alice follows the pheromone trails left by Bob shown in Fig.5.1. They are both on a rescue mission at a disaster site to find a victim and provide medical aid. They are both given the approximate map of the site. Bob uses one of the traditional pheromone deposition schemes to deposit pheromones and Alice follows sensing the trail. Both the robots (leader Bob, and follower Alice) have localization and mapping units along with sensors attached to them. The robots associate and match the registered features of the map with the features extracted from the sensor data in

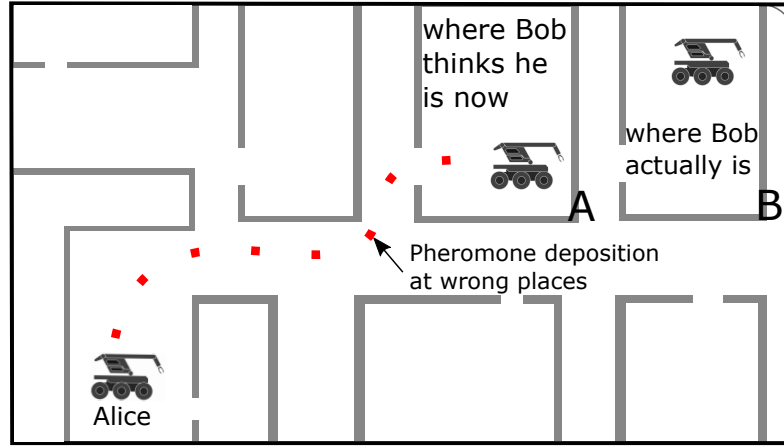


Fig. 5.1 Bob's localization failure results in its pheromone deposition at wrong places. Like Bob, the pheromone trailing Alice too will end up in the wrong place.

order to estimate their position in the map. If the estimation is good, then there is less error between the estimated and the true position of the robot. If the estimation is bad, the estimated position is far from the robots true position. The robots deposits the pheromones at their *estimated* position. Hence, in case of good estimation (and hence localization), pheromones are deposited at nearly the correct place. But, in case of bad estimation, the pheromones are deposited at wrongly estimated positions in the map. This difference in true and estimated position may arise due to many similar features in the map, noisy environment, noisy sensors, wheel slip, lack of good features, and others. Pheromone deposition at wrong places is further explained in Fig.5.2, in which, Robot's estimated position is Pos.E at coordinates $X=15, Y=10$. Robot's true position is Pos.T. at coordinates $X=25, y=10$. Hence, the pheromones are wrongly deposited by the robot at Pos.E ($X=15, Y=10$).

As shown in Fig.5.1, Bob gets lost as there are many similar features in the environment, and the disaster site is cluttered with noise. Bob 'estimates' that it is at location A. However, Bob's true position is B. But, since Bob does not model its positional uncertainty in depositing pheromones, it keeps depositing pheromones according to its estimated position in the map, which differs from its true position. Therefore, the pheromones deposited by Bob end up at wrong places as well. In Fig.5.1, pheromones deposited at Bob's wrongly estimated position are shown in red color.

Alice, the trail following robot must catch-up with Bob to provide assistance and follows the Bob's pheromone trails. Alice, which relies on pheromone trails finds the same amount of pheromones deposited by Bob everywhere and blindly keeps following

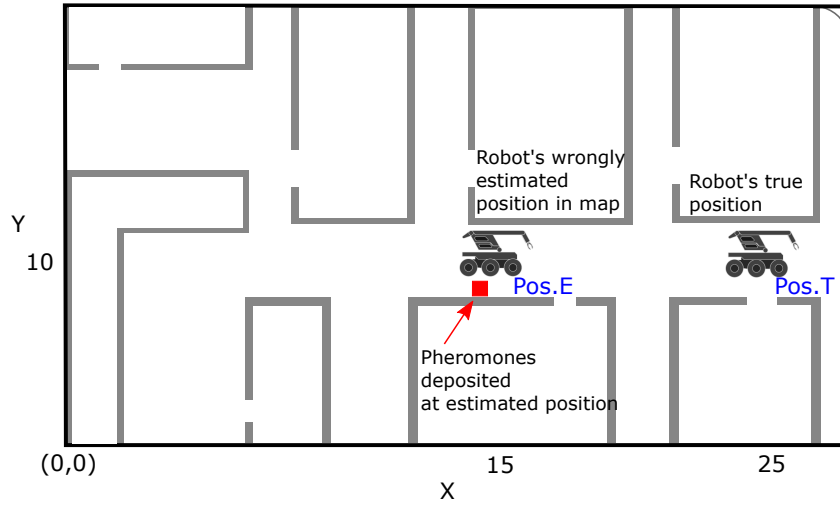


Fig. 5.2 Pheromone deposition at wrong places. Robots deposit pheromones at their estimated positions which might differ from their true position. In this example, Robot's estimated position is Pos.E at coordinates $X=15, Y=10$. Robot's true position is Pos.T. at coordinates $X=25, Y=10$. Hence, the pheromones are wrongly deposited at Pos.E ($X=15, Y=10$).

the trails deposited at Bob's wrongly estimated positions. Eventually, Alice ends up at the wrong position A shown in Fig.5.1, and does not find Bob which is truly located at position B. Alice too leaves its own pheromone trails for other robots. Irrespective of whether Alice's localization was good or not, Alice actually goes to position A following the Bob's trails, and hence, all the robots end up at wrong places of the map.

This is a classical example of 'blind leading the blind' case. The fault does not lie with Bob, or Bob's localization unit as good localization is difficult to achieve in cluttered environments. The real culprit is Bob's pheromone deposition module which *does not* model the Bob's positional uncertainty while depositing pheromones. Therefore, Alice and other robots have no clue if Bob was lost at some place in the map.

Only if Alice had some clue that Bob was actually lost at some place in the map, it would have been in a position to decide if it should follow the trails or explore the environment on its own based on other conditional factors.

Problem Statement: In the absence of modeling the positional uncertainty in pheromone deposition, pheromones will be deposited at the wrong places in the map. Other robots trailing the pheromones as well may end up in wrong areas of map which will reduce the efficiency of the multi-robot system.

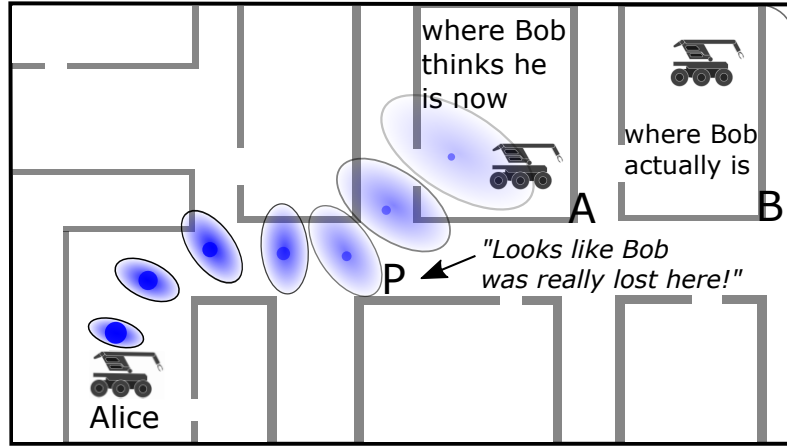


Fig. 5.3 Bob reduces pheromone deposition and disperses them depending on its uncertainty. Trailing Alice now has some idea that Bob was lost at point P, and is in a better position to take decision.

5.1.1 Main Idea of this Work

Pheromone deposition and positional uncertainty are (and must be) integrated and modeled together. As shown in Fig.5.3, if Bob's localization module estimates a large uncertainty in its current position in the map, Bob deposits less pheromones and disperses them across its current position. The amount of pheromones deposited is inversely proportional to the estimated positional uncertainty. If the robot has localized itself well in the map, large amount of pheromones are deposited confidently in a small area centered around the current position. This enables Alice to decide if it should follow the leader robot. If the trails are dispersed and weak, the follower-robot may decide not to follow the 'lost' leader-robot, and instead explore the map on its own.

5.1.2 New Contributions of this Work

Pheromone deposition with uncertainty modeling is proposed. Pheromone deposition scheme which can be integrated with estimation filters (like Extended Kalman Filter, Particle Filter) is proposed. More pheromones are deposited at locations where the robot's localization is comparatively better, and vice-versa. The pheromones are deposited as a gradient in the map which allows the robots to move towards the direction with maximum rate of change of pheromones. A pheromone evaporation considering the influence of ambient pheromones is also presented. The proposed method is tested in both simulation and real environment. Speed adaption by trailing robots with reference to the pheromones sensed is presented as a practical application.

5.2 Related Works

The induction of virtual pheromone trailing in robotics is not new and there is a plethora of works dealing with using this idea to achieve a complex task for multi-robot cooperation in the literature. The idea is inspired from biology and nature. Many insects (ex. ants, bees, wasps) deposit particular chemicals called as ‘pheromones’ [84] in order to attract other members of the same species. Sometimes they are also used to warn other species, or claim territory. With regards to robotics and pheromone trailing, an extensive review has been carried out in the works found in [117, 179, 205]. Regarding comparison with other works, it is suffice to mention that the proposed work is one the first works in the literature to model the positional uncertainty of the robot in the map, to the best of our survey. Ravankar et al. mentioned this idea in their work [146] (work in previous chapter), but their work mainly focused on combining pheromones and anti-pheromones [53, 126] in a single framework with no experimental verification and constant pheromone deposition only at certain points of the map. This chapter extends the work by mathematically describing the deposition of pheromones anywhere in the map. The work is also extended by concretely integrating the evaporation mechanism of the pheromones while also considering the effect of ambient pheromones. Moreover, a thorough test in both simulation and real environment with integration in SLAM has been provided in this chapter. Other than the works of Ravankar et al., most of the previous works consider a perfect localization model of the robot. Previous works on pheromone signalling in robotics has been mainly concentrated to mimic the swarm behaviour using attractive pheromones [125]. The applications included process control [49, 98], communication [56, 57], and to mimic swarm behaviour [133]. Works in [18, 39] have focused on applications of robot surveillance. An interesting work of using virtual pheromones with multi-robots is presented in [126, 127] to test the dispersive behaviour of multi-robots.

The proposed work does not deals with any sort of optimization techniques which involves pheromones (ex. ant colony optimization like in [174]). On the contrary, this work took an entirely fresh approach to deal with the problem of depositing pheromones in the map at a more fundamental level by also modeling the positional uncertainty of the robot.

5.3 A Brief Introduction to Robot State and Uncertainty Estimation Using Various Filters

An autonomous mobile robot in an unknown environment needs to build a map of its surrounding while simultaneously updating its position in the map, also known as the SLAM (Simultaneous Localization and Mapping) problem. If a map is available, the robot must localize itself in the map by associating features (ex. lines, corners, or color) in the map and the features extracted from the sensors attached to the robot to estimate its position. It is a challenging problem due to the uncertainties associated with sensors, robot motion, and dynamics of the environment. Probabilistic filters have been successfully employed to mathematically model this uncertainty by representing ambiguity and degree of belief using the calculus of probability.

The most general algorithm for calculating the probabilistic beliefs is given by the Bayes filter. The state of the robot (x_t) at time t is indicated by a vector comprising of its pose $[x \ y]^T$ and orientation (θ) as, $x_t = [x \ y \ \theta]^T$. Bayes filter is a recursive filter in which the belief $bel(x_t)$ at time t is calculated from the prior belief $bel(x_{t-1})$ at time $t - 1$. Given a control u_t at time t , Bayes filter calculates the predictive belief $\overline{bel(x_t)}$ just before the observation (z_t) using its prior belief ($bel(x_{t-1})$) as,

$$\overline{bel(x_t)} = \int p(x_t)|u_t, x_{t-1} \ bel(x_{t-1}) \ dx_{t-1}. \quad (5.1)$$

This is known as the prediction step. The next step is that of correction in which the predicted belief ($\overline{bel(x_t)}$) is multiplied with the probability that the measurement z_t may have been observed. Since the probability density function may not integrate to 1, the result is normalized by the normalization constant η ,

$$bel(x_t) = \eta \ p(z_t)|x_t \ \overline{bel(x_t)}. \quad (5.2)$$

There are many variants of Bayes filter which can estimate the state of the robot and the uncertainty associated with this estimate. Although a detailed explanation of these filters is out of the scope of this chapter, what is most relevant to this chapter is that all these filters can estimate the state and the associated uncertainty which will be used to deposit pheromones by the robots. A brief discussion of these filters is presented here with pointers for detailed explanation. Moreover, the discussion is limited to three

most widely used filters: Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), and Particle Filter (PF).

5.3.1 Extended Kalman Filter

EKF is a powerful mathematical tool to model the uncertainties of the sensors attached to the robot, and has been demonstrated successfully with laser range sensors, vision sensors, and inertial sensors, etc. for mapping and localization. The standard Kalman filter assumes linear state and measurement transitions with Gaussian noise. An EKF overcomes the non-linearity by employing Jacobians and (first order) Taylor expansion in the non-linear model. The belief at time t is represented by the mean μ_t and the covariance matrix Σ_t . There are two non-linear functions: the state transition function $g(\cdot)$ and the measurement transition function $h(\cdot)$, given as,

$$\begin{aligned} x_t &= g(u_t, x_{t-1}) + \epsilon_t, \\ z_t &= h(x_t) + \delta_t, \end{aligned} \tag{5.3}$$

where ϵ_t , and δ_t are the noises. The non-linear function $g(\cdot)$ is approximated by the value at μ_{t-1} and control u_t through a Taylor expansion:

$$\begin{aligned} g(u_t, x_{t-1}) &\approx g(u_t, \mu_{t-1}) + g'(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1}), \\ &= g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}), \end{aligned} \tag{5.4}$$

where G_t is the Jacobian given as the first derivative of g at μ_{t-1} . EKF implements the same linearization for measurement function h at the predicted mean $\bar{\mu}_t$:

$$\begin{aligned} h(x_t) &\approx h(\bar{\mu}_t) + h'(\bar{\mu}_t)(x_t - \bar{\mu}_t), \\ &= h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t), \end{aligned} \tag{5.5}$$

where H_t is the Jacobian given as the first derivative of h at $\bar{\mu}_t$. The details about the prediction and correction steps of EKF can be found in [32, 158, 163, 180].

5.3.2 Unscented Kalman Filter

Unlike EKF which uses Taylor series expansion to linearize the transformation of a Gaussian, an Unscented Kalman Filter (UKF) performs a stochastic linearization through statistical linear regression process. The main idea behind UKF is that it is easier to approximate the probability function than the nonlinear function. For an n -dimensional Gaussian with mean μ and covariance Σ , UKF extracts $2n + 1$ points

(also called sigma points $\chi^{[i]}$) from the Gaussian and passes these through g . The sigma points are chosen according to the following rules:

$$\begin{aligned}\chi^{[0]} &= \mu \\ \chi^{[i]} &= \mu + (\sqrt{(n + \lambda)\Sigma})_i \quad \text{for } i = 1, \dots, n \\ \chi^{[i]} &= \mu - (\sqrt{(n + \lambda)\Sigma})_{i-n} \quad \text{for } i = n + 1, \dots, 2n\end{aligned}\tag{5.6}$$

where $\lambda = \alpha^2(n + \kappa) - n$ and α and κ are scaling parameters that determine how far the sigma points are spread from the mean. Each sigma point $\chi^{[i]}$ has two associated weights: $w_m^{[i]}$ used when computing the mean, and $w_c^{[i]}$ used when computing the covariance. The sigma points are then passed through the non-linear function g ,

$$Y^{[i]} = g(\chi^{[i]})\tag{5.7}$$

Parameters μ and Σ of the resulting Gaussian are extracted from the mapped sigma points $Y^{[i]}$ according to,

$$\begin{aligned}\mu &= \sum_{i=0}^{2n} w_m^{[i]} Y^{[i]} \\ \Sigma &= \sum_{i=0}^{2n} w_c^{[i]} (Y^{[i]} - \mu')(Y^{[i]} - \mu')^T.\end{aligned}\tag{5.8}$$

A detailed explanation of UKF can be found in [32, 65, 113, 158, 163, 180].

5.3.3 Particle Filter

Particle Filter (PF) is a non-linear state estimator based on Bayesian filtering. It is a sequential Monte Carlo based technique which models the probability density using a set of discrete points. PF can represent a much broader space of distributions than, for example, Gaussians. Generally, robot localization involves the estimating the state $\mathbf{X}_t$ of the robot, where, $\mathbf{X}_t = [x, y, \theta]^T$ with measurements $\mathbf{Z}_t$ at time t . The state space and observation equations are given in the form,

$$\begin{aligned}\mathbf{X}_t &= \mathbf{f}_t(\mathbf{X}_{t-1}, \mathbf{u}_t) && \text{(state equation)} \\ \mathbf{Z}_t &= \mathbf{h}_t(\mathbf{X}_t, \mathbf{v}_t) && \text{(observation equation),}\end{aligned}$$

where, $\mathbf{X}_t$ is a state vector with noise $\mathbf{u}_t$, and $\mathbf{Z}_t$ is a set of observations with noise $\mathbf{v}_t$. $\mathbf{f}_t(\cdot)$ is a non-linear system transition function, and $\mathbf{g}_t(\cdot)$ is a non-linear measurement

function. By Bayes' formula, the posterior state distribution $p(\mathbf{X}_t|\mathbf{Z}_{1:t-1})$ is given as,

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t-1}) = \int p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1}) d\mathbf{X}_{t-1}$$

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) = \frac{p(\mathbf{Z}_t|\mathbf{X}_t)p(\mathbf{X}_t|\mathbf{Z}_{1:t-1})}{\int p(\mathbf{Z}_t|\mathbf{X}_t)p(\mathbf{X}_t|\mathbf{Z}_{1:t-1}) d\mathbf{X}_t}$$

Particles or samples of posterior distribution are represented as,

$$\mathbf{X}_t = X_t^{[1]}, X_t^{[2]}, X_t^{[2]}, \dots, X_t^{[M]}$$

where M denotes the total number of particles in the particle set $\mathbf{X}_t$. Each particle $X_t^{[m]}$ ($1 \leq m \leq M$) is a concrete instantiation of the state at time t and has a weight w_t^m associated with it. X_t approximates the distribution $p(X)$ by,

$$p(X) \approx \sum_{m=1}^M w_t^{[m]} \delta(X - X_t^{[m]})$$

where $\delta(\cdot)$ is the Dirac delta function. Initially, M particles are generated from the distribution $p(X)$ and each particle is assigned a uniform weight equal to $1/M$. PF approximates the state by drawing a set of M weighted random samples from the importance density function $\pi(\cdot)$. The normalized importance weight $\bar{w}_t^{[m]}$ and updated weight $w_t^{[m]}$ which governs the re-sampling is given by,

$$\bar{w}_t^{[m]} = \frac{p(X)}{\pi(X)} = \frac{w_t^{[m]}}{\sum_{m=1}^M w_t^{[m]}}$$

$$w_t^{[m]} \propto \frac{p(Z_t|X_t^{[m]})p(X_t^{[m]}|X_{t-1}^{[m]})}{\pi(X_t^{[m]}|X_{0:t-1}^{[m]}, Z_{0:t})} w_{t-1}^{[m]}.$$

A detailed explanation of particle filter can be found in [10, 110, 119, 163, 180].

Apart from the above mentioned filters, there are many other filters like histogram filter, information filter, and others which have been explained in detail in [32, 163, 180]. All of these filters can estimate the state of the robot and the uncertainty associated with the estimate. Readers may find a comparison of the merits and limitations of various filters in [91].

5.4 Pheromone Deposition with Uncertainty Modeling

Irrespective of the filter used, it is assumed that the state $[x \ y \ \theta]^T$ of the robot is estimated along with the uncertainty associated. Pheromone deposition in the map is first discussed in Section 5.4.1. Later, the pheromone evaporation mechanism is explained in Section 5.4.2.

5.4.1 Pheromone Deposition

The robot's uncertainty in state is generally represented by the covariance matrix (Σ_t),

$$\Sigma_t = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 \end{bmatrix} \quad (5.9)$$

If the uncertainty in robot's pose given by Σ_t is large, the robot should deposit less pheromones, and vice-versa. Hence, the pheromone deposition can be modeled as,

$$\text{Amount of pheromones} \propto \frac{1}{\text{Uncertainty given by } \Sigma_t}. \quad (5.10)$$

Σ_t also contains the uncertainty in the orientation (θ). However, the amount of pheromones deposited in the map are scalar in nature. The robot can deposit the same number of pheromones at the same spatial position with any orientation (correct or not) at that position. Hence, a sub-matrix (Σ_{xy}) of covariance Σ_t is considered without the orientation uncertainty, and the pheromone deposition is modeled as,

$$\Sigma_{xy} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}, \quad (5.11)$$

$$\text{Amount of pheromones} \propto \gamma \propto \frac{1}{\text{Uncertainty given by } \Sigma_{xy}}.$$

Here, γ is a factor which controls the amount of pheromones to be deposited.

The eigenvalues ($\lambda_1, \dots, \lambda_n$) and eigenvectors ($\vec{v}_1, \dots, \vec{v}_n$) of the matrix Σ_{xy} denotes the magnitude and direction of the variance, respectively.

The main idea is to deposit pheromones inversely proportional to the uncertainty given by the covariance matrix. The pheromone controller factor γ is thus inversely

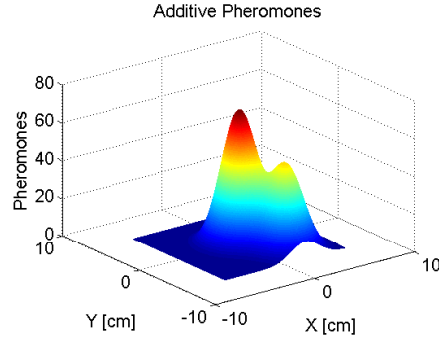


Fig. 5.4 Addition of Pheromone Fields in Vicinity

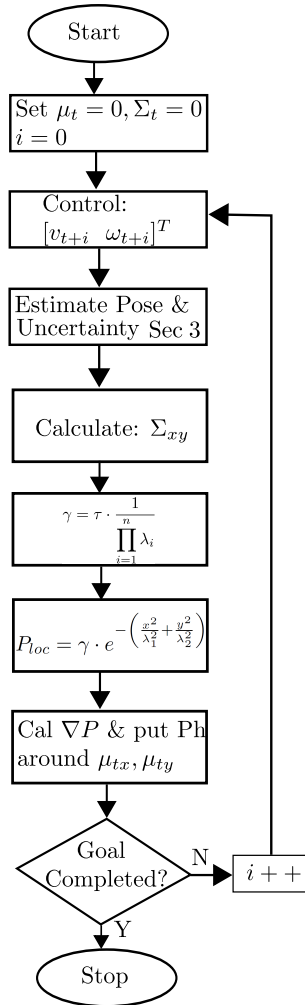


Fig. 5.5 Flowchart

proportional to the product of eigenvalues of the covariance matrix,

$$\gamma \propto \frac{1}{\prod_{i=1}^n \lambda_i}, \implies \gamma = \tau \cdot \frac{1}{\prod_{i=1}^n \lambda_i}, \quad (5.12)$$

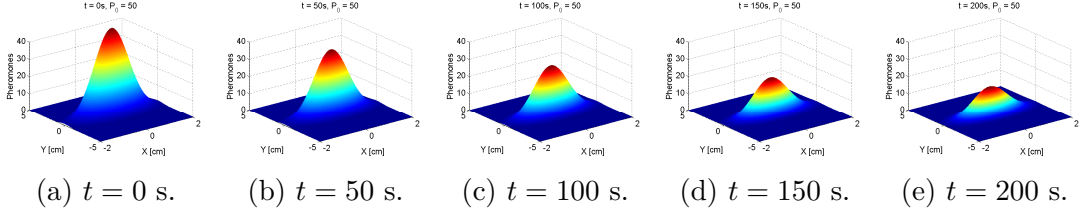


Fig. 5.6 Pheromone evaporation with $P_0 = 50$, $K = 0.0057$, $P_a = 0$ in Eq.(5.19).

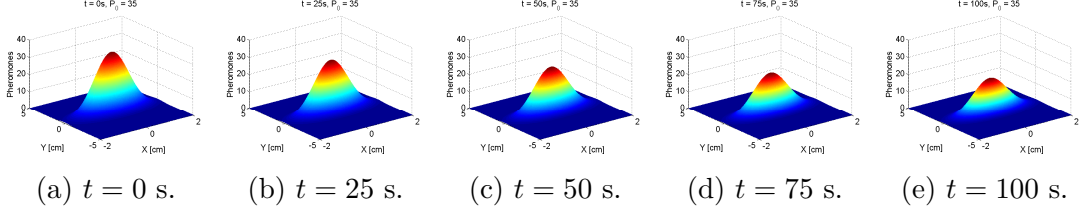


Fig. 5.7 Pheromone evaporation with $P_0 = 35$, $K = 0.0057$, $P_a = 0$ in Eq.(5.19).

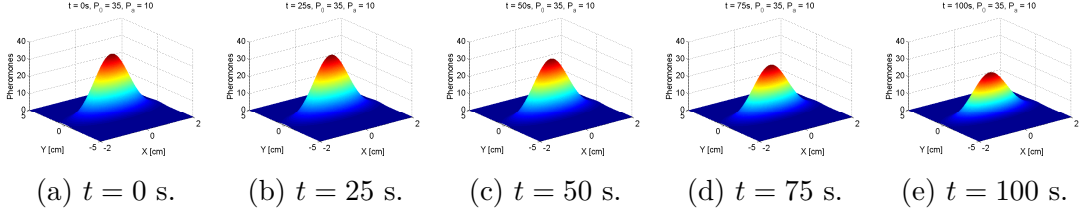


Fig. 5.8 Pheromone evaporation with $P_0 = 35$, $K = 0.0057$, $P_a = 10$ in Eq.(5.19).

and τ is a constant determined experimentally.

Other robots can also deposit pheromones on the top of already existing pheromones at any position in the map. The range of the sensor employed is denoted by z_{rng} . Pheromone deposition at a location is defined as the function,

$$P_{loc} = f(x, y). \quad (5.13)$$

Here, x and y represents the spatial coordinates of the robot's estimated position in the map. In the present work, the case of wheeled mobile robots and their motion in 2D space is discussed. The number of pheromones deposited at a certain location decays exponentially with distance from the robot's position, and the deposition is modeled as,

$$P_{loc} = \begin{cases} \gamma \cdot e^{-\left(\frac{x^2}{\lambda_1^2} + \frac{y^2}{\lambda_2^2}\right)} & \text{within } z_{rng} \\ 0 & \text{otherwise} \end{cases} \quad (5.14)$$

Pheromones are deposited only within the range of the sensor (z_{rng}). P_{loc} is a scalar field which associates amount of pheromones with the spatial location in the map. The gradient of this function generates a vector field which indicates the direction in which there is the largest increase in pheromones, and also the magnitude of those vectors in the vector field to the robot,

$$\nabla P = \frac{\partial P_{loc}}{\partial x} \hat{i} + \frac{\partial P_{loc}}{\partial y} \hat{j}, \quad (5.15)$$

where, $\hat{i}, \hat{j}$ are standard unit vectors. Substituting P_{loc} , we get,

$$\begin{aligned} \nabla P &= \\ & \frac{\partial}{\partial x} \gamma \cdot e^{-\left(\frac{x^2}{\lambda_1^2} + \frac{y^2}{\lambda_2^2}\right)} \hat{i} + \frac{\partial}{\partial y} \gamma \cdot e^{-\left(\frac{x^2}{\lambda_1^2} + \frac{y^2}{\lambda_2^2}\right)} \hat{j} \\ &= -\frac{2x\gamma}{\lambda_1^2} \cdot e^{-\left(\frac{x^2}{\lambda_1^2} + \frac{y^2}{\lambda_2^2}\right)} \hat{i} - \frac{2y\gamma}{\lambda_2^2} \cdot e^{-\left(\frac{x^2}{\lambda_1^2} + \frac{y^2}{\lambda_2^2}\right)} \hat{j} \\ &= -2\gamma \cdot e^{-\left(\frac{x^2}{\lambda_1^2} + \frac{y^2}{\lambda_2^2}\right)} \cdot \begin{bmatrix} x & y \\ \lambda_1^2 & \lambda_2^2 \end{bmatrix} \times [\hat{i} \quad \hat{j}]^T. \end{aligned} \quad (5.16)$$

As compared to a scalar representation of the pheromones in the map, the gradient vector field provides more information to the robots in pheromone trailing. For large area pheromone depositions, the gradient points the robot in the direction of the maximum rate of increase of the pheromones, and its magnitude is the slope of the graph in that direction, which enables the robot to properly track the pheromones. Moreover, the deposition is probabilistic and if the localization of the robot is good, more pheromones are deposited. However, if the robot fails to localize itself, less or no pheromones are deposited. The entire pheromone deposition scheme is shown in the flowchart in Fig.5.5.

