COOPERATIVE LOCALIZATION AND MULTI-ROBOT EXPLORATION

Ioannis Rekleitis

School of Computer Science
McGill University, Montréal

January 2003

A Thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

© Ioannis Rekleitis, 2003

Abstract

This thesis has two main contributions. The first contribution is the use of cooperative localization for decoupling the positional error of a moving robot from its environment. The second contribution is the development of efficient multi-robot exploration strategies for an unknown environment.

The proposed method is designed to be robust in the face of arbitrarily large odometry errors or objects with poor reflectance characteristics. Central to the exploration strategy is a sensor (robot tracker) mounted on a robot that could track a second mobile robot and accurately report its relative position. Our exploration strategies use the robot tracker sensor to sweep areas of free space between stationary and moving robots and to generate a graph-based description of the environment. This graph is used to guide the exploration process. Depending on the size of the environment relative to the range of the robot tracker, different spatial decompositions are used: a triangulation or a trapezoidal decomposition of the free space. Complete exploration without any overlaps is guaranteed as a result of the guidance provided by the dual graph of the spatial decomposition of the environment.

The uncertainty in absolute robot positions and the resulting uncertainty in the map is reduced through the use of a probabilistic framework based on particle filtering (a Monte Carlo simulation technique). Particle filtering is a probabilistic sampling technique used to efficiently model complex probability distributions that cannot be effectively described using classical methods (such as Kalman filters).

We present experimental results from two different implementations of the robot tracker sensor, in simulated and in real environments. The accuracy of the resulting map increases with the use of cooperative localization. Furthermore, the deterioration of the floor conditions did not affect the quality of the map verifying the decoupling of positioning error from the environment.

Résumé

Cette thèse apporte essentiellement deux contribuitions. La première consiste en l'emploi de la méthode de localization coopérative, pour obtenir le découplage entre l'erreur de la position du robot en mouvement et ce de l'environnement. La deuxième comporte le développemnt de stratégies efficaces d'exploration multi-robot d'un environnement inconnu.

La méthode proposée est conçue de manière à être robuste face à des erreurs odométriques arbitrairement grandes ou à des objets ayant de mauvaises caractéristiques de réflectance. L'élément central de la stratégie d'exploration est un senseur (suiveur de robot) monté sur un robot qui puisse repérer un deuxième robot mobile et mesurer sa position avec précision. Notre stratégie d'exploration utilise le senseur pour balayer les régions d'espace vide entre le robot stationnaire et le robot mobile, avec quoi on obtient un graphe qui décrit l'environnement. Ce graphe est utilisé pour guider la procédure d'exploration. En fonction de la taille de l'environnement par rapport au champ du suiveur de robot, différentes décompositions spatiales sont utilisées: soit une triangulation, soit une décomposition trapézoïdale de l'espace vide. Une exploration complète et sans recouvrements est garantie par le guidage fourni par le graphe dual de la décomposition spatiale de l'environnement.

L'incertitude dans la position absolue des robots et l'incertitude résultante dans la cartographie sont réduites par l'utilisation d'une méthode probabilistique basée sur le filtrage de particules (une technique de simulation Monte Carlo). Le filtrage de particules est une technique d'échantillonnage probabilistique utilisée pour modéliser efficacement des distributions de probabilités complexes qui ne peuvent pas être décrites efficacement par les méthodes classiques (tel que les filtres de Kalman).

Nous présentons des résultats expérimentaux provenant de deux suiveurs de robot différents, dans des environnements réels et simulés. La précision de la cartographie obtenue augmente avec l'utilisation et la localisation coopérative. De plus, la détérioration de l'état du sol n'a pas affecté la qualité de la cartographie, confirmant le découplage entre l'erreur de position et l'environnement.

Acknowledgements

Throughout my Ph.D. research, I have been very fortunate in having the support and encouragement of many people. I would like to personally thank everyone who helped make this thesis possible. First of all I would like to thank my two supervisors Gregory Dudek and Evangelos Milios. The combination of the two provided me with excellent scientific guidance and technical advice. Their supervision has been of the highest caliber and guided me through both theoretical problems and practical difficulties.

A special thanks goes to the members of the Mobile Robotics Lab (MRL), past and present. They have created, each in their own way, a special place to work, learn and live. In particular, I would like to thank Eric Bourque for his sharp commentaries and his (always sought after) technical expertise. To Robert Sim I am grateful for his contributions to the RoboDaemon, our collaborations in research and for the countless interesting and stimulating conversations. Saul, Abril and Richie made MRL a fun place to be and always provided intelligent input in the group meetings. From the older generations a special thanks goes to Paul Mackenzie for the Rhys line fitting software and especially for its current incarnation ALMS.

I would like to thank the people at the Centre for Intelligent Machines (CIM) for their support and friendliness. Special thanks to the administration past and present especially M. Gray and C. Davidson and to the system staff Jan, Danny and Michael that maintained excellent computing facilities. Among the people of CIM the people from the Ambulatory Robotics Lab of M. Buhler provided tremendous help, support and friendship - a heart-felt thanks, guys.

Many thanks to the people from the school of Computer Science. Among them a special thanks goes to the graduate secretary D. Anastasopoulos for her help with the various forms and deadlines.

Special thanks also go to the faculty members that helped me during my research, the list includes but is not limited to Frank Ferrie, Doina Precup, Kaleem Siddiqi, Michael Langer and Luc Devroye.

I would like to thank L. Solomon for being with me and also for taking me out of the Christmas blues.

Most of all I am grateful to my family that supported me any time I was in need, always standing by me, guiding me into searching for answers and striving for a better tomorrow.

TABLE OF CONTENTS

Abstract i

Résumé ii

Acknowledgements iv

LIST OF FIGURES x

LIST OF TABLES xxii

CHAPTER 1. Introduction 1

1. Motivation 1
2. The relevance of maps 3
3. Contributions of the Work 7
4. Exploration with the Robot Tracker 8
5. Cooperative Localization 9
6. Thesis Outline 10

CHAPTER 2. Background 11

1. Mapping 11
2. Estimation Theory 13
3. Dead Reckoning 14
4. Localization 15
5. Multiple robots 15
6. Geometry Overview 17

CHAPTER 3. Exploration within Sensor Range (Triangulation) 19

1. Outline of the triangulation algorithm 19
2. Wall Following 25
3. Complexity Analysis 27
4. Illustrative Example 29
5. More than two robots 29

CHAPTER 4. Exploration Beyond Sensor Range (Trapezoidation) 31
1. Introduction 31
2. Top-Down description of the Algorithm 32
3. Complexity Analysis 33
3.1. Complexity of Global Exploration 35
3.2. Complexity of the exploration over a single exchange 35
4. More than two robots 37
4.1. Motion Strategies 37

CHAPTER 5. Uncertainty Reduction 40
1. Bayesian Reasoning 40
2. Particle Filter 42
2.1. Prediction 44
2.2. Resampling 47
2.3. Update 49

CHAPTER 6. Robot Tracker Implementation 57
1. Introduction 57
2. Visual Tracker 57
2.1. Tracker Evaluation 58
3. Laser Tracker 59
3.1. Laser Sensor Accuracy 61
3.2. Laser Sensor Calibration 62
3.3. Target Calibration 70
4. Laser Sensor based Localization 77

CHAPTER 7. Experimental Results 81

1. Trapezoidation Algorithm 81
1.1. Simulation Results 82
1.2. Laboratory 83
2. Triangulation Algorithm 83
2.1. Simulation 84
2.2. Exploration with the two Superscout robots 87
3. More than two robots 105
3.1. Cooperative Localization 105
3.2. Sensing Modalities 107
4. Discussion 111

CHAPTER 8. Collaborative Exploration for Visual Map Construction 113
1. Introduction 113
2. Motivation 115
3. Application: Landmark Learning 116
4. Experimental Results:
Construction of Landmark-based Visual Maps 119
4.1. Experiment 1: 120
4.2. Experiment 2 123

CHAPTER 9. Conclusions 128
1. Future Work 130
2. Final Words 132

REFERENCES 133

APPENDIX A. Proof of Optimal Coverage 152
1. Proof 152

APPENDIX B. Odometry Error Study 160
1. Introduction 160
2. Odometry Study of a Differential Drive Robot 161
2.1. Rotation 161
3. Translation 164

4. Odometry Error Modeling 166
4.1. Rotation 167
4.2. Translation 168

APPENDIX C. Resampling Methods 173
1. Select with Replacement 173
2. Linear time Resampling 174
3. Resampling by Liu *et al.* 175
4. Variations on Resampling 175
4.1. Corrective Resampling 175
4.2. Maintaining the variance of the distribution 175

LIST OF FIGURES

1.1 Two robots exploring the environment employing cooperative localization. 2

1.2 The map of North America by Jansson c. 1625 A.D. Note that California is an island and the north-western part is void of details. 5

1.3 The system consists of two stations mounted on tripods. One tripod is positioned on a fixed location and acts as a reference point. The second tripod is on wheels and carries two sensors, one localization sensor that is able to infer the position of the moving tripod relative to the reference point and a laser range finder that is used in order to map the distance to various objects (walls, corners, etc.). 6

1.4 Area covered when one robot (light grey-green) moves and the other one is stationary. 8

2.1 A polygonal environment with reflex vertices/corners and one obstacle. With dashed lines are marked internal diagonals that form a triangulation. 17

2.2 A polygonal environment with reflex vertices/corners and one obstacle. With dashed lines are marked internal rays that form a trapezoidal decomposition of the interior of the polygon. 18

3.1 Thick line represent walls, dashed lines represent unexplored walls, grey area is explored free space, dashed lines inside the grey are internal diagonals. The stationary robot is red, the moving robot is green, and the red circles connected by arrows at the center of the triangles represent the dual graph. *Line of Visual Contact interrupted*: (a) **Case**

1: The stationary robot is at a non-reflex vertex and the moving robot encounters a reflex vertex that would interrupt the line of visual contact (b) **Case 2:** Occluding Vertex between the two robots. 21

3.2 Same notation as in Figure 3.1. Line of Visual Contact interrupted: (a) **Case 3:** Both robots are placed at reflex vertex such that any further exploration would break the line of visual contact (b) The moving robot explores perpendicular to the diagonal that connects the two reflects vertices. At the first wall found a Steiner point is introduced and an internal triangle is created. (c) **Case 4:** Occluding Edge next to the stationary robot.(d) One branch of the dual graph is completely mapped, the robots would proceed to the nearest open edge of the dual graph. 22

3.3 Illustration of the wall following. (a) The closest wall is the same and the robot has not reached the wall's end: continue forward at 60cm from the wall. (b) A new wall is discovered which is closer than the previous one: reached non-reflex vertex; map the corner and continue the exploration of the new wall. (c) The closest wall ended and the robot has moved past the end of the closest wall: it is a reflex corner; proceed as in sub-figure (d). (d) Map the reflex corner by moving around the end-point of the old closest wall. Move by intervals of θ at a distance less than 60cm and more than 30cm from the end-point. Robots drawn with dashed lines represent past positions. 26

3.4 Two robots (R_1,R_2) are exploring a simple environment. Only one branch of the dual graph is explored, the remaining areas marked as **U**. Robot R_2 explores first. 28

3.5 Triangulation with four robots (a) All robots are aligned, R_1 is stationary and robot R_4 is following the wall. (b) Robot R_4 moves along the wall. (c) Robot R_3 aligns itself with R_1 and R_4. (d) R_2 moves into line. Robots drawn with dashed lines represent past positions. 30

4.1 A top down description of the Trapezoidation algorithm. (a) The environment is divided into trapezoids. (b) The order in which the trapezoids are mapped is given by a traversal of the *Dual Graph.* (c) Each trapezoid is further divided into stripes with a width proportional to the sensing range R. (d) Each stripe is covered by areas of free space one next to the other. Each area of free space is explored by the motion of a single robot. Different motion strategies can be used, and the size of the area is controlled by the angle θ. 34

4.2 (a) Exploration of a stripe with 5 robots. The robots move at time T_1,T_2,T_3, and T_4. (b) Exploration of a stripe with 3 robots, covering space in diamond areas. The robot move at time T_1-T_{19}. 38

5.1 Arbitrary motion $[\Delta x, \Delta y]^T$ of robot R_i. At time $t = k - 1$ the pose is $[x, y, \hat{\theta}]^T$, after the motion at time $t = k$ the pose is $[x', y', \hat{\theta}_k]^T$. The robot first rotates to orientation $\hat{\theta}_k$ and then translates by ρ_k. 45

5.2 The effect of $\sigma_{trs}, \sigma_{drft}$ for the forward translation: (a) $\sigma_{trs} = 5cm/m, \sigma_{drft} = 1^\circ/m$ (b) $\sigma_{trs} = 1cm/m, \sigma_{drft} = 5^\circ/m$. 47

5.3 (a) Large trajectory, the uncertainty build up is represented by the spread of the particle cloud. (b) Series of forward translations and 360° rotations performed in our laboratory. The connected curved line represent the uncorrected odometer values (captured accurately by the cloud of particles), and the bottom line represents the actual trajectory. 48

5.4 The stationary robot with the robot tracker sensor observes the moving robot that carries the target. 50

5.5 The contribution of each measurement of the robot tracker in the weighting *pdf* of the moving robot. 51

5.6 The contribution of each measurement of the robot tracker in the weighting *pdf* of the moving robot. In contrast to Figure 5.5 the $\sigma_{\hat{\theta}}$ is not calculated to be proportional to distance between the two robots. 53

5.7 Observation. 54

5.8 (a)*Prediction* of the first step. (b) *Update* using the robot tracker. (c) *Prediction* of the second step. (d) *Update* using the robot tracker. . . 56

6.1 Robot Tracker: (a) The raw image of the moving robot as observed by the robot tracker. (b) The helical and cylindrical pattern detected in the image. (c) The distances estimated from the helix. 58

6.2 (a) The visual robot tracker system with the camera mounted on one robot and the helical target pattern mounted on the second robot. (b) Calibration data for the distance estimation relating observed image position to actual distances. 59

6.3 (a)The two robots: the observed robot with white target on the left and the observing robot with the laser range finder on the right. (b) Top view of the observed robot 6 canonical views that are qualitative different depending on the position of the observing robot. 60

6.4 Top view of the observing/observed robots. For illustration purposes we examine the case where the two planes meet at a 140°, view 3 and view 6. (a) *View 3* 140 degrees angle between the two target lines; the observing robot and the intersection point are on opposite sides of the line that passes between the two end points of the target. (b) *View 6* Same angle (140 degrees) but the observing robot and the intersection point are in the same side. 61

6.5 An environment with four walls. 63

6.6 Error estimation after translation (a) Using all the data. (b) Without any outliers. 65

6.7 The four walls providing three landmarks. (a) Before the rotation. (b) After the rotation. 67

6.8 Error in the odometry based orientation estimate versus landmark based orientation estimates. Top sub-plot presents the standard deviation of the three estimates of the orientation using three pairs of landmarks("+") and the maximum difference ("*"). Bottom sub-plot presents the difference between the mean estimate of orientation based on the landmarks and the orientation reported from odometry. 68

6.9 Iterative least squares calculation of the X (top sub-plot) and Y (bottom sub-plot) by removing one outlier at the time. The accuracy in both cases is better than one millimeter. 69

6.10 The robot is observed from *View 3*: relation between the detected target ($T = [x_t, y_t, \theta_{t_1}, \theta_{t_2}]$) and the pose of the robot ($P_r = [x_r, y_r, \theta_r]$). . . 72

6.11 The observed robot translates along its heading (specified by θ_r) by a distance d, the center of the target translates also along the same heading (θ_r) by the same distance d. 72

6.12 (a) The robot rotates around its center (R) by an angle Θ, the fixed target rotates accordingly (from A to B). The solid lines starting from A represent the target before the rotation and the dashed lines starting from B represent the target after the rotation. (b) Two possible centers (R,R') on the opposite sides of AB line. 73

6.13 The trajectory of the observing robot with the laser range finder and the detected laser points. In the center stands the observed robot which is visible only in the form of the three lines of the target. Enough walls existed around the two robots in order to provide adequate landmarks during the calibration process. 76

6.14 The Robot Pose is estimated based on data from the face 3 (the ellipse indicates the uncertainty of the robot pose estimation. The intersection points for the different faces are displayed as follows: face 1 as "+", face 2 as "$\triangleright$", face 3 as "*", face 4 as "x", face 5 as "$\triangle$", face 6 as "o". . . 77

6.15 Zoom at the intersection points of the Target (from Figure 6.14); the ellipse around each point represents the uncertainty over ten measurements. 78

6.16 The Robot Pose is estimated based on the calibration table of Table 6.7 (the ellipse indicates the uncertainty of the robot pose estimation. The intersection points for the different faces are displayed as follows: face 1 as "+", face 2 as "$\triangleright$", face 3 as "*", face 4 as "x", face 5 as "$\triangle$", face 6 as "o". 78

6.17 Zoom at the intersection points of the Target (from Figure 6.16). Ten measurements were collected from each position. 79

6.18 Localizing from the observed lines. 80

7.1 Exploration of one stripe (120 exchanges). The results are from a single run. 82

7.2 (a) Path of the moving robot as estimated by the visual tracker. Measurements taken in different positions validate the accuracy with precision of roughly 2cm. (b) The desired path of the moving robot. Although the robot can be driven along this path using open-loop control, dead reckoning error leads to a substantial discrepancy. (c) The error in positioning from the odometry estimations. 83

7.3 The paths of the two robots after the completion of the exploration. . 84

7.4 First three rows: Exploring an unknown environment, figures b and e illustrate the trajectory of *Robot 0*. Figures a,c,d,f,g,h illustrate the trajectory of *Robot 1*. Finally the third column (Figures c,f,i) presents the map up to that point. Last row: Close-up on the build up of the uncertainty when only odometry was used. The solid line is the odometry based estimation of the robots while the dashed line is the real position of the robots (see text Section 7.1.1). 85

7.5 **Exploring an unknown environment with two occluding vertices:** The first column illustrates the trajectory of *Robot 0*. (a,d,g,j,m). The second column illustrates the trajectory of *Robot 1* (b,e,h,k,n). Finally the third column presents the **map** up to that point (c,f,i,l,o). (See text Section 7.1.1). 88

7.6 Exploration inside two adjacent labs at the fourth floor of our building (a) Beginning of the exploration *Robot 1* is stationary at a reflex corner and *Robot 0* explores a wall across from it. (b) *Robot 0* has mapped inside of the lab and moves along the corridor towards the exit. (c) *Robot 1* starts exploring coming through the corridor. *Robot 0* is positioned at a reflex corner. (d) The last part of the exploration, *Robot 1* (on the right) is mapping the wall across from *Robot 0* that is placed at a reflex corner and provides corrections for *Robot's 1* pose estimation. 89

7.7 The exploration of the two laboratories. Red lines are the walls, blue dashed lines are diagonals, green dotted lines are gates to unexplored space. 90

7.8 (a) The laser data collected during the exploration. (b) Scan match applied to the laser data and their position corrected using Stephen Gutmann's scanstudio. (c) The triangulation map produced using only the sonar for mapping. The trajectory of *Robot 0* is marked in magenta and the trajectory of *Robot 1* is marked black. The walls are displayed in red and their lengths (in cm) is marked next to them. The internal diagonals that define the triangulation are marked as blue dashed lines. 91

7.9 Sonar data collected from *Robot 0* (on the left) and *Robot 1* (on the right), filtered at different ranges: (a,b) 65cm; (c,d) 120cm; (e,f) 250cm; (g,h) 400cm. The sonar points are marked blue and the path of the robot is marked red. 93

7.10 (a)The odometry uncertainty growth during the trajectory of *Robot 0*. (b) The odometry uncertainty growth during the trajectory of *Robot 1*. 94

7.11 (a)The cooperative localization estimates (*) versus the trajectory resulting from the motion commands (+) given to *Robot 0* during the exploration. (b) The same for *Robot 1*. 94

7.12 (a) The prediction phase of the particles for the trajectory of *Robot 0*. (b) The update phase of the particles for the trajectory of *Robot 0*. (c),(d) Prediction and update phases for *Robot 1*. 95

7.13 Exploration at the sixth floor of our building (a) *Robot 0* maps a reflex corner. (b) *Robot 0* follows a wall (c) *Robot 0* reached a reflex corner that interrupts line of sight. (d) *Robot 1* starts exploring after a roles exchange. 97