Multiple robots may deposit pheromones at the same or nearby locations and the two pheromone fields gets merged by Eq.(5.16). This is represented in Fig.5.4 which shows the additive pheromones resulting from deposition in the vicinity.

5.4.2 Pheromone Evaporation

Like in nature, the pheromones are modeled to evaporate with time. The rate of evaporation of pheromones is proportional to the difference in the pheromones deposited (P) and that of the ambient or nearby pheromones (P_a). In other words, pheromones deposited at a lonely location with no nearby (i.e. within the sensor range z_{rng})

pheromones evaporate faster. If the robot(s) deposit large number of pheromones around an area, it is an indication of a trail towards a source (ex. food) and the presence of ambient pheromones slow down the evaporation process. The rate of evaporation of pheromones is given by,

$$\frac{dP}{dt} = -K \cdot (P - P_a), \quad (5.17)$$

where, K is a constant. Equation (5.17) is a separable differential equation. Integrating this equation, we get,

$$\begin{aligned} \int \frac{1}{(P - P_a)} dP &= \int -K dt \\ \Rightarrow \ln |P - P_a| &= -Kt + c \\ \Rightarrow |P - P_a| &= e^{-Kt+c} \\ \Rightarrow |P - P_a| &= e^{-Kt} \cdot e^c \\ \Rightarrow |P - P_a| &= C \cdot e^{-Kt} \end{aligned} \quad (5.18)$$

which gives,

$$\Rightarrow P(t) = C \cdot e^{-Kt} + P_a, \quad (5.19)$$

which is a general solution of the differential equation, and C is a constant. The constant K is a controlling factor which governs the rate of evaporation of the pheromones in the map. The factors K and C are determined by the boundary conditions. At $t = 0$, when there are P_0 pheromones, from Eq.(5.19),

$$\begin{aligned} P(t = 0) &= P_0 = C + P_a \\ \Rightarrow C &= P_0 - P_a. \end{aligned} \quad (5.20)$$

At time $t_m > 0$, the pheromones (P_{t_m}) are given from Eq.(5.19) as,

$$P(t = t_m) = P_{t_m} = C \cdot e^{-Kt_m} + P_a. \quad (5.21)$$

Substituting the constant C given by Eq.(5.20) into the above equation,

$$P_{t_m} = (P_0 - P_a) \cdot e^{-Kt_m} + P_a, \quad (5.22)$$

solving which we get the expression for the constant K as,

$$K = -\frac{1}{t_m} \ln \left(\frac{P_{t_m} - P_a}{P_0 - P_a} \right). \quad (5.23)$$

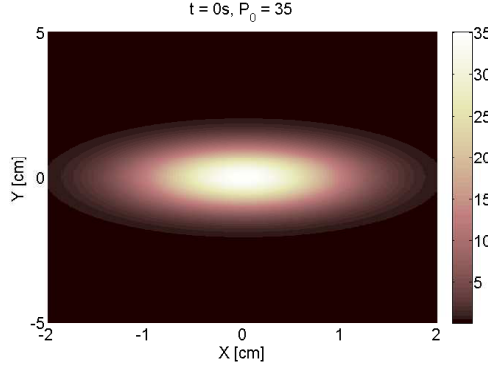


Fig. 5.9 Alternate view of pheromone deposition in case of Fig.5.6a. Pheromone deposition is in the error ellipse given by the eigenvalues and eigenvectors of the covariance matrix. The lighter color intensities show more pheromone deposition.

A threshold is fixed for the amount of pheromones deposited at a location and sensed by the robot below which the robot ‘rejects’ the trail. If a following robot comes across a very weak trail, it does not follow it and rather carries its own exploration. If P_0 is the total pheromone count in the beginning, this threshold is set to δP_0 , where $0 < \delta \leq 1$. For maximum amount of pheromones deposited at a location, the total time it takes for the pheromone to evaporate to the threshold value δP_0 is calculated. From Eq.(5.22),

$$\delta \cdot P_0 = (P_0 - P_a) \cdot e^{-Kt} + P_a. \quad (5.24)$$

Substituting the value of K from Eq.(5.23) into the above equation,

$$\begin{aligned} \delta P_0 &= (P_0 - P_a) \cdot \exp \left(\frac{t}{t_m} \ln \left(\frac{P_{t_m} - P_a}{P_0 - P_a} \right) \right) + P_a \\ \Rightarrow \ln \left(\frac{\delta P_0 - P_a}{P_0 - P_a} \right) &= \frac{t}{t_m} \ln \left(\frac{P_{t_m} - P_a}{P_0 - P_a} \right) \end{aligned} \quad (5.25)$$

Solving,

$$t = t_m \frac{\ln\left(\frac{\delta P_0 - P_a}{P_0 - P_a}\right)}{\ln\left(\frac{P_{t_m} - P_a}{P_0 - P_a}\right)}. \quad (5.26)$$

While modeling the system, Eq.(5.26) is useful in calculating the total time it takes for the pheromones to evaporate below a threshold level considering the size of the area, and the speed of the robot. The parameter δ tunes the rate of evaporation according to these characteristics of the environment and the robot.

Figure 5.6 shows the evaporation of pheromones with time when the robot deposits a large value of pheromones ($P_0 = 50$) with no ambient pheromones ($P_a = 0$). The value of parameter K is determined so that pheromones evaporate to a threshold value of 18 in around 300 seconds or 5 minutes (i.e. $P_{t=300} = 18$). At the same rate, Fig.5.7 shows the evaporation of pheromones with initially $P_0 = 35$ amount of pheromones deposited. It takes about 116 seconds for the pheromones to evaporate to the threshold value. This value is chosen considering the speed of the robots and the characteristics of the environment. Typically, in indoor environments a speed of 0.5 m/s is safe for the mobile robots. Considering the other factors like the dynamics of the environment, smoothness of the surface, etc. Eqs.(5.19), (5.23) and (5.26) are used to tune the values and determine appropriate parameters for the trailing robot to catch-up with the leader robot before the pheromones evaporates to a threshold value. Figure 5.8 shows the evaporation of pheromones with initially $P_0 = 35$ amount of pheromones deposited in presence of ambient pheromones with $P_a = 10$. It can be seen that in presence of ambient pheromones the evaporation is slower compared to that with no ambient pheromones as shown in Fig.5.7.

Figure 5.9 shows an alternate view of pheromone deposition in case of Fig.5.6a. Pheromone deposition is in the error ellipse given by the eigenvalues and eigenvectors of the covariance matrix. The lighter color intensities show more pheromone deposition. Compared to a circular deposition, an ellipse deposition precisely models the pheromone deposition in case of long corridors where the ratio of eigenvalues can be high.

5.5 Results

This section describes results in simulation and real environments. If the map of the environment is already available, then the robot needs to localize itself in the map.

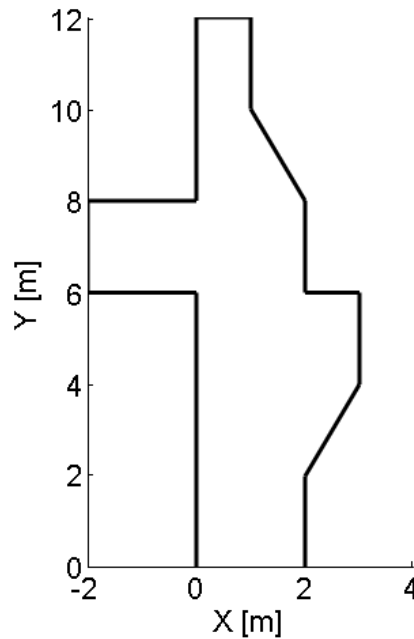


Fig. 5.10 Simulation Environment

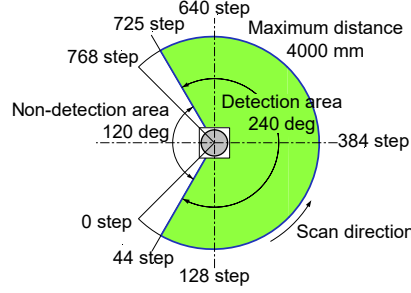
In some cases, information like floor plans can also be integrated to improve robot localization, as proposed in [92, 196]. However, it is not safe to assume that map is always available. Hence, mapping and localization needs to be carried simultaneously through a SLAM module. SLAM is an integral component of mobile robots, and in both the simulation and real environment test cases SLAM has been employed. However, in the availability of a map, the problem reduces to a localization only problem, and the proposed pheromone deposition scheme will work in both the cases.

Table 5.1 Laser range sensor for simulation

Laser range sensor	Simulated model of URG-04LX
Distance range	60 mm ~ 4000 mm
Accuracy	20 ~ 1000 mm : ± 25 mm 1000 ~ 4000 mm : ± 2.5 %
Distance resolution	3.84 mm ((4000-60) mm / 1024)
Scan angle	240 deg
Angular resolution	1.875 deg (240 deg / 128)



(a) URG-04LX



(b) Measurement range

Fig. 5.11 Figure of laser range sensor URG-04LX (HOKUYO AUTOMATIC CO., LTD.) and measurement range

5.5.1 Results in Simulation Environment

Figure 5.10 shows the simulation environment used in the experiment. It consists of a narrow passage with opening passages. The line segment based EKF-SLAM algorithm [143] (discussed in Chapter 7) was used to localize the robot in this environment. It extracts line features from the laser range sensor information and uses the line features to build the map as well as localize the robot in the map as it moves. Simulation based on URG-04LX (HOKUYO AUTOMATIC) laser range sensor (Fig.5.11) whose specifications are shown in Table.5.1 were performed. Gaussian noise of ± 25 mm and $\pm 2.5\%$ was added in distance range of 1000 mm and 4000 mm, respectively, to purposefully bring some uncertainty to robot's localization. Effects of errors caused by the shape of objects due to reflection etc. were not taken into consideration. Compared to the real LRS, the angular resolution was set to $1/8$ for faster computation. The simulation was carried on MATLAB 7.12 (R2011a) running on a Linux machine with Intel Core i7-4500U CPU, 1.80 GHz, and 16GB RAM.

The details of the SLAM algorithm can be found in [143], but what is relevant to the proposed idea is that the robot can estimate the errors in x , y , and orientation ϕ at

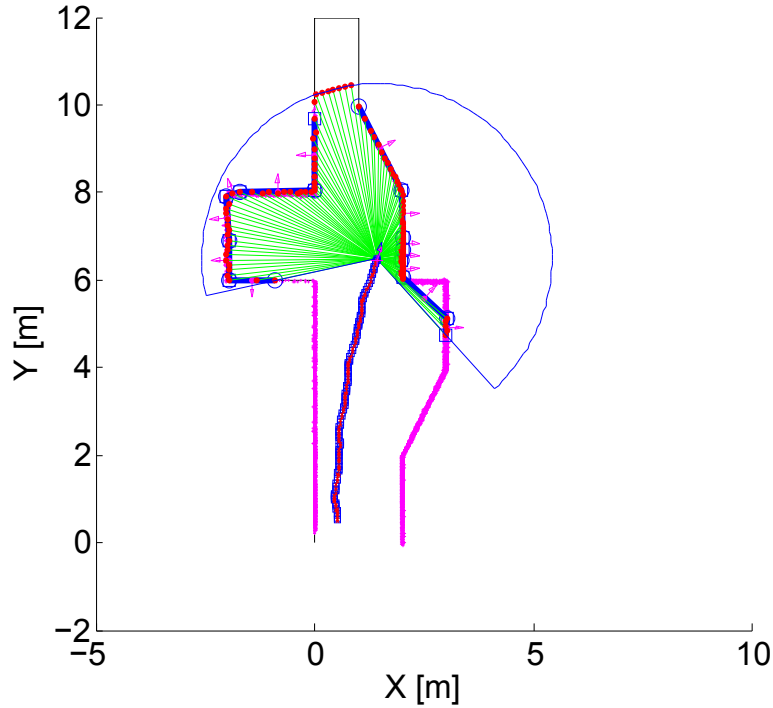
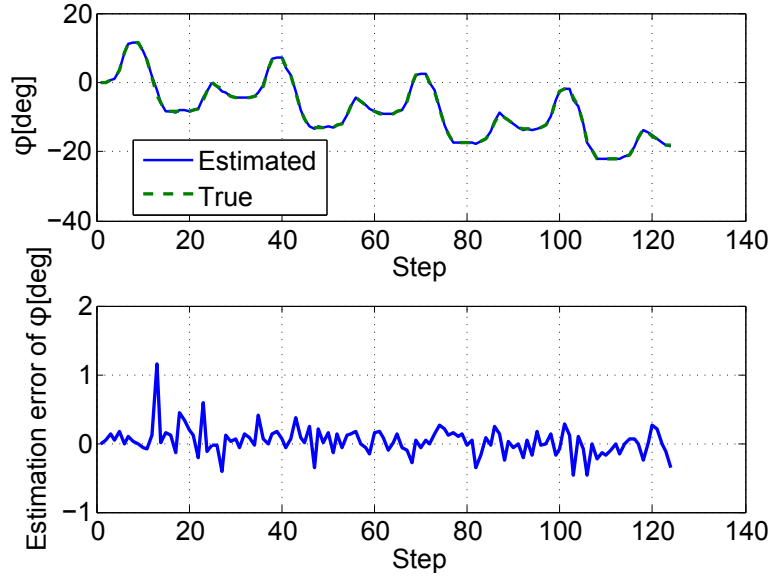
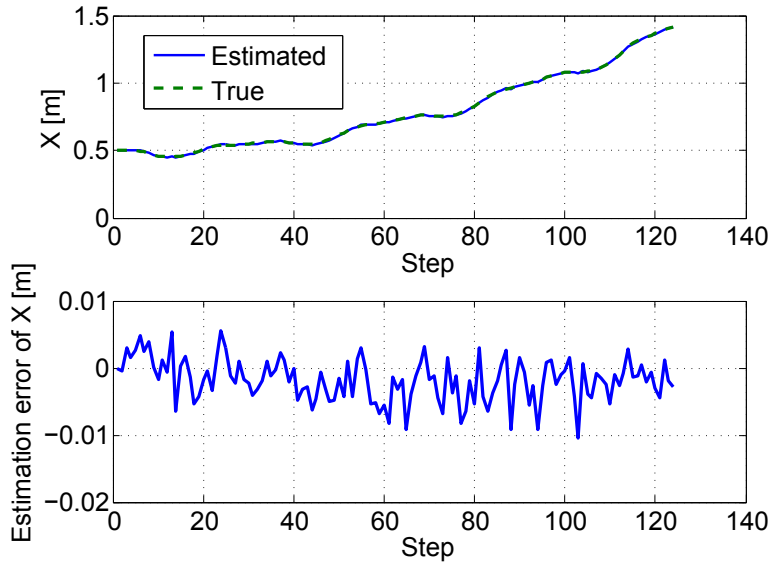


Fig. 5.12 Result of pheromone deposition while localizing in map

each step. Figure 5.12 shows the result of localization and mapping at a particular step in which the constructed map is shown by small '+' mark in pink color. Sensor location is shown by a larger '+' sign in pink color, and similarly sensor orientation is shown by an arrow.

Figure 5.13, Fig. 5.14, and Fig. 5.15, show the estimation errors in ϕ , x , and y , respectively, at each step of the robot, along with the respective true and the estimated values. As mentioned earlier, pheromone deposition is scalar in nature, i.e. the robot can deposit pheromones at the same place irrespective of its orientation. Hence, the orientation in uncertainty is not considered. The errors estimated in x and y are a direct reflection of the spatial uncertainty of the robot in the map. Figure 5.16 represents the combined error of this spatial uncertainty at each step of robot, where the robot took a total of 124 steps in the experiment. It can be seen that at the start of the operation, the uncertainty of the robot is zero, and it starts to vary as the robot moves in the environment.

Figure 5.17 shows the total number of pheromones deposited by the robot at each step. The parameter γ in Eq.(5.12) was set to 0.2. A zero uncertainty at the start of the operation would result in an infinite pheromone deposition by Eq.(5.12), but an upper

Fig. 5.13 Estimation error of sensor angle ϕ Fig. 5.14 Estimation error in X -axis

limit of 100 was set for the maximum number of pheromones which could be deposited in the map. For a comparative analysis, Fig.5.18 shows the positional uncertainty of the robot and the pheromones deposited at each step, in the same graph. The amount of pheromones are shown in a *log* scale, whereas, the uncertainty is scaled up three times. It can be seen in Fig.5.18 that at the start of the operation with no uncertainty, the robot deposits a maximum (100) number of pheromones. It can also be seen in

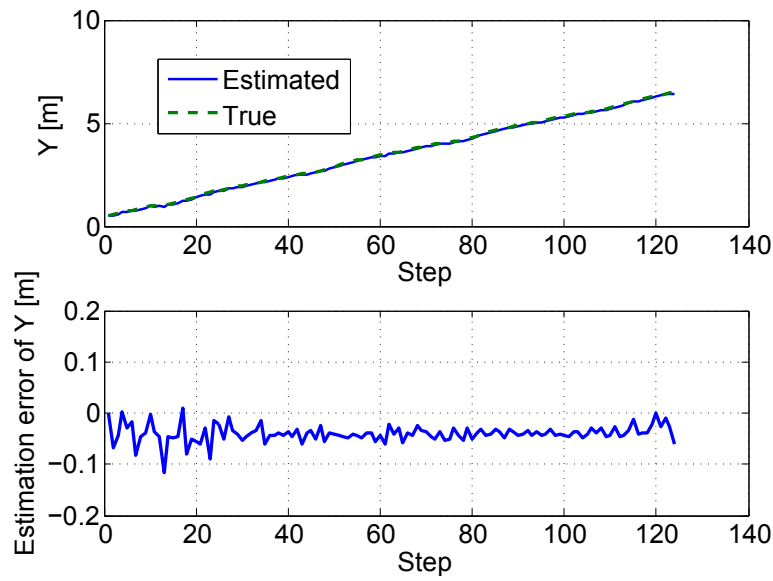
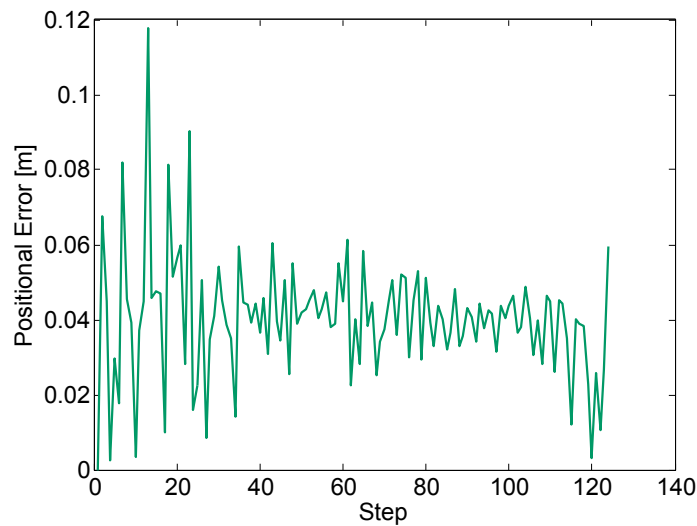
Fig. 5.15 Estimation error in Y -axis

Fig. 5.16 Combined error representing spatial uncertainty

Fig.5.18 that the peaks in uncertainty of the robot (shown in green color) corresponds to the drops in the amount of pheromones deposited (shown in red color). This means that robot had deposited less pheromones at uncertain locations. On the other hand, drops in the uncertainty maps well to the peaks in the pheromones deposited, at any step. This validates that the robot deposited more pheromones at locations with good localization.

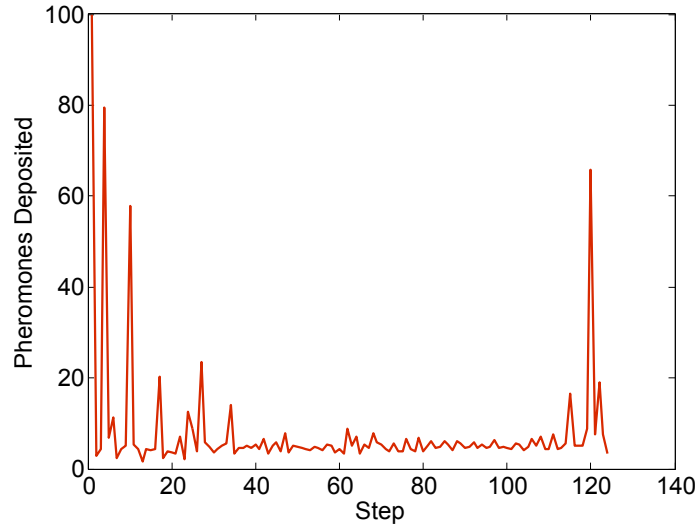


Fig. 5.17 Value of pheromones deposited

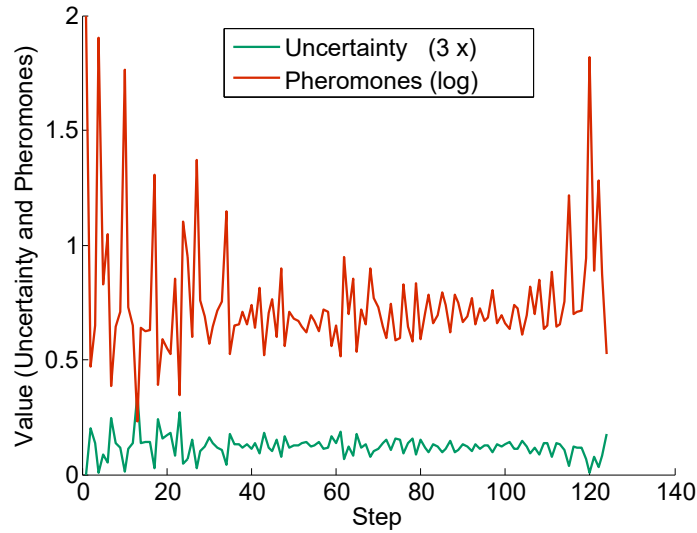


Fig. 5.18 Comparison of Pheromones and Uncertainty

5.5.2 Results in Real Environment

For the real environment, Pioneer-P3DX [131] and Kobuki Turtlebot [182] robot shown in Fig.2.9a and Fig.2.9b were used. Pioneer P3DX was the leader robot, and Turtlebot was programmed to follow the trails of Pioneer P3DX. Living creatures like ants deposit chemical pheromones in the environment. Solutions have been proposed to mimic biology by actually depositing and detecting chemicals by the robots as in [60, 134]. However, such attempts are limited by the cost of physical sensors and are not scalable.



Fig. 5.19 Real Environment

Virtual pheromones, which are mostly software counterparts of the chemicals, have also been attempted to be realized using Infra-Red sensors mounted on the top of mobile robots for pheromone communication. Other researcher have proposed to use RFID sensors [107, 175]. In this chapter, a virtual pheromone deposition similar to [176] is used. In the present work, virtual pheromones are deposited on a common map located on a central remote server which is accessible to all the mobile robots. All the robots and the server are on the same wireless network for communication. The map is discretized into small cells of unit area. The estimate robot pose $[x \ y]^T$ may have fractional values which are rounded off to the nearest integer values for deposition in cells. Since all the robots deposit the virtual pheromones on the central server, pheromone addition from multiple robots and evaporation easily becomes feasible. This flexibility, however, comes at a cost of frequent communication.

It is assumed that the leader and trailing mobile robots start from the same location which allows them to have the same ‘anchor’ in the global map. This allows the robots to create and maintain a correspondence between the spatial coordinates and features of the map. The situation when multiple robots initialize operation from different points of the map is a different area (multi-robot mapping and localization) of research and not considered in this chapter. For those scenarios, a relative formulation of the relationship between multiple poses of multiple robots needs to be maintained and solutions have been provided in various previous works [6, 71, 83, 87, 154].

Both the robots had SLAM units to build a map and localize themselves in the cluttered corridor environment shown in Fig.5.19 with many similar features. The total length of the corridor was about 36 meters, and 3 meters wide. Grid map of the environment

was made using particle filter and modified standard ROS (Robot Operating System [135]) mapping library [15]. Each robot maintained its own map, and the pheromones calculated were deposited in its own map and the server.

Motion Model

Both Pioneer-P3DX and Turtlebot are two wheeled differential drive robots. the motion model of the robot is first described. The distance between the left and the right wheel is W , and the robot state at position P , is given as $[x, y, \theta]$. From Fig.2.9c, the motion model is calculated as given in Section 2.7.

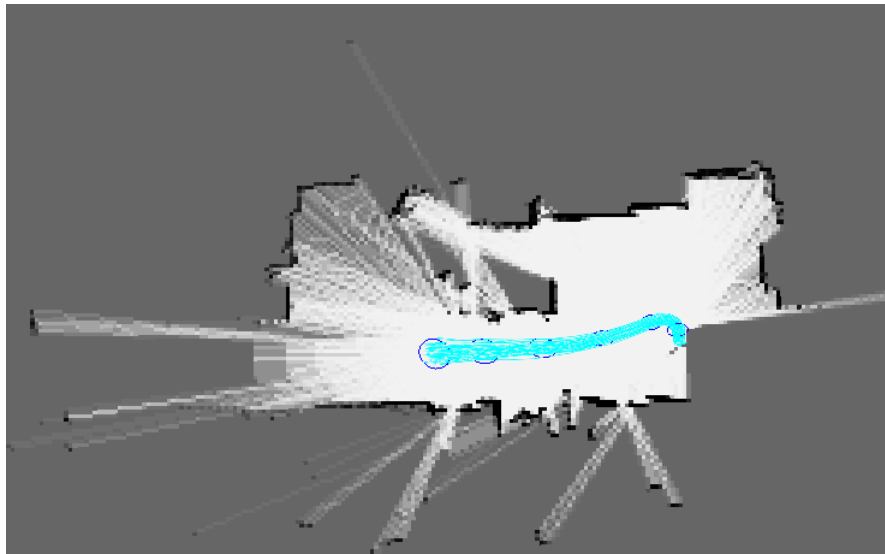


Fig. 5.20 Good localization at the start of the operation. Uncertainty in localization is represented by blue ellipses and particles in cyan.

Behavioral Difference of Follower Robot With & Without Proposed Method

A robot may get lost in actual scenarios of cluttered disaster sites due to lack of visibility, noise and many other factors. However, since the experiment environment (Fig.5.19) was simple, in order to realize a ‘leader lost’ scenario, the number of particles for the leader robot were purposefully set to a low value. Instead of depositing the pheromones at each and every step, the robot was programmed to deposit the pheromones at intermittent time intervals according to the estimated positional uncertainty in the map. Figure 5.20 shows the map created by the robot at the start of the operation. The blue ellipses corresponds to the spatial uncertainty in robot’s localization. Smaller

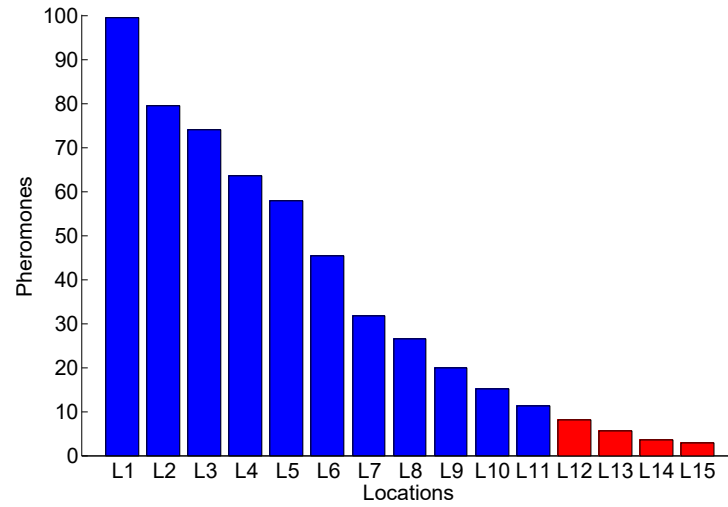


Fig. 5.21 Pheromones deposited at different locations shown in Fig.5.22. Blue areas show pheromones above threshold (10). Total pheromones at location L12 were 8.15.