7.14 The exploration hallways in the 6th floor. Red lines are the walls, blue dashed lines are diagonals, green dotted lines are gates to unexplored space. 98

7.15 Sonar data collected from *Robot 0* (on the left) and *Robot 1* (on the right), filtered at different ranges: (a,b) 65cm; (c,d) 120cm; (e,f) 250cm; (g,h) 400cm. The sonar points are marked blue and the trajectory for the robot is marked red. 99

7.16 (a) The laser data collected during the exploration. (b) Scan match applied to the laser data and their position corrected using Stephen Gutmann's scanstudio. 100

7.17 The triangulation map produced using only the sonar for mapping. The trajectory of *Robot 0* is marked in magenta and the trajectory of *Robot 1* is marked black. The walls are displayed in red and their lengths (in cm) is marked next to them. The internal diagonals that define the triangulation are marked as blue dashed lines. 101

7.18 (a) The cooperative localization estimates (*) versus the trajectory resulting from the motion commands (+) given to *Robot 0* during the exploration. (b) The same for *Robot 1*. 101

7.19 (a) The prediction phase of the particles for the trajectory of *Robot 0*. (b) The update phase of the particles for the trajectory of *Robot 0*. (c),(d) Prediction and update phases for *Robot 1*. The walls of the environment are indicated in red. 102

7.20 The exploration of the 6th floor with a small modification. Red lines are the walls, blue dashed lines are diagonals, green dotted lines are gates to unexplored space. 103

7.21 (a) The laser data collected during the exploration. (b) Scan match applied to the laser data and their position corrected using Stephen Gutmann's scanstudio. (c) The triangulation map produced using only the sonar for mapping. 104

7.22 The triangulation map produced using only the sonar for mapping. The trajectory of *Robot 0* is marked in magenta and the trajectory of *Robot 1* is marked black. The walls are displayed in red and their lengths (in cm) is marked next to them. The internal diagonals that define the triangulation are marked as blue dashed lines. 104

7.23 (a) The average error during the exploration of 50 triangles (over 100 experiments) without and with cooperative localization. (b)The path of the robot after the completion of the exploration. The outside solid line marks the position of the walls the moving robot followed. The actual path of the robot is the solid line, the odometry based estimate of position is the dotted line, while the tracker estimate is the dashed-dotted line. 106

7.24 (a) Estimation of the pose of robot R2 using only the distance from robot R1 (d1) and from robot R3 (d3). (b) Average error in position estimation using the distance between the robots only (3,4 and 10 robots; bars indicate standard deviation). 107

7.25 Average error in position estimation using the orientation of the moving robot is seen by the stationary ones. 108

7.26 The *pdf* of the moving robot (R2) at different phases of its estimation: (a) prediction using odometry only; (b) using the orientation from stationary robot R1; (c) using the orientation from stationary robot R3; (d) final *pdf*. 109

7.27 Average error in position estimation using both the distance between the robots and the orientation the moving robot is seen by the stationary ones. (a) Average error in positioning of the team of robots one trial (3,5 and 10 robots). (b) Average error in position estimation over twenty trials (3,5, 10 and 40 robots). 110

7.28 Average error in position estimation using full pose $[\rho, \theta, \phi]$. 111

8.1 Collaborative explorers 114

8.2 Mapping: **(a)** Continuous function such as: Radiation, Visual appearance, Elevation, Magnetic field, Temperature, etc. **(b)** Discrete function such as: Mine detection, Lost objects, Holes, Electrical outlets, etc. 115

8.3 The off-line training method. Images (large rectangles) are collected sampling the pose space. Landmarks are extracted from the images and matched across the samples. The *tracked landmarks* are parameterized as a function of pose and saved for future pose estimation. Figure courtesy of R. Sim. 117

8.4 The likelihood of an observation as a function of pose. Figure courtesy of R. Sim. 118

8.5 Pose estimation based on learned visual landmarks. Landmarks (small squares) are extracted from the current camera observation and matched to the previously learned tracked landmarks. Each match generates a pose estimate, which are filtered and combined to generate a final pose estimate. Figure courtesy of R. Sim. 120

8.6 Views of the two "rooms" as seen by the robot, and the floor plan of the two "rooms". 121

8.7 Odometric (x) vs Tracker-corrected (o) trajectories of the robot. . . . 121

8.8 Odometric error versus distance traveled. 122

8.9 Tracker estimates (o) vs Vision-based estimates (x) for training images. 122

8.10 Tracker estimates (o) vs Image-based estimates (x) for a set of 21 random positions. 123

8.11 Opposing views of the lab as seen by the exploring robot. 124

8.12 In this experiment the robot took pictures in four orientation; the higher number of rotations increased non-linearly the odometric error. (a) Odometric (denoted by dash-dotted line) vs Tracker-corrected (denoted by a solid line) trajectories of the robot. (b) Odometric error versus distance traveled. 124

8.13 Tracker estimates (o) vs Image-based estimates (x) for a set of 93 random positions. 126

8.14 The trajectory of the moving robot based on odometry estimates (triangles connected with a dashed line), the robot tracker cooperative localization ('+' connected with a solid line) and the image based localization ('o' connected with a dash-dotted line). 126

A.1 Equivalence among two pairs of tiles. 153

A.2 Shortest path along the arc connections. 153

A.3 Sequence of adjacent tiles. Dark lines define the wedges and each wedge is defined by three letters. Dashed lines define the sensor range boundary. 154

A.4 Graph of ΔA for $0 \leq \theta_i \leq 180°$, $\Delta A \geq 0$ for any pair of angles 156

A.5 Positioning of the wedge stripes. 158

A.6 Optimal tiling. 159

B.1 Measuring the odometry error on carpet. 161

B.2 The four walls providing three landmarks. (a) Before the rotation. (b) After the rotation. 162

B.3 Error in rotation relative to the odometer for different angles and for different speeds ("o" speed 10, "x" speed 50, "+" speed 90, lines connect the mean values). 163

B.4 Error in rotation relative to the intended pose for different angles and for different speeds (as in Figure B.3) 164

B.5 Error distribution from the odometry measurement for different surfaces (rotation of 90°). 165

B.6 Error distribution from the intended pose for different surfaces (rotation of 90°). 166

B.7 Error distribution after translation of 100cm. Tile floor, 165 samples. 167

B.8 Error distribution after translation of 120cm. Plastic surface, 43 samples. 168

B.9 Error distribution after translation of 120cm. Carpet surface, 43 samples. 169

B.10 One step in translation. 169

B.11 The standard deviations of 30000 particles as they move along the x axis for 100cm using different number of steps each time. The experiment is repeated 100 times. 172

LIST OF TABLES

4.1 Analytical complexity of two different path curves. 36

6.1 Accuracy of simple visual tracker 59

6.2 Maximum Difference and standard deviation of observing different sized landmarks. Distance is measured in centimeters and the angles in degrees. 62

6.3 Orientation estimation with two different methods (analytical and least squares estimation). For the analytical calculation the mean value and the standard deviation is reported. The estimates contain all data collected (no outliers removed). 66

6.4 The same as Table 6.3 but with the outliers removed. 66

6.5 Estimated values for x and y (the position of the laser in robot coordinates) using an analytical solution and a least squares solution. 70

6.6 The estimated five parameters for the target calibration. Moving the observed robot (N/M: Not Measured). 75

6.7 The estimated five parameters for the target calibration. The calculations are all relative to face 3 and present average values. 75

6.8 The estimated five parameters for the target calibration. Measuring the geometry of the body of the observed robot. 76

7.1 The length of the walls measured with tape and from the triangulation map (first two columns) together with the error. See Figure 2.2.1c for the correspondence between walls and lengths. 96

7.2 The mean error in position estimation after 40m travel over 20 trials. 111

B.1 Mean error and Standard Deviation along the X,Y-axis (in cm) and orientation Θ (in degrees) after the translation of 100cm for three different speeds. 165

When you set out on your journey to Ithaca,
pray that the road is long,
full of adventure, full of knowledge ...
–From the poem Ithaca by C. Cavafis 1911

I would like to dedicate this thesis to my family.

CHAPTER 1

Introduction

I am told there are people who do not care for maps, and find it hard to believe. The names, the shapes... the courses of the roads and rivers... are an inexhaustible fund of interest for any man with eyes to see or two pence worth of imagination to understand with...
– Attributed to Robert Louis Stevenson in Treasure Island, 1883.

1. Motivation

In mobile robotics there are three fundamental problems. The first problem can be described by the simple question "*Where am I?*" and refers to establishing the pose[1] of the robot in a global frame of reference; commonly called the *localization problem*. The second problem is expressed by the question "*How do I go from point A to point B?*" and is called the *path planning problem* [**92**]. Finally the third problem can be encapsulated by the question "*What does the world look like?*" commonly referred to as *mapping*. The localization problem is critical because an error in the estimated position would surely result into erroneous or even hazardous behavior, such as moving to the wrong place, collisions, performing a task at the wrong location. Clearly the first and third problems are interrelated because when an accurate description (map) of the environment exists then the robot can localize itself by matching its observations to the world model; also, if the robot knows its pose with high accuracy, then the observed features of the world can be combined seamlessly into a map. The process of trying to solve both problems at the same time is known as

[1]The *pose* of the robot is defined in $2D$ as the triplet $< x, y, \theta >$ where x and y are the coordinates of the robot on the plane and θ its orientation.

Simultaneous Localization And Mapping (SLAM) or *Concurrent Localization and Mapping* (CLM).

FIGURE 1.1. Two robots exploring the environment employing cooperative localization.

This thesis deals with the construction of accurate metric maps of an unknown environment by a team of mobile robots. By using multiple mobile robots (like the ones in Figure 1.1) we can achieve a high level of accuracy and robustness. The robots are equipped with different sensors that allow them to collect measurements describing both the environment and the robot pose; these measurements are invariably corrupted by noise. The main contribution of this thesis is a methodology for reducing the uncertainty introduced by noise and for recording the pose of the moving robots and the position of obstacles as accurately as possible. In order to achieve this goal the mobile robots must cooperate closely with each other (Figure 1.1).

Uncertainty is a central issue in perception for mobile robots. During exploration there are two different sources of uncertainty. First, there is uncertainty in the pose of the robot. Second, the sensors used to model the free and occupied space of the environment return measurements that are corrupted by noise. In particular, the pose of the robot can be estimated either by dead reckoning[2] or by using external reference points, but either estimation approach is corrupted by noise.

[2]To paraphrase Dunlap and Shufeldt, *dead reckoning* is a simple mathematical procedure for determining the present location of a vehicle by advancing some previous position through known course and velocity information over a given length of time [**54, 16, 47**].

This thesis provides algorithms that would guide a group of mobile robots (equipped with noisy sensors) to construct an accurate (up to a certain bound) representation of the environment. Central to this thesis is the use of a new sensor (which we refer to as the *Robot Tracker*) with the dual purpose of localization and mapping. At any time, one of the robots remains stationary while the other robot is moving. The stationary robot acts as an artificial landmark in order for the moving robot to measure its pose with respect to it. Therefore, a detectable landmark in the form of the other robot is provided without any modification of the environment. We call this approach, based on the use of the robot tracker, *Cooperative Localization*. We first developed this concept in 1997 [**132**] and coined this term in 1998 [**139**], which has more recently been taken up by other authors [**144, 172, 9, 59, 64, 161, 108, 58, 182**]. A related system was developed by Kurazume *et al.* in 1994 [**89, 88, 87**]. Moreover, when one robot is moving and maintaining an uninterrupted line of visual contact with the stationary robot, it effectively maps the area swept by the line of visual contact. As the two robots move through the environment, they map areas of free space using the fact that they are constantly able to *see* each other. Our sensing strategy is sufficiently robust to cope with environments that may have uneven or slippery terrains, or whose surface reflectance properties are not well suited to conventional sensors.

2. The relevance of maps

Across human history, the exploration of new areas is accompanied by the construction of representations (maps) that describe them. The act of mapping goes back to 2000–3000 B.C. in Mesopotamia. The earliest known map, found near Kirkuk (modern Iraq), dates to 2400–2200 B.C. [**22**]. Around the same time, in China nine copper or bronze vases were made bearing representations of the nine provinces under the Hsia dynasty [**6**]. Strabo's works "Geography" (around 25 B.C.) and Claudius Ptolemy's "Syntaxis" (around 90-168 A.D.) [**129, 128**], both from Alexandria, remained the most important cartographic guides in the western world for the next 1500 years. The creation of maps flourished with the exploration of the world, especially in the last 500 years. Maps have been used for exploration of new areas, navigation in unknown waters, searching for shipwrecks, even for fun [**126**]. From crude signs on a piece of stone to detailed topographical maps and three dimensional models of areas, it was a long journey. Among different civilizations different models/representations were used: interweaved branches representing sea routes in Marshall Islands, Micronesia; carved wooden charts mapping the coast line of Greenland (by the Inuit); clay tables in Mesopotamia; bronze or copper vases showing mountains, rivers and local products in China; up to the more common paper or cloth maps and the globes representing geographical, political, economical and cultural data [**6**].

The ability to build an internal representation of the environment is critical to most intelligent organisms. Experiments suggest multiple systems for internal representations both in humans and in animals. In mammals, experiments have shown that areas in the hippocampus are directly related to spatial relationships, and certain locations trigger consistent responses of particular brain cells [**119, 118, 120**]. The existence of a cognitive map in the brain that spatially corresponds to the environment has been shown in experiments in rodents, in which dilation of the environment results in stretching the pattern of the neural responses in the hippocampus [**115**]. Even though there is some criticism [**11**] of the importance and uniqueness of the cognitive maps in the brain, it is generally accepted that an internal model exists that corresponds to landmarks and to dead reckoning estimates [**62**]. Studies in human perception of places suggest that at an early stage a rough model is built with a $2\frac{1}{2}$D sketch [**109**] as input and then this model is used to construct higher abstract models necessary for tasks that need spatial orientation [**180**]. Experiments with artificial neural networks confirm the role of an internal representation (cognitive map) in the tasks of navigation and perception [**146**].

In robotics, building and/or using an internal model of the environment (a map), is as we saw earlier, one of the main problems in the field. It is nearly impossible for a robot to operate in an environment if it does not have a model to guide its actions. One approach for robot operation is to construct a minimal architecture that would react to the environment and operate under a small set of assumptions [**21, 70**]. Such an approach does not require an explicit internal representation of the environment, but is unfortunately rather limited in the tasks it could perform (e.g., learn that a specific route is blocked, compute optimal path, etc.). Another approach in mobile robotics is to provide the robot with a complete map of the environment at the beginning of operation. Unfortunately, in most applications accurate maps do not exist and, in the few places that such maps (such as blueprints) exist and are error free [3], they are usually in a format that is not easily transferable to the mobile robot. Furthermore, over time, even accurate maps became obsolete. Thus, it is not surprising that the need for a mobile robot to construct a accurate map of an unknown world as it moves through it became apparent in the early stages of mobile robotics. The first attempts were purely theoretical and performed well in an idealized environment with no uncertainty; however, when these methods were transferred to the real world, they displayed the same shortcomings as the early mapping attempts of the human explorers.

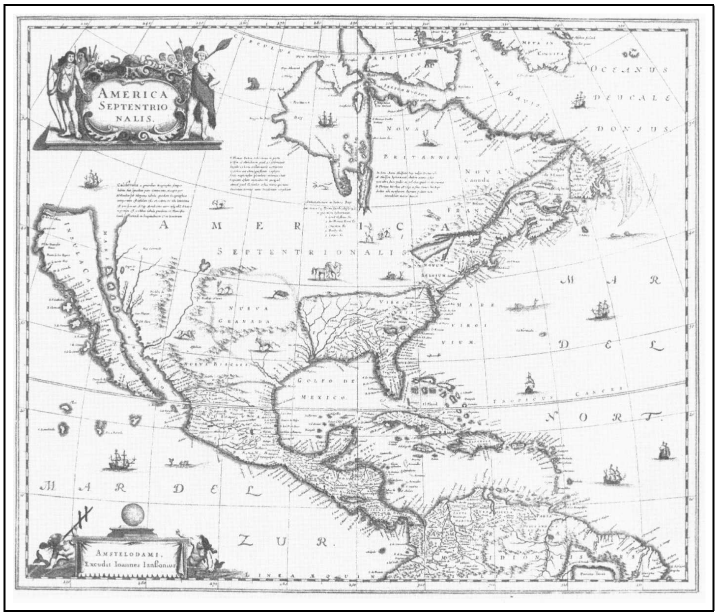


FIGURE 1.2. The map of North America by Jansson c. 1625 A.D. Note that California is an island and the north-western part is void of details.

When someone observes a map constructed by medieval cartographers (see for example Figure 1.2) two things are apparent: first, certain areas are mapped accurately and in great detail while others stay blank (sometimes filled with drawings of various interesting figures). Second, although the map may be topologically (i.e. qualitatively) correct, the distances between places are wrong. Both observations are easily explained by the methods that were used to construct the maps. Each map was usually compiled by one cartographer who had access to a limited number of locations, and had an approximate knowledge for the distances and the angles between these locations. The task of constructing a general map was only assisted by the ability to observe in the sky a set of common landmarks that could

[3] Architectural blueprints are rarely error free.

provide (with a small error) a common frame of reference [**129, 128**]. When surveying was developed, teams of people were able to map the world accurately, establishing complete models for vast areas.

In mobile robotics, when the area to be mapped is small the performance of simple algorithms is satisfactory, but when the robots have to traverse larger areas, the estimates of their position invariably become corrupted by the odometry error they accumulate. Thus, a single mobile agent can construct relatively accurate models of its immediate vicinity, but when it tries to place them in a general frame of reference the relationship among different locations and the distances among them are badly miscalculated. When multiple agents are sent out to explore independently, the task for combining the partial maps into a common reference becomes nearly impossible. The proposed methodology in this thesis addresses the above problems and provides a solution using intelligent cooperation among multiple mobile agents.

FIGURE 1.3. The system consists of two stations mounted on tripods. One tripod is positioned on a fixed location and acts as a reference point. The second tripod is on wheels and carries two sensors, one localization sensor that is able to infer the position of the moving tripod relative to the reference point and a laser range finder that is used in order to map the distance to various objects (walls, corners, etc.).

To exemplify the relevance of our approach, a surveying company called LaserCad[4] that constructs CAD models of buildings using aspects of Cooperative Localization in their approach (see Figure 1.3 for the system in action in McGill University). In particular they deploy two stations (manually), one stays fixed providing a landmark (equivalent to our stationary robot) and the second station is moved around and used to collect measurements with a laser range finder. After an area is surveyed they keep the station with the laser range finder fixed and they move the previously stationary station to a new position in order to provide a new landmark. In contrast to our approach they do not use both stations to collect measurements in the environment and they do not use the line of visual contact to sweep/map the space.

3. Contributions of the Work

The primary contribution of this thesis is the cooperative localization which enables the robots to map the free space and to decouple the positional error from the environment. It is the first time when the ability of two (or more) robots to observe each other is exploited in order to infer about the occupancy of the space between them. Another important contribution is the adaptation of a general probabilistic framework (in the form of a Monte-Carlo simulation) for the reduction of uncertainty both in the pose estimation and the resulting map in the multi-robot paradigm.

A non-comprehensive list of the contributions reported in this thesis follows:

1. The concept of *Cooperative Localization* [**132, 136, 135, 137**].
2. The concept of mapping free space by "sweeping" the line of visual contact [**132, 136, 135, 137**].
3. The development of an exploration algorithm (based on the triangulation of free space) for mapping areas of bounded size [**132, 135, 137**].
4. The development of an exploration algorithm (based on the trapezoidal decomposition of free space) for mapping unbounded areas. [**132, 136, 137**].
5. The proof of the optimal motion strategy for covering free unbounded space (in the asymptotic case) [**137**].

[4]Please refer to http://www.lasercad.qc.ca/ for more information.

6. The adaptation, development and analysis of a general probabilistic framework (in the form of a Monte-Carlo simulation) for the reduction of uncertainty both in the pose estimation and the resulting map for a multi-robot case.
7. The development and construction of an accurate robot tracker sensor [**140, 138**].
8. The introduction of a new methodology in the multi-robot field for estimating the bounds of the accumulated uncertainty based on the statistical properties of the robot tracker sensor and the number of robots [**140, 138**].
9. Experimental results indicating feasibility and performance of our approach in simulation and in the laboratory.
10. The development of a new strategy for accurately mapping a spatially varying property of interest over an unknown (possibly hazardous) environment [**134, 133**].
11. Development of software (more than 20K lines of code in C++) to execute the proposed algorithms.

4. Exploration with the Robot Tracker

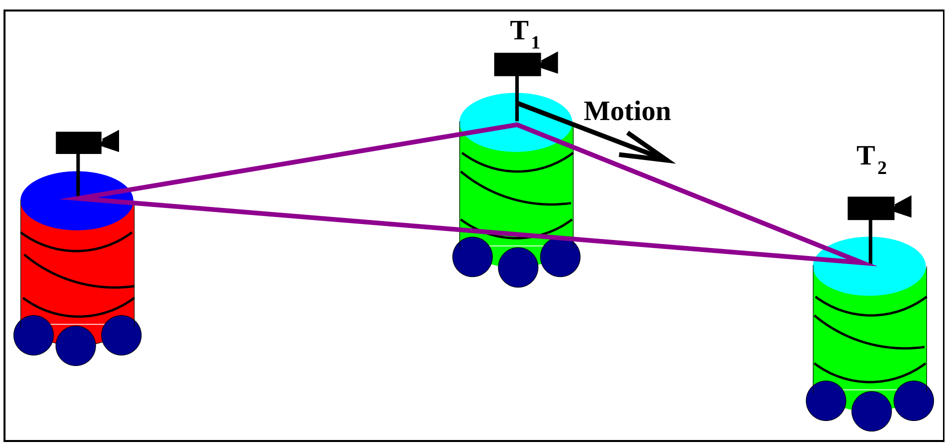


FIGURE 1.4. Area covered when one robot (light grey-green) moves and the other one is stationary.

As mentioned earlier, the use of a novel sensing modality (the robot tracker) allows the robots to map large areas of open space with limited uncertainty. Our mapping strategy exploits standard sensing technology in a novel way. Different sensors have been used in order to realize the robot tracker; an early implementation employed a camera on the observing robot and a spiral pattern target on the observed robot and provided an inexpensive solution; higher accuracy is achieved with the current implementation of a laser range finder and a three plane target (see Chapter 6 Sections 2 and 3 for details). The main idea in the exploration strategy is as follows. The two robots maintain an uninterrupted line of visual contact between them. When the moving robot proceeds along a trajectory, the line of visual contact sweeps a wedge defined by the lines connecting the stationary robot position to the initial and final positions of the moving robot (see Figure 1.4) and the trajectory of the moving robot. If an obstacle obstructs the line of visual contact, the moving robot backtracks and then proceeds to map around the interfering obstacle. This permits the robots to measure objects with reflectance properties that would be unmanageable with traditional sensors (like laser range finders). The order in which the robots move and exchange roles is determined by the environment (see the two algorithms in Chapter 3 and Chapter 4).

5. Cooperative Localization

Since sensing is being used to correct pose estimation errors, the determining source of error in the localization of the robots is the inaccuracy of the "robot tracker" sensor that is used to update/correct the position of the moving robot relative to the position of the stationary one. Therefore, if the two robots start with one stationary robot in an initial position $\hat{X}_{stat}(0)$ then the moving robot could localize itself with respect to that position, (see Figure 1.1). Note that, in practice, information from both sensing and odometry is combined using a probabilistic methodology.

There are three potential sources of information for the localization of the moving robot. First, the odometry measurements $\hat{X}_{odom}(t)$ provide a base estimate of the moving robot's position (with high uncertainty σ_o). Second, the different objects in the environment, when sensed from different positions, could provide updates to the robot position [**107, 167**]. Finally, the robot tracker provides pose measurements $\hat{X}_{track}(t)$ relative to the position of the stationary robot $\hat{X}_{stat}(t)$. Our approach utilizes the information from all three sources. In practice, over large scale environments, the position of (movable) objects changes over time and they cannot provide safe position updates. On the other hand, the estimate of the robot tracker is influenced only by the uncertainty in the position of the stationary robot

σ_s plus the error of the tracker measurement $\hat{X}_{track}(t)$. The accumulation of uncertainty in the position of the stationary robot depends only on the number of role exchanges the two robots had. Consequently, over large open spaces, where the odometry error grows without bound, the moving robot could always make reference to a stationary landmark (a role that is played by the second robot). Under certain assumptions regarding the noise functions we can optimally combine two information sources by weighting them as a function of their standard deviations. This is, in fact, the essence of the Kalman filter (an optimal estimator under appropriate conditions) [**63, 19**].

6. Thesis Outline

Chapter 2, which follows immediately, presents the relevant work and sets the framework in which our work is situated. The next two chapters contain two deterministic exploration algorithms for large areas; Chapter 3 presents the triangulation exploration algorithm for bounded areas and Chapter 4 introduces the exploration algorithm (trapezoidation) for unbounded areas based on the trapezoidal decomposition. Probabilistic reasoning and our approach to uncertainty reduction is presented in Chapter 5. Practical aspects of our work such as technical specification of the robot tracker sensor, calibration procedures, and programming methodology are described in Chapter 6. Chapter 7 presents experimental results both from laboratory experiments and from simulations. The results collected in the real world validate our approach, while simulated environments are used in order to examine in depth different aspects of our methodology. A different application of our methodology of cooperative localization is presented in Chapter 8. The visual map construction algorithm is used to examine the effectiveness of cooperative localization with very promising results. Finally, Chapter 9 presents conclusions and areas for future research. Appendix A contains the optimality proof of the trapezoidation algorithm with respect to distance traveled. Appendix B presents a study of the odometry error properties of the robots in our laboratory together with a comprehensive noise model and Appendix C contains a discussion on different resampling algorithms for the particle filter employed for uncertainty reduction.

CHAPTER 2

Background

In this chapter we examine relevant background for our work. It is worth noting that the subjects of localization and mapping cover a large fraction of the research in mobile robotics thus making a comprehensive survey of all the available work outside the scope of this thesis. Furthermore, a non comprehensive reference list [**171, 47, 16**] of work on multi-robot systems contains more than four hundred references. We present relevant work on exploration and mapping in Section 1, then a brief overview on estimation theory is presented in Section 2. Section 3 discusses the work on odometric error estimation and dead reckoning, and Section 4 presents an overview on localization. Section 5 examines relevant work on multi-robot systems. Finally, Section 6 contains basic definitions from computational geometry relevant to this thesis.

1. Mapping

Mapping via exploration is a fundamental problem in mobile robotics. The different approaches to mapping can be broadly divided into two categories: theoretical approaches that assume idealized robots and environments without uncertainty, and practical approaches that contend with issues of a real environment, often at the expense of theoretical rigor. The theoretical approaches provide lower bounds for the exploration problem while the practical approaches produce algorithms that operate in environments under uncertainty. Many algorithms have been proposed that explore the interior of a polygon or a collection of polygons under the assumption of perfect sensing and dead reckoning: the resulting map consists of a collection of linked lines. Representative of the above approach is the family of bug algorithms [**105, 104**]. Each obstacle in the environment is explored in turn by the mobile agent that circumnavigates it and then the agent moves to the next closest unexplored obstacle, the algorithm guarantees convergence. Other techniques [**131, 121**], also assume a polygonal world, which the robot maps by traversing the visibility graph ensuring every part of the polygon is visited. At the same geometric level of abstraction (a polygonal world) Choset proposed the use of Voronoi diagrams as a guide for an exploration strategy that navigated through the environment keeping a maximum distance from the obstacles [**33, 31**].

At a higher level of abstraction graph models for the world have been used for studying the task of exploration given minimal information for the environment. An abstract robot can travel across a graph by traversing edges and inquiring only about local information of the current vertex, such as the degree of the vertex or the existence of a marker on the current vertex. The robot can also carry the marker from one vertex to another. Algorithms exist for the exploration of a graph like world with a single abstract robot equipped with one marker [**49, 48**]. Extensions to the previous algorithms have been proposed where the robot has no marking ability and it relies only on the structure of the graph for the map construction [**44, 45, 141**]. In geometric worlds idealized models are used that deal with the world at a purely topological level [**40, 85**]. The resulting map is a graph where the vertices denote locations deemed to be more significant than average (such as corners, corridor intersections, etc). Using an automaton Pierce and Kuipers developed a learning algorithm that could learn not only the environment but an abstract model of its sensorimotor system [**125**]. Topological maps can be learned even with little available information [**148**].

Taking into account sensor uncertainty leads to a different approach to mapping. Several approaches centered on the exploration of an unknown world using a single sensor such as vision, sonar or a laser range finder [**20, 55, 10, 177, 90**]. Subsequently, data from different sensors were fused into a map in order to improve the efficiency and the accuracy of the map [**46, 3, 175, 23**]. Thrun *et al.* proposed an approach combining an occupancy grid with a topological map in order to construct a reliable map for a mobile robot exploring an office like environment. This approach is based computationally on a Partially Observable Markov Model [**167, 166**]. Thrun *et al.* extended their work to creating three dimensional maps [**99**] and to multi-robot mapping [**165**]. Durant-Whyte *et al.* pioneered the application of Kalman filters and target-tracking methods to the problem of robot localization [**95**] and introduced the methodology of simultaneous localization and mapping [**179, 41**]. Leonard *et al.* also have worked on mapping especially with out-door mobile robots and amphibian robots [**94, 93, 96**]. Mapping unstructured environments presents new challenges [**176**], especially for fast-moving, car-like vehicles [**82**]. One of the problems during mapping is the constant drift during the exploration that distorts the map [**69**].

Different forms of mapping have been proposed by Choset *et al.*, such as covering the environment optimally using the Boustrophedon Cellular Decomposition [**32, 2, 1**], with multiple robots [**91**], and also using the extended Voronoi diagram in mapping [**30**]. Furthermore research in mapping [**7, 84, 147**] includes the construction of visual maps [**76**], mapping in domestic environments [**183**], coverage in rectilinear environments [**25**] and in extreme conditions [**164**].

2. Estimation Theory

During the exploration of the unknown environment, the robots maintain a set of hypotheses with regard to their position and the position of the different objects around them. The input for updating these beliefs comes from the various sensors the robots poses. An " optimal estimator" [**63**] can be employed in order for the mobile robots to update their beliefs as accurately as possible. More precisely, the position of an obstacle observed in the past can be updated every time more data become available (a process called smoothing). Moreover, after an action, the estimate of the pose of the robot can be updated based on the data collected up to that point in time (a process called filtering).

Kalman filtering [**63, 19, 114**] is a standard approach for reducing the error, in a least squares sense, in measurements from different sources. In particular, in mobile robotics, Smith, Self and Cheeseman provided a framework for estimating the statistical properties of the error in robot positioning given different sets of sensor data [**159, 160**]. A variation is based on Extended Kalman filtering (EKF), where a nonlinear model of the motion and measurement equations is used [**95, 35**]. Roumeliotis *et al.* successfully employed Extended Kalman Filter in a variety of tasks such as localization and multi-robot mapping [**143, 142, 144**]. Kurazume *et al.* proposed the use of multiple robots, equipped with a sophisticated laser range finder, in order to localize, using some of them as movable landmarks [**89, 88, 87**]. The team of mobile robots uses a swarm behavior, using each other for localization. The fact that two robots could see each other was not used to infer that the space between them was empty.

One approach that has gained popularity lately falls under the category of Monte Carlo Simulation (see Doucet *et al.* [**43**] for an overview) and is known under different names in different fields. The technique we use was introduced as particle filtering by Gordon *et al.* [**66**] for tracking a moving target. In mobile robotics particle filtering has been applied successfully by different groups for single robots [**37, 38, 80, 173**], or for multiple robots [**39**], during navigation for online localization and for localization with a uniform prior (solving the kidnaped robot problem) [**168**], but also during exploration and mapping [**79**]. In vision this technique was introduced under the name of condensation [**77**] and particle filtering [**14**] for the estimation of optical flow in image sequences [**78**] and for tracking multiple moving objects in video sequences [**106, 163**].

3. Dead Reckoning

Dead reckoning is the procedure of modeling the pose (position and heading) of a robot by updating an ongoing pose estimate through some internal measures of velocity, acceleration and time [**17, 47**]. In most mobile robots this is achieved with the use of optical encoders on the wheels and is called odometric estimation. The estimate of the pose of the robot is usually corrupted with errors resulting from conditions such as: unequal wheel diameters, misalignment of wheels, finite encoder resolution (both space and time), wheel-slippage, travel over uneven surfaces [**17**]. The process of correcting the pose estimate is referred to as *localization*.

Borenstein and Fend in numerous studies present an analysis of the mechanical/kinematic causes of odometry error. Furthermore, they proposes a standard test (*UMB*test) for the estimation of systematic error [**18**]. Chong and Kleeman [**29**] use the *UMB*test for the elimination of systematic error and then calculate analytically the Covariance matrix for an extended Kalman Filter. Moon *et al.* [**116**] studied the effect of speed and acceleration in the kinematics of differential-drive robot, and proposed a method for maintaining a straight line trajectory. Roy et al [**145**] proposed an online calibration using external sensing in order to estimate the systematic error as a separate component for rotation and for translation.

4. Localization

There are two major approaches to localization of a mobile robot based on whether the full structure of the environment is used. For both approaches a variety of sensing methodologies can be used including computational vision, sonar or laser range finding [**47**].The first approach is to use landmarks in the environment in order to localize frequently and thus reduce the odometry error [**17**]. A common technique is to select a collection of landmarks in known positions and inform the robot beforehand [**61, 97, 65**]. Another technique is to let the robot select its own landmarks according to a set of criteria that optimize its ability to localize, and then use those landmarks to correct its position [**12**]. The second approach to localization is to perform a matching of the sensor data collected at the current location to an existing model of the environment. Sonar and laser range finder data have been matched to geometrical models [**95, 100, 101, 178, 107, 117**], and images have been matched to higher order configuration space models [**5, 53**] in order to extract the position of the robot. Borenstein suggested a two-part robot that would more accurately measure its position by moving one part at a time [**15**]. Also, Markov models have been used in order to describe the state of the robots during navigation [**83**].

The existence of clearly identifiable landmarks is an optimistic assumption for an unknown environment. Even in man-made environments, the cost of maintaining labels in prearranged positions may be prohibitive. Moreover, in large-scale explorations the robot may have to travel a large distance (larger than its sensor range) before being able to locate a distinct landmark.

5. Multiple robots

The advantages of collaborative behavior have been examined extensively in the context of biological systems [**170, 124**]. In the field of robotics, research on multi-robot systems is gaining popularity because the multi-robot approach provides distinct advantages over a single robot such as scalability, robustness and speed. On the other hand it increases the complexity of the system by adding more parameters into the problem space. Basic geometric formations of the robots such as lines and/or circles of robots have been studied for ideal robots [**162, 4**] and for real robots under physical constraints [**181**], as well as simple tasks such as box pushing [**112, 42**], forming a transportation line [**174, 60**] or moving in a convoy [**50**]. The behavior of a group of autonomous robots and the dynamics developed differ according to the communication model among them and what is usually called the sociology of the group. There are different levels of communication that reflect the level of cooperation between the agents [**51**], while sometimes the robots are even tied together [**72, 73**]. The behavior of the robots could range from a master-slaves relation, to full cooperation to complete indifference [**28, 86, 71, 110, 111**].

In the context of terrain coverage in particular, Balch and Arkin were among the first to quantitatively evaluate the utility of inter-robot communication [**8**]. Mataric was another pioneer in considering the utility of inter-robot communication and teamwork in space coverage [**113**]. Dudek, Jenkin, Milios and Wilkes proposed a multi-robot mapping strategy akin to that proposed here, but they only considered certain theoretical aspects of the approach as it applied to very large groups of robots. Several authors have also surveyed a range of possible approaches for collaborative robot interactions [**26, 51, 47**].

Exploration using multiple robots is characterized by techniques that avoid tightly coordinated behavior [**8, 130, 34**]. In earlier work multiple robots used each other to localize when the lack of landmarks made it otherwise impossible [**51**]. Until recently (and subsequent to publications [**132, 139**]) the use of localization among the group members using each robot's neighbors to correct the pose estimate during mapping in order to remove uncertainty from the resulting map has not been considered. The term cooperative localization has been used by a number of authors lately [**143, 172, 9, 59, 64, 161, 108, 58, 182, 74, 13**]. In particular, Fox *et al.* [**59**] presented some work on multi-robot mapping in which two robots exchange information opportunistically, when and if they met. Grabowski *et al.* used a team of cooperating miniature robots for exploration [**67**]. More organized approaches to multi-robot mapping have been proposed by Dellaert [**36**], Parker [**123**] and Simmons [**158**].

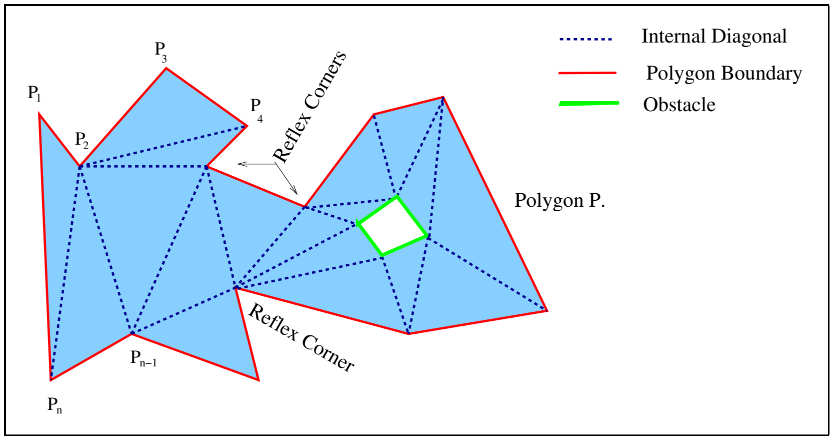


FIGURE 2.1. A polygonal environment with reflex vertices/corners and one obstacle. With dashed lines are marked internal diagonals that form a triangulation.

6. Geometry Overview

The environment is modeled as a simple polygon with holes [**127, 122**], the surrounding walls are represented as the polygon edges and any obstacles inside are represented as holes. In practice, a non-polygonal environment can always be described using a polygonal approximation. Such an approximation can be readily computed so that it is either conservative in the sense that the interior of the approximated free space is assured to be free, or it can be designed to be accurate in a least-squared sense, so that for a given number of vertices in the approximation the discrepancy between the polygonal model and the actual environment is minimized [**75, 56**].

The main terms used in the next two chapters are :

Interior of the Polygon: The free space of the environment where the robots explore. In Figure 2.1 the light blue shaded area.

Polygon Vertex: The corner where two walls meet.

Reflex Vertex: A *Polygon Vertex* with its internal angle (the angle in the interior of the polygon) strictly greater than 180 degrees. See Figure 2.1 for examples.

Internal Diagonal: A line segment connecting two non-consecutive polygon vertices completely contained in the interior of the polygon. Dark blue dashed line in Figure 2.1.

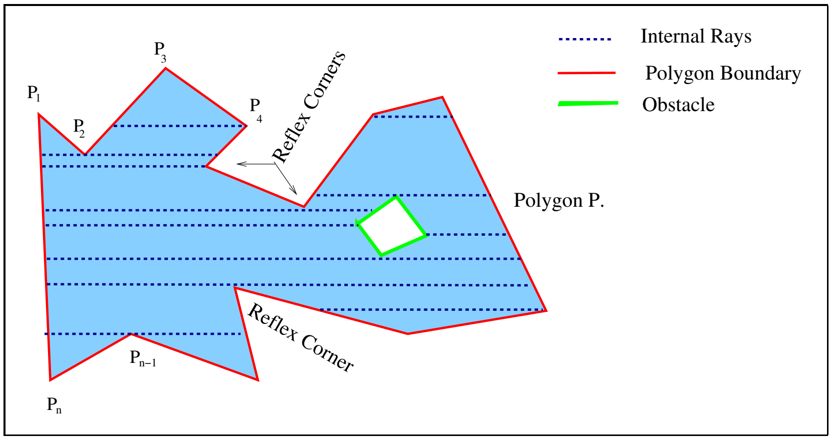


FIGURE 2.2. A polygonal environment with reflex vertices/corners and one obstacle. With dashed lines are marked internal rays that form a trapezoidal decomposition of the interior of the polygon.