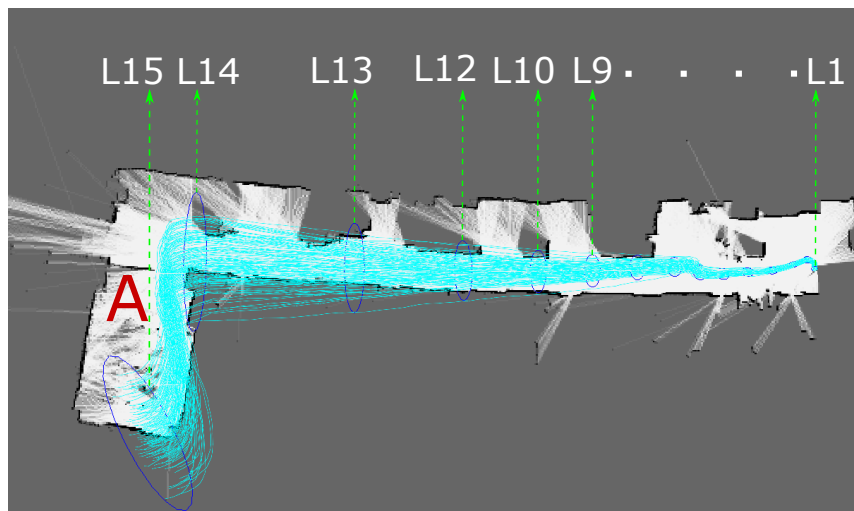


Fig. 5.22 Leader (P3DX) robot's uncertainty at different locations L1 to L15. Uncertainty in localization is represented by blue ellipses and particles in cyan. The uncertainty well within the bounds of the corridor until location L12. After location L12 there is bad localization and minimum number of pheromones are deposited. The robot goes to location A which is highly uncertain location due to large uncertainty of the leader robot.

ellipses reflects smaller uncertainty, and vice versa. Each particle represents a 'belief' of the robots pose. The particles are shown in cyan color. The concentration of particles over an area is an indication of how good the localization is. If the particles are densely

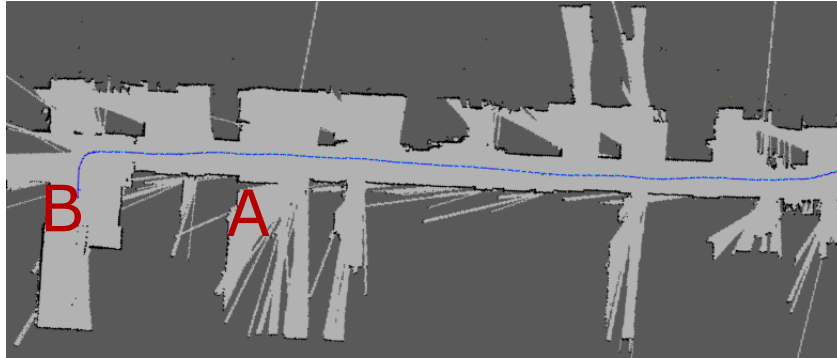


Fig. 5.23 Trajectory of the follower robot (Turtlebot). ‘Leader lost’ is detected at location L12, and Turtlebot goes to location B. Blindly following the pheromones using the traditional pheromone deposition schemes would have caused Turtlebot to end up at uncertain location A, which is avoided using the proposed uncertainty modeling.

concentrated in an area, it means that there is less error in estimating the robot’s position in the map. More dispersed particles indicates that the robot is unable to calculate a good estimate of its position.

Particle filter can represent multi-modal probability densities. In the present work, it is assumed that the particle filter models a single hypothesis as the indoor environment used in the experiment has many unique features so that the particles converge quickly after a few scans. In case the environment is devoid of unique features, the particle filter may generate multiple hypothesis representing multiple possible locations of the robot. In that case, the pheromones will be deposited at multiple estimated positions of the map, some of which will be wrong locations. Once unique features are found and the particles have converged to a single hypothesis, the wrongly deposited pheromones needs to be removed or evaporated quickly. This can be achieved by tracking the number of hypothesis generated by the particle filter. If it is more than one, the locations of pheromone deposition can be buffered in memory, and later when the particles converge to a single hypothesis the pheromones deposited at wrong locations can be eliminated. However, this case is not considered in the present work and is planned to be addressed in a future extension of the work.

The robot assumes its position to be the mean location of all the particles. The calculation of covariance matrix in case of particle filter can be found in [10, 110, 119, 163, 180]. It can be seen that at the start of the operation, there is less uncertainty and all the particles are concentrated over a smaller area as indicated by the size of the ellipses in Fig.5.20.

Figure 5.22 shows the uncertainty of the robot at different locations of the map from location L1 to L15. In the 3 meter width corridor, the positional uncertainty of the robot was well within the bounds until location L12. After location L12, the robot was unable to localize well and the uncertainty exceeds the width of the corridor. Figure 5.21 shows the total number of pheromones deposited at locations L1 to L15. For a long corridor such as the one used in the experiment, the threshold pheromone value was set to 10 for the trailing robots to catch-up with the leader robots. The total number of pheromone deposition at location L11 was 11.35, and at location L12 was 8.15 pheromones which is below the set threshold.

As indicated in Fig.5.22, the leader robot explores location A in the map. Since the positional uncertainty of the leader robot is too high, there is a high probability that the robot is exploring wrong area of the site. Figure 5.23 shows the trajectory of the follower Turtlebot robot. From the point of view of the trailing Turtlebot robot which sensed a very less amount of pheromones after location L12, Pioneer-P3DX robot was considered ‘lost’. It can be seen from Fig.5.21 that the total amount of pheromones deposited is a direct indication of the uncertainty of P3DX for the trailing Turtlebot robot. Turtlebot abandons following the pheromones after location L12. As shown in Fig.5.23, Turtlebot explores location B and not A.

Notice that, with the traditional pheromone deposition schemes which do not model the uncertainty, Turtlebot would have blindly followed the pheromone trails of the leader robot and ended up at location A of Fig.5.23 which has high probability of being a wrong area due to the high positional uncertainty of the leader robot. Moreover, all the subsequent follower robots too would have ended in the uncertain location A of the site. However, with the proposed uncertainty modeling, the follower robot can know that the leader was lost at a particular position, and can instead explore other areas of the site (like the location B in the experiment).

5.5.3 Results of Speed Adaptation of Trailing Robots using the Proposed Method

While there are many aspects of how the proposed technique improves the efficiency of the system, some practical examples are considered. In search and rescue operations, an indication of higher uncertainty of leader robots is useful for a trailing robot if it fails to find the leader robot at some location, and can instead rely on some other important factors rather than the virtual trails themselves. Another important application is in speed adaptation by a trailing robot depending on the intensity of

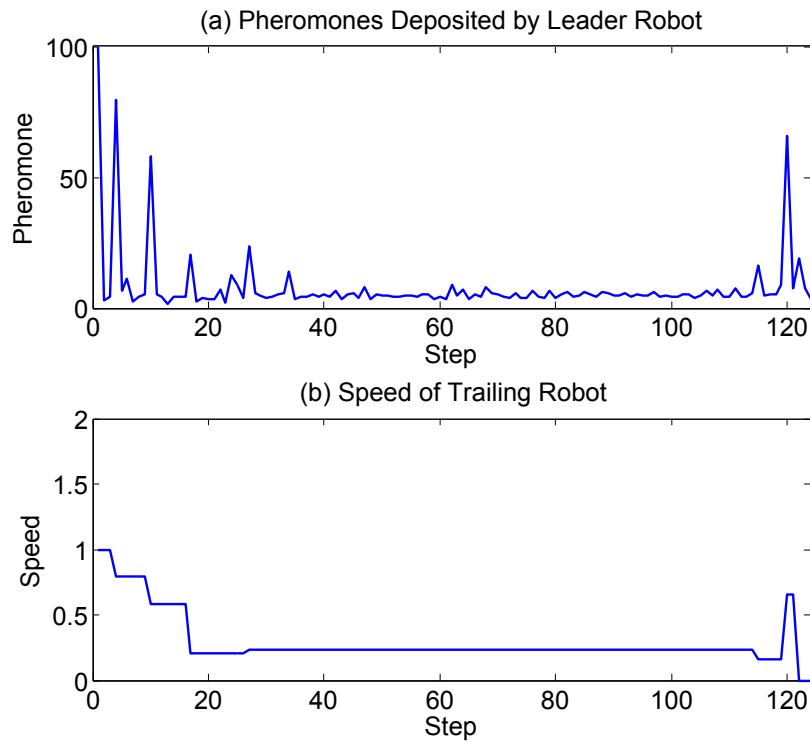


Fig. 5.24 Speed variation of the second trailing robot with respect to the pheromones deposited by the first robot (in simulation experiment). (a) Pheromones deposited by the first robot. (b) Speed of the second trailing robot.

the pheromones encountered at different locations. A trailing robot can increase its speed if it consequently encounters high deposition of pheromones in the map, as it is an indication of being on a good track and the proximity of other robots. On the contrary, if a trailing robot encounters weak and dispersed pheromone trails then it is an indication of the uncertainty of the leader robots and a large separation between

Table 5.2 Pheromone range and corresponding robot speed

Pheromone Range	Speed (m/s)
60-100	1
30-60	0.65
15-30	0.45
0-15	0.15

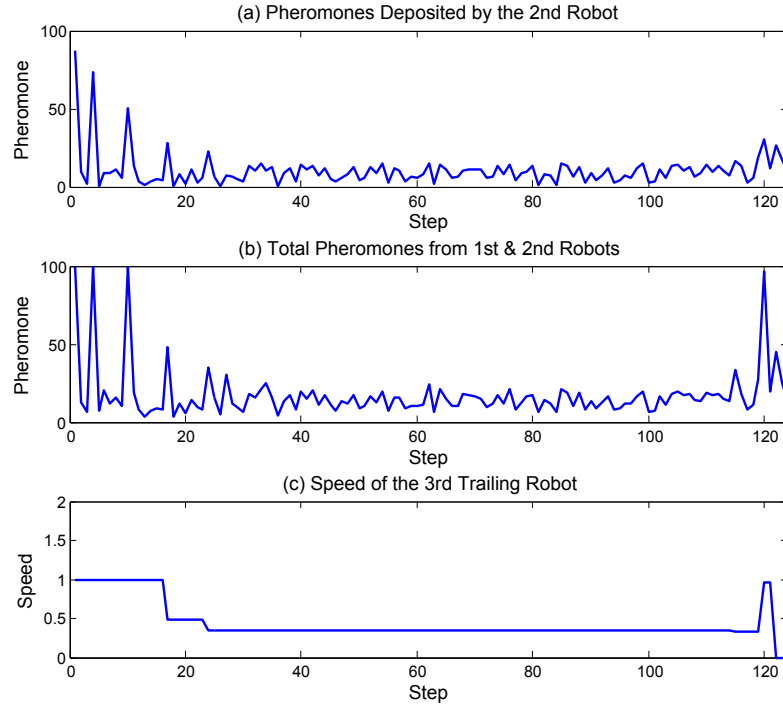


Fig. 5.25 Speed variation of the third trailing robot with respect to the total pheromones deposited by the first and second robots (in simulation experiment). (a) Pheromones deposited by the second robot. (b) Total pheromones deposited by the first and second robots. (c) Speed of the third trailing robot.

the two. In this case, a trailing robot is supposed to lower its speed for more robust search of the leader robot.

Although experiments regarding pheromone based speed control of trailing robot are discussed in this section, it does not imply that it should be the only criterion for speed control. Robots must control their speed taking many factors into consideration like the static and dynamic obstacles in the environment, width of the path, visibility (in case of vision sensors), width of the path, steepness and smoothness of the path, intersecting paths with other robots or people, and others.

Figure 5.24a shows the amount of pheromones deposited by the leader robot (first robot). This pheromone deposition is the result of SLAM employed by the robot in Section 5.5.1. Figure 5.24b shows how the second robot adapts its speed according to the sensed pheromones deposited by the first robot. The maximum speed of the robot was set to 1 m/s. It can be seen that initially (step number 1 to 18), a large pheromones are sensed by the trailing robot and hence its speed is high. However, as the uncertainty increases, the trailing robot reduces its speed, accordingly. Figure 5.25

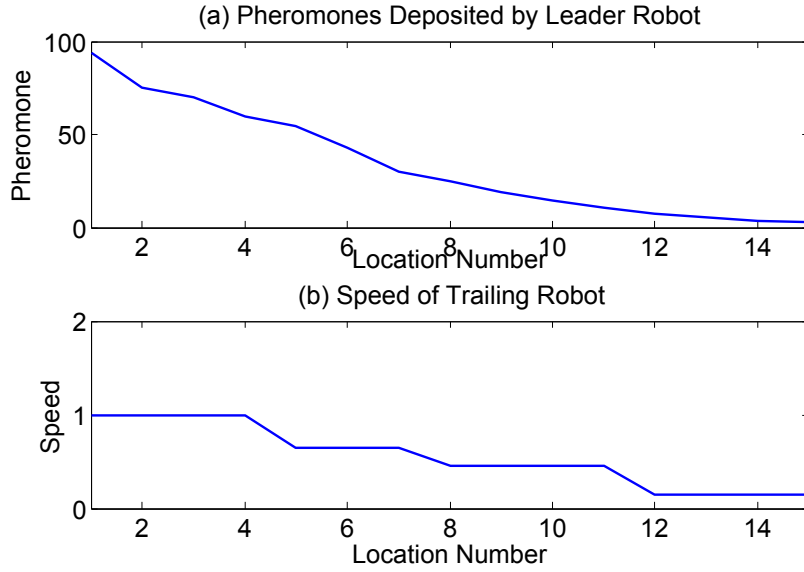


Fig. 5.26 Speed variation of the trailing Turtlebot robot with respect to the pheromones deposited by Pioneer P3DX robot (in real experiment). (a) Pheromones deposited by Pioneer P3DX robot. (b) Speed of trailing Turtlebot robot.

shows the speed adaptation by a third trailing robot. Notice that, the second trailing robot also has a SLAM component and will also deposit pheromones on the top of the first robot's pheromones. Figure 5.25a shows the pheromone deposition by the 2nd robot. Figure 5.25b shows the total pheromones from the first and the second robot with a maximum pheromone threshold of 100. Figure 5.25c shows the adaptation in speed by the third robot according to the total pheromones sensed at each step. Figure 5.26 shows the example of speed adaptation in real environment. Figure 5.26a shows the total pheromones deposited at the 15 locations on the map of Fig.5.22. Figure 5.26b shows the adaptation in speed by the trailing robot. For the real environment, the speed was set to specific values according to a range of pheromones described in Table 5.2.

Notice that, a robot can not only adapt its speed, but can also turn on/off some sensor modules based on confidence or uncertainty of pheromones sensed at different locations. For example, when it is found that the leader robot is lost, the trailing robot can change parameters of SLAM or sensors (like increase the FPS of camera), or switch on/off other sensors for more robust search. On the contrary, it can also turn off some features to save computation. A particularly useful case in case of particle filters is to increase the number of particles for more robust localization based on the number of pheromones encountered.

5.6 Summary and Discussion

Pheromone trailing is a well studied phenomenon in robotics and a wide range of applications have successfully been proposed. Traditionally, pheromones have been thought of as static quantities with binary qualities, i.e. they just indicate the presence or absence of something. A presence of pheromones indicates that the robot took this path, and its absence means that this path has been unexplored. Although a very powerful technique, traditional pheromone trail techniques have largely been limited to simulations. This is mainly because the dynamics and challenges of mobile robots working in fields are immense and hard to model. One such factor is a good localization of the robot in such dynamic environments which is difficult to achieve. Since the location at which pheromones are deposited is very crucial for a multi-robot system employing virtual pheromone trailing, it is strongly suggested that the positional uncertainty of the robot needs to be modeled in pheromone deposition. This modeling, as proposed in the current work, provides extra information to the trailing robot about the preceding robot. A ‘confident’ deposition concentrated in a small area indicates that the preceding robot was certain about its location at that place. Whereas, low and dispersed pheromones indicates that the preceding robot was not sure about its location while depositing pheromones.

A mathematical model of uncertainty modeling in pheromone deposition was presented. The proposed model was tested in both simulation and real environment. The proposed technique will work irrespective of the SLAM technique employed with the robot. Calculating the robot’s positional uncertainty is computationally expensive. However, SLAM is an indispensable component of every mobile robot and it is rare to see mobile robots relying solely on wheel encoder data. The proposed technique can be used in conjunction to any SLAM module to deposit pheromones. The various parameters can experimentally be found using the mathematical model for various environments.

The proposed model can easily be extended to a 3D case in which an unmanned air vehicle deposits pheromones in 3D space. Moreover, pheromone deposition in case of multiple hypothesis generation by particle filter is also considered as potential future work.

Chapter 6

Hitchhiking in Multi-Robot System

Overview

Virtual trailing based multi-robot task coordination and navigation as discussed in Chapters 4 and 5 are useful over large distances, particularly when the leader robot is not in the field of view (FoV) of the follower robot. However, there are cases when the leader robot is in the FoV of the trailing robot. In such scenarios, a more straightforward solution is to directly follow the robot. In these cases, such direct following can be incorporated with other mechanisms for performance efficiency in multi-robot systems. A robot must localize, do path planning, obstacle avoidance, and update the map if there are new features in the environment while navigating to its goal location. When two or more robots have the same goal location, it is a waste of computing if all the robots performs all these tasks individually. To this end, this chapter introduces a novel concept of ‘hitchhiking’ as a means of efficiency gains. The work in this chapter builds upon the work in previous chapters (Chapter 4 and 5) and is interdependent on, and assumes the availability of SLAM module (Chapter 7), path planning module (Chapter 2), and resource sharing (Chapter 3).

Chapter Abstract

Hitchhiking is a typical phenomenon found only in humans. It is a means of transportation gained by asking other people for a ride. A multi-robot system is developed in this work which is the first of its kind to incorporate hitchhiking in robotics and discuss its advantages. The hitchhiker robot can save computation required for (a) path planning, (b) localization, (c) dynamic obstacle avoidance, and (d) map update. The system has an award mechanism in which the hitchhiker ‘owes’ to

the driver robot which receives points. The admissible characteristics of the hitchhiker and the driver robots are discussed. Important points about when should hitchhiking be allowed and when not are also discussed.

6.1 Introduction

In order to set the scene for incorporating hitchhiking in robotics, consider the following example:

Example: Bob lives in a small village called Dinan in France and wants to go to Paris to see the Eiffel tower, but is short of resources. On the highway to Paris in his village of Dinan, Bob tries hitchhiking at a gas station where many riders stop. Alice, who owns a Ferrari, happens to stop at that gas station and agrees to give a ride to Bob. Alice is not actually going to the location of the Eiffel tower itself, but drops Bob at a location in Paris which is nearest to the Eiffel tower and gives some instructions. Bob thanks Alice, and walks his way to enjoy the beautiful Eiffel tower.

It can be seen that Bob had many advantages in this scenario:

- Bob had to do no path planning from his village of Dinan to the Paris city. Bob only had to plan a path from the location Alice dropped him to the Eiffel tower.
- During his journey from Dinan to Paris, Bob did not worry about his current location in the map, as he trusted Alice who is an accomplished driver.
- Since Alice was riding the car, Bob had to do absolutely nothing to avoid hitting the pedestrians or other vehicles, stop at the red signal, and so on.
- Bob saved money as well as energy which he would have spent if he was actively riding to Paris during the process.

Now consider a multi-robot system deployed at a large infrastructure for object delivery, cleaning, patrolling, and other tasks. Each robot of the multi-robot system needs to navigate from one location to other to perform its task. The paths of the robot often overlap completely or partially. Moreover, each robot is equipped with its own navigation unit, localization unit, dynamic obstacle avoidance unit, and SLAM unit. In the context of the Bob's hitchhiking example, follow questions arise:

Why should two (or more) robots with completely or partially overlapping paths perform their own path planning, collision avoidance, and localization separately? Can't one robot just hitchhike another robot which is going to the same goal location, and skip most of the redundant operations to save energy?

6.2 Hitchhiking in Robots

The following terminologies are defined and used throughout the chapter:

- **Driver**: Driver robot is defined as the robot which is the leader robot in the multi-robot scenario. The Driver does all the path planning, dynamic obstacle avoidance, map updating, and assists the follower robot in the hitchhiking process.
- **Hitchhiker**: Hitchhiker is defined as the robot which follows the driver robot through visual servoing.

6.2.1 Hitchhiking Mechanism



Fig. 6.1 QR code and camera system for coupling.

Hitchhiking is achieved through visual servoing. A complete description of visual servoing is beyond the scope of the main idea of the proposed work. However, a brief description is provided.

Visual servoing [74], also known as vision-based robot control is a technique which uses feedback information extracted from a vision sensor (like camera) to control the motion of a robot. Visual Servoing control techniques are broadly classified into the following types[23]:

- Image-based Visual Servoing (IBVS)
- Position/pose-based Visual Servoing (PBVS)
- Hybrid approach based Visual Servoing

IBVS was proposed by Weiss and Sanderson in [157]. The control law is based on the error between current and desired features on the image plane, and does not involve any estimate of the pose of the target. The features may be the coordinates of visual features, lines or moments of regions. IBVS has difficulties with motions very large rotations, which has come to be called camera retreat. PBVS [195] is a model-based technique (with a single camera). This is because the pose of the object of interest is estimated with respect to the camera and then a command is issued to the robot controller, which in turn controls the robot. In this case the image features are extracted as well, but are additionally used to estimate 3D information (pose of the object in Cartesian space), hence it is servoing in 3D. Hybrid approaches [86] use a combination of the 2D and 3D servoing.

Visual servoing is generally implemented through a marker (mostly artificial) and a camera. Each robot has an QR-code which is fixed on the back side. Each robot also has a forward facing camera. The QR-code and camera setup is shown in Fig.6.1.

6.2.2 Four Steps of Hitchhiking

Each robot of the multi-robot system is provided with a unique ID (R_{id}). They also have respective G_{loc} which is the goal location, T_p which is the task priority, and Pf is the profile of the robot. A profile comprises of the specification of the robot. The specification comprise mainly of the type of sensors attached to the robot, accuracy of the odometers and sensors, and robustness of the SLAM module. A profile provides logical parameters to compare the performance of two robots to navigate towards a goal.

Hitchhiking is carried out in the four steps which are graphically shown in Fig.6.2 and explained below:

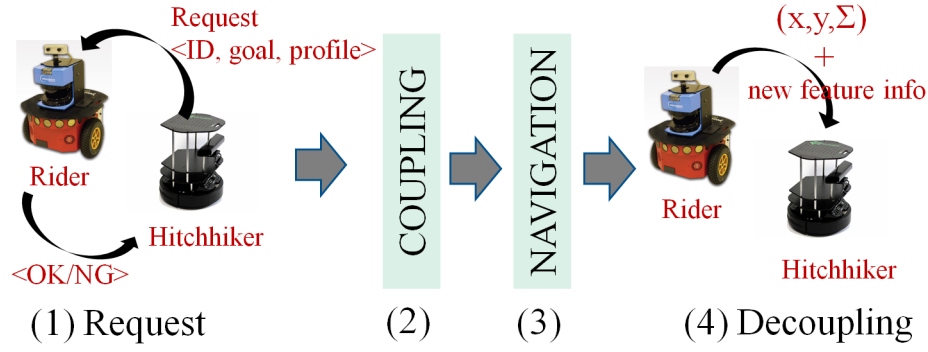


Fig. 6.2 Hitchhiking steps.

1. **Handshake:** The hitchhiker and the (potential) driver robots first exchange information. The hitchhiker sends a request to a potential driver. The request comprises of:

$$\text{Request} = \{R_h, G_h, T_h, Pf_h\}, \quad (6.1)$$

where, R_h is the ID of the robot, G_h is the goal location, T_h is the task priority, and Pf_h is the profile of the hitchhiking robot. The potential driver robot checks if its goal location (G_d) is in proximity of G_h . If yes, then the driver responds with an acknowledge message (ACK) to the hitchhiker. If the goals are different, then no-acknowledge (NACK) message is communicated and the hitchhiking process is terminated. This process is called as a handshake.

2. **Coupling:** The next step is coupling between the hitchhiker and the driver. Coupling is defined as the process by which the hitchhiker comes behind the driver robot and the QR-code behind the driver is recognized to initiate visual servoing. In order to assist coupling, the environment is marked with special pre-defined markers known to all the robots. Certain positions with markers can also be reserved to assist coupling.
3. **Navigation:** Once the robots are coupled, the driver initiates navigation towards the goal. During this time, the hitchhiker shuts down all the processes except visual servoing. In other words, the hitchhiker shuts down the SLAM, path planning, dynamic obstacle avoidance, and map update module. It simply follows the driver robot using visual servoing. The driver robot does all the heavy lifting.
4. **Decoupling:** Once the two robots have reached the destination, the decoupling process is initiated. Decoupling is defined as the opposite process of coupling.

During decoupling, the driver gives the position location (i.e. the estimated x, y coordinates in the map) and the uncertainty associated with it (Σ). This information must be given to the hitchhiker as it requires this information to do other tasks or start navigation from the point of decoupling to another location. Moreover, during navigation if the driver robot has encountered new static obstacles in the map and has updated its map, then the location of the new static obstacles are also transferred to the hitchhiker.

The hitchhiker can thus save a lot of computation and energy which would have been spend in path planning, obstacle avoidance, mapping and localization from the hitchhiking point to the goal location. At the same time, the hitchhiker has all the necessary information at the goal location like estimated positional coordinates with certainty and the location of new obstacles in the map.

6.3 Hitchhiking Points

Although hitchhiking can be initiated at any place in the map, it is generally not a good idea to do so because of the following two reasons:

1. **Loss of Efficiency:** Notice that, hitchhiking itself consumes computation and time. By allowing hitchhiking anywhere in the map, the robots may not find fixed markers in the environment to assist coupling. In the absence of such markers, coupling might be very difficult for the robots and they may consume more energy and time. Another problem is that the robots must always be alert of an incoming request all the time. This degrades the performance of the entire system and even defies the whole point of using hitchhiking in the first place.
2. **Problem of Obstacles:** Since it consumes time for the two robots to handshake and couple, hitchhiking at an inappropriate place might be a hindrance in the pathways for other robots and people.

Thus, it is better to have certain fixed ‘hitchhiking points’ in the environment which are laid with artificial markers to assist hitchhiking. It eliminates the aforementioned two problems and gives an estimate to the robots about a possibility of hitchhiking.

6.3.1 When To Hitchhike and When Not

As a rule, hitchhiking should be done whenever computation power can be saved and efficiency gains are possible. The three good candidate scenarios for hitchhiking are

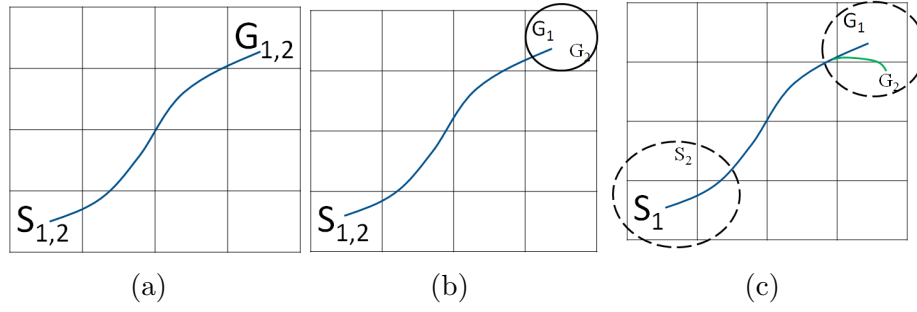


Fig. 6.3 Different hitchhiking scenarios. (a) When the start and goal locations of the two robots are the same. (b) When the goal locations of the two robots are nearby. (c) Hitchhiking in between.

shown in Fig.6.3. Figure 6.3a is the best possible scenario when the start and goal locations of the hitchhiker and driver are exactly the same, and the distance to be travelled is large. Fig.6.3b shows a scenario in which the start location of the two robots is the same but the goal locations are different, however, in proximity. Figure 6.3c shows the scenario of hitchhiking in between when neither the start nor the goal locations are the same. Hitchhiking can be beneficial in this case too, provided the distance to be travelled is large.