Triangulation: The decomposition of the inside of the polygon into triangles. A triangulation of a simple polygon consists of n-2 triangles or n-3 non-intersecting diagonals where n is the number of vertices in the simple polygon.

Trapezoidal decomposition: The decomposition of the inside of the polygon into trapezoids and triangles by rays; each ray starts at a corner and all rays are parallel to each other. See Figure 2.2 for an example.

CHAPTER 3

Exploration within Sensor Range (Triangulation)

In the previous chapters we introduced the idea of mapping free space by sweeping the line of visual contact that connects two robots (Chapter 1) and presented relevant background (Chapter 2). In this chapter we are going to discuss a new algorithm for mapping the interior of an environment systematically (i.e. for performing complete exploration of a bounded environment without exploring any areas more than once). This algorithm works under the assumption that the two robots can maintain visual contact and effectively track each other across any open space in the environment. Our algorithm is based on a polygonal approximation of the environment and the two robots map the interior of the polygon (see Chapter 2 section 6) that represents the free space. Both robots use a traditional range finder in order to detect and circumnavigate obstacles during exploration. In addition, each robot has a robot tracker sensor that is used to detect interfering obstacles when the line of visual contact is interrupted. The exploration strategy is based on a triangulation of the free space.

1. Outline of the triangulation algorithm

The exploration algorithm is based on the following idea. At any single time one robot is positioned at a vertex (corner) of the environment operating as a landmark, while the other robot moves along the perimeter of the environment maintaining visual contact with the stationary robot (see Figure 3.1). More precisely, as the moving robot follows one wall of the environment, it "sweeps" the line of visual contact across the triangle defined by the corner where the stationary robot is positioned and the two ends of the wall. Thus, the robot establishes the position of the wall and the occupancy status of the swept free space inside the triangle. The two robots progressively map the environment by dividing it into triangles of free space, thus constructing a triangulation of the environment. Both robots run the same exploration algorithm, taking turns between (a) moving, thus mapping the free space, and (b) being stationary, thus providing a fixed localization reference for the moving robot [1]. First, a few definitions are presented (in addition to the ones in Chapter 2 section 6), then the major operations of the algorithm are discussed; finally, an outline of the exploration algorithm is presented.

Map: A set of triangles residing entirely within the polygon, which cover completely the interior of the environment (polygon) without overlaps (*Triangulation*).

Unfinished Triangle: A triangle that is not completely mapped; in other words, one of the wall sides is not fully explored (one end-point not mapped yet).

Dual Graph: A graph $(\mathcal{V}, \mathcal{E})$ such that every vertex $v_i \in \mathcal{V}$ corresponds to a triangle T_i, and an edge $e_{ij} \in \mathcal{E}$ between two vertices v_i, v_j exists iff their triangles T_i, T_j share an internal diagonal.

Internal Triangle: A triangle constructed by three internal diagonals of the polygon. The corresponding *Dual Graph Vertex* has degree three.

Open Edge: An edge in the dual graph that connects a mapped triangle with an *Unfinished Triangle*.

Degree of a triangle: The degree of the corresponding vertex in the dual graph, equal to the number of triangle sides that are internal diagonals of the polygon.

Steiner Point: A point that is not part of the input set of points. In our case a pseudo vertex that introduces an extra internal triangle in the triangulation.

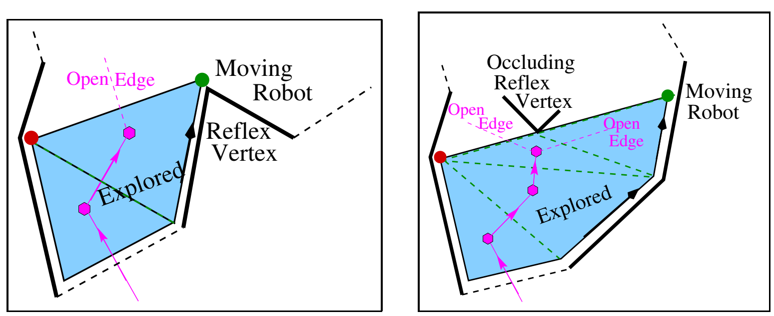
As mentioned earlier the basic operation is the mapping of a triangle of free space as the moving robot travels along one (or two) sides of each triangle. After the triangle is mapped it is included in the map and the corresponding node is added to the *dual graph*. *Internal diagonals* that separate fully mapped triangles from unexplored (or partially explored) areas (unfinished triangles) correspond to *open edges* in the *dual graph*; these *open edges* guide the next step of the exploration (which triangle to map next). The exploration continues as long as there are areas of free space to be mapped and the line of visual contact between

[1]In the following we assume no three vertices are collinear. If this was the case, it would involve a minor but tedious change to the algorithm.

the robots is uninterrupted. Adjacent mapped triangles form a chain of nodes in the *dual graph* which ends either in an *internal triangle* (where there is a bifurcation and the graph splits into two paths), or in a triangle with degree one (where two of its sides are walls, as in the top left triangle in Figure 3.2d). The above two cases represent the end of a depth first search branch of the dual graph.

There are two instances in which the moving robot stops exploring and a decision is made. One instance when the exploration halts is when a triangle with only one internal diagonal (where the corresponding node in the dual graph has degree one) is fully mapped. In that case, the two robots search the *dual graph*, select the closest *open edge*, travel to the two ends of the corresponding diagonal, and resume the exploration.

The second instance when the exploration stops is when the line of visual contact is interrupted. There are four distinct cases where the line of visual contact is interrupted (see Figures 3.1a,b,3.2a,c). In these cases the moving robot cannot continue its previous course and it has to make a decision where to move next in order to maintain visual contact with the stationary robot.



(a) (b)

FIGURE 3.1. Thick line represent walls, dashed lines represent unexplored walls, grey area is explored free space, dashed lines inside the grey are internal diagonals. The stationary robot is red, the moving robot is green, and the red circles connected by arrows at the center of the triangles represent the dual graph. *Line of Visual Contact interrupted*: (a) **Case 1:** The stationary robot is at a non-reflex vertex and the moving robot encounters a reflex vertex that would interrupt the line of visual contact (b) **Case 2:** Occluding Vertex between the two robots.

Case 1: The stationary robot is located at a non-reflex vertex while the moving robot reaches a reflex vertex. If the moving robot continues to follow the next wall then the line of visual contact is interrupted (see Figure 3.1a). In this case the

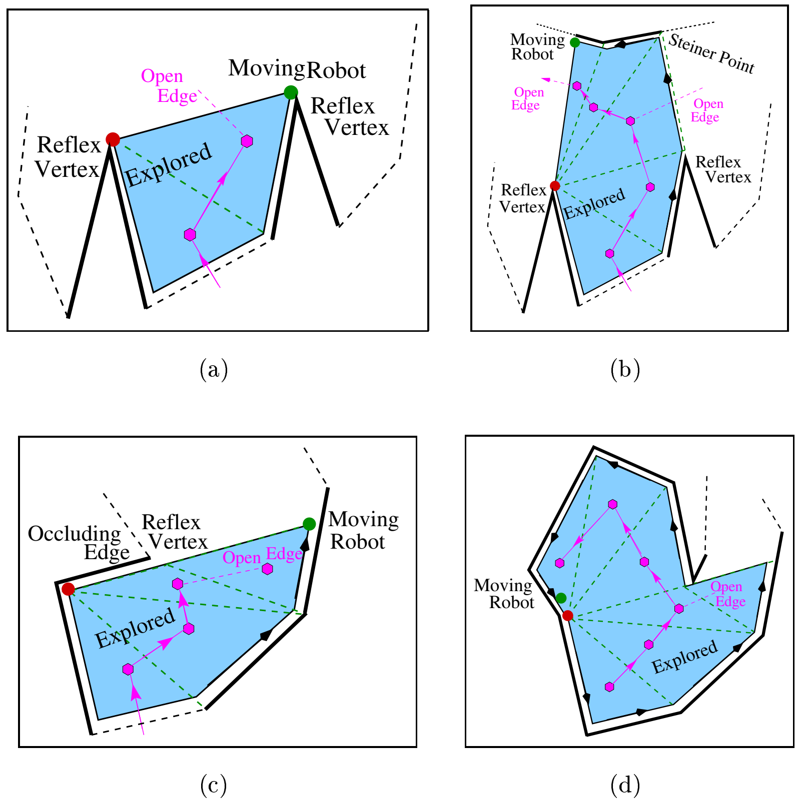


FIGURE 3.2. Same notation as in Figure 3.1. Line of Visual Contact interrupted: (a) **Case 3:** Both robots are placed at reflex vertex such that any further exploration would break the line of visual contact (b) The moving robot explores perpendicular to the diagonal that connects the two reflects vertices. At the first wall found a Steiner point is introduced and an internal triangle is created. (c) **Case 4:** Occluding Edge next to the stationary robot.(d) One branch of the dual graph is completely mapped, the robots would proceed to the nearest open edge of the dual graph.

two robots simply switch roles, the moving robot sends a signal to the stationary robot to start exploring and then becomes stationary and the stationary robot (which was waiting, see Algorithm 3) continues the exploration.

Case 2: During the mapping of a triangle a reflex vertex located between the two robots interrupts the line of visual contact (see Figure 3.1b). First, the partially mapped triangle is stored as an unfinished triangle. Then, the moving robot travels towards the stationary robot until the reflex vertex is encountered and mapped. Consequently, an *internal triangle* is constructed defined by the reflex vertex and the first *internal diagonal* of the unfinished triangle (see Figure 3.1b, Algorithm 2). The *internal triangle* is connected with three triangles the previous fully mapped triangle, and two *unfinished triangles* located at the two sides of the reflex vertex (see Figure 3.1b).

Case 3: Both robots are located next to reflex corners and any motion along the unexplored walls would result in a break of the line of visual contact (see Figure 3.2a). In this case, the moving robot explores in a direction perpendicular to the *internal diagonal* between the two reflex vertices until a wall is encountered (see Figure 3.2b). A Steiner point is introduced and an *internal triangle* is created from the Steiner point and the two reflex vertices. Two *unfinished triangles* are attached to it on the two sides of the Steiner point (see Algorithm 2). Then the exploration continues in the *unfinished triangle* that is closer to the robots [2].

Case 4: During the exploration an occluding edge interrupts the line of visual contact (see Figure 3.2c). This is a sub-case of the *occluding reflex vertex* case where the stationary robot is placed next to the edge adjacent to the occluding vertex. It is treated differently in order to eliminate redundant traveling. The two robots exchange roles and the previously stationary robot receives a command to explore only up to the *occluding reflex vertex* and add the triangle to the map. Then the two robots exchange roles again and continue the exploration.

These four cases cover all possible configurations of interruptions in the line of visual contact. A formal description of the algorithm is presented in Algorithm 1 and an outline of the wall following strategy is presented next.

[2] The introduction of the Steiner point increases the number of triangles by one. In practice it is only necessary in the case where a reflex vertex interrupts the line of sight before the moving robot has encountered a vertex on the top wall (see Figure 3.2b). If a corner is encountered by the moving robot on the wall where the Steiner point was placed then the corner of the internal triangle that was at the Steiner point is moved to that corner and the Steiner point is erased.

```
while Dual Graph contains Open Edges do
  while (No Occlusion) AND (Dual Graph has Open Edges) do
    Assign closest Open Edge to Unfinished Triangle
    Explore Unfinished triangle
  end while{Occlusion has occurred Or branch of Dual Graph is completed}
  if Occlusion is of Case 1 then { One Robot at reflex vertex, Fig. 3.1a}
    SIGNAL(OtherRobot,Continue){Exchange role with the other Robot.}
    WAIT { see Algorithm 3}
  else if Occlusion is of Case 2 then
     { Occluding Vertex between Robots, Fig. 3.1b }
    Mark current position as a temporary Polygon Vertex
    repeat
      Go Towards the Stationary Robot
    until Occluding Polygon Vertex Encountered
    Map the Occluding Polygon Vertex
    CreateInternalTriangle() { see Algorithm 2}
  else if Occlusion is of Case 3 then
     {Both Robots at Reflex Vertices, Fig. 3.2a,b}
    while New Polygon Vertex Not Found do
      Explore perpendicular to the line of visual contact
    end while
    Map Steiner point as a Polygon Vertex
    CreateInternalTriangle() {see Algorithm 2}
  else if Occlusion is of Case 4 then {Occluding Edge, Fig. 3.2c}
    SIGNAL(OtherRobot,ExploreOccludingEdge)
    WAIT {see Algorithm 3}
  end if
  if Current branch of Dual Graph ends then {see Fig. 3.2d}
    Traverse the Dual Graph towards the closest Open Edge
  end if
end while{The Map is complete}
```

Algorithm 1: Triangulation Algorithm; procedures are noted as underlined text, comments are inside curly brackets "{comment}".

```
procedure CreateInternalTriangle()
Create an Internal Triangle with two Open Edges
Add node to the Dual Graph
Connect the Unfinished Triangle to the Internal Triangle
 via the first Open Edge
Continue the exploration following the second Open Edge
```

Algorithm 2: Create Internal Triangle

```
procedure Wait()
repeat
   Check Condition from Signal
until Condition is set
if Condition=ExploreOccludingEdge then
   Map the Occluding Edge up to the next corner
   Create Triangle containing the occluding edge
   Add Triangle to the Map.
   SIGNAL(OtherRobot,Continue)
   WAIT()
end if
```

Algorithm 3: Wait

2. Wall Following

When the line of sight between the two robots is uninterrupted the moving robot explores the environment one triangle at a time by following the closest wall from one corner (end point) to the other. In our implementation of the algorithm the robots follow the walls at a distance of sixty centimeters using a sonar range finder sensor in order to sense the wall. Lines are fitted to the sonar points, using the MacKenzie-Dudek algorithm [**52, 107**], and then the newly sensed lines are merged with the existing map.

An outline of the mapping procedure follows. If the closest wall is the same (see Figure 2a, robots drawn with dashed lines represent past positions) and the robot has not moved past its end (the robots position projects inside the line segment of the wall) then continue exploring the same wall. If a new wall is detected at distance d closer than sixty centimeters (see Figure 2b) then a non-reflex corner is reached and the old wall is fully mapped. The intersection point of the old closest wall with the new closest wall is calculated and then it is used to mark the second end point of the old wall and the first end point of the new wall. Finally, the robot continues the exploration of the new wall by moving away from the first

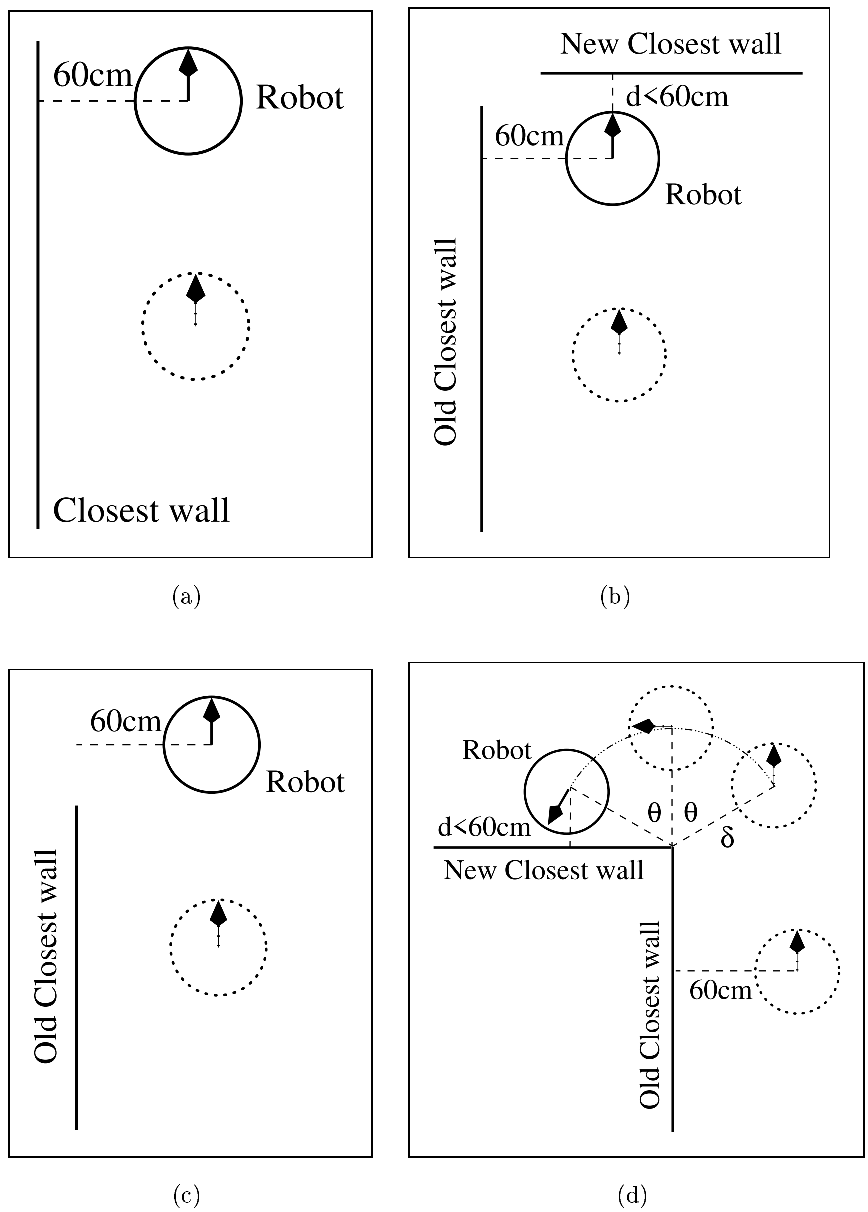


FIGURE 3.3. Illustration of the wall following. (a) The closest wall is the same and the robot has not reached the wall's end: continue forward at 60cm from the wall. (b) A new wall is discovered which is closer than the previous one: reached non-reflex vertex; map the corner and continue the exploration of the new wall. (c) The closest wall ended and the robot has moved past the end of the closest wall: it is a reflex corner; proceed as in sub-figure (d). (d) Map the reflex corner by moving around the end-point of the old closest wall. Move by intervals of θ at a distance less than 60cm and more than 30cm from the end-point. Robots drawn with dashed lines represent past positions.

end point, again at a distance of sixty centimeters. If the closest wall is the same but the robot has moved past the end of it (see Figure 2c) then this indicates that the robot has reached a reflex vertex. In order to map the reflex corner the following procedure is used (see Figure 2d). The robot moves in a circular path at a distance δ from the end point of the closest wall until it finds a new closest wall. The circular motion is done in steps defined by an angle θ, see Figure 2d (in the current implementation $\theta = 15^\circ$ and $\delta = 45cm$). Then the intersection point between the old and the new closest walls is calculated. The old closest wall becomes fully mapped and then, if the reflex vertex does not interrupt the line of visual contact between the two robots, the moving robot continues the exploration by mapping the new closest wall. Otherwise the line of visual contact is interrupted and the robots follow Algorithm 1.

3. Complexity Analysis

In order to analyze the complexity of the exploration we need to distinguish between two qualitatively different stages of exploration, the *local* and the *global* exploration phases. The strategy used during the local exploration phase governs the coverage of individual triangles, while the strategy used during the global exploration phase determines the sequence in which these triangles are visited by using the dual graph. As noted earlier, the exploration strategy is guided by the dual graph of the triangulation and the two robots visit every triangle in a depth first traversal, thus passing through each triangle at most twice (the first time exploring, the second time moving through towards the unmapped parts of the environment). Therefore, the complexity of the global exploration is the complexity of a depth first traversal of the dual graph (of size O(N) where N is the number of vertices in the polygonal environment). The complexity of local exploration, i.e. the mapping of a single triangle, is proportional to the distance traveled by the moving robot which is the length of the wall mapped. Therefore, the total length traveled during the mapping phase is equal to the perimeter of the environment. In the next section we are going to illustrate the previously described algorithms using a simple environment.

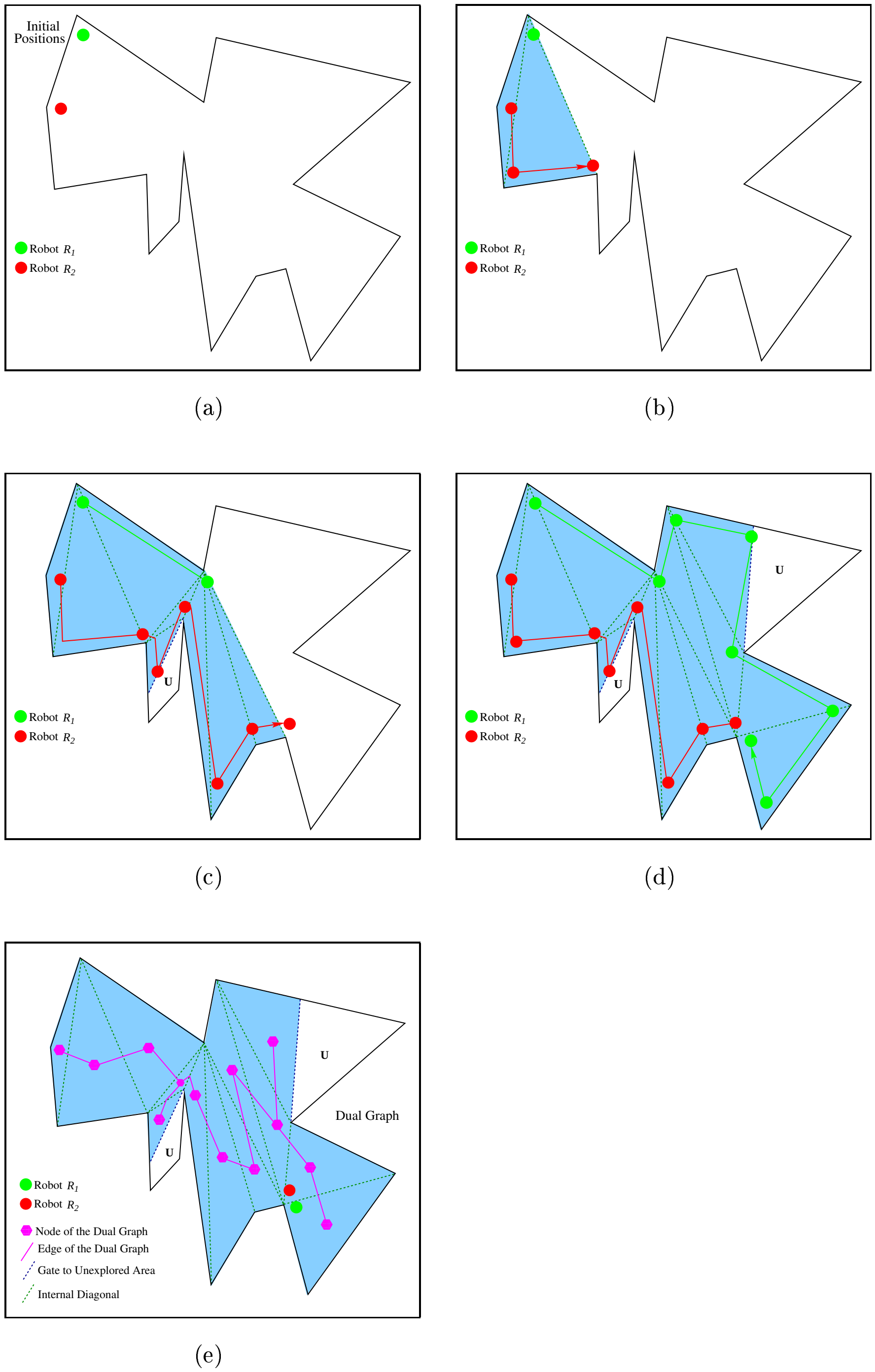


FIGURE 3.4. Two robots (R_1,R_2) are exploring a simple environment. Only one branch of the dual graph is explored, the remaining areas marked as **U**. Robot R_2 explores first.

4. Illustrative Example

In this section we present the triangulation exploration algorithm on the simple environment that appears in Figure 3.4. The two robots start at the upper left wall, Figure 3.4a, and they are noted R_1 in green and R_2 in red. First the robot R_2 explores two walls (see Figure 3.4b) reaching a reflex corner: the mapped area is shaded blue. The two robots exchange roles and robot R_1 starts the exploration (see Figure 3.4c). After a single triangle robot R_1 reaches a reflex corner and the two robots again exchange roles. During the exploration of the next triangle by robot R_2 a reflex vertex interrupts the line of visual contact and robot R_2 proceeds to map the reflex vertex, create an internal triangle and continue the exploration on the left branch of the dual graph. The area that was not mapped is marked by **U**. After mapping three more triangles robot R_2 is forced to stop and exchange roles with robot R_1. Figure 3.4d illustrates the mapping by robot R_1. The line of visual contact is interrupted and an internal triangle is created. The robot R_1 continues the exploration of the right branch of the dual graph up to the point that the two robots meet and the branch of the dual graph is fully explored. Finally, Figure 3.4e presents the triangulation up to this point; the two internal triangle have created two bifurcations on the dual graph and in each case only one branch is mapped. The two robots move next through the explored areas to the closest opening that leads to an unexplored space (marked as **U**) on the map, and from there they resume the exploration.

5. More than two robots

An immediate extension of the triangulation algorithm can be obtained by the addition of more robots. The robots form a chain, where the first and the last robot play the roles of the two robots of the previously described algorithm and the other robots (in between) act as relays. If the range of the robot tracker sensor is R then the use of N robots extends the sensor range to $(N-1)R$. Figure 3.5 illustrates the above idea. Robot R_1 is stationary and robot R_4 is following the wall (Figure 3.5a presents an initial configuration). Next the exploring robot (R_4) is moving along the wall (Figure 3.5b). Note that robots drawn with dashed lines represent past positions. In Figures 3.5c,d the robots R_3 and R_2 move respectively to form a line again. With the addition of more robots the range of the robot tracker sensor could be extended in order to map larger environments.

In this chapter we discussed an algorithm for exploring an unknown environment that is bounded by the range of the sensor. If the environment is larger than the range of the robot tracker sensor then this algorithm can not be used. Thus a different exploration algorithm is proposed based on the trapezoidal decomposition of the environment, as described in the next chapter.

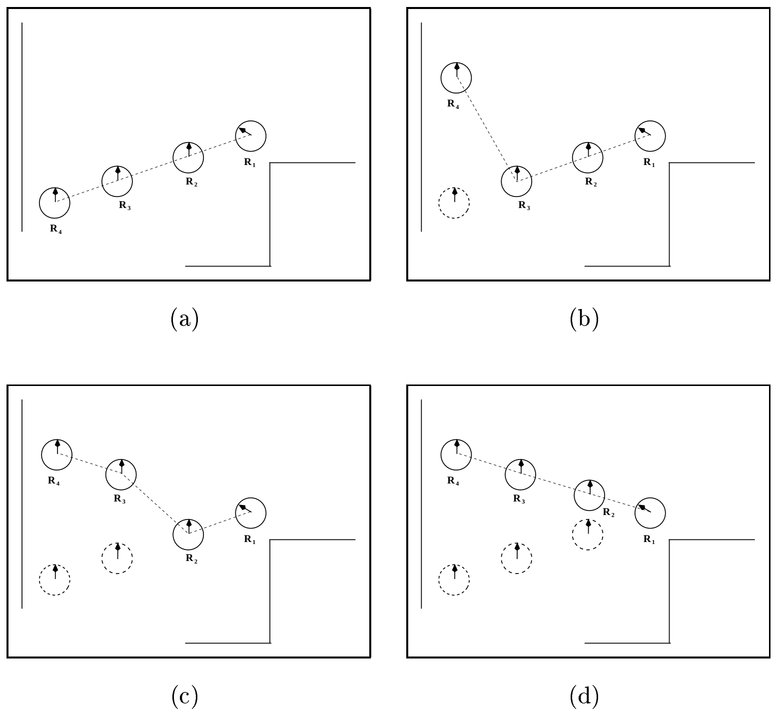


FIGURE 3.5. Triangulation with four robots (a) All robots are aligned, R_1 is stationary and robot R_4 is following the wall. (b) Robot R_4 moves along the wall. (c) Robot R_3 aligns itself with R_1 and R_4. (d) R_2 moves into line. Robots drawn with dashed lines represent past positions.

CHAPTER 4

Exploration Beyond Sensor Range (Trapezoidation)

...In that Empire, the craft of Cartography attained such Perfection that the Map of a Single province covered the space of an entire City, and the Map of the Empire itself an entire Province. In the course of Time, these Extensive maps were found somehow wanting, and so the College of Cartographers evolved a Map of the Empire that was of the same Scale as the Empire and that coincided with it point for point.
–From Travels of Praiseworthy Men (1658) by J.A. Suarez Miranda in Jorge Luis Borges's "Of Exactitude in Science" A Universal History of Infamy (1972).

1. Introduction

In the previous chapter we presented an algorithm for exploring an unknown environment in which the range of the robot tracker sensor is long enough to reach across it. In this chapter we are going to examine an exploration strategy for mapping beyond the range of the sensor. In environments consisting of large areas of open space (eg. warehouses, docking areas, open fields) it is quite common for the robots to be unable to follow a wall or to detect any landmarks because the environment is larger than the range of the sensors. In such environments the moving robot uses the stationary robot as a portable marker for relocalizing and mapping. We present different motion strategies for the complete mapping of the environment. The core idea of the algorithm is the mapping of an area of free space by one moving robot while the other robot is stationary. The purpose of the algorithm described below is to determine the sequence in which the free areas are explored, without duplication and ensuring full coverage of the free space. In a bottom up description of the algorithm there are the following steps. One robot moves and sweeps the line of visual contact across the free space, thus mapping a single region of free space. Then the two robots exchange roles in order to explore a chain of free-space areas which forms a stripe; a series of stripes are connected together to form a trapezoid. After several iterations of these two steps the entire collection of the trapezoids provides the trapezoidal decomposition of the entire free space – a complete spatial decomposition of the interior of the environment.

2. Top-Down description of the Algorithm

The proposed algorithm is based on the trapezoidal spatial decomposition of a polygon [**122, 127**]. A top down description of this algorithm is illustrated in Figure 4.1a-d. More specifically, the two robots explore the world using a trapezoid decomposition of the free space as their guide, as can be seen in Figure 4.1a. Each trapezoid is mapped completely before the two robots proceed to map the next one. The order in which the trapezoids are mapped is given by a depth-first traversal of the embedded dual graph (see Figure 4.1b). Every trapezoid corresponds to a vertex in the graph; vertices corresponding to adjacent trapezoids are connected with an edge in the graph. Therefore, after one trapezoid is mapped the two robots proceed to map the trapezoid that corresponds to the adjacent node in the dual graph. The sensing range of the robot tracker provides a limit on the space that can be explored at any single time, before the two robots have to switch roles. Thus, if a trapezoid is larger than the range of the robot tracker, then it is broken down into stripes with a width that depends on the sensing range **R** (see Figure 4.1c).

The exploration of a single stripe can be accomplished using various motion strategies. At the top of Figure 4.1d, two different motion strategies are displayed. The first motion strategy (Strategy **A**) is quite intuitive. In each exchange, one of the robots moves on a straight line (dotted line in figure 4.1d) sweeping (and hence mapping) a triangular region. When the distance between the two robots reaches the robot tracker sensor range the two robots exchange roles. The second motion strategy (Strategy **B**) is proven optimal (see

Appendix A) with respect to the area covered over distance traveled. In each exchange, one of the robots traverses the two chords shown as dashed lines in figure 4.1d, sweeping a diamond shaped area. The moving robot travels across the first chord and at an angle $\theta/2$ changes heading and follows the second chord (see Figure 4.1d).

As we discussed above Strategy **A** is simpler and requires a smaller number of direction changes, but unfortunately, the width of the stripe (d) produced is suboptimal ($d < R$), and thus a larger number of stripes is needed in order to cover the same area. Strategy **B** is optimal in terms of path traveled over area covered (see Appendix A) because at any single time the width of the stripe covered (d) is the maximum possible ($d = R$). At the bottom of Figure 4.1d, the mapping of free space is presented over a single exchange. Angle θ is an input parameter that can be chosen to minimize a cost function as a function of θ. In the case of reflex corners of the polygonal boundary of the large open space being mapped, one trapezoid splits into two new trapezoids, and the two agents decide which branch of the embedded graph to follow.

When a sequence of explored regions are linked to each other as the exploration progresses, different types of stripes are created. In the case of the coverage of a triangular area, the two robots travel in parallel lines separated by d, and the stripe mapped has the same width d (see Figure 4.1d, Motion Strategy A). In the case where each robot covers a diamond area, the trajectory of each robot would be a zig-zag line creating a stripe with width R, equal to the sensing range of the robot tracker (see Figure 4.1d, Motion Strategy B). A sequence of stripes connected together (lengthwise) map a single trapezoid (see Figure 4.1c). At the end of each stripe the two robots follow the walls and reposition themselves to explore the next stripe.

3. Complexity Analysis

Similar to the triangulation algorithm (see Chapter 3), in order to analyze the complexity of the exploration we need to distinguish between two qualitatively different stages of exploration, the *local* and the *global* exploration phases. The local exploration strategy guides the path traveled for the mapping of a convex segment of free space defined by

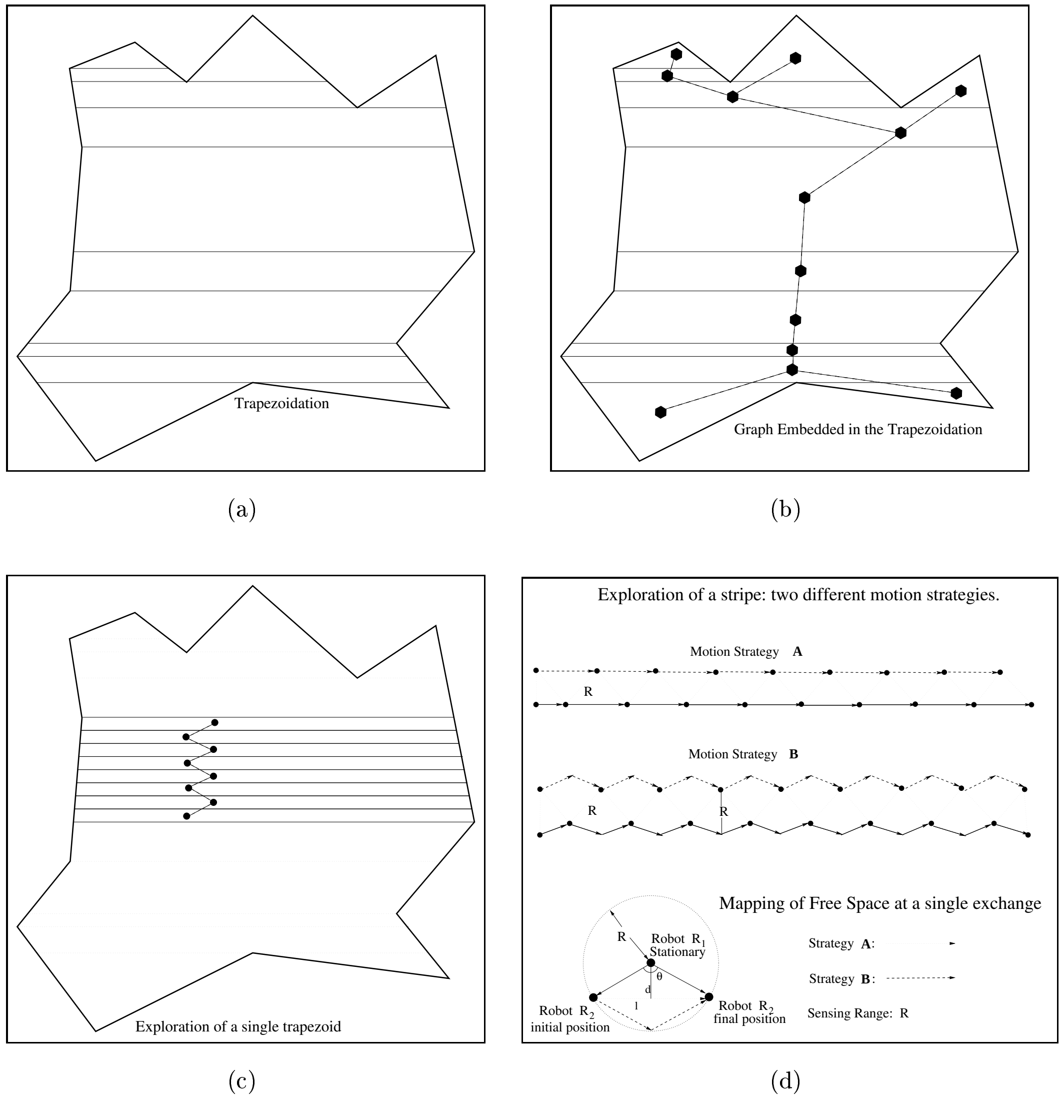


FIGURE 4.1. A top down description of the Trapezoidation algorithm. (a) The environment is divided into trapezoids. (b) The order in which the trapezoids are mapped is given by a traversal of the *Dual Graph*. (c) Each trapezoid is further divided into stripes with a width proportional to the sensing range R. (d) Each stripe is covered by areas of free space one next to the other. Each area of free space is explored by the motion of a single robot. Different motion strategies can be used, and the size of the area is controlled by the angle θ.

the exploration algorithm (a triangle, or a trapezoid [1]). The global exploration strategy determines the order in which these areas are explored.

[1]The trapezoidal decomposition divides the interior of a polygon into trapezoids and triangles.

3.1. Complexity of Global Exploration. As noted earlier, the exploration strategy is guided by the dual graph of the trapezoidal decomposition used. More specifically, the two robots explore one trapezoid at a time and then proceed to map the next trapezoid by following the dual graph in a depth first traversal. Every trapezoid is visited at most twice, the first time when it is being mapped and the second time when the two robots pass through in a shortest path traversal to move to the next unexplored area. In general the complexity is proportional to the number of edges of the polygon, the size of the environment, and the range of the tracking sensor (which determines how many stripes are created in each trapezoid).

3.2. Complexity of the exploration over a single exchange. In contrast to the triangulation algorithm where the robots move along the walls, when the trapezoidation algorithm is used, the moving robot could follow different trajectories as long as it stays inside the sensing range of the stationary robots. Alternative motion strategies present certain advantages and disadvantages. More precisely, there are different factors that affect the cost of the exploration depending on the configuration of the different robots. In terms of cost we consider the distance traveled, the number of rotations and the number of role exchanges between the two robots. Moreover all of the above can be translated into time cost on how long it would take for the exploration. More precisely, every time the two robots exchange roles, the moving robot uses the stationary one to correct its position and then the stationary one starts exploring. Each of these operations introduces a time delay, therefore the number of exchanges increases the cost. In addition, every time one of the robots has to change directions the rotation adds to the total cost. Finally, the total path traveled has to be taken into consideration. For the two different motion strategies (diamond area covered, and triangular area covered) examined earlier, the total mechanical effort can be computed as shown in Table 4.1. The cost is calculated for the exploration of a rectangle X by Y, when the robot tracker sensor range is R.