However, hitchhiking is not good for short distances. This is because, time and energy are required for handshaking, coupling, and decoupling. Moreover, a hitchhiker always hitchhikes a robot with better or same profile. In other words, ‘*you don’t trust suspicious drivers*’. This is natural as a robot with a robust and accurate SLAM module should never hitchhike another robot with inaccurate sensors and SLAM profile as doing so has potential of task failure for both the robots. On the other hand, a hitchhiker with a lower profile can always benefit from a driver with good profile in terms of accuracy of maps, navigation, obstacle avoidance, and localization.

6.4 Problem of ‘Driver Lost’ Scenario

One potential problem in hitchhiking is that while navigation the driver might be ‘lost’. This is particularly dependent on the robustness of the visual servoing employed. Visual servoing, particularly image based visual servoing are not very robust to large rotations. A detailed explanation of problems in IBVS and PBVS can be found summarized in the work of C. Francois [22]. Any of the problems described in [22] can result in a driver lost scenario in which the follower robot is left behind while the driver robot navigates to its goal.

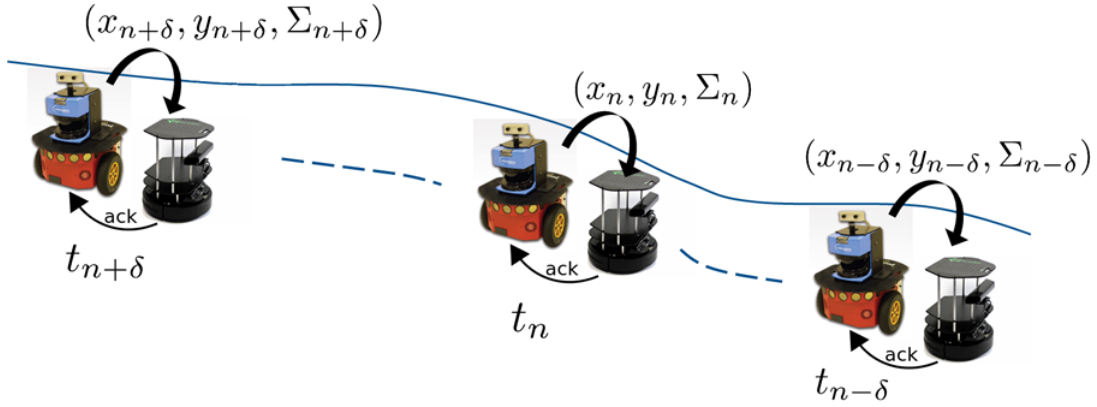


Fig. 6.4 Intermittent information transfer to prevent ‘driver-lost’ scenario.

The hitchhiker has just visual servoing in operation and temporarily disables other modules of SLAM and path planning. Therefore, if the driver is lost it is very difficult for the hitchhiker to navigate to its goal. It is completely unaware of its current position in the map. This problem is similar to the famous ‘*kidnapped robot problem*’ for which many solutions are available in literature [17, 37, 106].

However, a scheme is designed in this work to prevent the kidnapped robot problem in the first place. This is achieved by transferring the estimated position and associated uncertainty information to the hitchhiker intermittently in intervals of δ s. This is graphically shown in Fig.6.4 where a driver is shown transferring information intermittently. With this scheme in implementation, even if the driver robot is lost due to failure of visual servoing, the hitchhiker still has enough information to localize itself in the map and carry its task on its own.

As shown in Fig.6.4, the hitchhiker acknowledges the receipt of the intermittent information (x, y, Σ) . Through this acknowledgement message (ACK), the driver robot can know that the hitchhiker is following well and continue its navigation. In the absence of ACK message, the driver can stop for the hitchhiker to catch-up. The place where it stops might be in the middle of the way with no markers. Hence, if it takes time to couple again and there is a hitchhiking point nearby, the driver can instead broadcast a message to inform the hitchhiker to couple at that point. However, if the driver has a time critical task at hand, it can instead broadcast a ‘leave’ message and move towards its goal. The hitchhiker robot can later localize and continue towards the goal with its own modules.

6.5 Award Mechanism

An award mechanism is designed in which the driver robot is awarded with points for assisting other robots via hitchhiking. The points are directly proportional to the distance travelled.

$$\text{Points} = \kappa \cdot \text{dist}(\text{hitchhike_start}, \text{hitchhike_end}). \quad (6.2)$$

where κ is a constant and the function `dist` calculates the distance between the start and end locations of hitchhiking. Each robot tries to increase the points whenever possible. The overall result is that the efficiency of the entire system is improved. The following constraints check that the task performance is not compromised:

- Hitchhiking is not allowed over short distances.
- Hitchhiking is only allowed when the driver robot has better or same profile.

6.6 Experimental Results

Two experiments in real indoor environment were performed to test the validity of the proposed hitchhiking in multi-robot scenarios. The robots used were Pioneer-P3DX [131] and Kobuki Turtlebot [182] robot shown in Fig.2.9a and Fig.2.9b. Pioneer P3DX was the driver robot, and Turtlebot initiated hitchhiking in both the cases. Both Pioneer-P3DX and Turtlebot are two wheeled differential drive robots and their motion model is given in Section 2.7.

For visual servoing, the open source Visp (Visual Servoing Platform) library [108, 187] was used as it is fast and robust for online tracking. Visp sets the initial pose by extracting the location of the four QR-code corners using a Perspective-n-Point (PnP) algorithm [67] from which a model based tracker is initialized to extract the black area around the QR-code. A hybrid approach for tracking edges and keypoint features is employed. The ROS implementation provides ‘`visp_auto_tracker`’ node to track the pose in program and details can be found in [108, 169].

6.6.1 Experiment 1

Experiment 1 was carried out in the corridor of the engineering lobby shown in Fig.6.5. In order to test the new obstacle update, the boxes were positioned in the environment as new obstacles and were not reflected in the maps which was build by the two robots. Turtlebot waited and initiated the hitchhiking process. The hitchhiking point with a



Fig. 6.5 Experiment 1: Environment with obstacles marked in red.



Fig. 6.6 Experiment 1. (a) Coupling between the two robots. (b) Turtlebot following P3DX through visual servoing.

(plus sign) marker on the pillar is shown in Fig.6.6a. P3DX robot aligned itself with the marker on the pillar and the Turtlebot aligned itself with the QR-code on the back of P3DX for visual servoing. The navigation started and Turtlebot followed the driver P3DX as shown in Fig.6.6b. The hitchhiker Turtlebot had temporarily shut down its modules except servoing and was successfully able to navigate towards its goal. Figure 6.7 shows the results of the map update process of the experiment. Figure 6.7a was the map without the obstacles. Both Turtlebot and P3DX had this map before the commencement of the experiment. Figure 6.7b shows the map build by the driver P3DX. The positions of the new obstacles are clearly shown. The driver successfully transferred the coordinates of the new obstacles to the hitchhiker during the decoupling process.

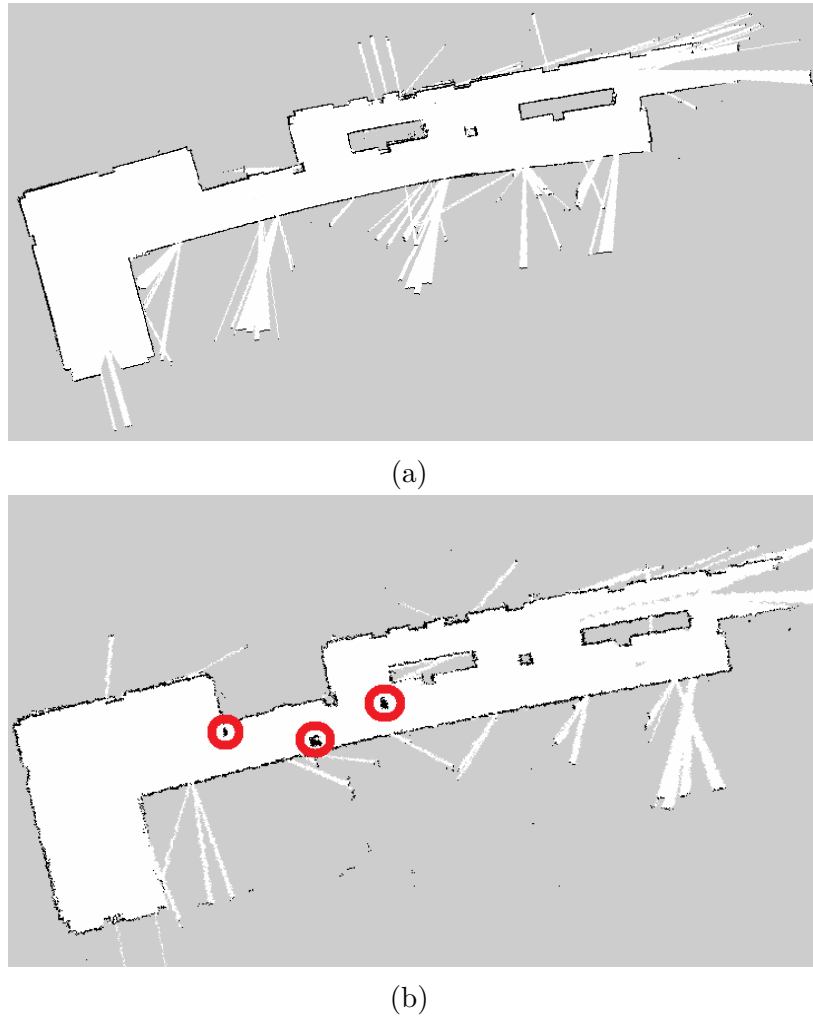


Fig. 6.7 Experiment 1. (a) Initial map without obstacles. (b) Map with obstacles. Driver robot transfers the obstacle coordinates (marked in red) to the hitchhiker robot.

6.6.2 Experiment 2

Experiment 2 was carried in a long corridor shown in Fig.6.8. In order to test the new obstacle update, the boxes were positioned in the environment as new obstacles and were not reflected in the maps which was build by the two robots. Turtlebot waited and initiated the hitchhiking process. Although the hitchhiking point was marked using color stripes on the floor, they were not used and instead the left pillar was used as a marker. The coupling process is shown in Fig.6.9a. The navigation started and Turtlebot followed the driver P3DX as shown in Fig.6.9b. The hitchhiker Turtlebot had temporarily shut down its modules except servoing and was successfully able to navigate towards its goal.



Fig. 6.8 Experiment 2: Environment with obstacles marked in red.



Fig. 6.9 Experiment 2. (a) Coupling between the two robots. (b) Turtlebot following P3DX through visual servoing.

Figure 6.10 shows the results of the map update process of the experiment. Figure 6.10a was the map without the obstacles. Both Turtlebot and P3DX had this map before the commencement of the experiment. Figure 6.10b shows the map build by the driver P3DX. The positions of the new obstacles are clearly shown. The driver successfully transferred the coordinates of the new obstacles to the hitchhiker during the decoupling process.

In experiment, apart from the occupancy grid map based map update, 3D map update was also performed by the driver robot. The initial map of the environment before the commencement of hitchhiking and before placing new obstacles is shown in Fig.6.11a and Fig.6.12a. The final updated with the locations of the new obstacles is shown in Fig.6.11b and Fig.6.12b.

Notice that not only the locations of the new obstacles, but high quality 3D maps [137, 142] can also be transferred to the hitchhiker during decoupling. This is particularly

useful if the hitchhiker's profile is poor and it is not equipped with accurate 3D sensors or powerful computational units. In that case, a driver with good sensors and powerful computer can build high quality maps and give to the hitchhiker robot.

6.7 Summary

The work in this chapter introduced a novel idea of hitchhiking in multi-robot systems. When two or more robots have the same start and goal locations, it is a waste of computation power if all the robots perform their own path planning, obstacle avoidance, dynamic map update, and localization separately. However, this chapter showed that hitchhiking can be more efficient in such scenarios. The process of hitchhiking was systematically explained in four steps. Later, the advantages of having hitchhiking points in the map were discussed. Solutions to the problem of 'driver lost' scenario were provided. The experimental results demonstrated that hitchhiking is feasible in a multi-robot system with potential benefits.

Virtual pheromone is useful for scenarios where the distance between the leader and follower robots is large enough such that the leader is not visible to the follower robot. However, when two robots are in proximity, a much straightforward way is to follow the leader robot via. visual servoing. Hitchhiking is more than a simple leader-follower system. It is a mechanism to save computation power and robust operations. It is particularly useful in multi-robot systems in which some of the robots have inaccurate sensors and less computational capabilities. Such robots can achieve a lot of efficiency gains by hitchhiking.

This chapter demonstrated the feasibility of hitchhiking in multi-robot systems. However, a lot of work still needs to be done. Particularly, the power profile of the different modules needs to be estimated to calculate exactly how much energy is saved. The efficiency gains are obvious for very large distance hitchhiking. However, the threshold distance above which hitchhiking should be permitted needs to be calculated for crowded and non-crowded environments. These are considered as future works.



(a)



(b)

Fig. 6.10 Experiment 2. (a) Initial map without obstacles. (b) Map with obstacles. Driver robot transfers the obstacle (marked in red) coordinates to the hitchhiker robot.

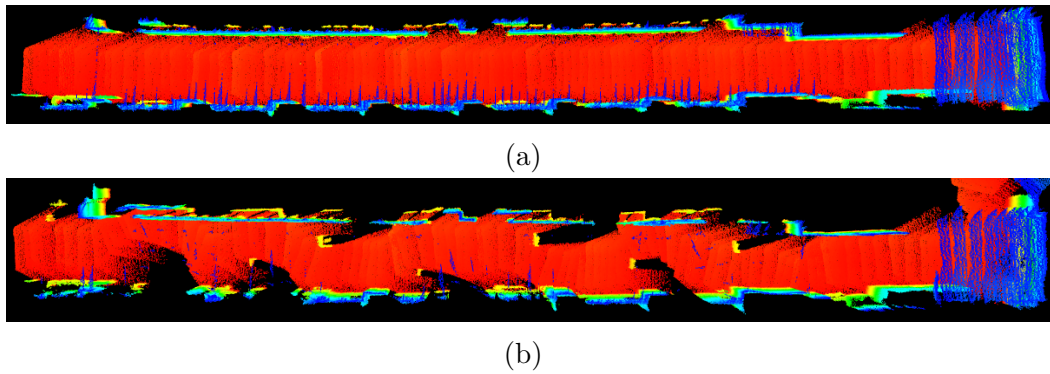


Fig. 6.11 Experiment 2. (a) Initial map without obstacles. (b) Map with obstacles. Driver robot transfers the obstacle coordinates to the hitchhiker robot.

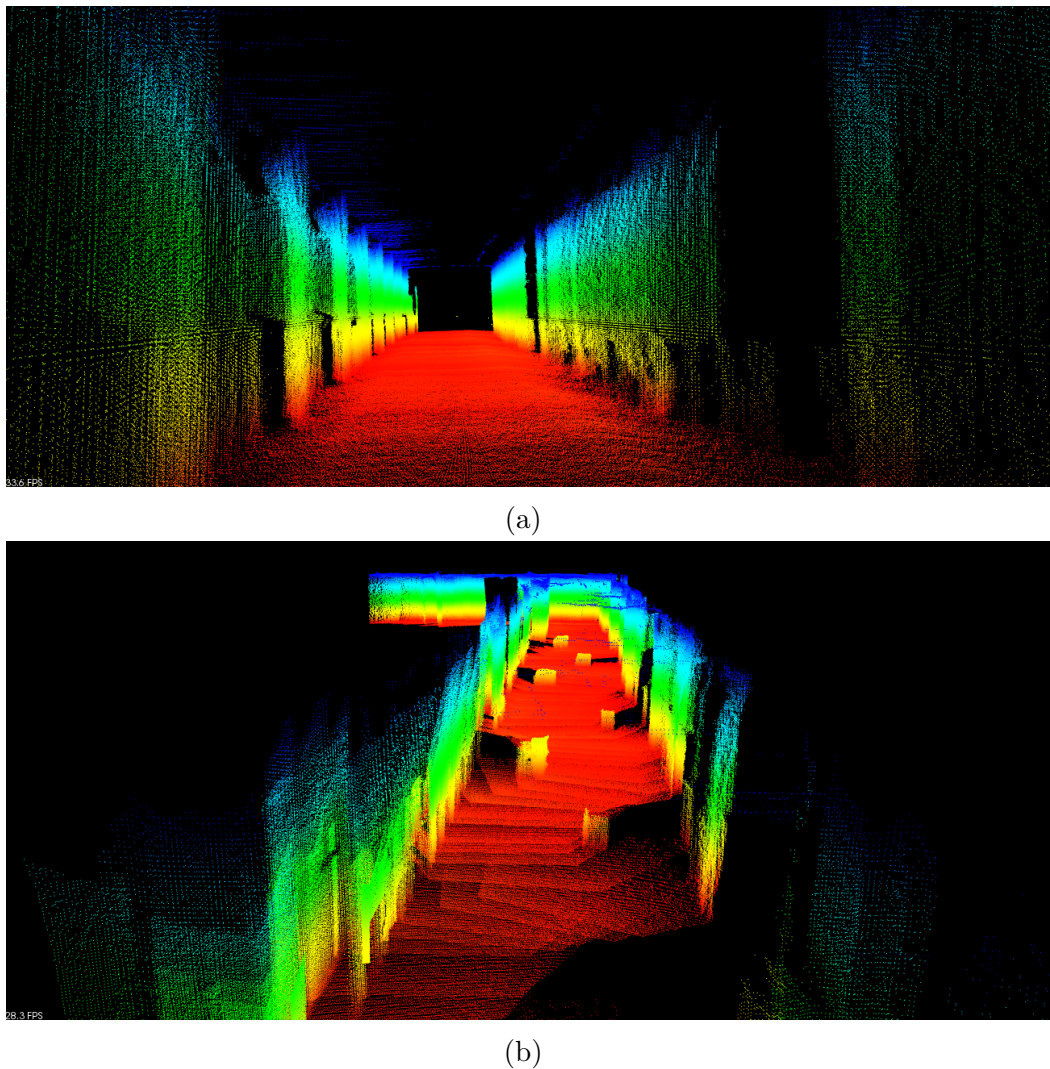


Fig. 6.12 Experiment 2. 3D Map. (a) Initial map without obstacles. (b) Map with obstacles. Driver robot transfers the obstacle coordinates to the hitchhiker robot.

Chapter 7

SLAM Based on a Novel Hopping-points SVD and Hough Transform Based Line Detection Algorithm

Overview

This chapter describes novel contributions in the field of simultaneous localization and mapping (SLAM) which forms the core module of works presented in all the previous chapters as also shown in Fig.1.3. In all the previous chapters, it was assumed that multiple robots had a SLAM component. Many algorithms have been proposed for SLAM and line segment based SLAM is popular as it can be implemented with laser range sensors which requires less computational power for processing than say vision based sensors. This chapter proposes an efficient algorithm for line detection from LRS data using a novel hopping-points Singular Value Decomposition (SVD) and Hough transform based algorithm, in which, SVD is applied on intermittent LRS points to accelerate the algorithm.

Chapter Abstract

Line detection is an important problem in computer vision, graphics, and autonomous robot navigation. Lines detected from laser range sensor (LRS) mounted on a robot can be used as features to build a map of the environment, and later to localize the robot in the map, in a process termed as Simultaneous Localization and Mapping (SLAM). I

propose an efficient algorithm for line detection from LRS data using a novel hopping-points Singular Value Decomposition (SVD) and Hough transform based algorithm, in which, SVD is applied on intermittent LRS points to accelerate the algorithm. A reverse-hop mechanism ensures that the end points of the line segments are accurately extracted. Line segments extracted from the proposed algorithm are used to form a map and subsequently LRS data points are matched with the line segments to localize the robot. The proposed algorithm eliminates the drawbacks of point based matching algorithms like Iterative Closest Points (ICP) algorithm whose performance degrades with increasing number of points. I tested the proposed algorithm for mapping and localization in both simulated and real environment, and found it to accurately detect lines and build map with good self-localization.

7.1 Introduction

Simultaneous Localization and Mapping (SLAM) is an indispensable component of any autonomous robot which deals with making a map of the surrounding and localizing itself in the map at the same time. The robot perceives the exterior environment from sensors such as laser range sensor (LRS), RGB-camera, stereo camera, inertial sensors, GPS, etc. The robot has wheel encoders which gives an estimates about how much the robot has moved in the environment. The sensor data and encoder data are both used to build a map and estimate the position of the robot in the map. It is a challenging problem mainly because the sensors and encoders are prone to errors, which is further complicated by the dynamics of the environment. These errors keep on accumulating with time as the robot moves in the environment. Hence, a robot must incrementally build the map and keep on correcting it and its position at each step. Various approaches to solve this problem have been proposed and the earliest work can be tracked in the works of Smith et. al.[165]. Probabilistic approaches including Extended Kalman Filter (EKF) SLAM have been proposed in [180] which uses particle filters. Similar approaches using different sensors like camera [34], and depth sensors [44] have also been proposed.

In most of the approaches used to solve SLAM, a robot needs to match its current data with the previously saved map data to estimate its position and then apply correction. Matching can either be point based, or feature based. In the former, feature points obtained from sensors such as LRS are directly matched such as in [28–30]. One of the most common point matching algorithm is ICP (Iterative Closest Point)[13]. ICP is a classical rigid point set registration algorithm and many variants of ICP have been

proposed in [19, 103, 114, 128]. When given two point clouds, ICP algorithm tries to iteratively minimize the least square difference between them to determine rotation and translation. A major drawback of ICP is that it is computationally complex and the performance deteriorates for large number of points. ICP has a computational complexity of $\mathcal{O}(m^d n^d)$ [47], where m and n are the number of points in the two point sets in a d dimensional space. Faster variants of ICP which employ KD-tree [12] such as [61] have also been proposed and accelerating ICP is an active area of research.

On the other hand, feature based matching utilizes map features like line segments, corners, color histogram information, scale invariant features (SIFT) [101] and others to match the sensor data. Compared to point based matching, feature based mapping generally require less memory and are relatively computationally efficient, and have extensively been utilized in SLAM. For distance sensors such as LRS, line segment based SLAM has been studied extensively employing line extraction algorithms such as Split and Merge [16] and incremental [1] algorithms. Among these, the most prominent algorithms for line extraction includes Hough transform [33, 42, 52] and RANSAC [50]. Other techniques includes Expectation-Maximization algorithm [129, 155], line regression [9], and clustering based line map generation [140, 151, 152]. Mapping based on line segments have also been reported in [59, 129, 185]. Line segment extraction and its integration in particle filter slam has been proposed in [4, 5]. A good summary of various line extraction techniques can be found in [121].

This chapter proposes a novel line detection algorithm using SVD and Hough transform with good computational efficiency. In order to speed up the line extraction process, I propose a hopping mechanism in which SVD is applied on intermittent points. A reverse hop mechanism ensures that the end points of each line segment are accurately extracted. Moreover, the line extraction algorithm is integrated in a SLAM framework and the map comprising of the extracted lines are used to match the current LRS data for efficient localization.

This chapter is structured as follows: Section 7.2 explains the problem of straight line detection from LRS data using Hough transform. Section 7.3 explains incorporating SVD in the equations of Hough transform to accurately detect lines. The forward and reverse hopping scheme to accelerate the algorithm are explained in Section 7.3.3. The proposed line extraction algorithm is integrated in SLAM and explained in Section 7.4. Mapping is explained in Section 7.4.1 and localization in Section 7.4.2. Results are discussed in Section 7.5. Section 7.5.1 discuss simulation results, and Section 7.5.2

shows the results of the experiments in real environment. Finally, Section 8.2 concludes the chapter.

7.2 Straight Line Detection from LRS data with Hough Transform

A typical indoor environment of a building is shown in Fig.7.1. It consists of straight passages which are diverting and opening in left and right directions. A robot with LRS sensor mounted on it moves through the passage and the sensor records the points on the edges of the walls of the environment which have been indicated by heavy black lines. How far an LRS can detect these points depends on the specifications of the sensor. In general, every LRS sensor is characterized by a maximum detect distance $d_{\max}$, the minimum detect distance $d_{\min}$, the maximum scan angle $\theta_{\max}$, and the minimum scan angle $\theta_{\min}$. The center of the scan angle is, therefore, $\theta_{\text{center}} = (\theta_{\min} + \theta_{\max})/2$. I set this center on the coordinate system $O-XY$ with absolute rotating angle ϕ_k . This is shown in Fig.7.2 which represents the k th scan step. In Fig.7.2, $o-xy$ is a local coordinate system fixed on the LRS and the y -axis agrees with θ_{center} .

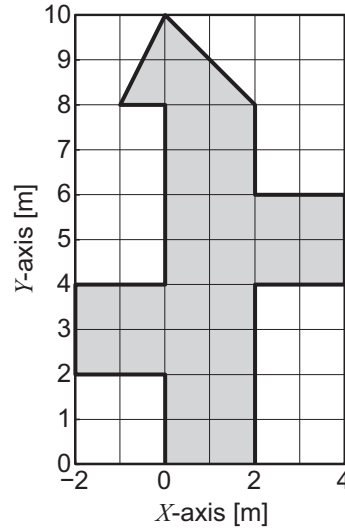


Fig. 7.1 Environment for simulation

As shown in Fig.7.2, LRS outputs a point $\mathbf{p}_k(j)$ at the j th scan at an angle $\theta_k(j)$ ($1 \leq j \leq N$) measured from x -axis. Here, the number N is the resolution with respect to the angle of scanning. Figure 7.3 shows LRS output at position $X_k = 1$ m, $Y_k = 1$ m, and angle $\phi_k = 20$ deg in the map shown in Fig.7.1.

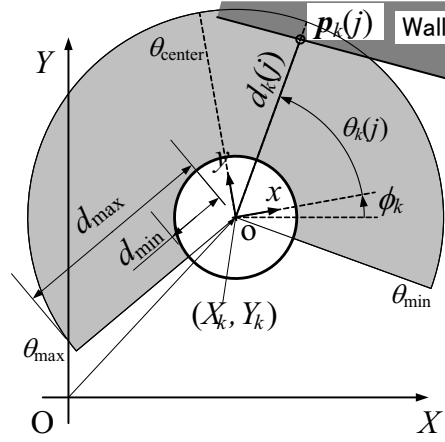


Fig. 7.2 Local coordinate for laser range sensor

Position of the detected wall is indicated by heavy line in Fig.7.2 and is expressed as,

$$\mathbf{p}_k(j) = \begin{bmatrix} p_{xk}(j) \\ p_{yk}(j) \end{bmatrix} = \begin{bmatrix} d_k(j) \cos(\theta_k(j)) \\ d_k(j) \sin(\theta_k(j)) \end{bmatrix} \quad (7.1)$$

on the coordinate system $o-xy$.

The vector $\mathbf{p}_k(j)$ ($1 \leq j \leq N$) represents the N number of points on the coordinate system $o-xy$. Hough transform [70] can be applied to detect a straight line from the measured N points.

The Hough transform of the j th point $(p_{xk}(j), p_{yk}(j))$ is expressed as,

$$\rho(\theta) = p_{xk}(j) \cos(\theta) + p_{yk}(j) \sin(\theta). \quad (7.2)$$

Hough transform changes the points from Cartesian (x, y) space to (ρ, θ) space. It can be seen that the value of ρ is a function of θ and hence Hough transform on a set of points which lie on the same line intersect at a common point in the (ρ, θ) accumulator space. Hough transform employs a voting scheme and the point (ρ, θ) with the maximum number of votes is the point which corresponds to the straight line passing thorough the points.

If the sensor is accurate without any errors, and there is no noise, all the points on the line will intersect at exactly the same point (with maximum votes) in the (ρ, θ)

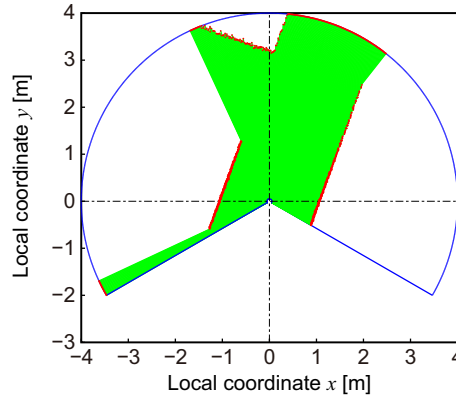


Fig. 7.3 Example of laser range sensor data

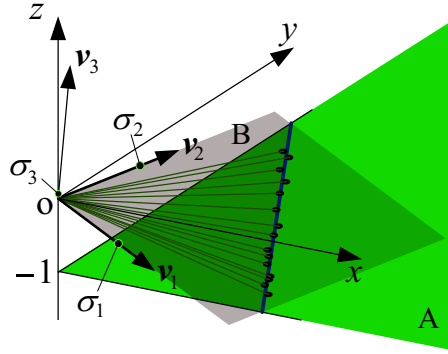


Fig. 7.4 Detection of a line from points with SVD

accumulator space. However, non-linear points, or linear points in presence of sensor and environmental noise do not intersect at a single (ρ, θ) point, and multiple points with the same maximum votes may be generated. In order to extract the lines, we have to estimate (ρ, θ) points with maximum votes. Hence, when N measured points makes multiple lines, a two dimensional search is required in θ and ρ plane which is computationally expensive.

7.3 SVD Based Line Detection Algorithm

This section describes the proposed algorithm for detecting straight line segments using SVD and Hough transform.

7.3.1 Straight Line Detection with SVD

We can re-write the equation of Hough transform (7.2) as,

$$p_{xk}(j) \cos(\theta) + p_{yk}(j) \sin(\theta) - \rho(\theta) = 0. \quad (7.3)$$

This equation can be further expressed as,

$$\begin{bmatrix} p_{xk}(j) & p_{yk}(j) & -1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) & \rho(\theta) \end{bmatrix}^T = 0. \quad (7.4)$$

A regression line which starts from the i th point and comprises of a total of l points is determined by the combination of ρ and θ , and satisfies the equation,

$$\mathbf{A}_i(l) \mathbf{X}(\rho, \theta) = \mathbf{0} \quad (1 \leq i \leq N-2, 3 \leq l \leq N-i+1), \quad (7.5)$$

where,

$$\begin{aligned} \mathbf{A}_i(l) &= \begin{bmatrix} p_{xk}(i) & p_{yk}(i) & -1 \\ \vdots & \vdots & \vdots \\ p_{xk}(i+l-1) & p_{yk}(i+l-1) & -1 \end{bmatrix}, \\ \mathbf{X}(\rho, \theta) &= \begin{bmatrix} \cos(\theta) & \sin(\theta) & \rho(\theta) \end{bmatrix}^T \neq \mathbf{0} \end{aligned} \quad (7.6)$$

Singular Value Decomposition (SVD) is employed to solve this equation. SVD of the matrix $\mathbf{A}_i(l)$ is expressed as

$$\mathbf{A}_i(l) = \mathbf{U} \mathbf{S} \mathbf{V}^T, \quad (7.7)$$

where $\mathbf{U}$ and $\mathbf{V}^T$ are the orthonormal matrices consisting of left and right singular vectors. $\mathbf{S}$ is the diagonal matrix with singular values arranged in decreasing order along the diagonal as follows:

$$\begin{aligned} \mathbf{U} &= [\mathbf{u}_1 \cdots \mathbf{u}_l], \mathbf{S} = \begin{bmatrix} \text{diag}(\sigma_1 \ \sigma_2 \ \sigma_3) \\ \mathbf{0} \end{bmatrix} (\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0), \\ \mathbf{V} &= [\mathbf{v}_1 \ \mathbf{v}_2 \ \mathbf{v}_3]. \end{aligned} \quad (7.8)$$

Here, the matrices $\mathbf{U}$ and $\mathbf{V}$ satisfy $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ and $\mathbf{V}^T \mathbf{V} = \mathbf{I}$. The matrix $\mathbf{I}$ represents identity matrix. When minimum singular value σ_3 is sufficiently-small and has near-zero

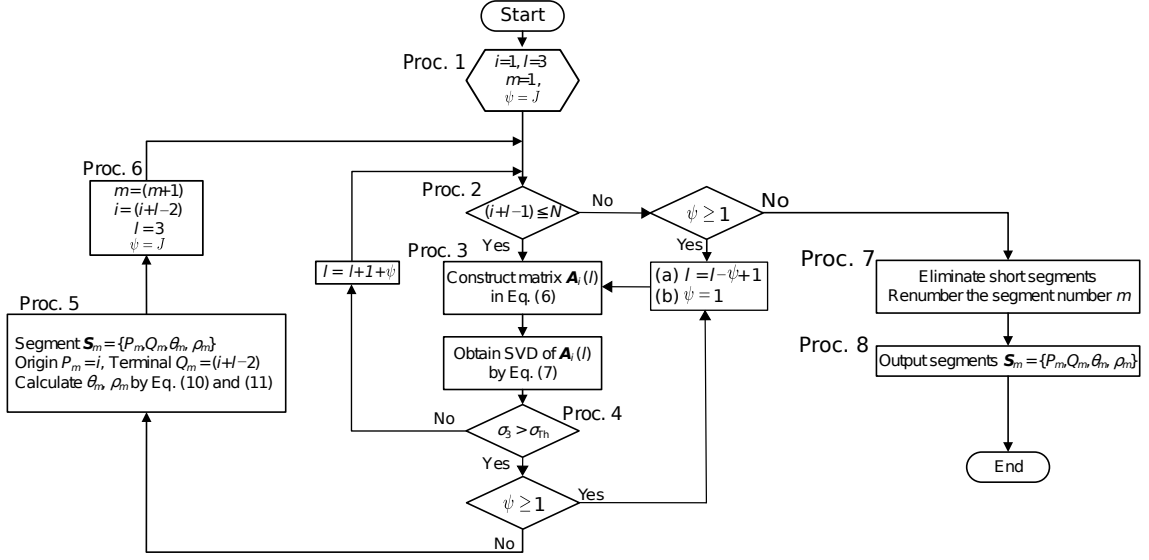


Fig. 7.5 Flow chart of line-segment detection

value, the least square solution of Eq.(7.5) is given by the right singular vector $\mathbf{v}_3$ corresponding to σ_3 .

When sufficiently small singular value cannot be obtained, it means that it is difficult to determine a regression line of l points, starting from the i th point. It also implies that the coefficient of correlation of these points gives a near-zero value.

Figure 7.4 gives a graphical intuition of the proposed method. Points indicated by $\bullet$ are the points measured by LRS. The detected straight line is shown by a bold line. The z -axis is normal to the x - y plane. We can interpret Eq.(7.3) as expressing a plane of $x = p_{xk}(j)$, $y = p_{yk}(j)$, $z = -1$ on the coordinate system $o-xyz$. In the proposed method, placing the measured points $(p_{xk}(j), p_{yk}(j))$ on the plane $z = -1$ by Eq.(7.4), singular vectors $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ are derived by SVD.

A straight line is detected as a line of intersection of the plane which consists of the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, and the plane $z = -1$. The minimum singular value σ_3 indicates the variation in measured points along the direction of the singular vector $\mathbf{v}_3$.

A threshold σ_{Th} is defined for σ_3 . When σ_3 is smaller than σ_{Th} , the solution $\mathbf{X}(\rho, \theta)$ of Eq.(7.5) is expressed as,

$$\mathbf{X}(\rho, \theta) = \alpha \mathbf{v}_3. \quad (7.9)$$

where, α is an undetermined factor. Expressing the singular vector $\mathbf{v}_3$ as,

$$\mathbf{v}_3 = [v_{31} \ v_{32} \ v_{33}]^T, \quad (7.10)$$

we get,

$$\cos(\theta) = \alpha v_{31}, \quad \sin(\theta) = \alpha v_{32}, \quad \rho = \alpha v_{33} \quad (7.11)$$

from Eq.(7.6) and α is determined as,

$$\alpha = \frac{1}{\sqrt{v_{31}^2 + v_{32}^2}} \quad (7.12)$$

from the relationship $\cos^2(\theta) + \sin^2(\theta) = 1$.

7.3.2 Line Segment Detection Algorithm

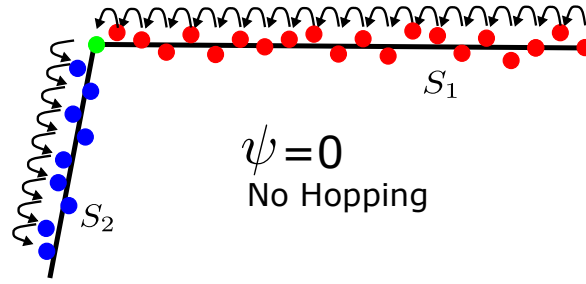
In this subsection I describe the entire process of line segment detection. Flow-chart of the algorithm is shown in Fig.7.5. The parameter ψ in Fig.7.5 deals with ‘hopping’ feature of the algorithm and is explained in detail in Section 7.3.3. First, the algorithm is explained without the hopping feature in which value of ψ is set to 0 (i.e. $J = 0$). Line detection process comprises of the following steps:

Proc.1 In order to detect the m th line segment which consists of total l points and starts from the i th point, I set the initial parameters as, $i = 1$, $l = 3$ and $m = 1$. This means that the process will detect the first ($m = 1$) line segment which starts from $i = 1$ and has just 3 points initially (number of points will be incremented later).

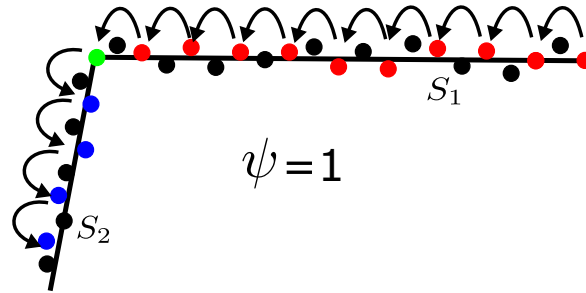
Proc.2 Since the maximum number of points in an LRS scan is N , we check if $(i + l - 1)$ is greater than N . If it is greater than N , it means that the algorithm has already been applied to the all measured point, and line segments have been extracted. So, we finish the line segment detection and jump to **Proc.7**.

Proc.3 A matrix $\mathbf{A}_i(l)$ is constructed from Eq.(7.6), and apply SVD given by Eq.(7.7) to obtain the minimum singular value σ_3 .

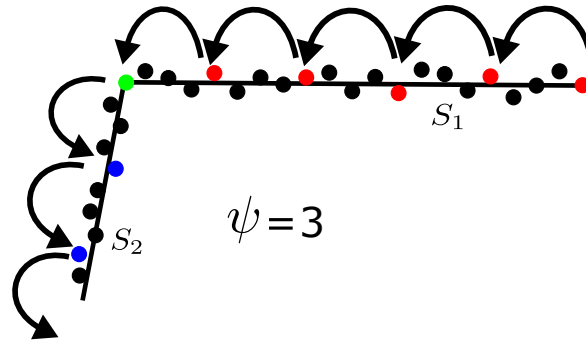
Proc.4 If the minimum singular value σ_3 is less than σ_{Th} , the points from the i th point to the $(i + l - 1)$ th point are estimated to form a straight line. The next $(i + l + \psi)$ th point ($\psi = 0$, for non-hopping case) might also exist on the same



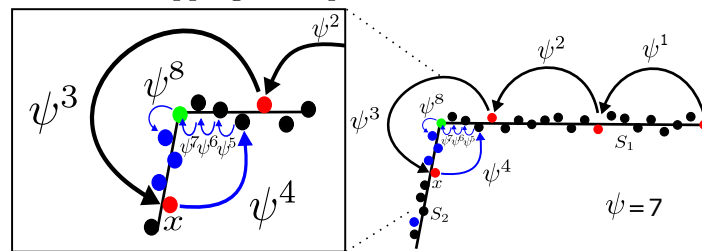
(a) No hopping with $\psi = 0$. SVD is applied on each and every point.



(b) Line detection with $\psi = 1$. SVD is applied on every intermittent point.



(c) Line detection with $\psi = 3$. SVD is applied after skipping three points.



(d) Hopping with $\psi = 7$ and reverse hopping. Reverse hop occurs at point x and the end point show in green is accurately detected.

Fig. 7.6 Explanation of hopping point SVD based line segment detection.

straight line, and hence $l = (l + 1)$ is set and the algorithm jump back to **Proc.2**. If a point is reached where σ_3 is greater than σ_{Th} , the next **Proc.5** is executed.

Proc.5 A value of σ_3 greater than σ_{Th} means that the $(i + l - 1)$ th point cannot be estimated on the straight line constructed by points from the i th point to the $(i + l - 2)$ th point. A new line segment starts from this point. Hence, the m th line segment is first extracted which contains the points from the i th point to the $(i + l - 2)$ th point. The origin is set to the point $P_m = i$, and the terminal point $Q_m = (i + l - 2)$. The slope θ_m and the distance ρ_m from the origin point of the line segment are determined by Eqs.(7.11) and (7.12). Hence, the m th line segment S_m is detected and stored as four parameters $\{P_m, Q_m, \theta_m, \rho_m\}$.

Proc.6 To start detecting the next line segment, segment number m is incremented by one to $m + 1$. The end point of the previous line segment (S_m) marks the beginning of the $(m + 1)$ th line segment, and hence, it is set to $(i + l - 2)$. The total number of points l of the new $(m + 1)$ th line segment is initialized to 3, and the line extraction process is repeated from **Proc.2** to extract the next line segment.

Proc.7 Short line segments are eliminated and line segments S_m are renumbered. Concretely, the total number of points of the detected line segment S_m is calculated as $(P_m - Q_m + 1)$ points. Only longer line segments S_m which contain more than L_{Th} points are kept. Here, L_{Th} is a pre-determined threshold which controls the length of the lines to be detected. Since some shorter line segments gets deleted, the line segment number m is renumbered.

Proc.8 All the extracted line segments S_m are outputted and the algorithm stops.

The accuracy of the line segments are determined by the threshold σ_{Th} and L_{Th} for the minimum singular value σ_3 and the length of the line segment. Notice that, if σ_{Th} is set to an extremely small value, many short line segments will be extracted as LRS point data contains measurement error and noise, which is not desired.

7.3.3 Accelerating Line Detection Algorithm using Hopping-points SVD

In the proposed line detection method, SVD is applied to each and every point, and the lower singular value (σ_3) is checked against a threshold value (σ_{Th}) for each point.

SVD is computationally expensive and the computational complexity of an $m \times n$ matrix is $\mathcal{O}(mn^2)$. However, we need not apply SVD to each and every point. Instead, SVD is applied on intermittent points governed by a hopping factor ψ . The basic idea is that if a set of points lie on the same regression line, it can be assumed that the consecutive points too will lie on the same line and are skipped. However, the intermittent points are still included and SVD is applied to check if the hypothesis still holds and whether the points can still be skipped. Even after skipping ψ points, $\sigma_3 < \sigma_{Th}$ indicates that the hypothesis holds and the skipped points also lie on the same regression line. Concretely, skipping the i th point means that SVD is not applied on matrix $\mathbf{A}_i(l)$ in Eq.7.6 for $i > 1$ points. However, in the next hop at the j th point ($j > i > 1$) the matrix $\mathbf{A}_i(l)$ in Eq.7.6 includes all the points from 1 to j . Parameter ψ determines the number of points which are skipped before applying SVD.

This process is graphically explained in Fig.7.6. The normal SVD on every point is shown in Fig.7.6a. Figure 7.6b shows SVD applied on intermittent points with $\psi = 1$, i.e one point (shown in black) is skipped. Figure 7.6c shows the hopping mechanism with $\psi = 3$ and three points are skipped. In both Fig.7.6b and Fig.7.6c, notice that constructed matrix $\mathbf{A}_i(l)$ from Eq.(7.6) contains information even of skipped points also, and SVD is applied by Eq.(7.7) to obtain the minimum singular value σ_3 . Skipping points accelerates the process. Similar approach has been used in other works [139], however, to realize other objectives like noise removal.

A ‘reverse-hop’ mechanism ensures that end points of line segments are accurately extracted. In both Fig.7.6b and Fig.7.6c, the hop accurately terminates at the end-points of the line segments S_1 and S_2 (indicated in green color). However, this is not true in case of Fig.7.6d with a larger value of $\psi = 7$. In Fig.7.6d, ψ^i represents the i th hop. It can be observed that in the 3rd hop, i.e. ψ^3 , the actual end point of the segment S_1 is missed, as a larger value of σ_3 will be detected at point x as shown in Fig.7.6d. To correct this, whenever σ_3 is detected to be greater than σ_{Th} , a check is performed if the value of hopping parameter (ψ) is greater than or equal to 1. In that case, a ‘reverse-hop’ is performed and the algorithm goes back to the next point of the previous hop’s initial point and ψ is set to 0. A reverse-hop is shown in Fig.7.6d by the ψ^4 , which takes it to the next point of the previous hop position. The value of ψ is set to 0, and hence, no hopping is performed in ψ^5, ψ^6 and, ψ^7 . The condition $\sigma_3 > \sigma_{Th}$ is detected in ψ^8 , and the end point of S_1 is correctly detected. After successfully extracting line segment S_m , ψ is set back to $\psi = J$ in Proc.6 i.e. hopping is reset for the next segment.

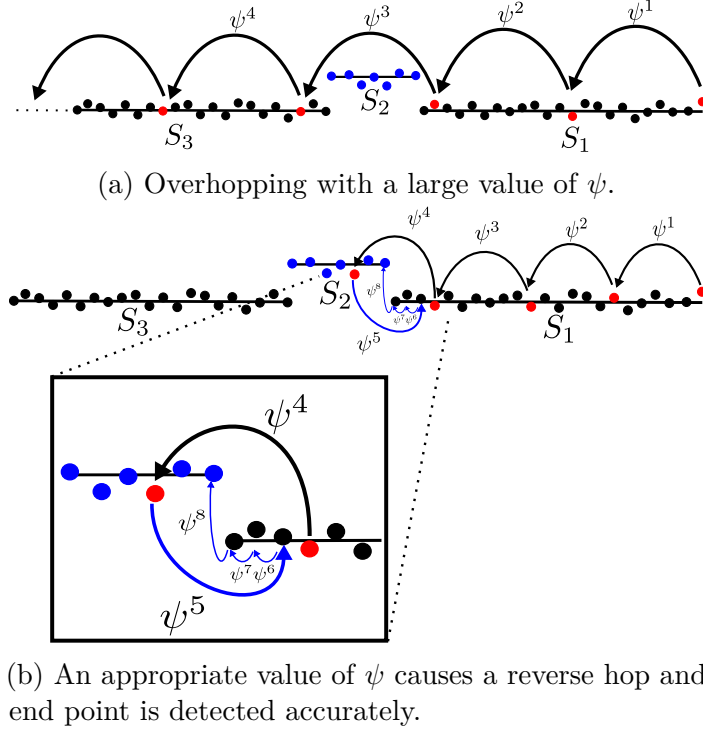


Fig. 7.7 Hopping with large and appropriate hop values.

If the time taken to compute SVD over n points of a line segment is given by a function $t(n)$, the algorithm gets speeded-up by a factor of,

$$\frac{\sum_{j=1}^{n-1} t(j)}{\sum_{j=1}^{h_{\text{major}}} t(1 + (j-1)(\psi+1)) + \sum_{j=h_{\text{major}}+1}^{h_{\text{total}}} t(j)}, \quad (7.13)$$

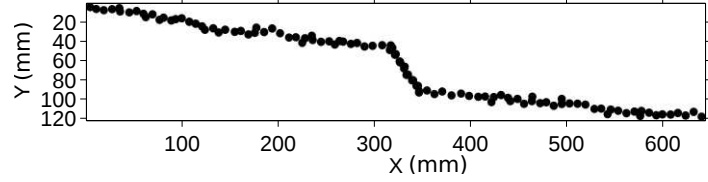
where, h_{major} is the total number of long hops, and h_{total} is the total number of hops (long hops when $\psi \geq 1$ and short hops when $\psi = 0$), and is given by,

$$h_{\text{major}} = \left\lceil \frac{n-1}{\psi+1} \right\rceil, \quad (7.14)$$

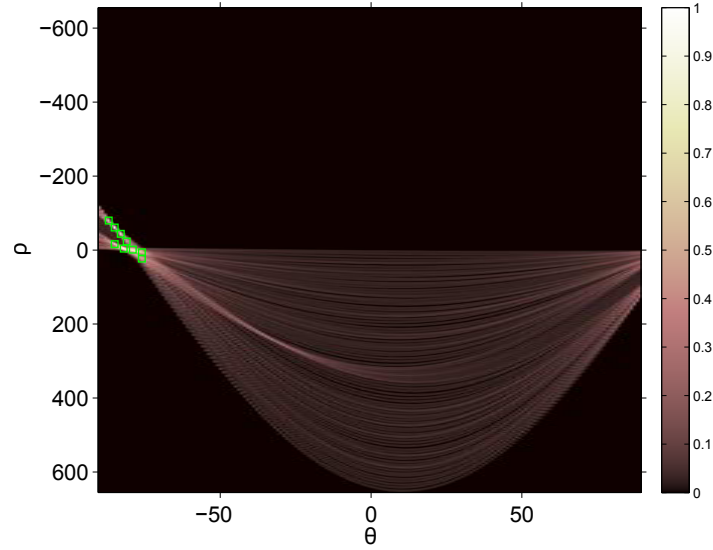
$$h_{\text{total}} = \left\lceil \frac{n-1}{\psi+1} \right\rceil + \underbrace{(n-1) \bmod (\psi+1)}_{\text{minor singular hops without skipping}}.$$

Notice that when $\psi = 0$, the algorithm is the same when no hopping is applied. Setting ψ to a large value might seem advantageous but it may lead to ‘overhopping’ and some shorter line segment detection might be skipped. This may lead to a drop in

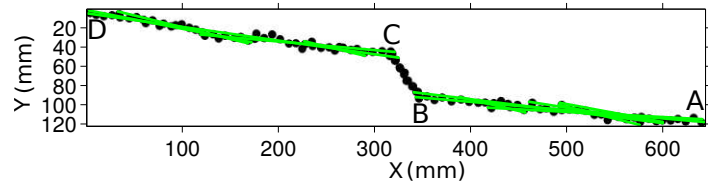
the accuracy of the line detector as end points of line segments might not be detected correctly. As shown in Fig.7.7a, segment S_2 might be omitted in the hop ψ^3 . Setting ψ to a value less than $L_{Th}/2$ prevents skipping shorter segments, and the reverse-hop mechanism ensures accurate end point detection as shown in Fig.7.7b.



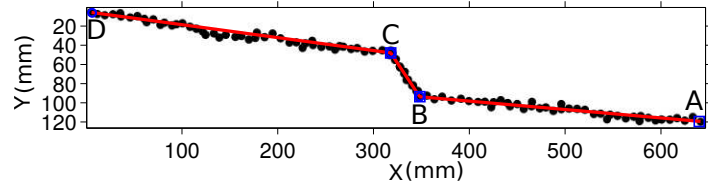
(a) A typical LRS scan data with three line segments.



(b) Hough space of the data with multiple points with maximum votes shown in green boxes.



(c) Multiple lines extracted by Hough transform.



(d) Line segment detected by the proposed method. Start point and end point of each segment are shown by blue boxes and circles, respectively.

Fig. 7.8 Proposed line segment detection with Hough transform.

7.3.4 Comparison with Hough Transform

Figure 7.8a shows a typical LRS data in which points are not necessarily linear due to sensor and environment noise. Figure 7.8b shows the corresponding Hough (ρ, θ) space which is obtained after applying Hough transform [33, 42, 52] on the LRS data. Since the points are non-collinear, they generate multiple peaks in the (ρ, θ) space with maximum number of votes. A computationally expensive two dimensional search is required to detect the maximum votes in Hough space. Multiple lines are extracted corresponding to these peaks in Hough space as shown in Fig.7.8c. Applying inverse Hough transform on these (ρ, θ) pairs results in generation of multiple lines within segments AB and CD, most of which are overlapping. Shorter segment BC is not extracted as it has fewer votes in (ρ, θ) space. In order to detect segment BC, maximum vote value needs to be lowered. However, that results in even more number of shorter segments to be extracted within segments AB and CD. This has adverse effects on robot's localization as there are multiple lines corresponding to the same feature. In contrast, the proposed method detects the segments accurately as shown in Fig.7.8d. In the proposed method parameter σ_{Th} governs the permissible deviation of points along a segment and L_{Th} controls the length of the shortest segment to be detected. Shorter segment BC is also accurately detected in Fig.7.8d using the proposed method. Regarding the execution time, Hough transform took 157.42 ms to detect the segments in Fig.7.8c. On the other hand, by using the proposed method with hopping ($\psi = 5$), accurate line segment extraction could be done in 80.6 ms. Both the computations were performed on MATLAB 7.12 (R2011a) running on a Linux machine with Intel Core i7-4500U CPU, 1.80 GHz, and 16GB RAM.

Like Hough transform, the proposed method is sensitive to outliers. RANSAC [50] gives better performance than the proposed method in case of data with many outliers. However, RANSAC requires a large number of iterations, and requires to set many parameters like minimum number of points, threshold distance between points and fitting line, and inliers ratio, which are often problem specific thresholds [50, 194]. In presence of too many outliers, the proposed method can easily be combined with noise removal techniques [46, 152] to detect line segments from LRS data.

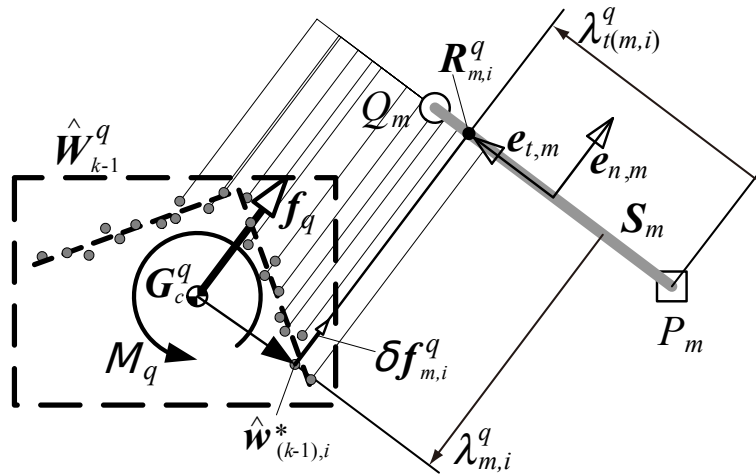


Fig. 7.9 Matching algorithm

7.4 Integration of Line Extraction Algorithm in SLAM Framework

This section explains the integration of the proposed line extraction algorithm in the framework of simultaneous localization and mapping. As shown in Fig.7.2, at the k th scan step, LRS is at the absolute attitude angle of ϕ_k , and position (X_k, Y_k) . Initially, when $k = 1$, the position is given by X_1 , Y_1 and ϕ_1 . At the k th scan step, the true values of the absolute attitude angle ϕ_k and position (X_k, Y_k) are assumed to be unavailable. Instead, the estimated values of absolute attitude angle $\hat{\phi}_k$ and position $\hat{X}_k, \hat{Y}_k$, are calculated. When the robot starts its operation, the position at the 1st scan step is assumed as the anchor point and both the true and estimated values are set to be equal, i.e. $\hat{X}_1 = X_1$, $\hat{Y}_1 = Y_1$, and $\hat{\phi}_1 = \phi_1$.

7.4.1 Mapping from LRS Information

The output data of the LRS at the k th scan step is given by $\mathbf{p}_k(j)$ ($1 \leq j \leq N$) of Eq.(7.1).