3.2.1. *Cost Analysis.* The factors that affect the cost of the exploration are: the number of exchanges, the total path traveled and the number of rotations. For a specific set of robots the cost of the above factors can be determined beforehand. Specifically, the total cost of the exploration can be computed as the weighted sum of: the total path traveled (P_θ) multiplied by the cost of path traveled (C_p in sec/m), the total number of

Covering	Triangle Area	Diamond Area
Path length P_θ	$2Y + \frac{XY}{R}\frac{2}{\cos\frac{\theta}{2}}$	$2Y + \frac{XY}{R}\frac{2}{\cos\frac{\theta}{4}}$
# of exchanges E_θ	$\frac{XY}{R^2}\frac{2}{\sin\theta}$	$\frac{XY}{R^2}\frac{1}{\sin\frac{\theta}{2}}$
# of turns R_θ	$2\frac{Y}{R\cos\frac{\theta}{2}}$	$2\frac{Y}{R} + 2\frac{XY}{R^2\sin\frac{\theta}{2}}$

TABLE 4.1. Analytical complexity of two different path curves.

exchanges (E_θ) multiplied by the cost for an exchange (C_e in sec/exchange), and the total number of rotations (R_θ) multiplied by the cost of rotation (C_r in sec/rotation). The factors (C_p, C_e, C_r) can be estimated before the exploration, while the sensing range (R) of the robot tracker is known. Equations 4.1 and 4.2 provide the total cost $C_{total}(\theta)$ as a function of angle θ for the exploration, using diamond area and triangular area covering respectively (as shown in Figure 4.1d), of a rectangle $X \times Y$ as a function of θ, using the cost estimates and the analytical results from table 4.1. The optimal θ for the exploration is the one that minimizes $C_{total}(\theta)$.

$$\begin{aligned} C_{total,diamond}(\theta) &= C_pP_\theta + C_eE_\theta + C_rR_\theta = \\ &= C_p(2Y + \frac{2XY}{R\cos\frac{\theta}{4}}) + (\frac{C_eXY}{R^2\sin\frac{\theta}{2}}) + C_r\theta(\frac{2Y}{R} + \frac{2XY}{R^2\sin\frac{\theta}{2}}) \end{aligned} \tag{4.1}$$

$$\begin{aligned} C_{total,triangle}(\theta) &= C_pP_\theta + C_eE_\theta + C_rR_\theta = \\ &= C_p(2Y + \frac{2XY}{R\cos\frac{\theta}{2}}) + C_e(\frac{2XY}{R^2\sin\theta}) + C_r\theta(\frac{2Y}{R\cos\frac{\theta}{2}}) \end{aligned} \tag{4.2}$$

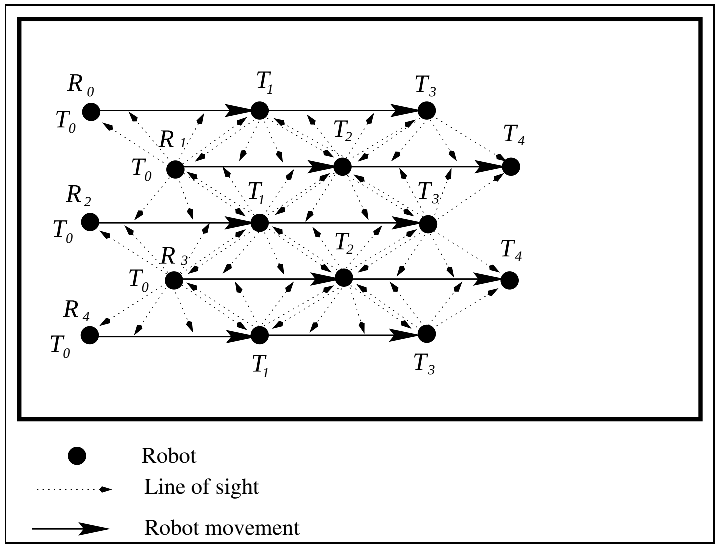
For one of our robots, a Nomad 200, the cost factors, for a typical experimental arrangement are: $C_p = 4.1$ sec/m, $C_e = 7$ sec/exchange, $C_r = 4.65$ sec/rad. The optimal angle θ is 180°, for the diamond area motion strategy. For the same costs the optimal angle θ is 90° for the triangular area motion strategy. As expected the total cost is lower for the motion strategy that covers the diamond area than that which covers a single triangular area.

4. More than two robots

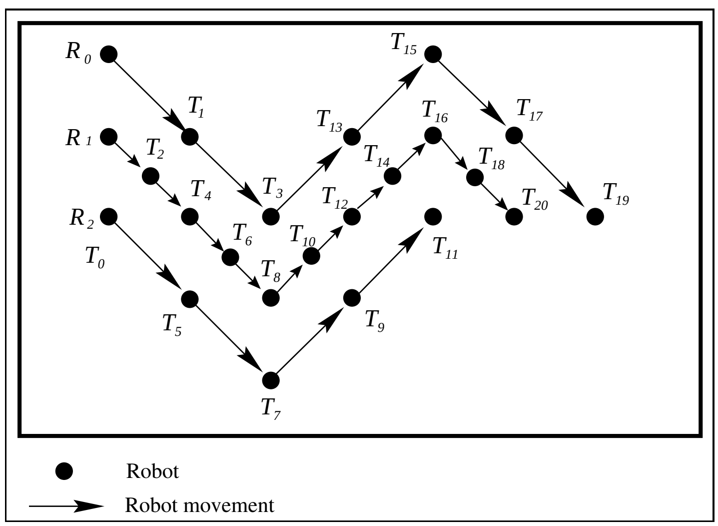
The above strategies could be extended by the addition of more robots. By forming a chain of robots that "sweeps" through the free space the range of the tracker is multiplied by the number of robots, thus covering a much larger area in a single sweep. In addition, every robot could estimate its pose with respect to more than one stationary robot, therefore gaining higher precision in its measurements. Two motion strategies are proposed for groups of more than two robots with respective advantages. Using the first motion strategy, only one robot moves during the exploration while the stationary ones that are still visible are used as landmarks. This method ensures minimum uncertainty accumulation as, at any given time, the moving robot would correct its odometry error with respect to more than one landmark. Using the second motion strategy, the robots divide into two teams and they interchange roles: while the first team is moving the second team works as a set of landmarks. This method explores an environment in less time but fewer robots are available as landmarks.

4.1. Motion Strategies. As mentioned earlier an immediate extension of the trapezoidation algorithm can be obtained by the addition of more robots. When the two robots sweep one stripe of width d, then by adding an extra robot we could double the area swept per step of the algorithm. In the original algorithm, every robot has only one device to track the other robots; in this case a scheduling algorithm should be applied to determine the order in which the robots are moving. If we add a second tracking device, one robot could track robots on both sides, allowing a parallel cover of double the area at the same time.

In the example of Figure 4.2a we use five robots ($R_0 \ldots R_4$) that are positioned in two lines at time T_0. First the robots R_0, R_2, R_4 move forward, tracked by R_1 and R_3 accordingly, mapping the four triangles as free space (time T_1), then both R_0, R_2 track R_1, which moves forward (time T_2); while R_2, R_4 track R_3, which moves forward (time T_2). Then it is time for the other column of robots (R_0, R_2, R_4) to advance marking more area as free space (time T_3). The tracking is marked with the dotted lines of sight. The same pattern is followed as the two columns alternatively advance, marking a stripe of free space much wider than that possible with only two robots.



(a)



(b)

FIGURE 4.2. (a) Exploration of a stripe with 5 robots. The robots move at time T_1,T_2,T_3, and T_4. (b) Exploration of a stripe with 3 robots, covering space in diamond areas. The robot move at time T_1-T_{19}.

The second part of the algorithm concerning the exploration strategy for the whole space and the order in which the trapezoids should be explored is identical to the previous algorithm where only two robots were used.[2]

[2]There is a possible speedup by splitting up the group in order to explore different parts in critical points, but that would in the end spread the robots too thin.

Moving only one robot at a time can also be easily extended to multiple robots. The robots start exploration aligned with each other in a straight line, at distance R from each other, where R is the tracker sensor range. The first robot and the last robot in the line act out the algorithm for two robots, while the role of the other robots is simply to provide a communication path between them. As such, the first robot in the line remains stationary, and the rest of the robots are moving such that the distance between two robots is never more than R. The width of the explored stripe is $(n-1)R$, where n is the number of robots. A pictorial representation of this strategy can be seen in Figure 4.2b, the robots R_0 and R_2 sweep a stripe using the diamond pattern and the Robot R_1 stays between them.

In this chapter an algorithm for mapping areas beyond the range of the robot tracker sensor was discussed. Together with the previous chapter they provide an approach to mapping the free space by sweeping the line of visual contact. In practice, the robot tracker sensor, as well as any other sensor used, suffers from noise. In the next chapter we propose a probabilistic framework for dealing with the uncertainty/noise that corrupts the measurements.

CHAPTER 5

Uncertainty Reduction

In theory, there is no difference between theory and practice.
In practice, there is.
–Attributed to Yogi Berra and Jan L.A. van de Snepscheut.

In the two previous chapters we described two deterministic algorithms for the methodical exploration of an unknown environment by a team of autonomous mobile robots. During exploration the robots collect information from various sources (sonar sensors, odometers, robot-tracker sensors, etc), information that is always corrupted by noise. In order to cope with the effects of the noise we adopt a Bayesian probabilistic framework that integrates information over time and takes into account multiple, competing hypotheses in order to produce a map with higher accuracy.

The following subjects are presented in this chapter. Section 1 provides an outline of the Bayesian framework used in the rest of the chapter. Section 2 contains a detailed description of the Monte-Carlo Simulation method (particle filtering) we used in order to implement the Bayesian framework.

1. Bayesian Reasoning

The Bayesian approach provides a general framework for the estimation of the state of our system (the current pose of all robots) in the form of a probability distribution function (*pdf*), based on all the available information.

For the linear-Gaussian estimation problem[1] the required *pdf* remains Gaussian and the Kalman filter provides a provably optimal solution [**81, 19, 142**]. In the non-linear Gaussian case the Extended Kalman Filter (EKF) has been successfully used by linearizing the control equations [**159, 160**]. For non-linear, non-Gaussian models two difficulties must be resolved: how to represent a general *pdf* using finite computer storage and how to perform the integrations involved in updating the *pdf* when new data are acquired. During the exploration the uncertainty build-up in the pose estimate of each robot translates into uncertainty in the resulting map. In order to improve the accuracy of the map the pose of the robot has to be estimated at discrete time steps. This is an instance of the discrete time estimation problem and can be formulated in state-space notation (see also Gordon *et al.* [**66**]).

The i^{th} robot pose at time $t = k$ is represented by the state vector $\mathbf{x}_k^i = [x_k^i, y_k^i, \hat{\theta}_k^i]^T$, $\mathbf{x}_k^i \in \Re^2 \times S^1$. Each robot takes action (α_k^i) and its pose evolves according to Equation 5.1. [2]

$$\mathbf{x}_k = f_\alpha(\mathbf{x}_{k-1}, v_k) \tag{5.1}$$

where, f_α is the system transition function that models how action α probabilistically modifies the pose of the robot and how it is affected by the noise v_k. The actual transfer function f_α is not analytically available; instead, a simulation (as described in Appendix B Section 4) that models the effect of noise and provides an approximation $\hat{f}_\alpha \approx f_\alpha$ is used.

After each action is performed the robot acquires one (or more) sensor readings. Every sensor measurement available at time $t = k$ is included in a sensor data vector noted as $\mathbf{z}_k^i$. These measurements are related to the state vector via the observation equation 5.2.

$$\mathbf{z}_k = g_k(\mathbf{x}_k, u_k) \tag{5.2}$$

where g_k is the measurement function and u_k is the noise model.

It is assumed that the initial *pdf* $P(\mathbf{x}_0)$ is known and that the available information at time $t = k$ is the set of measurements and the set of actions up to that time. In order for

[1] Where the noise probability distribution functions are Gaussian and the model of the system is linear.
[2] The superscript "i" that indicates the robot to which we refer is dropped for clarity of presentation for the rest of the discussion.

the robot to decide the next action it needs to know its current pose or, since knowledge of the true pose is not feasible due to noisy measurements, at least the *pdf* of its pose given the previous actions and observations ($P(\mathbf{x}_k|\mathbf{x}_0, \alpha_j, \mathbf{z}_j : j = 1 \ldots k)$). This can be achieved recursively, by first predicting the prior probability of $\mathbf{x}_k$ from the previous pose $\mathbf{x}_{k-1}$ (presuming it is available) and the action taken α_k (see Equation 5.3) and then updating using the latest sensor data $\mathbf{z}_k$ in order to obtain the posterior distribution of the pose $\mathbf{x}_k$ of the moving robot given all available information.

$$P(\mathbf{x}_k|\mathbf{x}_0, \alpha_k, \underbrace{\alpha_j, \mathbf{z}_j}_{j=1\ldots k-1}) = \int P(\mathbf{x}_k|\alpha_k, \mathbf{x}_{k-1}) P(\mathbf{x}_{k-1}|\mathbf{x}_0, \underbrace{\alpha_j, \mathbf{z}_j}_{j=1\ldots k-1}) d\mathbf{x}_{k-1} \tag{5.3}$$

Note that the $P(\mathbf{x}_k|\alpha_k, \mathbf{x}_{k-1})$ can be derived by the system model (Equation 5.1), the known characteristics of the noise v_{k-1} and the $P(\mathbf{x}_{k-1}|\mathbf{x}_0, \alpha_j, \mathbf{z}_j : j = 1 \ldots k-1)$, which is the posterior of $\mathbf{x}$ at time $t = k-1$.

When new sensory information becomes available we can use Bayes rule in order to update the *pdf* of the moving robot with the latest observations (Equation 5.4). The conditional probability of the sensor measurement $\mathbf{z}_k$ given the pose $\mathbf{x}_k$ from which it was obtained can be estimated by the sensing function g_k and the noise model v_k. Finally the normalizing denominator can be obtained through Equation 5.5.

$$P(\mathbf{x}_k|\mathbf{x}_0, \alpha_j, \mathbf{z}_j : j = 1 \ldots k) = \frac{P(\mathbf{z}_k|\mathbf{x}_k) P(\mathbf{x}_k|\mathbf{x}_0, \alpha_j, \mathbf{z}_j : j = 1 \ldots k-1)}{P(\mathbf{z}_k|\mathbf{x}_0, \alpha_j, \mathbf{z}_j : j = 1 \ldots k-1)} \tag{5.4}$$

$$P(\mathbf{z}_k|\mathbf{x}_0, \alpha_j, \mathbf{z}_j : j = 1 \ldots k-1) = \int P(\mathbf{z}_k|\mathbf{x}_k) P(\mathbf{x}_k|\mathbf{x}_0, \alpha_j, \mathbf{z}_j : j = 1 \ldots k-1) d\mathbf{x}_k \tag{5.5}$$

Many different methods can be used in order to estimate the *pdf* of the moving robot, an overview of which is given in Chapter 2 Sections 4 & 2. In our work we applied a Monte Carlo simulation technique called *particle filtering* which is described next.

2. Particle Filter

The main objective of particle filtering is to "track" a variable of interest as it evolves over time, typically with a non-Gaussian and potentially multi-modal *pdf*. The basis of the method is to construct a sample-based representation of the entire *pdf*. A series of actions are taken, each one modifying the state of the variable of interest according to some model. Moreover at certain times an observation arrives that constrains the state of the variable of interest at that time.

Multiple copies (particles) of the variable of interest are used, each one associated with a weight that signifies the quality of that specific particle. An estimate of the variable of interest is obtained by the weighted sum of all the particles. The particle filter algorithm is recursive in nature and operates in two phases: *prediction* and *update*. After each action, each particle is modified according to the existing model (*prediction* stage), including the addition of random noise in order to simulate the effect of noise on the variable of interest. Then, each particle's weight is re-evaluated based on the latest sensory information available (*update* stage). At times the particles with (infinitesimally) small weights are eliminated, a process called resampling.

More formally, the variable of interest (in our case the pose of the moving robot $\mathbf{x}^k = [x^k, y^k, \hat{\theta}^k]^T$) at time $t = k$ is represented as a set of M samples (the "particles") ($S_i^k = [\mathbf{x}_j^k, w_j^k] : j = 1 \ldots M$), where the index j denotes the particle and not the robot, each particle consisting of a copy of the variable of interest and a weight (w_j^k) that defines the contribution of this particle to the overall estimate of the variable.

If at time $t = k$ we know the *pdf* of the system at the previous instant (time $t = k - 1$) then we model the effect of the action to obtain a prior of the *pdf* at time $t = k$ (prediction). In other words, the *prediction* phase uses a model in order to simulate the effect an action has on the set of particles with the appropriate noise added. The *update* phase uses the information obtained from sensing to update the particle weights in order to accurately describe the moving robot's *pdf*. Algorithm 4 presents a formal description of the particle filter algorithm and the next two subsections discuss the details of prediction and update.

Given a particle distribution, we often need to take actions based on the robot pose. Three different methods of evaluation have been used in order to obtain an estimate of the pose. First, the weighted mean ($P_{est} = \sum_{j=1}^{M} w_j \mathbf{x}_j$) can be used; second, the best particle (the P_j such that $w_j = max(w_k) : k = 1 \ldots M$) and, third, the weighted mean in a small window around the best particle (also called robust mean) can be used. Each method has its advantages and disadvantages: the weighted mean fails when faced with multi-modal

Require: A set of Particles for Robot i at time 0: $S_i^0 = [\mathbf{x}_j, w_j : j = 1 \ldots M]$.
$W = w_j : j = 1 \ldots M$
while (Exploring) **do**
$k = k + 1$;
if (ESS(W) $< \beta * M$) **then** {`Particle Population Depleted (Equation 5.10)`}
Index=Resample(W);
$S_i^k = S_i^k(Index)$;
end if
for ($j = 1$ to M) **do** {`Prediction after action` α}
$\mathbf{x}_j^{k+1} = \hat{f}(\mathbf{x}_j^k, \alpha)$
end for
s=Sense()
for ($j = 1$ to M) **do** {`Update the weights`}
$w_j^{k+1} = w_j^k * \mathcal{W}(s, \mathbf{x}_j^{k+1})$
end for
for ($j = 1$ to M) **do** {`Normalize the weights`}
$w_j^{k+1} = \frac{w_j^{k+1}}{\sum_{j=1}^{M} w_j^{k+1}}$
end for
end while
{`ESS is the Effective Sample Size, see Equation 5.10`}

Algorithm 4: Particle Filter Algorithm; procedures are noted as underlined text, Comments are inside curly brackets "{`comment`}".

distributions, while the best particle introduces a discretization error. The best method is the robust mean but it is also the most computationally expensive.

2.1. Prediction. In order to predict the probability distribution of the pose of the moving robot after a motion we need to have a model of the effect of noise on the resulting pose. Many different approaches have been used (see Borenstein *et al.* [**16, 18**] for an overview), most of which use an additive Gaussian noise model for the motion. Any arbitrary motion $[\Delta x, \Delta y]^T$ can be performed as a rotation followed by a translation (a piecewise linearisation, see Figure 5.1). The robot's initial pose is $[x, y, \hat{\theta}]^T$. First the robot rotates by $\delta\hat{\theta} = \hat{\theta}_k - \hat{\theta}$, where $\hat{\theta}_k = \arctan(\Delta y / \Delta x)$ to face the destination position, and then it translates forward by distance $\rho = \sqrt{\Delta x^2 + \Delta y^2}$ [3]. If the starting pose is $[x, y, \hat{\theta}]^T$, the resulting pose $[x', y', \hat{\theta}_k]^T$ is given in Equation 5.6. Consequently, the noise model is applied separately to each of the two types of motion because they are assumed independent.

[3] In our experimental setup the Nomadic Technologies Superscout II robots used are controlled by the same rules.

$$\begin{bmatrix} x' \\ y' \\ \hat{\theta}' \end{bmatrix} = \begin{bmatrix} x + \rho\cos(\hat{\theta}_k) \\ y + \rho\sin(\hat{\theta}_k) \\ \hat{\theta}_k \end{bmatrix} \tag{5.6}$$

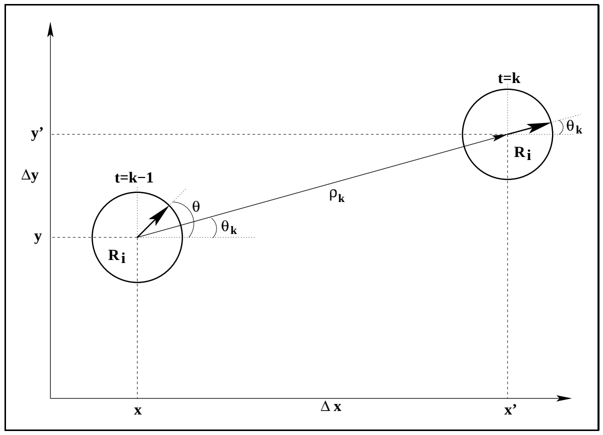


FIGURE 5.1. Arbitrary motion $[\Delta x, \Delta y]^T$ of robot R_i. At time $t = k-1$ the pose is $[x, y, \hat{\theta}]^T$, after the motion at time $t = k$ the pose is $[x', y', \hat{\theta}_k]^T$. The robot first rotates to orientation $\hat{\theta}_k$ and then translates by ρ_k.