Let $\hat{\mathbf{V}}_k$ represent the map of the surrounding environment. It consists of LRS data given by,

$$\hat{\mathbf{V}}_k = \begin{bmatrix} \hat{\mathbf{p}}_k(1) & \cdots & \hat{\mathbf{p}}_k(j) & \cdots & \hat{\mathbf{p}}_k(N) \end{bmatrix}, \quad (7.15)$$

where $\hat{\mathbf{p}}_k(j)$ is the surrounding environment data obtained from the estimated position by applying coordinate transform from coordinate system $o-xy$ fixed on the LRS to the coordinate system $O-XY$ for the environment data $\mathbf{p}_k(j)$. In order to carry out

this transformation, the rotational transform matrix $\mathbf{T}(\hat{\phi})$ and the parallel translation vector $\hat{\mathbf{X}}_k$ are given as,

$$\mathbf{T}(\hat{\phi}) = \begin{bmatrix} \cos(\hat{\phi}) & -\sin(\hat{\phi}) \\ \sin(\hat{\phi}) & \cos(\hat{\phi}) \end{bmatrix}, \quad \hat{\mathbf{X}}_k = \begin{bmatrix} \hat{X}_k \\ \hat{Y}_k \end{bmatrix}. \quad (7.16)$$

Hence, we obtain $\hat{\mathbf{p}}_k(j)$ as,

$$\begin{aligned} \hat{\mathbf{p}}_k(j) &= \begin{pmatrix} \hat{p}_{Xk}(j) \\ \hat{p}_{Yk}(j) \end{pmatrix} = \mathbf{T}(\hat{\phi})\mathbf{p}_k(j) + \hat{\mathbf{X}}_k \\ &= \begin{bmatrix} d_k(j) \cos(\hat{\phi} + \theta_k(j)) + \hat{X}_k \\ d_k(j) \sin(\hat{\phi} + \theta_k(j)) + \hat{Y}_k \end{bmatrix}. \end{aligned} \quad (7.17)$$

We can represent the estimated global map ($\hat{\mathbf{W}}_k$) made by accumulating the LRS data at k scan steps as,

$$\hat{\mathbf{W}}_k = [\hat{\mathbf{W}}_{k-1} \quad \hat{\mathbf{V}}_k] = [\hat{\mathbf{w}}_{k,1} \quad \hat{\mathbf{w}}_{k,2} \quad \cdots \quad \hat{\mathbf{w}}_{k,N_{\text{map}}}], \quad (7.18)$$

where $\hat{\mathbf{w}}_{k,i}$ is the i th column vector of matrix $\hat{\mathbf{W}}_k$, and N_{map} represents total count of points stored in $\hat{\mathbf{W}}_k$.

At the initial condition, estimated global map is given by $\hat{\mathbf{W}}_1 = \hat{\mathbf{V}}_1$. Also, at the very first scan, the true and estimated map is the same, i.e. $\hat{\mathbf{V}}_1 = \mathbf{V}_1$.

The robot updates the estimated global map at each scan. In order to do so, estimated values of absolute position $\hat{X}_k$, $\hat{Y}_k$ and absolute attitude angle $\hat{\phi}_k$ are necessary. The next subsection describes how to calculate $\hat{X}_k$, $\hat{Y}_k$, and $\hat{\phi}_k$.

7.4.2 Self-Localization by Matching LRS Data with Estimated Global Map

For very short distances, the estimated value of the absolute position and absolute attitude angle of the LRS can be obtained from wheel encoders. However, for long distances, the odometry errors keep on integrating and we cannot rely only on encoder data. Therefore, correction of the estimated values by using landmarks is necessary. Outline of correction of the estimation value of self-localization method is shown in Fig.7.9.

After executing one LRS scan at the k th step, I apply the algorithm mentioned in subsection 7.3.2 to the N points of data $\mathbf{p}_k(j)$ ($1 \leq j \leq N$) of Eq.(7.1). A total of N_{Seg}

line segments are extracted, out of which the shorter segments are removed and $\mathbf{S}_m$ line segments are stored, where $1 \leq m \leq N_{\text{Seg}}$.

$\hat{X}_k$, $\hat{Y}_k$ and $\hat{\phi}_k$ of the LRS at the k th scan step are estimated by matching the line segments $\mathbf{S}_m (1 \leq m \leq N_{\text{Seg}})$ with the global map $\hat{\mathbf{W}}_{k-1}$ at the $(k-1)$ steps given by Eq.(7.18) as,

$$\begin{cases} \hat{\mathbf{X}}_k &= \hat{\mathbf{X}}_{k-1} + \mathbf{T}(\hat{\phi}_{k-1})\Delta\mathbf{X}(k|k-1) \\ \hat{\phi}_k &= \hat{\phi}_{k-1} + \Delta\phi(k|k-1) \end{cases}, \quad (7.19)$$

by using the transform matrix $\mathbf{T}(\hat{\phi}_{k-1})$ and estimated position vector $\hat{\mathbf{X}}_{k-1}$ of Eq.(7.16). Here, $\Delta\mathbf{X}(k|k-1)$ and $\Delta\phi(k|k-1)$ represent the displacements of the position and the angle of the LRS from the $(k-1)$ th to the k th scan step.

In order to estimate $\Delta\mathbf{X}(k|k-1)$ and $\Delta\phi(k|k-1)$, first, the map $\hat{\mathbf{w}}_{(k-1),i}^*$ on the coordinate system $o-xy$ fixed on the LRS is obtained by applying the translational and rotational transformation as,

$$\hat{\mathbf{w}}_{(k-1),i}^* = \mathbf{T}(-\hat{\phi}_{k-1}) (\hat{\mathbf{w}}_{(k-1),i} - \hat{\mathbf{X}}_{k-1}), \quad (7.20)$$

to all points $\hat{\mathbf{w}}_{(k-1),i}$ in the estimated global map $\hat{\mathbf{W}}_{k-1}$ of Eq.(7.18).

Points $\hat{\mathbf{w}}_{(k-1),j}^*$ within the LRS range are filtered from the map points $\hat{\mathbf{w}}_{(k-1),i}^*$, and transformation of translational displacement $\Delta\mathbf{X}_q(k|k-1)$ and rotational displacement $\Delta\phi_q(k|k-1)$ are applied iteratively on all the points. At the q th iteration count, the points are defined as,

$$\hat{\mathbf{w}}_{(k-1),j}^q = \mathbf{T}(-\Delta\phi_q(k|k-1))\hat{\mathbf{w}}_{(k-1),j}^* - \Delta\mathbf{X}_q(k|k-1). \quad (7.21)$$

In addition, the matrix, which is constructed by arranging the vector $\hat{\mathbf{w}}_{(k-1),j}^q$ in a row, is defined as,

$$\hat{\mathbf{W}}_{k-1}^q = \begin{bmatrix} \hat{\mathbf{w}}_{(k-1),1}^q & \hat{\mathbf{w}}_{(k-1),2}^q & \cdots & \hat{\mathbf{w}}_{(k-1),N_{\text{map}}^*}^q \end{bmatrix}. \quad (7.22)$$

Here, center of gravity $\mathbf{G}_c^q$ of the all point in the matrix $\hat{\mathbf{W}}_{k-1}^q$ can be obtained as,

$$\mathbf{G}_c^q = \frac{1}{N_{\text{map}}^*} \sum_{i=1}^{N_{\text{map}}^*} \hat{\mathbf{w}}_{(k-1),i}^q. \quad (7.23)$$

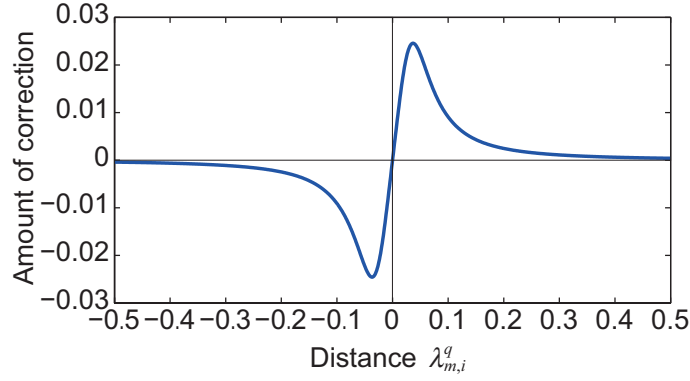


Fig. 7.10 Example of $\delta \mathbf{f}_{m,i}^q$ ($\gamma_f = 10^{-4}$, $\epsilon = 10^{-4}$, $\kappa = 3$)

In this study, amount of correction of translational displacement generated by a line segment $\mathbf{S}_m$ for a point $\hat{\mathbf{w}}_{(k-1),i}^q$ in $\hat{\mathbf{W}}_{k-1}^q$ is given by,

$$\delta \mathbf{f}_{m,i}^q = \begin{cases} \gamma_f \frac{\lambda_{m,i}^q}{|\lambda_{m,i}^q|^{\kappa+\epsilon}} \mathbf{e}_{n,m} & (\lambda_{t(m,P_m)} \leq \lambda_{t(m,i)}^q \leq \lambda_{t(m,Q_m)}) \\ \mathbf{0} & (\lambda_{t(m,i)}^q < \lambda_{t(m,P_m)}, \quad \lambda_{t(m,Q_m)} < \lambda_{t(m,i)}^q) \end{cases}, \quad (7.24)$$

as shown in Fig.7.9, where the vector $\mathbf{e}_{n,m}$ represents a unit normal vector to line segment $\mathbf{S}_m$ expressed as,

$$\mathbf{e}_{n,m} = [\cos \theta_m \quad \sin \theta_m]^T, \quad (7.25)$$

and $\lambda_{m,i}^q$ represents distance between the line segment $\mathbf{S}_m$ and the point $\hat{\mathbf{w}}_{(k-1),i}^q$ expressed as,

$$\lambda_{m,i}^q = -(\mathbf{e}_{n,m}^T \hat{\mathbf{w}}_{(k-1),i}^q - \rho_m). \quad (7.26)$$

As shown in Fig.7.9, $\mathbf{R}_{m,i}^q$ is a point on segment $\mathbf{S}_m$ which is perpendicular from the point $\hat{\mathbf{w}}_{(k-1),i}^q$. Moreover, $\lambda_{t(m,i)}^q$ in Fig.7.9 represents distance between start point $\mathbf{p}_k(P_m)$ of the segment $\mathbf{S}_m$ and point $\mathbf{R}_{m,i}^q$ which is given as,

$$\mathbf{R}_{m,i}^q = \hat{\mathbf{w}}_{(k-1),i}^q + \lambda_{m,i}^q \mathbf{e}_{n,m}, \quad (7.27)$$

and the distance $\lambda_{t(m,i)}^q$ is given by,

$$\lambda_{t(m,i)}^q = \mathbf{e}_{t,m}^T (\mathbf{R}_{m,i}^q - \mathbf{p}_k(P_m)). \quad (7.28)$$

Here, the vector $\mathbf{e}_{t,m}$ is the unit vector along the same direction as the segment $\mathbf{S}_m$ and expressed as,

$$\mathbf{e}_{t,m} = [-\sin\theta_m \ \cos\theta_m]^T. \quad (7.29)$$

Similarly, it follows from Eq.(7.28) that the distance $\lambda_{t(m,Q_m)}$ from the start point P_m to the end point Q_m is given by,

$$\lambda_{t(m,Q_m)} = \mathbf{e}_{t,m}^T (\mathbf{p}_k(Q_m) - \mathbf{p}_k(P_m)). \quad (7.30)$$

Obviously, the distance $\lambda_{t(m,P_m)}$ is zero, as it is the distance from the start point P_m to itself on the line segment $\mathbf{S}_m$ as,

$$\lambda_{t(m,P_m)} = \mathbf{e}_{t,m}^T (\mathbf{p}_k(P_m) - \mathbf{p}_k(P_m)) = 0. \quad (7.31)$$

Therefore, the amount of correction of translational displacement $\delta \mathbf{f}_{m,i}^q$ in Eq.(7.24) becomes the normal vector to the line segment $\mathbf{S}_m$ when the point $\mathbf{R}_{m,i}^q$ exists on the segment $\mathbf{S}_m$, and becomes zero vector when the point $\mathbf{R}_{m,i}^q$ does not exist on the segment. Notice that, in Eq.(7.24), the condition $(\lambda_{t(m,P_m)} \leq \lambda_{t(m,i)}^q \leq \lambda_{t(m,Q_m)})$ ensures that point lies on the segment, and the condition $(\lambda_{t(m,i)}^q < \lambda_{t(m,P_m)}, \ \lambda_{t(m,Q_m)} < \lambda_{t(m,i)}^q)$ ensures that the point does not lie on the segment.

Equation (7.24) is introduced to simulate a force such as the gravitational or magnetic force which decreases rapidly with increasing distance between two objects, where γ_f , ε and κ are the parameters. Figure 7.10 is an example of $(\mathbf{e}_{n,m}^T \delta \mathbf{f}_{m,i}^q)$ with respect to $\lambda_{m,i}^q$ given by Eq.(7.24). It can be observed that the amount of correction closes to 0 to decrease the influence on the amount of correction when the distance between line segment and a point is increased, i.e. for points which are too far from the line. On the other hand, the amount of correction is almost linear within a region of small distance. Amount of correction of rotational displacement is defined as,

$$\delta M_{m,i}^q = \gamma_M (\hat{\mathbf{w}}_{(k-1),i}^q - \mathbf{G}_c^q) \otimes \delta \mathbf{f}_{m,i}^q, \quad (7.32)$$

which is the moment around center of gravity $\mathbf{G}_c^q$ due to amount of correction of translational displacement $\delta \mathbf{f}_{m,i}^q$. Here, operator $\otimes$ gives the norm of the vector given by cross product of two vectors.

Amount of correction for the map $\hat{\mathbf{W}}_{k-1}^q$ is given by the average of all corrections from N_{Seg} segments $\mathbf{S}_m (1 \leq m \leq N_{Seg})$ to each point of map $\hat{\mathbf{W}}_{k-1}^q$ as follows:

$$\begin{aligned} \mathbf{f}_q &= \frac{1}{N_{Seg} N_{map}^*} \sum_{m=1}^{N_{Seg}} \sum_{i=1}^{N_{map}^*} \delta \mathbf{f}_{m,i}^q, \\ M_q &= \frac{1}{N_{Seg} N_{map}^*} \sum_{m=1}^{N_{Seg}} \sum_{i=1}^{N_{map}^*} \delta M_{m,i}^q. \end{aligned} \quad (7.33)$$

Initially at the start of robot operation, I set $\Delta \mathbf{X}_1(k|k-1) = \mathbf{0}$, and $\Delta \phi_1(k|k-1) = 0$. Later, $\Delta \mathbf{X}_q(k|k-1)$ and $\Delta \phi_q(k|k-1)$ are updated by the following equations,

$$\Delta \mathbf{X}_{q+1}(k|k-1) = \xi \Delta \mathbf{X}_q(k|k-1) - \eta \mathbf{f}_q - \zeta \sum_{i=1}^{q-1} \mathbf{f}_i \quad (7.34)$$

$$\Delta \phi_{q+1}(k|k-1) = \xi \Delta \phi_q(k|k-1) - \eta M_q - \zeta \sum_{i=1}^{q-1} M_i \quad (7.35)$$

where, $\mathbf{f}_q$ and M_q are given by Eq.(7.33). Here, the speed of the convergence can be tuned by parameters ξ , η and ζ . The update is finished at the time when $|\mathbf{f}_q| < \rho_f$ and $|M_q| < \rho_M$ are satisfied, where ρ_f and ρ_M represents the thresholds of convergence. Finally, the estimated values of $\Delta \mathbf{X}(k|k-1)$ and $\Delta \phi(k|k-1)$ can be obtained by,

$$\Delta \mathbf{X}(k|k-1) = \Delta \mathbf{X}_{q+1}(k|k-1), \quad (7.36)$$

$$\Delta \phi(k|k-1) = \Delta \phi_{q+1}(k|k-1), \quad (7.37)$$

The calculations are executed recursively returning to Eq.(7.21). Moreover, self-localization can be achieved by Eq.(7.19). Simultaneous Localization And Mapping (SLAM) is also achieved by executing alternately the map updating in subsection 7.4.1 and self-localization in subsection 7.4.2.

7.5 Results and Discussion

This section discusses results in simulation and real environments. For the real environment, Pioneer-P3DX [131] robot shown in Fig.2.9a was used. It is a two wheeled differential drive robot. The motion model of the robot is first described. The distance between the left and the right wheel is W_r , and the robot state at position P , is given as $[x, y, \theta]$. From Fig.2.9c, the motion model is calculated as given in Section 2.7.

The flowchart of mapping and localizing process integrated with the proposed line extraction method is shown in Fig.7.11. For each LRS scan at the k th step, the line segments are extracted and shorter segments are removed using the algorithm in subsection 7.3.2. For the very first scan, the true and estimated map is the same. For subsequent scans, the global map $\hat{\mathbf{W}}_{k-1}$ at the $(k-1)$ th step given by Eq.(7.18) is extracted and the detected line segments are matched against the map to estimate the position $\Delta \mathbf{X}(k|k-1)$ (by Eq.(7.34)) and angle $\Delta \phi(k|k-1)$ (by Eq.(7.35)). Based on this estimation, map is updated and the process is repeated for the LRS scan at the $(k+1)$ th step.

7.5.1 Simulation Results

Simulation based on URG-04LX (HOKUYO AUTOMATIC) laser range sensor (Fig.5.11) whose specification have been summarized in Table 7.1 was performed. Figure 5.11(b) shows the measurement range of the LRS. Gaussian noise of ± 25 mm and $\pm 2.5\%$ was added in distance range of 1000 mm and 4000 mm, respectively. Effects of errors caused by the shape of objects due to reflection etc. were not taken into consideration. Compared to the real LRS, the angular resolution was set to $1/8$ for faster computation. The simulation was carried on MATLAB 7.12 (R2011a) running on a Linux machine with Intel Core i7-4500U CPU, 1.80 GHz, and 16GB RAM.

Taking the distance resolution of the sensor into consideration (Table 7.1), the value of threshold σ_{Th} in the line detection algorithm (Fig.7.5) was set to $25/1024$. The map of the sensor environment used for self positioning was constructed by randomly extracting 75% of points from the world map which are within the sensor range. The values of parameters in the correction Eq.(7.24) were set to $\kappa = 3, \varepsilon = 0.0001, \gamma_f = 0.001$, and parameters γ_M in Eq.(7.32) was set to $\gamma_M = 10$. The convergence parameters of Eqs.(7.34) and (7.35) were set to $\xi = 0.99, \eta = 1, \zeta = 0.01$, and the parameters

Table 7.1 Laser range sensor for simulation

Laser range sensor	Simulated model of URG-04LX
Distance range	60 mm $\sim$ 4000 mm
Accuracy	20 $\sim$ 1000 mm : ± 25 mm 1000 $\sim$ 4000 mm : ± 2.5 %
Distance resolution	3.84 mm ((4000-60) mm / 1024)
Scan angle	240 deg
Angular resolution	1.875 deg (240 deg / 128)

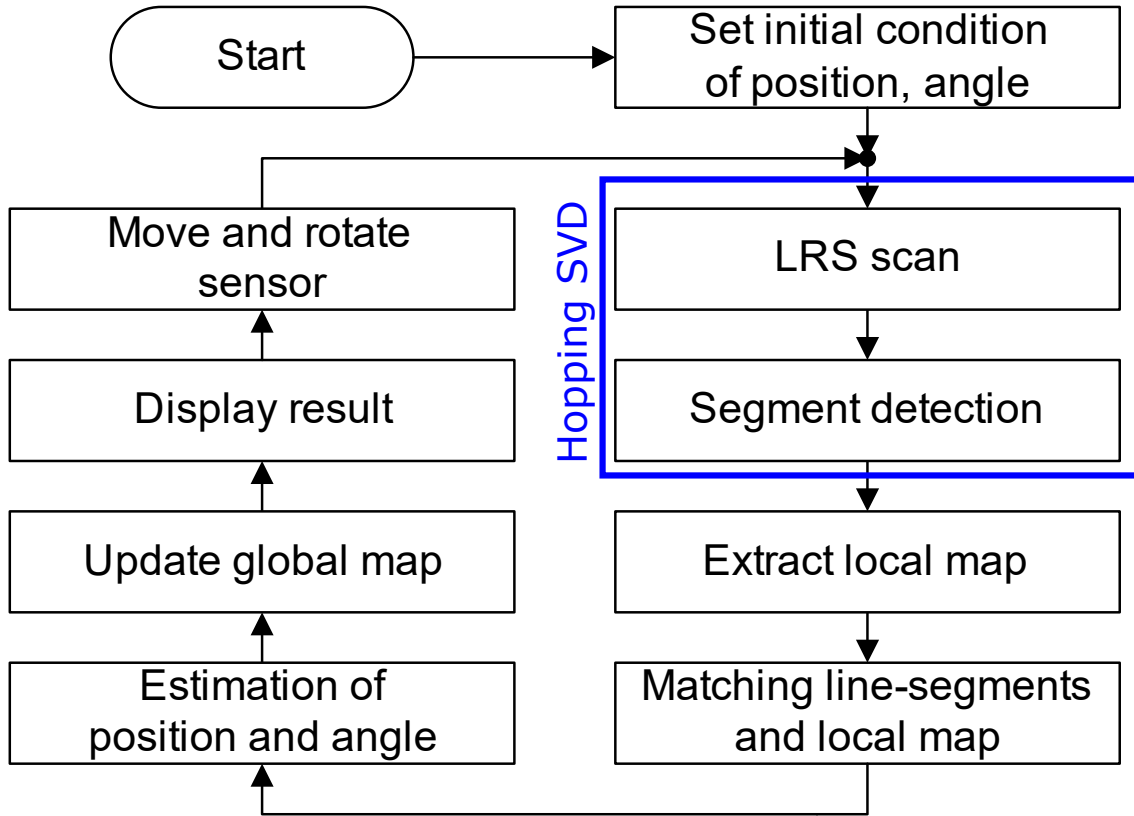


Fig. 7.11 Flow chart of the SLAM process integrated with proposed line segment detection. Marked portion shows the proposed line extraction accelerated by hopping.

of convergence condition were set to $\rho_f = 10^{-5}$, $\rho_M = 10^{-5}$. The values of these parameters were chosen as a trade-off between computation time and accuracy of localization, and determined by experimental trials. The hopping factor ψ was set to a small value of 1.

An example of map matching result in the sensor's $o-xy$ coordinate is shown in Fig.7.12. The result of simultaneously applying mapping and localization by the proposed method while moving the sensor in the upward direction for around 6 meters is shown in Fig.7.13. Estimation is performed for each point and map is updated.

The starting point of the extracted line is shown by $\square$, and the end point by $\bigcirc$ in Fig.7.12 and Fig.7.13. The final constructed map is shown by small '+' mark in pink color. Sensor location is shown by a larger '+' sign in pink color, and similarly sensor orientation is shown by an arrow. Not only accurate mapping but accurate robot localization was also achieved with the proposed method. The total execution time was 1015 seconds which included the calculations and display of graphs. It can be seen

that setting the threshold σ_{Th} to a very small value caused many smaller line segments to be extracted.

Figure 7.14, Fig.7.15, and Fig.7.16 represent the estimation error in X -axis, Y -axis, and sensor angle ϕ , respectively. Figure 7.17 represents the number of iterations q in Eqs.(7.34) and (7.35) at step k . For a complex environment, the upper limit for the total number of iterations was set to 1000. It can still be seen that map construction and localization was done accurately. From Fig.7.17, it can be seen that although there are steps where the total iterations were limited to 1000, but on an average, the iteration count was limited to 200 iterations, and less than 500 iterations. An angular estimation error of 0.5 deg is shown in Fig.7.16. Similarly, from Fig.7.14 and Fig.7.15, the error was limited to 5 cm for a 6 m run. The accuracy can still be improved by lowering the values of threshold ρ_f and ρ_M at the cost of the efficiency of the SLAM process.

The proposed hopping point SVD based line extraction algorithm was tested for a large set of almost linear 2280 points generated by URG-04LX in three subsequent scans ($760 \times 3=2280$, 760 points per scan) against different values of hopping factor ψ . As shown in Fig.7.5, the proposed algorithm applies SVD on data generated from the start point of each detected segment to its end point, after which the process is repeated from the beginning for the new line. The worst case scenario for the algorithm is the case when all the 2280 points are almost linear, and has been considered for calculating the speed-up. The execution time for different values of ψ is summarized in Table 7.2. It can be seen that the speed-up is nearly linear.

Table 7.2 Execution time for line detection using hopping point SVD on 2280 (760×3) points and speed-up for various values of ψ .

Parameter	$\psi = 0$	$\psi = 1$	$\psi = 2$	$\psi = 3$	$\psi = 4$	$\psi = 5$	$\psi = 6$
Time (s)	28.36	14.90	10.01	7.68	6.08	5.09	5.25
Speed up	1.00	1.90	2.83	3.68	4.66	5.56	5.40

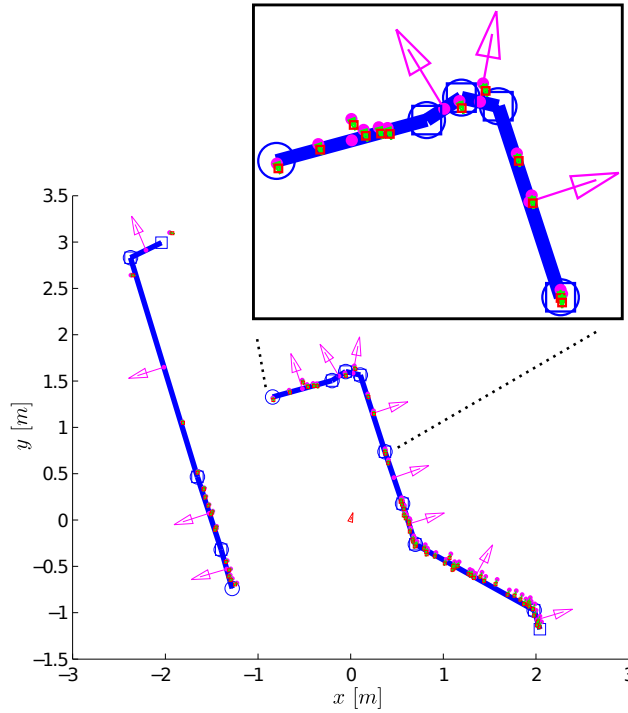


Fig. 7.12 Simulation result of estimation of sensor position with map matching

7.5.2 Experiments in Real Environment

The robot P3-DX (Fig.2.9a) was used in the experiment and the motion model has been explained earlier. The robot was equipped with the same sensor used in simulation, and moved in the environment shown in Fig.7.18a and data was collected. The environment is one of the corridors of our university cluttered with real objects such as boxes and tables. The corridor environment shown in Fig.7.18c is approximately 4 m in breadth and 35 m in length.

Figure 7.18b shows the dead reckoning of the collected sensor data. Figure 7.18c shows the final map constructed by the proposed method. The line segments are indicated in black bold line in the map. Many line segments are detected, and the start point of each line segment is indicated by a blue $\square$, and the end point by red $\circ$ in Fig.7.18c. Notice that, when the end point of a line segment is the starting point of the new line segment, the blue $\square$ and the $\circ$ overlap each other. Some minor segment were also detected by the robot in the map as the environment was cluttered. When the objects are too far from the LRS and no enough data is available, a bunch of very small line segments which are not joined to each other can also be seen. However, this can be corrected by moving the robot sufficiently close to the far away objects and collecting

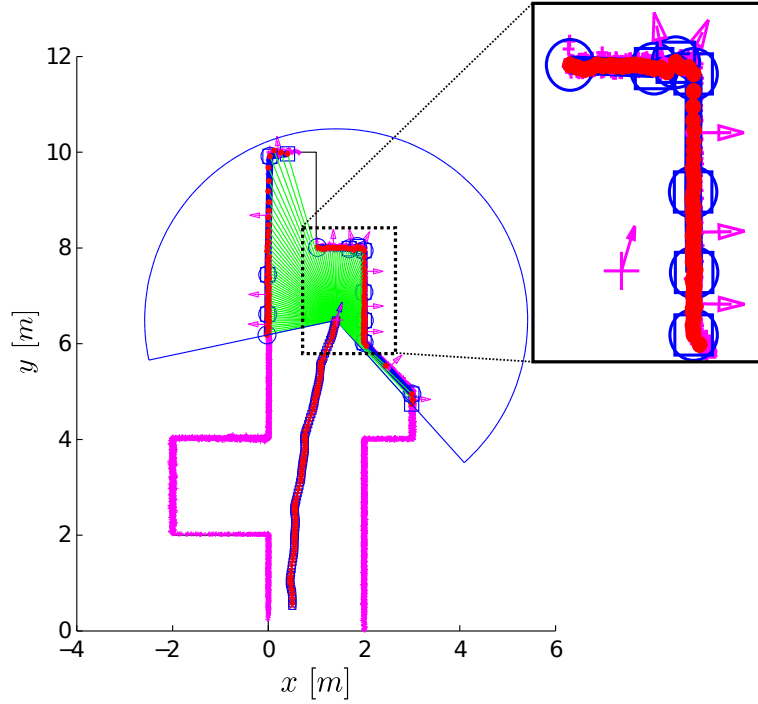
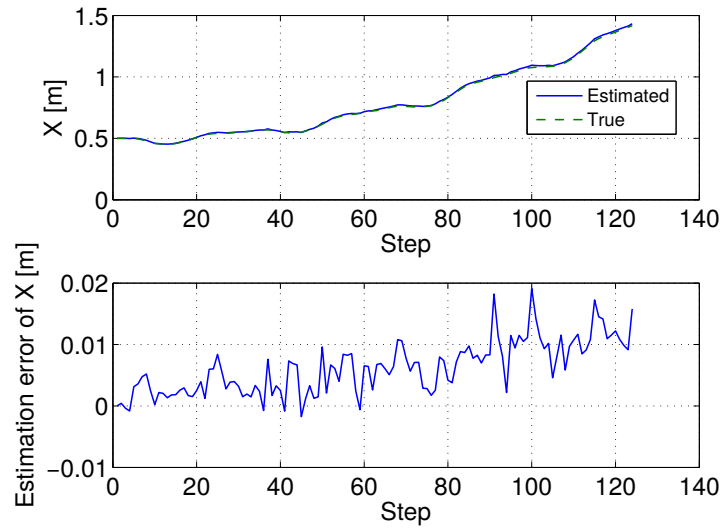
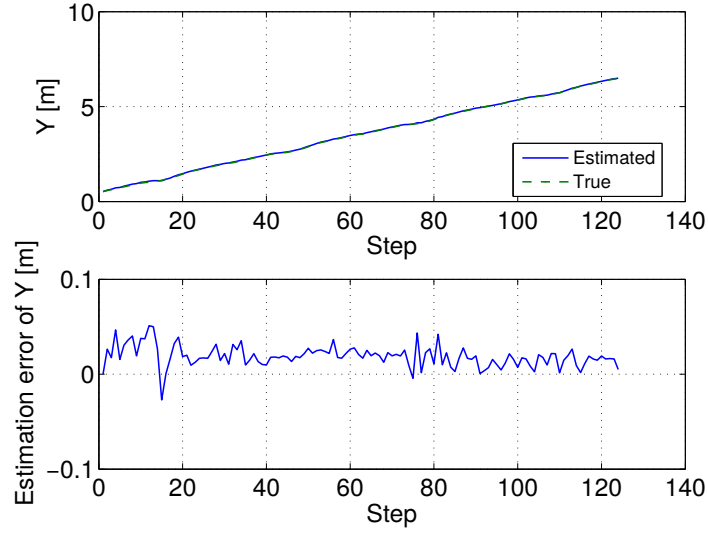
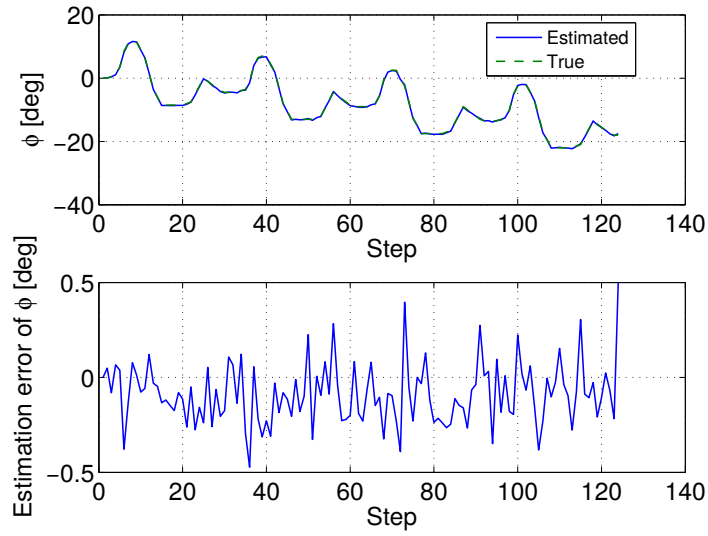


Fig. 7.13 Result of simulation of simultaneous localization and mapping

Fig. 7.14 Estimation error in X -axis

enough data. Line map was obtained by the proposed method with a runtime of 1453.2 seconds. A bulk of computation ($1302 \text{ s} \approx 89\%$) was consumed in map updating and localization with a large iterations for convergence of Eq.(7.34) and Eq.(7.35). As

Fig. 7.15 Estimation error in Y -axisFig. 7.16 Estimation error of sensor angle ϕ

explained in Section 7.4.2, this convergence can be tuned by parameters ξ , η and ζ . Although not realized in the current work, the proposed line extraction algorithm has a scope of parallelization with different CPU threads working on different sections of line segments which can later be joined.

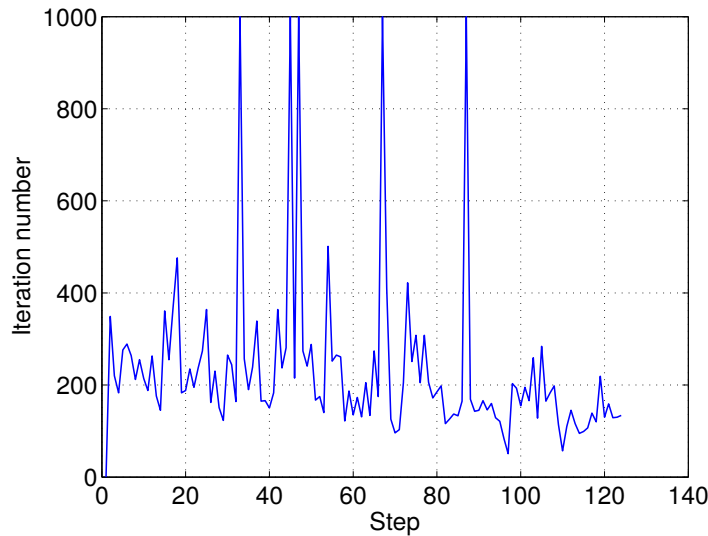


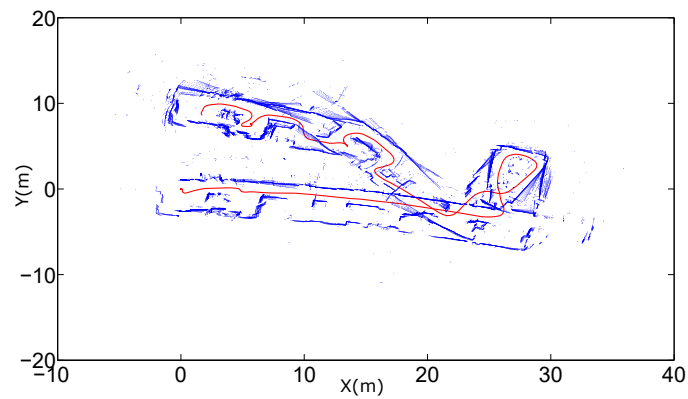
Fig. 7.17 Result of simulation of iteration number for estimation of sensor position

7.6 Summary

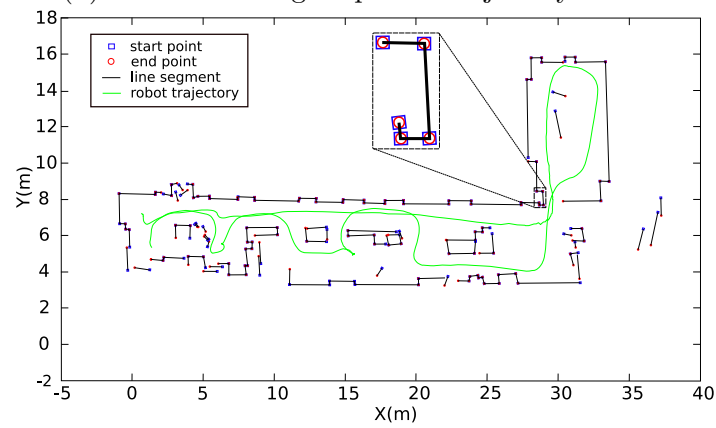
While there exists a large plethora of literature on building maps using traditional techniques like the Extended Kalman Filter and other probabilistic methods which have been successfully demonstrated in various scenarios, this chapter took a fresh approach of using SVD and Hough transform for line detection from LRS data to build maps. The algorithm was accelerated using a novel hopping points method in which SVD is applied on intermittent points governed by a hopping parameter ψ . A reverse hop mechanism ensures that the end points of the line segment are detected accurately. The proposed line detection method was integrated in a SLAM framework and it was shown how lines detected using the proposed method can be used as landmarks for the robot to localize itself in the map and build a map of the environment at the same time. By appropriately setting various parameter values, the proposed algorithm gives a flexibility to obtain maps with desired accuracy and speed. The proposed method was tested in both simulation and real environments and found that the proposed method can generate accurate maps with less error. Although the parameters of the proposed method needs to be determined experimentally, once they have been determined for a given sensor characteristics, the parameters actually help in tuning the convergence speed and accuracy of mapping and localization. The proposed algorithm has a scope of parallelism and different line segments can be detected separately and later joined in the single map. Our future work deals in exploiting this parallelism and testing the



(a) Corridor environment for experiment.



(b) Dead reckoning map with trajectory in red.



(c) Corrected line map with robot trajectory in green.

Fig. 7.18 Result with experiment environment.

algorithm in more complex and noisy environments. A robust comparison with other line segment detection techniques is also currently under work.

Chapter 8

Discussion, Conclusion, and Future Works

The work in this thesis presented novel algorithms to address problems in multi-robot systems in order to achieve autonomy. Novel algorithms corresponding to each of the major components of a multi-robot system were presented with experimental results. This chapter discusses the implications of the results of the presented work, concludes the work in this thesis, and finally shows directions of future works.

8.1 Discussion

The complexity and problems of the underlying system are proportional to the number of service robots employed in the system. Introducing multiple robots introduces many problems like collision free path planning, task management, and resource sharing. However, the potential benefits of using multiple robots for services are tremendous. Nevertheless, the problems must aptly be addressed with efficient solutions. To this end, this thesis presented algorithms to achieve high level of autonomy in multi-robot systems to perform certain tasks easily and efficiently.

The work in this thesis puts emphasis on usability and practicality of the proposed algorithms in real world scenarios. An effort was made to make the best use of previously proposed works in the literature. For example, in case of the problem of multi-robot path planning, the proposed work focussed on developing algorithms which can work on the top of already existing algorithms. The proposed SHP algorithm in the thesis works as an ‘extension’ and can be used with all of the previously proposed and very efficient algorithms like PRM, A*, D*, or RRT planners. Regarding smooth path generation too, practical usage in real world is emphasized. Robots must keep

a safe distance from either of the walls of the corridor. Other path smoother which employs techniques like interpolation, unnecessarily brings the robot close to either of the walls, sacrificing practicality just in order to preserve certain mathematical properties. Service robots in real work move at a relatively slow velocities and safety is often the prime concern. SHP prioritizes safety as well as avoid deadlocks and resolves conflicts by considering factors like task priorities.

It must be noted that most of the previous works in autonomous robots assume that robots will be charged by someone. However, in order to achieve “*life long autonomy*”, robots must device a way to recharge automatically. Particularly important is to manage a limited number of charging points intelligently with a large number of robots. To this end, this thesis made contribution by proposing a resource sharing manager which can easily be used to share other common resources.

Introducing more robots also introduces the problem of explicitly programming and instructing the robot to perform certain task. Each robot must explicitly be programmed to go to a certain place which does not happen to be the work area of another robot at that time. The complexity of performing such a task manually by a human operator in a very large workplace is immense. To this end, the pheromone and anti-pheromone based hybrid signalling mechanism proposed in the thesis comes to rescue. Unlike traditional virtual pheromone deposition schemes, the thesis contributed by integrating EKF SLAM in the virtual pheromone deposition process. Novel ideas of ‘area capture’ was proposed which removes the complexity of programming robots explicitly. Instead, the robots can now be given simple instructions like, “*Go and clean the entire area*”. The robots can move to diverse areas with a complete knowledge of which paths have been taken for better exploration, and which areas have already been captured for better efficiency. Using this signalling, robots may even ‘call’ other robots for cooperation.

An important contribution has been made in virtual pheromone trailing by modelling positional uncertainty in virtual pheromone deposition in the map. Pheromone trailing is a well studied phenomenon in robotics and a wide range of applications have successfully been proposed. Traditionally, pheromones have been thought of as static quantities with binary qualities, i.e. they just indicate the presence or absence of something. A presence of pheromones indicates that the robot took this path, and its absence means that this path has been unexplored. Although a very powerful technique, traditional pheromone trail techniques have largely been limited to simulations. This is mainly because the dynamics and challenges of mobile robots working in fields are

immense and hard to model. One such factor is a good localization of the robot in such dynamic environments which is difficult to achieve. Since the location at which pheromones are deposited is very crucial for a multi-robot system employing virtual pheromone trailing, the thesis emphasizes that the positional uncertainty of the robot needs to be modeled in pheromone deposition. This modelling, as proposed in the current work, provides extra information to the trailing robot about the preceding robot. A ‘confident’ deposition concentrated in a small area indicates that the preceding robot was certain about its location at that place. Whereas, low and dispersed pheromones indicates that the preceding robot was not sure about its location while depositing pheromones.

Virtual pheromone based task coordination is good particularly when the leader robots are not in the field of view of the trailing robots. The case when the leader robot is in the field of view was also considered and a novel idea of hitchhiking was introduced in multi-robot systems. When two or more robots have the same start and goal locations, it is a waste of computation power if all the robots perform their own path planning, obstacle avoidance, dynamic map update, and localization separately. However, it was demonstrated that such drawbacks are eliminated by hitchhiking. Notice that, even in day-to-day lives, people too follow similar strategies, even in indoor environments. If there are two people who want to go to a particular room and only the first person is sure about how to go there, the second person relies entirely upon the first person for navigation. The second person does not search its own path. Moreover, if the environment is crowded, the first person makes sure that the second person is not lost. The work in thesis contributed by incorporating this symbiotic behaviour in multi-robot systems.

Finally, contribution has been made in simultaneous localization and mapping by mobile robots. While there exists a large plethora of literature on building maps using traditional techniques like the Extended Kalman Filter and other probabilistic methods which have been successfully demonstrated in various scenarios, this thesis took a fresh approach of using SVD and Hough transform for line detection from LRS data to build maps. SLAM forms the core of most of the autonomous mobile robots. It is also tightly coupled with path planning, task coordination, and resource sharing which are dependent on SLAM for efficient work. A robust implementation and know-how of underlying SLAM also helps to fine-tune the higher level modules. For example, in this thesis, pheromone trailing was improved by modelling positional uncertainty which directly appears in the underlying SLAM algorithm.

The thesis not only presented novel contributions of the individual components, but also discussed their interdependence. To this end, the thesis makes a contribution by reiterating and emphasizing that components of a multi-robot system must never be thought of as separate entities working solely on their own, but instead, the synchronization and dependencies between the different components must be carefully considered in designing a multi-robot system.

8.2 Conclusion

The thesis first addressed the important problem of multi-robot collision free and smooth path planning, for which, novel smooth hypocycloidal paths (SHP) were proposed. SHP paths are at a safe distance from the environmental obstacles and avoid sharp turns, unlike the paths generated by D* algorithm or PRM algorithm. It was shown that the nodes of the curved paths can be used as points for the robot to retreat in order to avoid collision and the results were experimentally verified with performance benefits over previously proposed methods. The cases of robot retreat with different priorities were discussed. In case of dynamic obstacle, a decoupled approach is used to update other robots with the coordinates of the new and dynamic obstacles in the map.

The problem of common resource sharing was presented in the form of a assigning manager. Long term autonomy of service robots requires frequent recharging. To this end, the case of sharing limited number of docking stations for a large number of robots was presented. The manager maintained a modified queue based on a score which was calculated from the task priority and the current power level of the robot. The manager could easily be extended to incorporate other factors. The resource itself could be other entities like common narrow paths. A monitoring and controlling GUI software was designed to give a complete picture about useful information like the status of the docking points, tasks assigned to robots, and their current power levels. A virtual pheromone signalling based task cooperation of multiple robots was presented. A novel idea of a hybrid pheromone and anti-pheromone signalling mechanism for map exploration by multiple service robots was proposed. The hybrid mechanism involves anti-pheromones which repel robots across diverse portions of map, whereas pheromones attract robots to particular regions. The signalling mechanism was integrated with EKF localization which allows robots to capture areas of sub-areas of the map which enables the robot to work without interference from other robots. Results with both anti-pheromone and pheromone signalling showed that the proposed mechanism can be

effectively used to explore the map using multiple robots, for tasks which require diverse map exploration, or collaborative performance, without any explicit programming.

The pheromone trailing mechanism which is a well studied phenomenon in robotics and forms the basis of a wide range of applications in swarm robots was further improved by proposing a mathematical model of uncertainty modelling in pheromone deposition. The model was tested in both simulation and real environments and significantly improved the behaviour of trailing robots for a cooperative task in a multi-robot system.

A novel idea of multi-robot cooperation through hitchhiking was presented. A robot must localize, do path planning, obstacle avoidance, and update the map if there are new features in the environment while navigating to its goal location. When two or more robots have the same goal location, it is a waste of computing if all the robots performs all these tasks individually. To this end, a novel concept of ‘hitchhiking’ was introduced as a means of efficiency gains. The hitchhiker robot can save computation required for (a) path planning, (b) localization, (c) dynamic obstacle avoidance, and (d) map update. The system has an award mechanism in which the hitchhiker ‘owes’ to the driver robot which receives points. The admissible characteristics of the hitchhiker and the driver robots were discussed. Important points about when should hitchhiking be allowed and when not were also discussed.

Finally, the thesis presented a novel line segment based SLAM algorithm based on SVD and Hough transform. The algorithm was accelerated using a novel hopping points method in which SVD is applied on intermittent points governed by a hopping parameter. A reverse hop mechanism ensured that the end points of the line segment are detected accurately. The proposed line detection method was integrated in a SLAM framework and it was shown how lines detected using the proposed method could be used as landmarks for the robot to localize itself in the map and build a map of the environment at the same time.

8.3 Future Works

The work proposed in this thesis is ongoing and there are many potential areas of improvement. Most of these improvements come from the various ‘assumptions’ which were made in the work in this thesis. These are considered as future works and listed below:

1. **Natural Language Interface**: How does the operator instruct the multiple robots? In the proposed work, the instruction was made by setting specific

parameters of the program. However, a non-technical person must be able to control and instruct the multi-robot system. This demands for a high level natural language interface which can take input from the operator as text or speech and convert the input to program-readable instruction. Many open-source libraries which rely on machine learning solutions are available for such tasks and needs to be integrated into the system for smooth control.

2. **Inter-Robot Networking**: Robots in the proposed work assumes the availability of network through which they can communicate with each other. However, scenarios in which the robots go out of the network range, or when the network connection is broken down needs to be considered.
3. **Interactivity with the Environment**: Multiple robots in the current work simply avoid obstacles and plan a safe path. However, there are situations, like in a disaster scenario in which such a clear path is not available. In such cases, multiple robots must interact with the environment and move the rubble or the obstacles to make way. Such ‘interaction’ with the environment is considered as a potential future work.
4. **Fuzziness of tasks**: Although computers work on binary values, for example, a task was done or it was not done, human beings also work on probabilistic and fuzzy values like ‘almost done’. In the proposed work, the robots works on the binary model but ‘fuzziness’ needs to be incorporated to increase their efficiency.
5. **Heterogeneous Multi-Robots**: The current work assumes homogeneous robots. Depending upon the use case scenarios, the system performance can greatly be leveraged by introducing heterogeneous robots in the system. For example, introducing drones in a search and rescue mission can greatly benefit the system by having a bird’s eye view of the world. Such a view is not possible by employing only ground vehicles. Integrating heterogeneous systems is possible in the presented framework and is considered as potential future work.
6. **Algorithm Profiling**: In context of work related to hitchhiking in multi-robot systems, the algorithms needs to be profiled and energy gains needs to be calculated. Moreover, the work needs to be extended to test hitchhiking in case of more than two robots.

7. **Test in Dynamic Environments:** The algorithms needs to be tested and improved to also incorporate the movement of people in the environment. The work in this thesis considered mainly indoor environments. Outdoor environments poses a different set of challenges. Work needs to be done to consider the case of robots which provide services both indoor and outdoors.
8. **Incorporating Machine Learning and Deep Learning:** Machine learning, and particularly deep learning is gaining keen attention in robotics field for navigation, feature extraction, and manipulation. Currently, these methods are limited by the amount of training data and computational power. However, in limited areas their performance has been found to be at par with human beings. Future work will consider incorporating such learning strategies.

References

- [1] Adams, M. and Kerstens, A. (1998). Tracking naturally occurring indoor features in 2-d and 3-d with lidar range/ amplitude data. *The International Journal of Robotics Research*, 17(9):907–923.
- [2] Ahmad, A. and Ali, J. M. (2008). G3 transition curve between two straight lines. In *Computer Graphics, Imaging and Visualisation, 2008. CGIV '08. Fifth International Conference on*, pages 154–159.
- [3] Ahmad, A., Gobithasan, R., and Ali, J. M. (2007). G2 transition curve using quartic bezier curve. In *Computer Graphics, Imaging and Visualisation, 2007. CGIV '07*, pages 223–228.
- [4] An, S.-Y., Kang, J.-G., Lee, L.-K., and Oh, S.-Y. (2012). Line segment-based indoor mapping with salient line feature extraction. *Advanced Robotics*, 26(5-6):437–460.
- [5] An, S.-Y., Kang, J.-G., Lee, L.-K., and young Oh, S. (2010). Slam with salient line feature extraction in indoor environments. In *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*, pages 410–416.
- [6] Andersson, L. A. A. and Nygard, J. (2008). C-sam: Multi-robot slam using square root information smoothing. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2798–2805.
- [7] Arcelli, C. and Di Baja, G. S. (1985). A width-independent fast thinning algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-7(4):463–474.
- [8] Arduino Board (2016). Arduino embedded board. Available at <https://www.arduino.cc/> [Accessed: 10-10-2016].
- [9] Arras, K. O. and Siegwart, R. Y. (1997). Feature extraction and scene interpretation for map-based navigation and map building. In *Proc. of SPIE, Mobile Robotics XII*, pages 42–53.
- [10] Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.
- [11] Bai, X., Latecki, L., and yu Liu, W. (2007). Skeleton pruning by contour partitioning with discrete curve evolution. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(3):449–462.

- [12] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.
- [13] Besl, P. and McKay, N. D. (1992). A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256.
- [14] Bhattacharya, P. and Gavrilova, M. (2008). Roadmap-based path planning - using the voronoi diagram for a clearance-based shortest path. *Robotics Automation Magazine, IEEE*, 15(2):58–66.
- [15] Blanco, J.-L., Jimenez, J. G., and Fernandez-Madrigal, J.-A. (2013). A robust, multi-hypothesis approach to matching occupancy grid maps. *Robotica*, 31(5):687–701.
- [16] Borges, G. A. and Aldon, M.-J. (2004). Line extraction in 2d range images for mobile robotics. *J. Intell. Robotics Syst.*, 40:267–297.
- [17] Bukhori, I., Ismail, Z. H., and Namerikawa, T. (2015). Detection strategy for kidnapped robot problem in landmark-based map monte carlo localization. In *2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, pages 75–80.
- [18] Calvo, R., de Oliveira, J., Figueiredo, M., and Romero, R. (2011). Bio-inspired coordination of multiple robots systems and stigmergy mechanisms to cooperative exploration and surveillance tasks. In *Cybernetics and Intelligent Systems (CIS), 2011 IEEE 5th International Conference on*, pages 223–228.
- [19] Censi, A. (2008). An icp variant using a point-to-line metric. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 19–25.
- [20] Chang, S.-R. and Huh, U.-Y. (2014). A collision-free g2 continuous path-smoothing algorithm using quadratic polynomial interpolation. *International Journal of Advance Robot Systems*, 11:194, doi:10.5772/59463.
- [21] Chang, S.-R. and Huh, U.-Y. (2015). Curvature-continuous 3d path-planning using qpml method. *International Journal of Advance Robot Systems*, 11:194, doi:10.5772/60718.
- [22] Chaumette, F. (1998). *Potential problems of stability and convergence in image-based and position-based visual servoing*, pages 66–78. Springer London, London.
- [23] Chaumette, F. and Hutchinson, S. (2006). Visual servo control, part i: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90.
- [24] Choi, J.-W., Curry, R., and Elkaim, G. (2008). Path planning based on bezier curve for autonomous ground vehicles. In *World Congress on Engineering and Computer Science 2008, WCECS '08. Advances in Electrical and Electronics Engineering - IAENG Special Edition of the*, pages 158–166.
- [25] Corke, P. (2011). *Robotics, Vision and Control: Fundamental Algorithms in Matlab*, volume 73. Springer, doi = 10.1007/978-3-642-20144-8.

- [26] Cost of Computing (2016). Wikipedia. Available at <https://en.wikipedia.org/wiki/FLOPS> [Accessed: 30-12-2016].
- [27] Couture-Beil, A. and Vaughan, R. T. (2009). Adaptive mobile charging stations for multi-robot systems. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1363–1368.
- [28] Cox, I. J. (1988). Blanche: an autonomous robot vehicle for structured environments. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 978–982 vol.2.
- [29] Cox, I. J. (1989). Blanche: Position estimation for an autonomous robot vehicle. In *Intelligent Robots and Systems '89. The Autonomous Mobile Robots and Its Applications. IROS '89. Proceedings., IEEE/RSJ International Workshop on*, pages 432–439.
- [30] Cox, I. J. (1991). Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *Robotics and Automation, IEEE Transactions on*, 7(2):193–204.
- [31] DARPA Grand Challenge (2016). Wikipedia. Available at <http://mars.nasa.gov/mer/home/> [Accessed: 30-11-2016].
- [32] Daum, F. (2005). Nonlinear filters: beyond the kalman filter. *IEEE Aerospace and Electronic Systems Magazine*, 20(8):57–69.
- [33] Davies, E. R. (2004). *Machine Vision: Theory, Algorithms, Practicalities*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- [34] Davison, A., Reid, I., Molton, N., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067.
- [35] Delingette, H., Hebert, M., and Ikeuchi, K. (1991). Trajectory generation with curvature constraint based on energy minimization. In *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pages 206–211 vol.1.
- [36] Dellinger, D., Sanders, P., Schultes, D., and Wagner, D. (2009). Engineering route planning algorithms. In Lerner, J., Wagner, D., and Zweig, K., editors, *Algorithmics of Large and Complex Networks*, volume 5515 of *Lecture Notes in Computer Science*, pages 117–139. Springer Berlin Heidelberg.
- [37] Desrochers, B., Lacroix, S., and Jaulin, L. (2015). Set-membership approach to the kidnapped robot problem. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3715–3720.
- [38] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

- [39] Doi, S. (2013). Proposal and evaluation of a pheromone-based algorithm for the patrolling problem in dynamic environments. In *Swarm Intelligence (SIS), 2013 IEEE Symposium on*, pages 48–55.
- [40] Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):pp. 497–516.
- [41] Dubins, L. E. (1961). On plane curves with curvature. *Pacific J. Math.*, 11(2):471–481.
- [42] Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15.
- [43] Elbanhawi, M., Simic, M., and Jazar, R. N. (2015). Continuous path smoothing for car-like robots using b-spline curves. *Journal of Intelligent & Robotic Systems*, 80(1):23–56.
- [44] Engelhard, N., Endres, F., Hess, J., Sturm, J., and Burgard, W. (2011). Real-time 3d visual slam with a hand-held camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, Vasteras, Sweden.
- [45] Epperson, J. F. (1987). On the runge example. *Am. Math. Monthly*, 94(4):329–341.
- [46] Ester, M., peter Kriegel, H., S, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press.
- [47] Ezra, E., Sharir, M., and Efrat, A. (2008). On the performance of the icp algorithm. *Comput. Geom. Theory Appl.*, 41(1-2):77–93.
- [48] Farin, G. (2002). *Curves and Surfaces for CAGD: A Practical Guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition.
- [49] Filipescu, A., Susnea, I., Filipescu, S., and Stamatescu, G. (2009). Wheeled mobile robot control using virtual pheromones and neural networks. In *Control and Automation, 2009. ICCA 2009. IEEE International Conference on*, pages 157–162.
- [50] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.
- [51] Florea, B.-F., Grigore, O., and Datcu, M. (2015). Pheromone averaging exploration algorithm. In *Advanced Robotics (ICAR), 2015 International Conference on*, pages 617–622.
- [52] Forsberg, J., Larsson, U., and Wernersson, A. (1995). Mobile robot navigation using the range-weighted hough transform. *Robotics Automation Magazine, IEEE*, 2(1):18–26.
- [53] Fossum, F., Montanier, J.-M., and Haddow, P. (2014). Repellent pheromones for effective swarm robot search in unknown environments. In *Swarm Intelligence (SIS), 2014 IEEE Symposium on*, pages 1–8.