2.1.1. *Rotation.* When the robot performs a relative rotation by $\delta\hat{\theta}$ the noise from the odometry error is modeled as a Gaussian with mean (M_{rot}) experimentally established (see appendix B) and sigma proportional to $\delta\hat{\theta}$. More formally, if at time $t = k$ the robot has an orientation $\hat{\theta}_k$ then after the rotation (time $t = k+1$) the orientation of the robot is given by Equation 5.7. Therefore, to model the rotation of $\delta\hat{\theta}$, the orientation $\hat{\theta}_j$ of each particle j is updated by adding $\delta\hat{\theta}$ plus a random number drawn from a normal distribution with mean M_{rot} and standard deviation $\sigma_{rot}\delta\hat{\theta}$ ($N(M_{rot}, \sigma_{rot}\delta\hat{\theta})$, where σ_{rot} is in degrees per 360°).

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \delta\hat{\theta} + N(M_{rot}, \sigma_{rot}\delta\hat{\theta}) \tag{5.7}$$

2.1.2. *Translation.* Modeling the forward translation is more complicated [4]. There are two different sources of error, the first related to the actual distance traveled and the second related to changes in orientation during the forward translation. During the translation the orientation of the robot changes constantly resulting in a deviation from the desired

[4]For a detailed description of the model please refer to appendix B sections 3,4.2.

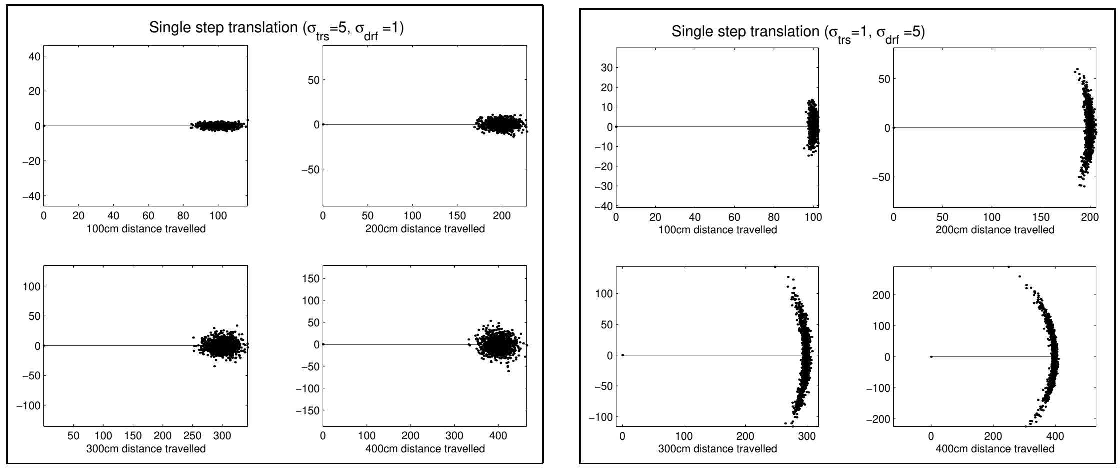
direction of the translation; such effect is called drift and we model it by adding a small amount of noise to the orientation of the robot before and after each step. As well, if the intended distance is ρ, the actual distance traveled is given by ρ plus some noise following a Gaussian distribution. Experimental results provide the expected value and the standard deviation for the drift and pure translation. Because it is very difficult to analytically model the continuous process, a simulation is used that discretizes the motion to K steps, where K is chosen to be low enough for computational efficiency but high enough in order to describe the effect of noise in forward translation. If $[\sigma_{translation}, \sigma_{drift}]$ are experimentally obtained values per distance traveled then at each step of the simulation the standard deviation used is given in Equation 5.8. Algorithm 5 provides a formal description of the prediction phase of a set of particles S for a forward translation by distance ρ.

$$\begin{aligned} \sigma_{trs} &= \sigma_{translation}\sqrt{K} \\ \sigma_{drft} &= \sigma_{drift}\sqrt{\frac{K}{2}} \end{aligned} \tag{5.8}$$

Input: Set of M Particles::S; Translation distance::ρ
$\delta\rho = \frac{\rho}{K}$;
for ($j = 1$ to M) **do** { `For each particle`}
  for ($k = 1$ to K) **do** { `At each of K steps`}
    E_{trs} =<u>rand_N</u>($M_{trs} * \delta\rho, \sigma_{trs} * \delta\rho$);
    E_{drft} =<u>rand_N</u>($M_{drft} * \delta\rho, \sigma_{drft} * \delta\rho$);
    $\hat{\theta}[j] = \hat{\theta}[j] + E_{drft}$;
    $x[j] = x[j] + (\delta\rho + E_{trs} * \cos(\hat{\theta}[j])$;
    $y[j] = y[j] + (\delta\rho + E_{trs} * \sin(\hat{\theta}[j])$;
    E_{drft} =<u>rand_N</u>($M_{drft} * \delta\rho, \sigma_{drft} * \delta\rho$);
    $\hat{\theta}[j] = \hat{\theta}[j] + E_{drft}$;
  end for
  $S'[j] = [x[j], y[j], \hat{\theta}[j]]^T$;
end for
Return(S')

Algorithm 5: Forward Translation with Noise; <u>rand_N</u>(M, σ) is a pseudo-random number generator drawing samples from a Normal distribution with mean M and standard deviation σ; procedures are noted as underlined text, Comments are inside curly brackets "{`comment`}". The variables M_{trs} and M_{drft} represent the mean error and are experimentally derived.

Figure 5.2 presents a graphical illustration of the effect of the two noise parameters $(\sigma_{trs}, \sigma_{drft})$ in the predictive model. In both cases the robot makes a single forward motion



(a) (b)

FIGURE 5.2. The effect of $\sigma_{trs}, \sigma_{drft}$ for the forward translation: (a) $\sigma_{trs} = 5cm/m, \sigma_{drft} = 1°/m$ (b) $\sigma_{trs} = 1cm/m, \sigma_{drft} = 5°/m$.

of 100cm (upper left sub-plot), 200cm (upper right sub-plot), 300cm (lower left sub-plot), and 400cm (lower right sub-plot). In Figure 5.2a the uncertainty in the distance traveled is the dominant uncertainty and thus the particles spread a lot more in the direction of the motion. In contrast, in Figure 5.2b, where the drift noise dominates, the particles spread in a circular pattern. Appendix B contains a detailed experimental study of these parameters using the Nomadic Technologies Superscout II mobile platform.

2.2. Resampling. One of the problems that appear with the use of particle filters is the depletion of the population after a few iterations. Most of the particles have drifted far enough for their weight to become too small to contribute to the *pdf* of the moving robot [5]. If we consider the current set of particles $S_k = \{\mathbf{x}_i^k, w_i^k\} : k = 1 \ldots M$ as a discrete representation of the *pdf* of the moving robot-pose, a new representation $S'_k = \{\mathbf{x'}_i^k, w'^k_i\} : k = 1 \ldots M$ is needed such that $\mathbf{x}_i^k = \mathbf{x'}_i^l$ for k, l in $[1, M]$ and weights ($w'^k_i = 1/M$) that represent the same *pdf*.

Liu *et al.* [**98**] refer to two different measures that estimate the number of *near-zero-weight* particles: one is the coefficient of variation cv_t^2 (see Equation 5.9) and the second is the effective sample size ESS_t (see Equation 5.10).

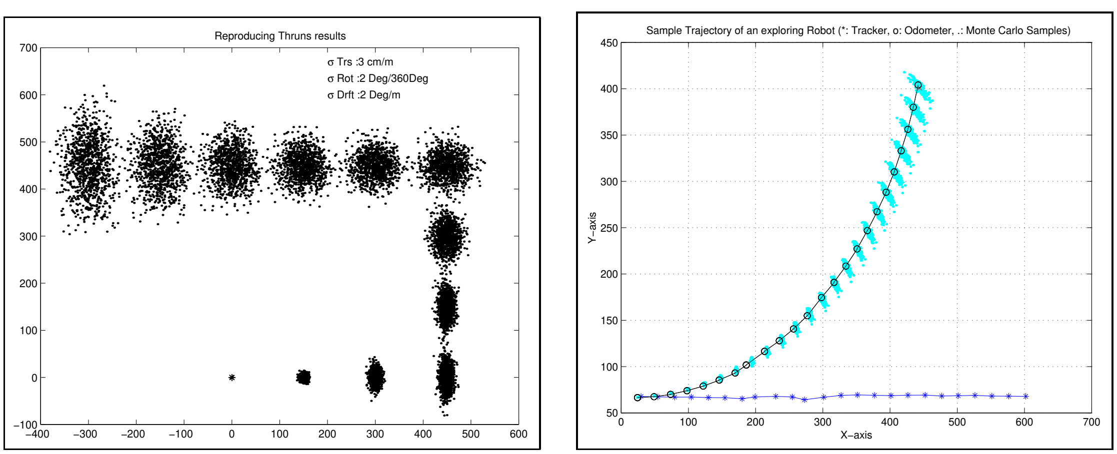
[5]For most practical implementations the weights become zero due to rounding off.

$$cv_t^2 = \frac{var(w_t(i))}{E^2(w_t(i))} = \frac{1}{M}\sum_{i=1}^{M}(Mw(i)-1)^2 \tag{5.9}$$

$$ESS_t = \frac{M}{1+cv_t^2} \tag{5.10}$$

When the effective sample size drops below a certain threshold, usually a percentage of the number of particles M, then the particle population is resampled, eliminating (probabilistically) the ones with small weights and duplicating the ones with higher weights.

Different methods have been proposed for resampling; three of the most common ones are discussed in Appendix C. In every case the input is an array of the weights of the particles and the output is an array of indices of which particles are going to propagate forward. The requirement is that the *pdf* reconstructed by the resampled population is very close to the one before the resampling. Experimental tests showed no noticeable improvements over the simple select with replacement scheme. In *Select with Replacement* each particle is selected to continue with a probability equal to its weight. We used the approach of Carpenter *et al.* [**27**] that runs in linear time in the number of particles (see Appendix C for a description of the algorithm).



(a) (b)

FIGURE 5.3. (a) Large trajectory, the uncertainty build up is represented by the spread of the particle cloud. (b) Series of forward translations and 360° rotations performed in our laboratory. The connected curved line represent the uncorrected odometer values (captured accurately by the cloud of particles), and the bottom line represents the actual trajectory.

Figure 5.3 presents two examples of complex motions and illustrates the performance of the prediction stage of the particle filter. In figure 5.3a, the robot moves forward three times, rotates ninety degrees, then translates forward three more times, after which it rotates again by ninety degrees and translates forward five times. As can be seen the uncertainty grows unbounded. Sub-figure 5.3b presents experimental validation of our predictive model. In this case the predictive model was guided by a set of motion commands that were used in an experiment in our laboratory (for the full description of this experiment please refer to chapter 8). In short, the experiment consisted of forward translations, each one followed by four rotations by ninety degrees (in order to sense the environment in four different directions). The connected circles in sub-figure 5.3b represent the uncorrected odometer values. In fact, the actual trajectory of the robot was kept in a straight line but the odometry estimates did deviate due to noise. The predictive model was constructed using the noise statistical parameters collected in our laboratory (see Appendix B). The predicted cloud of particles can be seen around the recorded values following the trajectory with high accuracy.

2.3. Update. After an action (the motion of one of the robots) the robot tracker sensor is employed in order to estimate the pose of the moving robot [6]. The calculations are dependent on the configuration of the robot tracker employed. The next two sections present the update of the weights of the particles of the moving robot for the laser/target robot tracker combination for two different cases. First we derive the update equations when the laser range finder is mounted on the stationary robot (subsection 2.3.1); second for when the laser range finder is mounted on the moving robot and the target is mounted on the stationary robot (subsection 2.3.2).

2.3.1. *Pose Estimation, stationary robot observing moving robot.* If the pose of the stationary robot $\mathbf{x}_s = [x_s, y_s, \hat{\theta}_s]^T$ (with laser range finder) and the pose of the moving robot $\mathbf{x}_m = [x_m, y_m, \hat{\theta}_m]^T$ (with target) are known, then the robot tracker sensor measurement $\mathbf{z} = [\rho, \hat{\theta}, \hat{\phi}]^T$ can be calculated by Equation 5.11:

[6]Additional sources of information (e.g. consistency of sensed parts of the environment with the map up to this point) can also be used during the update stage.

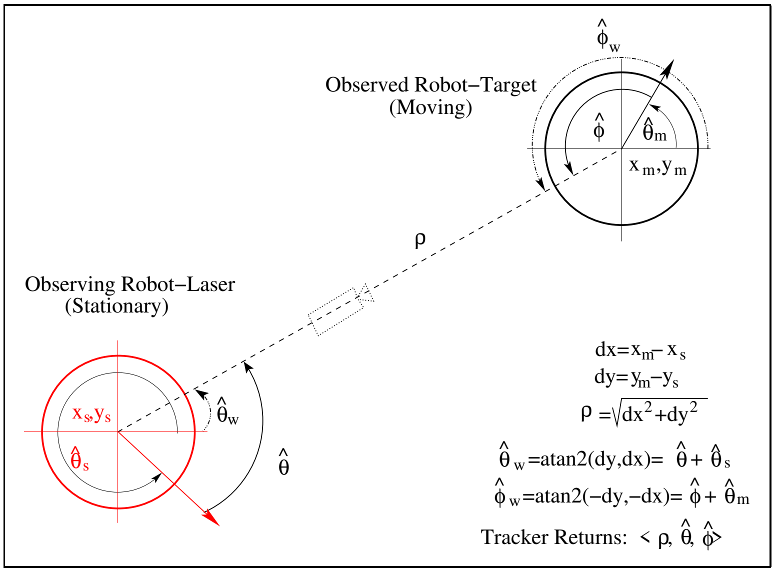


FIGURE 5.4. The stationary robot with the robot tracker sensor observes the moving robot that carries the target.

$$\mathbf{z} = \begin{bmatrix} \rho \\ \hat{\theta} \\ \hat{\phi} \end{bmatrix} = \begin{bmatrix} \sqrt{dx^2 + dy^2} \\ atan2(dy/dx) - \hat{\theta}_s \\ atan2(-dy/-dx) - \hat{\theta}_m \end{bmatrix} \tag{5.11}$$

where $dx = x_m - x_s$ and $dy = y_m - y_s$.

If the known information is the pose of the stationary robot ($\mathbf{x}_s$) (with the laser range finder) and the robot tracker measurement is ($\mathbf{z} = [\rho, \hat{\theta}, \hat{\phi}]^T$) then the *estimate* of the pose of the moving (target) robot ($\mathbf{x_{m_{est}}}(k+1)$) is given in Equation 5.12:

$$\mathbf{x_{m_{est}}}(k+1) = \begin{bmatrix} x_{m_{est}} \\ y_{m_{est}} \\ \hat{\theta}_{m_{est}} \end{bmatrix} = \begin{bmatrix} x_s + \rho \cos(\hat{\theta}_s + \hat{\theta}) \\ y_s + \rho \sin(\hat{\theta}_s + \hat{\theta}) \\ \pi + \hat{\theta} + \hat{\theta}_s - \hat{\phi} \end{bmatrix} \tag{5.12}$$

The Equations 5.11 and 5.12 are equivalent. Consequently, the above equations can be used in order to calculate the weight of each particle of the moving robot, assuming a Gaussian error model for each component of the sensor data $(\rho, \hat{\theta}, \hat{\phi})$, in two different ways. First, let the i^{th} particle at time k be $\mathbf{x}^k_{m_i} = [x_{m_i}, y_{m_i}, \hat{\theta}_{m_i}]^T$. Then if the pose of the stationary robot is known $\mathbf{x}_s = [x_s, y_s, \hat{\theta}_s]^T$ the estimated tracker measurement $\mathbf{z}_i$ for particle i is given in Equation 5.13:

$$\mathbf{z}_i = \begin{bmatrix} \rho_i \\ \hat{\theta}_i \\ \hat{\phi}_i \end{bmatrix} = \begin{bmatrix} \sqrt{dx_i^2 + dy_i^2} \\ atan2(dy_i, dx_i) - \hat{\theta}_s \\ atan2(-dy_i, -dx_i) - \hat{\theta}_{m_i} \end{bmatrix} \tag{5.13}$$

where $dx_i = x_{m_i} - x_s$ and $dy_i = y_{m_i} - y_s$.

The weight for particle i then is proportional to the probability of $\mathbf{x}_{m_i}^{k+1}$ given $\mathbf{x}_s$ and $\mathbf{z}_i$ (see Equation 5.14). As can be seen in Equation 5.13 the value of $\hat{\phi}_i$ is affected by the complete pose of particle i (both position and orientation). Therefore the error in position (x_{m_i}, y_{m_i}) is used twice. In Equation 5.14 the constants $\sigma_\rho, \sigma_{\hat{\theta}}, \sigma_{\hat{\phi}}$ are the presumed standard deviation of the robot trackers measurement noise and they signify the confidence with which we weight each measurement.

$$P(\mathbf{x}_{m_i}^{k+1}|\mathbf{x}_s, \mathbf{z}) = \frac{1}{\sqrt{2\pi}\sigma_\rho} e^{\frac{-(\rho-\rho_i)^2}{2\sigma_\rho^2}} \frac{1}{\sqrt{2\pi}\sigma_{\hat{\theta}}} e^{\frac{-(\hat{\theta}-\hat{\theta}_i)^2}{2\sigma_{\hat{\theta}}^2}} \frac{1}{\sqrt{2\pi}\sigma_{\hat{\phi}}} e^{\frac{-(\hat{\phi}-\hat{\phi}_i)^2}{2\sigma_{\hat{\phi}}^2}} \tag{5.14}$$

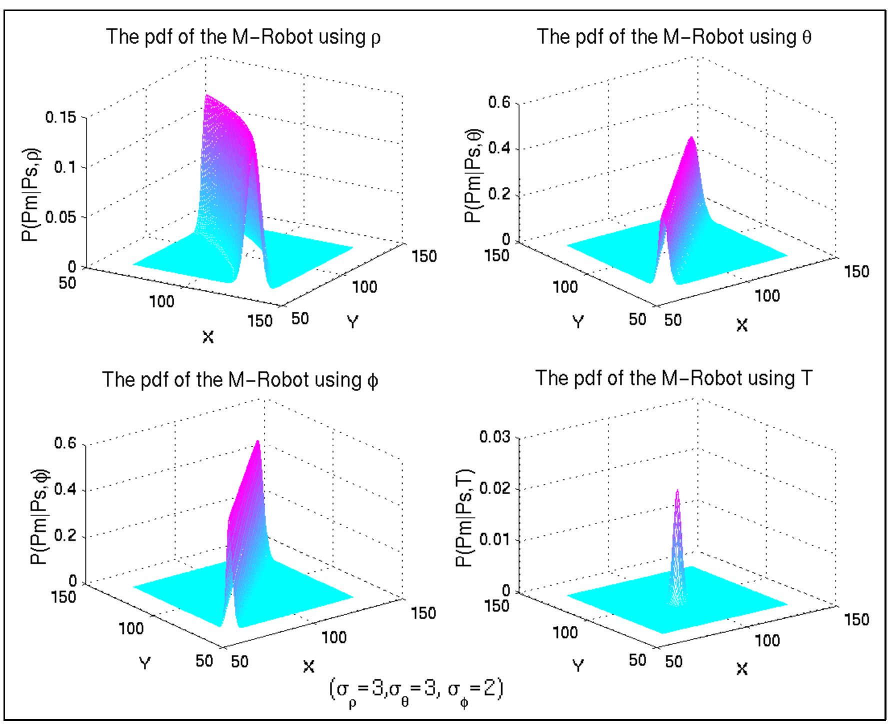


FIGURE 5.5. The contribution of each measurement of the robot tracker in the weighting *pdf* of the moving robot.

Figure 5.5 illustrates the spatial variation of Equation 5.14. In particular the spatial variation of the contribution of each component $(\rho, \hat{\theta}, \hat{\phi})$ to the weighting function is presented in the first three sub-plots, and the spatial variation of the weighting function is presented on the lower right sub-plot. For clarity of presentation, the pose of the observing robot is set at $\mathbf{x}_s = [0,0,0]^T$ and the pose of the moving robot at $\mathbf{x}_m = [100,100,45]^T$, and using Equation 5.11 the tracker measurement $\mathbf{z} = [\rho, \hat{\theta}, \hat{\phi}]^T$ is calculated. Then the spatial variation of the different terms of the product in Equation 5.14 is plotted keeping the moving robots orientation at the correct value (45°).