- [54] Fraichard, T. and Scheuer, A. (2004). From reeds and shepp's to continuous-curvature paths. *Robotics, IEEE Transactions on*, 20(6):1025–1035.
- [55] Fu, S., ying Liu, H., fang Gao, L., and xian Gai, Y. (2007). Slam for mobile robots using laser range finder and monocular vision. In *Mechatronics and Machine Vision in Practice, 2007. M2VIP 2007. 14th International Conference on*, pages 91–96.
- [56] Fujisawa, R., Imamura, H., Hashimoto, T., and Matsuno, F. (2008). Communication using pheromone field for multiple robots. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1391–1396.
- [57] Fujisawa, R., Shimizu, Y., and Matsuno, F. (2011). Effectiveness of tuning of pheromone trail lifetime in attraction of robot swarm. In *System Integration (SII), 2011 IEEE/SICE International Symposium on*, pages 702–707.
- [58] Fujita, Y., Nakamura, Y., and Shiller, Z. (2003). Dual dijkstra search for paths with different topologies. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 3, pages 3359–3364 vol.3.
- [59] Garulli, A., Giannitrapani, A., Rossi, A., and Vicino, A. (2005). Mobile robot slam for line-based environment representation. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 2041–2046.
- [60] Genovese, V., Dario, P., Magni, R., and Odetti, L. (1992). Self organizing behavior and swarm intelligence in a pack of mobile miniature robots in search of pollutants. In *Intelligent Robots and Systems, 1992., Proceedings of the 1992 IEEE/RSJ International Conference on*, volume 3, pages 1575–1582.
- [61] Greenspan, M. and Yurick, M. (2003). Approximate k-d tree search for efficient icp. In *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pages 442–448.
- [62] Guo, Y. and Parker, L. (2002). A distributed and optimal motion planning approach for multiple mobile robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 3, pages 2612–2619.
- [63] Habib, Z. and Sakai, M. (2003). Family of g2 cubic transition curves. In *Geometric Modeling and Graphics, 2003. Proceedings. 2003 International Conference on*, pages 117–122.
- [64] Hamasaki, S., Tamura, Y., Yamashita, A., and Asama, H. (2011). Prediction of human's movement for collision avoidance of mobile robot. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 1633–1638.
- [65] Hao, Y., Xiong, Z., Sun, F., and Wang, X. (2007). Comparison of unscented kalman filters. In *2007 International Conference on Mechatronics and Automation*, pages 895–899.
- [66] Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107.

- [67] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.
- [68] Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2012). Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663.
- [69] Holger Karl, A. W. (2005). *Robot Motion Planning*. Kluwer Academic Publishers, Wiley Publishers.
- [70] Hough, P. (1962). Method and means for recognizing complex patterns, us patents 3069654.
- [71] Howard, A. (2006). Multi-robot simultaneous localization and mapping using particle filters. *Int. J. Rob. Res.*, 25(12):1243–1256.
- [72] Howard, C. R. (2011). *The Transition-Curve Field-Book*. Nabu Press, isbn:1245461192 edition.
- [73] Huh, U.-Y. and Chang, S.-R. (2013). A g2 continuous path-smoothing algorithm using modified quadratic polynomial interpolation. *International Journal of Advance Robot Systems*, 11:25, doi:10.5772/57340.
- [74] Hutchinson, S., Hager, G. D., and Corke, P. I. (1996). A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670.
- [75] Hwang, Y. and Ahuja, N. (1992). A potential field approach to path planning. *Robotics and Automation, IEEE Transactions on*, 8(1):23–32.
- [76] Iftikhar, K. and Khan, M. T. (2016). Resource sharing and deadlock avoidance in multi robot systems using market based approach. In *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 1–7.
- [77] Internet of Things (2016). Wikipedia. Available at https://en.wikipedia.org/wiki/Internet_of_things [Accessed: 10-12-2016].
- [78] Internet Society (2016). Internet of things: An overview. Available at <http://www.internetsociety.org/sites/default/files/ISOC-IoT-Overview-20151221-en.pdf> [Accessed: 10-12-2016].
- [79] iRobot (2016). Roomba cleaning robot. Available at <http://store.irobot.com/default/robot-vacuum-roomba/> [Accessed: 12-12-2016].
- [80] Ishigami, G., Nagatani, K., and Yoshida, K. (2007). Path planning for planetary exploration rovers and its evaluation based on wheel slip dynamics. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2361–2366.
- [81] Jain, A. K. (1989). *Fundamentals of digital image processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

- [82] Jiang, R., Tian, X., Xie, L., and Chen, Y. (2008). A robot collision avoidance scheme based on the moving obstacle motion prediction. In *Computing, Communication, Control, and Management, 2008. CCCM '08. ISECS International Colloquium on*, volume 2, pages 341–345.
- [83] Kaess, M., Ranganathan, A., and Dellaert, F. (2008). isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378.
- [84] Karlson, P. and Luscher, M. (1959). ‘pheromones’: a new term for a class of biologically active substances. *Nature* 183, 55 - 56, 183:55–56.
- [85] Kavraki, L., Svestka, P., Latombe, J.-C., and Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580.
- [86] Kermorgant, O. and Chaumette, F. (2011). Combining ibvs and pbvs to ensure the visibility constraint. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2849–2854.
- [87] Kim, B., Kaess, M., Fletcher, L., Leonard, J., Bachrach, A., Roy, N., and Teller, S. (2010). Multiple relative pose graphs for robust cooperative mapping. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3185–3192.
- [88] Kim, M., Kim, H. W., and Chong, N. Y. (2007). Automated robot docking using direction sensing rfid. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4588–4593.
- [89] Knightscope (2016). Knightscope robot. Available at <http://knightscope.com/> [Accessed: 12-12-2016].
- [90] Komoriya, K. and Tanie, K. (1989). Trajectory design and control of a wheel-type mobile robot using b-spline curve. In *Intelligent Robots and Systems '89. The Autonomous Mobile Robots and Its Applications. IROS '89. Proceedings., IEEE/RSJ International Workshop on*, pages 398–405.
- [91] Kurt-Yavuz, Z. and Yavuz, S. (2012). A comparison of ekf, ukf, fastslam2.0, and ukf-based fastslam algorithms. In *2012 IEEE 16th International Conference on Intelligent Engineering Systems (INES)*, pages 37–43.
- [92] Lan, K. C. and Shih, W.-Y. (2013). Using floor plan to calibrate sensor error for indoor localization. In *2013 International Conference on ICT Convergence (ICTC)*, pages 718–723.
- [93] Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA.
- [94] Lavalle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical report.
- [95] LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, U.K. Available at <http://planning.cs.uiuc.edu/> [Accessed: 11-02-2016].

- [96] LaValle, S. M. and Kuffner, J. J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400.
- [97] Lavalle, S. M., Kuffner, J. J., and Jr. (2000). Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308.
- [98] Lee, D.-S. and Lee, A.-C. (2009). Pheromone propagation controller: The linkage of swarm intelligence and advanced process control. *Semiconductor Manufacturing, IEEE Transactions on*, 22(3):357–372.
- [99] Liscano, R. and Green, D. (1989). Design and implementation of a trajectory generator for an indoor mobile robot. In *Intelligent Robots and Systems '89. The Autonomous Mobile Robots and Its Applications. IROS '89. Proceedings., IEEE/RSJ International Workshop on*, pages 380–385.
- [100] Lockwood, E. H. (1961). *Book of Curves*. Cambridge University Press. Cambridge Books.
- [101] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.
- [102] Lozano-Pérez, T. and Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570.
- [103] Lu, F. and Milios, E. E. (1994). Robot pose estimation in unknown environments by matching 2d range scans. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 935–938.
- [104] Luo, R., Huang, C., and Huang, C. (2010). Search and track power charge docking station based on sound source for autonomous mobile robot applications. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1347–1352.
- [105] Luo, R. and Su, K. (2003). A multiagent multisensor based real-time sensory control system for intelligent security robot. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 2394–2399 vol.2.
- [106] Majdik, A., Popa, M., Tamas, L., Szoke, I., and Lazea, G. (2010). New approach in solving the kidnapped robot problem. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–6.
- [107] Mamei, M. and Zambonelli, F. (2005). Physical deployment of digital pheromones through rfid technology. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pages 281–288.
- [108] Marchand, E., Spindler, F., and Chaumette, F. (2005). Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics Automation Magazine*, 12(4):40–52.

- [109] Martinez-Marin, T. and Duckett, T. (2004). Robot docking by reinforcement learning in a visual servoing framework. In *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*, volume 1, pages 159–164 vol.1.
- [110] Maskell, S. and Gordon, N. (2001). A tutorial on particle filters for on-line nonlinear/non-gaussian bayesian tracking. In *Target Tracking: Algorithms and Applications (Ref. No. 2001/174)*, IEE, volume Workshop, pages 2/1–2/15 vol.2.
- [111] Mathematical Interpolation (2016). Wikipedia. Available at <https://en.wikipedia.org/wiki/Interpolation> [Accessed: 11-02-2016].
- [112] Meer, R. K. V., Breed, M. D., Espelie, K. E., and Winston, M. L. (1998). *Pheromone Communication in Social Insects: Ants, Wasps, Bees, and Termites*. Westview Press.
- [113] Menegaz, H. M. T., Ishihara, J. Y., Borges, G. A., and Vargas, A. N. (2015). A systematization of the unscented kalman filter theory. *IEEE Transactions on Automatic Control*, 60(10):2583–2598.
- [114] Minguez, J., Lamiraux, F., and Montesano, L. (2005). Metric-based scan matching algorithms for mobile robot displacement estimation. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3557–3563.
- [115] Minten, B., Murphy, R., Hyams, J., and Micire, M. (2001). Low-order-complexity vision-based docking. *Robotics and Automation, IEEE Transactions on*, 17(6):922–930.
- [116] Mirror Website (2016). Southmead hospital robot. Available at <http://www.mirror.co.uk/lifestyle/health/you-might-think-gluten-free-8870780> [Accessed: 12-12-2016].
- [117] Mohan, Y. and Ponnambalam, S. (2009). An extensive review of research in swarm robotics. In *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 140–145.
- [118] Moore’s Law (2016). Wikipedia. Available at https://en.wikipedia.org/wiki/Moores_law [Accessed: 3-12-2016].
- [119] Mustiere, F., Bolic, M., and Bouchard, M. (2006). Rao-blackwellised particle filters: Examples of applications. In *2006 Canadian Conference on Electrical and Computer Engineering*, pages 1196–1200.
- [120] NASA Jet Propulsion Laboratory, California Institute of Technology (2016). Mars lander. Available at <http://mars.nasa.gov/mer/home/> [Accessed: 30-11-2016].
- [121] Nguyen, V., Martinelli, A., Tomatis, N., and Siegwart, R. (2005). A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1929–1934.

- [122] nVidia Jetson TK1 Board (2016). nvidia. Available at <http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html> [Accessed: 12-10-2016].
- [123] Oliveira, J., Calvo, R., and Romero, R. (2014). Integration of virtual pheromones for mapping/exploration of environments by using multiple robots. In *Biomedical Robotics and Biomechatronics (2014 5th IEEE RAS EMBS International Conference on)*, pages 835–840.
- [124] Pandaboard Board (2016). Pandaboard. Available at <http://pandaboard.org/> [Accessed: 12-10-2016].
- [125] Payton, D., Estkowski, R., and Howard, M. (2001). Compound behaviors in pheromone robotics. *Robotics and Autonomous Systems*, 44:229–240.
- [126] Pearce, J., Rybski, P., Stoeter, S., and Papanikolopoulos, N. (2003). Dispersion behaviors for a team of multiple miniature robots. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 1158–1163 vol.1.
- [127] Pearce, J. L., Powers, B., Hess, C., Rybski, P. E., Stoeter, S. A., and Papanikolopoulos, N. (2006). Using virtual pheromones and cameras for dispersing a team of multiple miniature robots. *Journal of Intelligent and Robotic Systems*, 45(4):307–321.
- [128] Pfister, S., Kriechbaum, K., Roumeliotis, S., and Burdick, J. (2002). Weighted range sensor matching algorithms for mobile robot displacement estimation. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, pages 1667–1674 vol.2.
- [129] Pfister, S., Roumeliotis, S., and Burdick, J. (2003). 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*, volume 1, pages 1304–1311 vol.1.
- [130] Piazzzi, A., Guarino Lo Bianco, C., Bertozzi, M., Fascioli, A., and Broggi, A. (2002). Quintic g2-splines for the iterative steering of vision-based autonomous vehicles. *Intelligent Transportation Systems, IEEE Transactions on*, 3(1):27–36.
- [131] Pioneer P3-DX (2016). Pioneer p3-dx robot. Available at http://www.mobilerobots.com/Mobile_Robots.aspx [Accessed on 23-01-2016].
- [132] Pollack, M. and Wiebenson, W. (1960). Solutions of the shortest-route problem-a review. *Operations Research*, 8(2):224–230.
- [133] Purnamadjaja, A. and Russell, R. (2004). Pheromone communication: implementation of necrophoric bee behaviour in a robot swarm. In *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*, volume 2, pages 638–643 vol.2.
- [134] Purnamadjaja, A. H. and Russell, R. A. (2007). Guiding robots' behaviors using pheromone communication. *Autonomous Robots*, 23(2):113–130.

- [135] Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.
- [136] Raspberrypi Board (2016). Raspberrypi. Available at <https://www.raspberrypi.org/> [Accessed: 12-10-2016].
- [137] Ravankar, A. (2015). Probabilistic approaches and algorithms for indoor robot mapping in structured environments [an abstract of dissertation and a summary of dissertation review].
- [138] Ravankar, A., Hoshino, Y., Emaru, T., and Kobayashi, Y. (2012). Robot mapping using k-means clustering of laser range sensor data. *Bulletin of Networking, Computing, Systems, and Software*, 1(1).
- [139] Ravankar, A., Kobayashi, Y., Lv, J., Emaru, T., and Hoshino, Y. (2014). An embarrassingly parallel hoppingwindow noise removing algorithm for lidar based robot mapping. In *SICE Annual Conference (SICE), 2014 Proceedings of the*, pages 307–312.
- [140] Ravankar, A., Kobayashi, Y., Ravankar, A., and Emaru, T. (2015a). A connected component labeling algorithm for sparse lidar data segmentation. In *Automation, Robotics and Applications (ICARA), 2015 6th International Conference on*, pages 437–442.
- [141] Ravankar, A., Kobayashi, Y., Ravankar, A., Emaru, T., and Jixin, L. (2015b). Real time parallel implementation of dynamic feature detection on embedded robot platforms. *Robomech*, 2015.
- [142] Ravankar, A., Ravankar, A., Kobayashi, Y., Jixin, L., and Emaru, T. (2015c). Vision based localization and mapping for indoor robots using rgb-d sensor. *Robomech*, 2015.
- [143] Ravankar, A., Ravankar, A. A., Hoshino, Y., Emaru, T., and Kobayashi, Y. (2016a). On a hopping-points svd and hough transform based line detection algorithm for robot localization and mapping. *International Journal of Advanced Robotic Systems*, 13(3):98.
- [144] Ravankar, A., Ravankar, A. A., Kobayashi, Y., and Emaru, T. (2016b). Avoiding blind leading the blind. *International Journal of Advanced Robotic Systems*, 13(6):1729881416666088.
- [145] Ravankar, A., Ravankar, A. A., Kobayashi, Y., and Emaru, T. (2016c). Intelligent robot guidance in fixed external camera network for navigation in crowded and narrow passages. *The 3rd International Electronic Conference on Sensors and Applications, Sciforum Electronic Conference Series*, 3(8).
- [146] Ravankar, A., Ravankar, A. A., Kobayashi, Y., and Emaru, T. (2016d). On a bio-inspired hybrid pheromone signalling for efficient map exploration of multiple mobile service robots. *Artificial Life and Robotics*, pages 1–11.

- [147] Ravankar, A., Ravankar, A. A., Kobayashi, Y., and Emaru, T. (2016e). Path smoothing extension for various robot path planners. In *2016 16th International Conference on Control, Automation and Systems (ICCAS)*, pages 263–268.
- [148] Ravankar, A., Ravankar, A. A., Kobayashi, Y., and Emaru, T. (2016f). Shp: Smooth hypocycloidal paths with collision-free and decoupled multi-robot path planning. *International Journal of Advanced Robotic Systems*, 13(3):133.
- [149] Ravankar, A., Ravankar, A. A., Kobayashi, Y., Jixin, L., Emaru, T., and Hoshino, Y. (2015d). An intelligent docking station manager for multiple mobile service robots. In *Control, Automation and Systems (ICCAS), 2015 15th International Conference on*, pages 72–78.
- [150] Ravankar, A., Ravankar, A. A., Kobayashi, Y., Jixin, L., Emaru, T., and Hoshino, Y. (2015e). A novel vision based adaptive transmission power control algorithm for energy efficiency in wireless sensor networks employing mobile robots. In *2015 Seventh International Conference on Ubiquitous and Future Networks*, pages 300–305.
- [151] Ravankar, A. A., Hoshino, Y., Emaru, T., and Kobayashi, Y. (2011). Map building from laser range sensor information using mixed data clustering and singular value decomposition in noisy environment. In *System Integration (SII), 2011 IEEE/SICE International Symposium on*, pages 1232–1238.
- [152] Ravankar, A. A., Hoshino, Y., Ravankar, A., Jixin, L., Emaru, T., and Kobayashi, Y. (2015f). Algorithms and a framework for indoor robot mapping in a noisy environment using clustering in spatial and hough domains. *International Journal of Advanced Robotic Systems*, 12.
- [153] Robot, P. (2016). Pepper Robot by Softbank. Available at <https://www.ald.softbankrobotics.com/en/cool-robots/pepper> [Accessed: 11-12-2016].
- [154] Roumeliotis, S. I. and Bekey, G. A. (2002). Distributed multirobot localization. *IEEE Transactions on Robotics and Automation*, 18(5):781–795.
- [155] Sack, D. and Burgard, W. (2004). A comparison of methods for line extraction from range data. In *In Proc. of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*.
- [156] Sakellariou, G., Shanahan, M., and Kuipers, B. (2004). Skeletonisation as mobile robot navigation. In *In Proceedings of Towards Autonomous Robotic Systems (TAROS-04)*, pages 149–155.
- [157] Sanderson, A. C. and Weiss, L. E. (1983). *Adaptive Visual Servo Control of Robots*, pages 107–116. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [158] Scheduling, S., Dissanayake, G., Nebot, E. M., and Durrant-Whyte, H. (1999). An experiment in autonomous navigation of an underground mining vehicle. *IEEE Transactions on Robotics and Automation*, 15(1):85–95.
- [159] Shiroy Koibito Park, Sapporo (2016). Pepper at park. Available at <http://www.shiroikoibitopark.jp/english/> [Accessed: 11-12-2016].

- [160] Shu-Xi, W. (2012). The improved dijkstra's shortest path algorithm and its application. *Procedia Engineering*, 29:1186 – 1190. 2012 International Workshop on Information and Electronics Engineering.
- [161] Silva, G., Costa, J., Magalhaes, T., and Reis, L. (2010). Cyberrescue: A pheromone approach to multi-agent rescue simulations. In *Information Systems and Technologies (CISTI), 2010 5th Iberian Conference on*, pages 1–6.
- [162] Silverman, M. C., Nies, D., Jung, B., and Sukhatme, G. S. (2002). Staying alive: A docking station for autonomous robot recharging. In *ICRA*, pages 1050–1055. IEEE.
- [163] Simon, D. (2006). *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience.
- [164] Slashgear Website (2016). Knightscope robot. Available at <https://c.slashgear.com/wp-content/uploads/2014/11/K5-2.0-Green.jpg> [Accessed: 12-12-2016].
- [165] Smith, R., Self, M., and Cheeseman, P. (1988). A stochastic map for uncertain spatial relationships. In *Proceedings of the 4th international symposium on Robotics Research*, pages 467–474, Cambridge, MA, USA. MIT Press.
- [166] Softbank Pepper Robot (2016). Pepper robot by softbank. Available at <http://www.softbank.jp/en/robot/> [Accessed: 11-12-2016].
- [167] Song, G., Wang, H., Zhang, J., and Meng, T. (2011). Automatic docking system for recharging home surveillance robots. *Consumer Electronics, IEEE Transactions on*, 57(2):428–435.
- [168] Spero, D. J. and Jarvis, R. A. (2002). Path planning for a mobile robot in a rough terrain environment. In *Robot Motion and Control, 2002. RoMoCo '02. Proceedings of the Third International Workshop on*, pages 417–422.
- [169] Spindler, F. and Novotny, F. (2017). Visp auto tracker. Available at http://wiki.ros.org/visp_auto_tracker [Accessed: 1-01-2017].
- [170] Srivastava, S. and Roychowdhury, J. (2008). Independent and interdependent latch setup/hold time characterization via newton raphson solution and euler curve tracking of state-transition equations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(5):817–830.
- [171] Stentz, A. (1995). The focussed d* algorithm for real-time replanning. In *In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659.
- [172] Stentz, A. and Mellon, I. C. (1993). Optimal and efficient path planning for unknown and dynamic environments. *International Journal of Robotics and Automation*, 10:89–100.
- [173] Stereo Labs (2016). Zed 2k stereo camera. Available at <https://www.stereolabs.com/> [Accessed on 25-11-2016].

- [174] Sundareswaran, K., Vigneshkumar, V., Sankar, P., Simon, S. P., Nayak, P. S. R., and Palani, S. (2016). Development of an improved p and o algorithm assisted through a colony of foraging ants for mppt in pv system. *IEEE Transactions on Industrial Informatics*, 12(1):187–200.
- [175] Susnea, I., Vasiliu, G., and Filipescu, A. (2008). Rfid digital pheromones for generating stigmergic behaviour to autonomous mobile robots. In *Proceedings of the 4th WSEAS/IASME International Conference on Dynamical Systems and Control, CONTROL'08*, pages 20–24, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS).
- [176] Susnea, I., Vasiliu, G., Filipescu, A., Serbencu, A., and Radaschin, A. (2009). Virtual pheromones to control mobile robots. a neural network approach. In *2009 IEEE International Conference on Automation and Logistics*, pages 1962–1967.
- [177] Svestka, P. and Overmars, M. H. (1996). Coordinated path planning for multiple robots. Technical Report UU-CS-1996-43, Department of Information and Computing Sciences, Utrecht University.
- [178] Takahashi, A., Hongo, T., Ninomiya, Y., and Sugimoto, G. (1989). Local path planning and motion control for agv in positioning. In *Intelligent Robots and Systems '89. The Autonomous Mobile Robots and Its Applications. IROS '89. Proceedings., IEEE/RSJ International Workshop on*, pages 392–397.
- [179] Tan, Y. and yang Zheng, Z. (2013). Research advance in swarm robotics. *Defence Technology*, 9(1):18 – 39.
- [180] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.
- [181] Touhara, K. (2013). *Pheromone Signaling Methods and Protocols*. Humana Press.
- [182] TurtleBot 2 (2015). Turtlebot 2 robot. Available at <http://turtlebot.com/> [Accessed: 10-10-2015].
- [183] UHG-08LX Technical Specifications (2015). Uhg-08lx technical specifications. Available at <http://www.autonomoustuff.com/hokuyo-uhg-08lx.html> [Accessed: 2-12-2015].
- [184] URG-04LX Technical Specifications (2015). Urg-04lx technical specifications. Available at http://www.hokuyo-aut.jp/02sensor/07scanner/download/pdf/URG-04LX_spec_en.pdf.
- [185] Vandorpe, J., Van Brussel, H., and 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*, volume 1, pages 901–908 vol.1.
- [186] Vaz, P., Ferreira, R., Grossmann, V., and Ribeiro, M. (1997). Docking of a mobile platform based on infrared sensors. In *Industrial Electronics, 1997. ISIE '97., Proceedings of the IEEE International Symposium on*, volume 2, pages 735–740 vol.2.

- [187] Visp (2017). Visp: Visual servoing platform. Available at <https://visp.inria.fr/> [Accessed: 1-01-2017].
- [188] Waring, E. (2015). *Problems concerning Interpolations*. The Royal Society Publishing, London, U.K. Available at <http://rstl.royalsocietypublishing.org/content/69/59.full.pdf+html> [Accessed: 15-01-2016].
- [189] Weisstein, E. W. (2016a). Cornu spiral from mathworld—a wolfram web resource. Available at <http://mathworld.wolfram.com/CornuSpiral.html> [Accessed: 15-2-2016].
- [190] Weisstein, E. W. (2016b). Fresnel integrals from mathworld—a wolfram web resource. Available at <http://mathworld.wolfram.com/FresnelIntegrals.html> [Accessed: 25-2-2016].
- [191] Weisstein, E. W. (2016c). Hypocycloid from mathworld—a wolfram web resource. Available at <http://mathworld.wolfram.com/Hypocycloid.html> [Accessed: 3-3-2016].
- [192] Wikipedia (2016). Darpa robotics challenge. Available at https://en.wikipedia.org/wiki/DARPA_Robotics_Challenge [Accessed: 31-12-2016].
- [193] Wikipedia (2016a). Microsoft kinect. Available at <https://en.wikipedia.org/wiki/Kinect> [Accessed on 25-11-2016].
- [194] Wikipedia (2016b). Ransac. Available at <https://en.wikipedia.org/wiki/RANSAC> [Accessed on 25-03-2016].
- [195] Wilson, W. J., Hulls, C. C. W., and Bell, G. S. (1996). Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5):684–696.
- [196] Wu, C., Yang, Z., Liu, Y., and Xi, W. (2012). Will: Wireless indoor localization without site survey. In *INFOCOM, 2012 Proceedings IEEE*, pages 64–72.
- [197] Wu, Y.-C., Teng, M.-C., and Tsai, Y.-J. (2009). Robot docking station for automatic battery exchanging and charging. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pages 1043–1046.
- [198] Yang, D.-H. and Hong, S.-K. (2007). A roadmap construction algorithm for mobile robot path planning using skeleton maps. *Advanced Robotics*, 21(1):51–63.
- [199] Yang, K. and Sukkarieh, S. (2010). An analytical continuous-curvature path-smoothing algorithm. *Robotics, IEEE Transactions on*, 26(3):561–568.
- [200] Yi, J., Zhang, J., Song, D., and Jayasuriya, S. (2007). Imu-based localization and slip estimation for skid-steered mobile robots. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2845–2850.
- [201] Yokoyama, M., Matsubashi, Y., Sano, A., and Ko, M. T. (2015). Slip control of a wheeled mobile robot with a movable auxiliary mass. In *Advanced Intelligent Mechatronics (AIM), 2015 IEEE International Conference on*, pages 1008–1013.

-
- [202] Zaitsev, S. (2015). *Jsmn Open Source JSON C Parser*. Website. Available at <https://bitbucket.org/zserge/jsmn/wiki/Home> [Accessed: 9-10-2015].
- [203] ZakeriNejad, N., Bakhtiary, A. H., and Motlagh, M. R. J. (2010). Use of particle swarm optimization in path planning for rough terrain. In *Robotics in Alpe-Adria-Danube Region (RAAD), 2010 IEEE 19th International Workshop on*, pages 205–209.
- [204] Zhang, T. Y. and Suen, C. Y. (1984). A fast parallel algorithm for thinning digital patterns. *Commun. ACM*, 27(3):236–239.
- [205] Zhang, Y., Wang, S., and Ji, G. (2015). A comprehensive survey on particle swarm optimization algorithm and its applications. *J. Mathematical Problems in Engineering*, 2015(38).