Experimental results have shown that the accuracy of the position of the robot is (*almost*) fixed (independent of the distance at which the observed robot is seen). Unfortunately, the tracker measurements are in polar coordinates and thus for a fixed error in the angle $(\hat{\theta})$ the longer the distance (ρ) the higher the error. In practice, it is necessary to calculate $\sigma_{\hat{\theta}}$ as a function of ρ:

$$\sigma_{\hat{\theta}} = h(\rho, \sigma_{\hat{\theta}}) = asin(\sigma_{\hat{\theta}}/\rho) \tag{5.15}$$

If the $\sigma_{\hat{\theta}}$ is kept at a fixed value then the weighting function is spread out, as can be seen in the upper right sub-plot in Figure 5.6, and the prediction is less accurate. Figure 5.6 presents the spatial variation of the weighting function for the same condition as in Figure 5.6, except $\sigma_{\hat{\theta}}$ is not scaled.

An alternative weighting function is to use the difference in Cartesian coordinates and the orientation estimate in order to weight the particle $\mathbf{x}_{m_i}^k$ given $\mathbf{x}_s$ and $\mathbf{z}_i$ (see Equation 5.16).

$$P(\mathbf{x}_{m_i}^{k+1}|\mathbf{x}_s, \mathbf{z}) = \frac{1}{\sqrt{2\pi}\sigma_\rho} e^{\frac{-(dx-dx_i)^2}{2\sigma_\rho^2}} \frac{1}{\sqrt{2\pi}\sigma_\rho} e^{\frac{-(dy-dy_i)^2}{2\sigma_\rho^2}} \frac{1}{\sqrt{2\pi}\sigma_{\hat{\phi}}} e^{\frac{-(\hat{\theta}_m-\hat{\theta}_{m_i})^2}{2\sigma_{\hat{\theta}'}^2}} \tag{5.16}$$

The second approach is to use Equation 5.12 and weight every particle depending on how far it is from the estimated pose of the moving robot (see Equation 5.17). Where if $\mathbf{x_{m_{est}}}(k+1) = [x_{m_{est}}, y_{m_{est}}, \hat{\theta}_{m_{est}}]$ is the estimate pose and $\mathbf{x}_{m_i}^k = [x_{m_i}, y_{m_i}, \hat{\theta}_{m_i}]^T$ is the "i^{th}" particle then $d_i = \sqrt{(x_{m_{est}} - x_{m_i})^2 + (y_{m_{est}} - y_{m_i})^2}$. The disadvantage of this approach is that $\sigma_d, \sigma_{\hat{\theta}}$ do not represent the sensor's noise model.

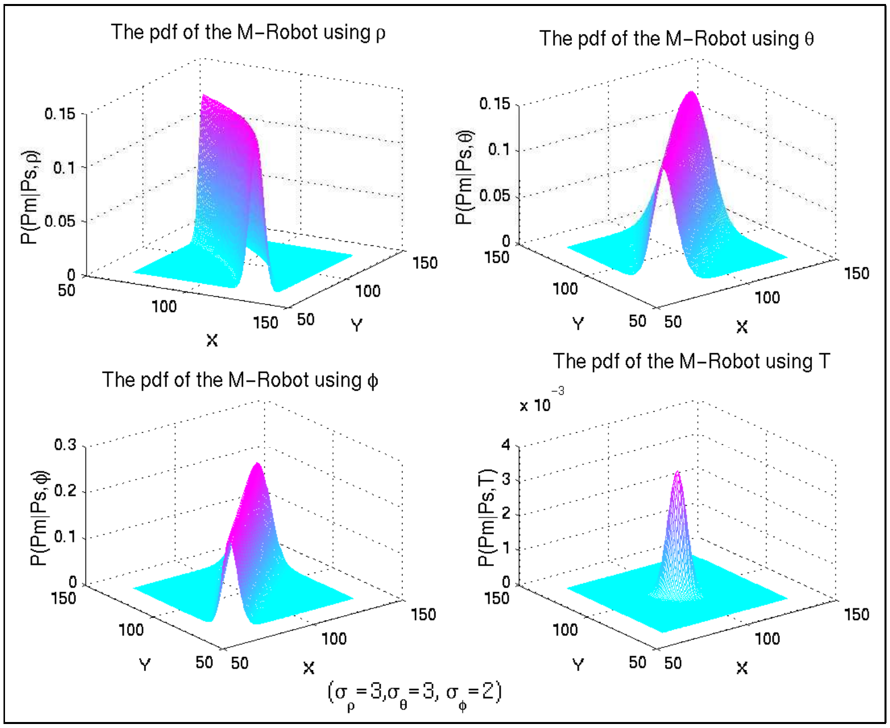


FIGURE 5.6. The contribution of each measurement of the robot tracker in the weighting *pdf* of the moving robot. In contrast to Figure 5.5 the $\sigma_{\hat{\theta}}$ is not calculated to be proportional to distance between the two robots.

$$P(\mathbf{x}_{m_i}^{k+1}|\mathbf{x}_s,\mathbf{z}) = \frac{1}{\sqrt{2\pi}\sigma_d} e^{\frac{-(d_i)^2}{2\sigma_d^2}} \frac{1}{\sqrt{2\pi}\sigma_{\hat{\theta}}} e^{\frac{-(\hat{\theta}_{m_{est}}-\hat{\theta}_{m_i})^2}{2\sigma_{\hat{\theta}}^2}} \tag{5.17}$$

During the estimation of the weight the pose of the stationary robot $\mathbf{x}_s$ is used. As the actual pose is not known, different estimates $\bar{\mathbf{x}}_s$ can be employed. The following options have been considered:

- The best particle (the one with maximum weight):

$$\bar{\mathbf{x}}_s = \mathbf{x}_s^{max}$$

- Weighted Mean:

$$\bar{\mathbf{x}}_s = \sum_{j=1}^{M} \mathbf{x}_j^s w_j$$

- Use every particle of the stationary robot ($O(n^2)$):

$$P(\mathbf{x}_{m_i}^{k+1}|\mathbf{x}_s,\mathbf{z}) = \Sigma_{j=1}^{n} P(\mathbf{x}_{m_i}^{k+1}|\mathbf{x}_s^j,\mathbf{z})$$

- Robust Mean: Select only the particles that are less than ϵ from the particle with maximum weight. The advantage of this method is that it selects the mode of the distribution and reduces the discretization error (which occurs when only a single particle is used).

$$\bar{\mathbf{x}}_s = \sum_{j=1}^{K} \mathbf{x}_s^j w_j : |\mathbf{x}_s^j - \mathbf{x}_s^{max}| \leq \epsilon$$

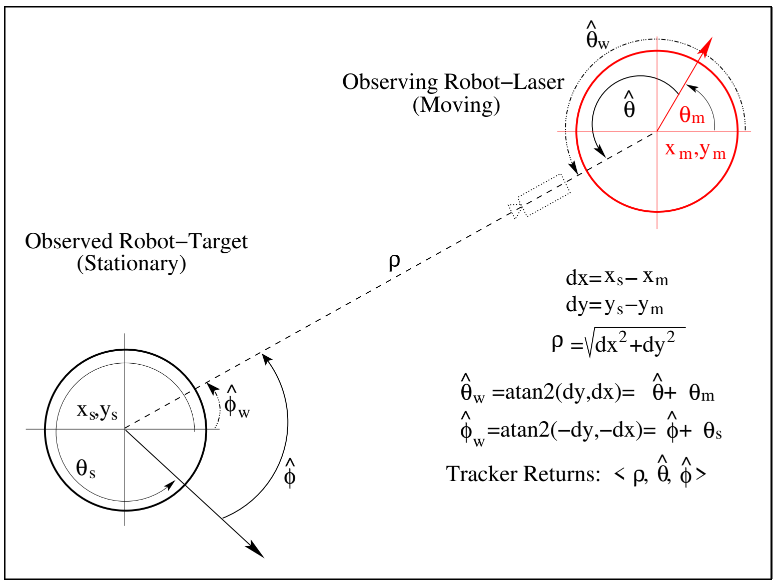


FIGURE 5.7. Observation.

2.3.2. *Pose Estimation, moving robot observing stationary robot.* This time the stationary robot has the target. If the poses of the two robots ($\mathbf{x}_s = [x_s, y_s, \hat{\theta}_s]^T$ and $\mathbf{x}_m = [x_m, y_m, \hat{\theta}_m]^T$) are known then the robot tracker sensor measurement ($\mathbf{z} = [\rho, \hat{\theta}, \hat{\phi}]^T$) can be calculated by Equation 5.18 (exactly as in the previous case Equation 5.11).

$$\begin{bmatrix} \rho \\ \hat{\theta} \\ \hat{\phi} \end{bmatrix} = \begin{bmatrix} \sqrt{dx^2 + dy^2} \\ atan2(dy, dx) - \hat{\theta}_m \\ atan2(-dy, -dx) - \hat{\theta}_s \end{bmatrix} \tag{5.18}$$

where $dx = x_s - x_m$ and $dy = y_s - y_m$ [7].

[7] Note that dx, dy are different from Equation 5.11.

If the pose of the stationary robot ($\mathbf{x}_s$) (carrying the target) and the robot tracker measurement ($\mathbf{z} = [\rho, \hat{\theta}, \hat{\phi}]^T$) are known then the *estimate* of the pose of the moving (carrying the laser) robot ($\mathbf{x}_m$) is given in Equation 5.19.

$$\mathbf{x_{m_{est}}}(k+1) = \begin{bmatrix} x_{m_{est}} \\ y_{m_{est}} \\ \hat{\theta}_{m_{est}} \end{bmatrix} = \begin{bmatrix} x_s + \rho * \cos(\hat{\phi} + \hat{\theta}_s) \\ y_s + \rho * \sin(\hat{\phi} + \hat{\theta}_s) \\ \pi + \hat{\phi} + \hat{\theta}_s - \hat{\theta} \end{bmatrix} \tag{5.19}$$

Applying the same methodology as in the previous section the weight update functions are identical with the ones in Equations 5.14, 5.16, 5.17.

Figure 5.8 presents an illustration of the above described process over two iterations. The first column present the *prediction* phase and the second column the *update* phase. The moving robot starts at position [0,0], and the stationary robot is located at [0,100]. At figure 5.8a the moving robot moves by [100cm,100cm] and the particles form a cloud of approximately 20cm in radius. Figure 5.8b presents the update phase based on the tracker sensor measurement (darker color represents higher weights). At the second step the robot moves by [100cm,-100cm] and figure 5.8c presents the cloud of particles. It is worth noting that the particles with higher weights (darker grey) have spread out. Finally, figure 5.8d presents the second update phase where again the particles closer to the sensed pose have higher weights.

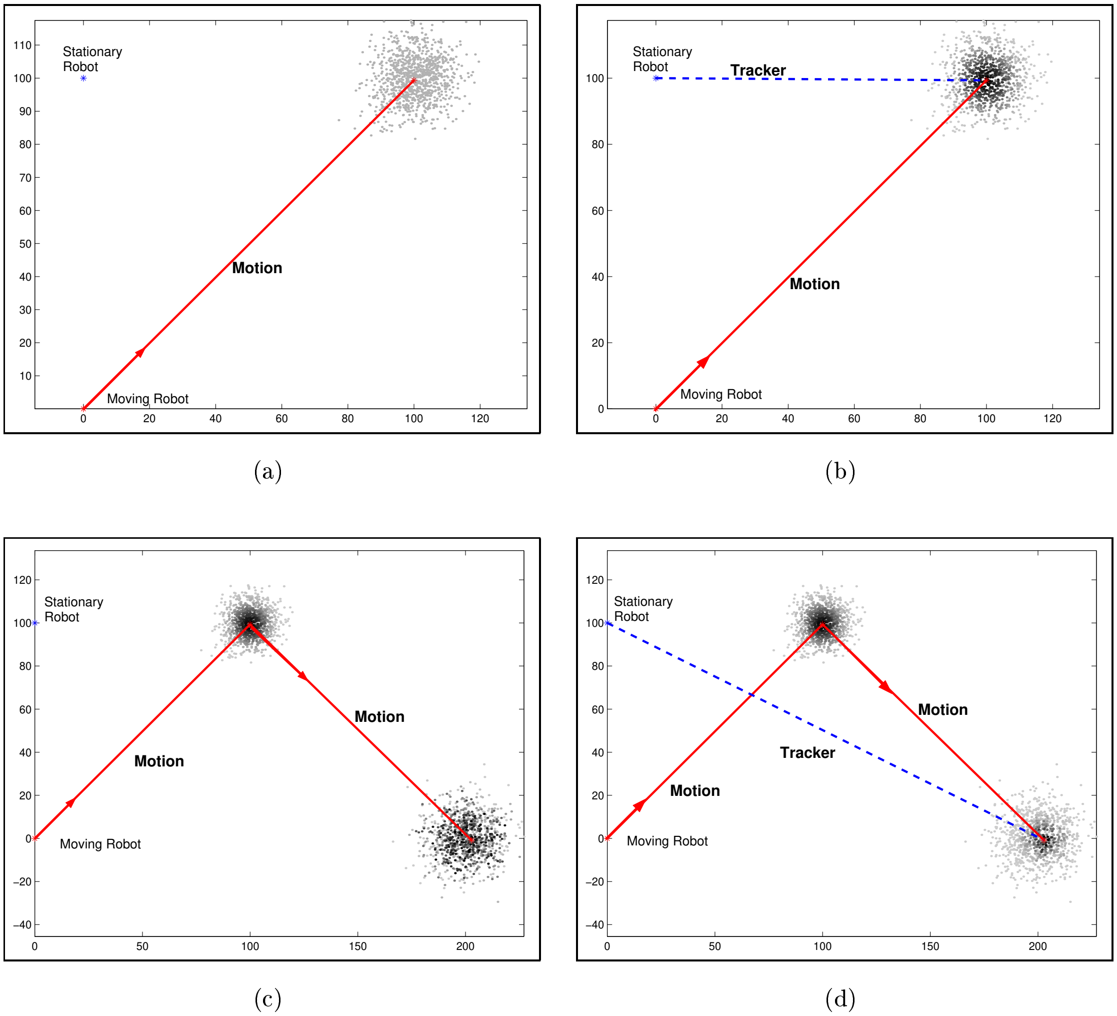


FIGURE 5.8. (a)*Prediction* of the first step. (b) *Update* using the robot tracker. (c) *Prediction* of the second step. (d) *Update* using the robot tracker.

CHAPTER 6

Robot Tracker Implementation

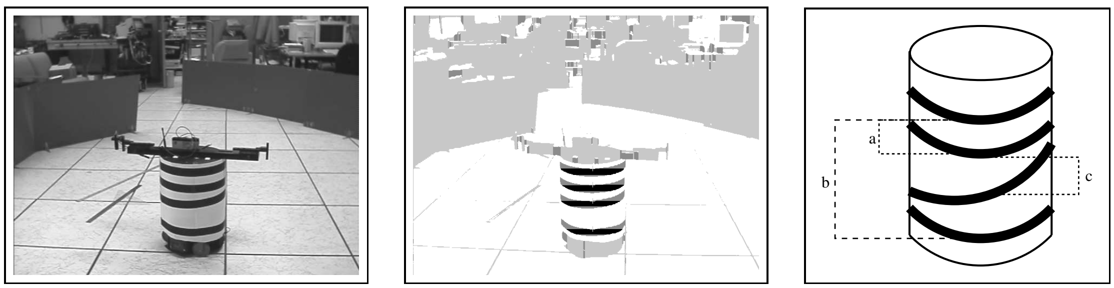
1. Introduction

As we saw in the previous chapters, central to our approach is the role of the robot tracker sensor, which serves the dual role of estimating the pose of the moving robot and of detecting obstacles between the two robots. This chapter is independent of the rest of the thesis and discusses the implementation details for the construction of two different robot tracker sensors. Section 2 presents the implementation details for the construction of a vision-based robot tracker sensor together with an analysis of its accuracy. Section 3 discusses an improved robot tracker sensor based on a laser range finder and a geometric target. The laser sensor calibration and the target calibration are discussed in this section together with an error analysis.

2. Visual Tracker

A variety of sensing technologies could be used for the robot tracker. Our first implementation of the robot tracker sensor was based on visual observation of a helical target pattern on the other robot [**50**]. One robot is equipped with a camera that allows it to observe its partner. The other robot is marked with a special pattern for pose estimation.

Several possible designs for the pattern are possible, but they should satisfy two key requirements: the pattern should be robustly detectable and it should allow the distance and orientation of the robot carrying it to be estimated. In this particular implementation, the first part of the pattern is a series of horizontal circles that project into an almost linear pattern in the image. This allows the robot to be easily discriminated from background objects: the particular ratio of distances between the circles (b/a in Figure 6.1c) is extremely unlikely to occur in the background by chance. Thus, the presence of the robot is established by a set of lines (curves) with the appropriate length-to-width ratio, and the appropriate inter-line ratios. The second component of the pattern is a helix that wraps once around the robot. The elevation of the center of the helix, relative to the surrounding bands (c/b in Figure 6.1c), allows the relative orientation of the robot to be inferred (see Figure 6.2a, 6.1). In practice, this allows the robot's pose to be inferred with an accuracy of a few centimeters in position and 3 to 5 degrees in heading.



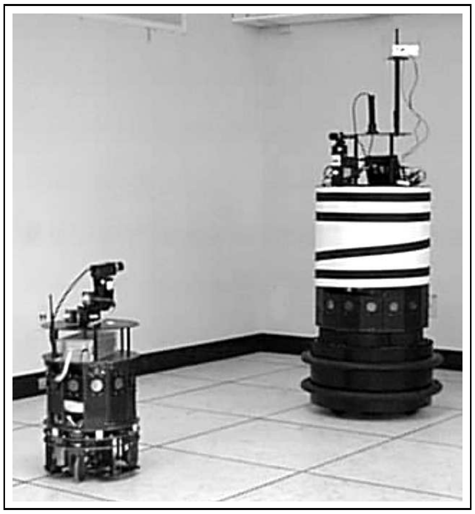
(a) (b) (c)

FIGURE 6.1. Robot Tracker: (a) The raw image of the moving robot as observed by the robot tracker. (b) The helical and cylindrical pattern detected in the image. (c) The distances estimated from the helix.

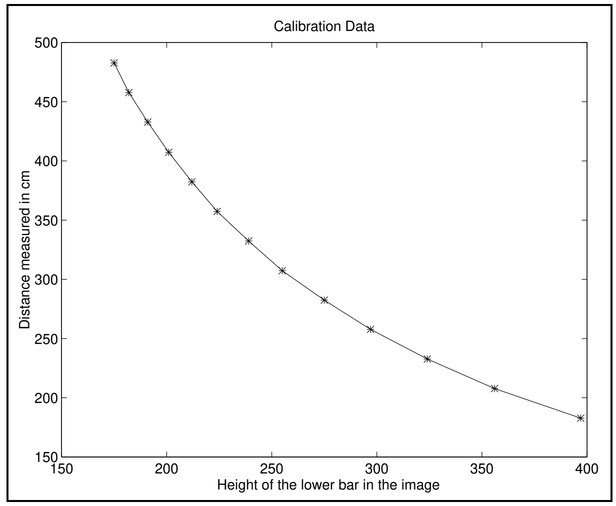
By mounting the observing camera above (or below) the striped pattern of the other robot, the distance from one robot to the other can be inferred from the height of the stripe pattern in the image, due to perspective projection (scaling of the pattern could also be used). The difference in height between the observing camera and the target can be selected to provide the desired tradeoff between range of operation and accuracy. One advantage of the helical target for orientation estimation is that it functions correctly even at very large distances, although with reduced accuracy, of course [1].

2.1. Tracker Evaluation. The accuracy of the visual tracker is shown in Table 6.1. While the relationship between the appearance of the target and the actual distance can be computed analytically, this would presuppose an accurate knowledge of the camera parameters. In order to relax this requirement, as well as to accommodate potential deviations

[1]Note that constraints due to specific task (such as mine sweeping) can sometime introduce additional constraints on the maximum inter-robot separation or optimal sensor geometry.



(a)



(b)

FIGURE 6.2. (a) The visual robot tracker system with the camera mounted on one robot and the helical target pattern mounted on the second robot. (b) Calibration data for the distance estimation relating observed image position to actual distances.

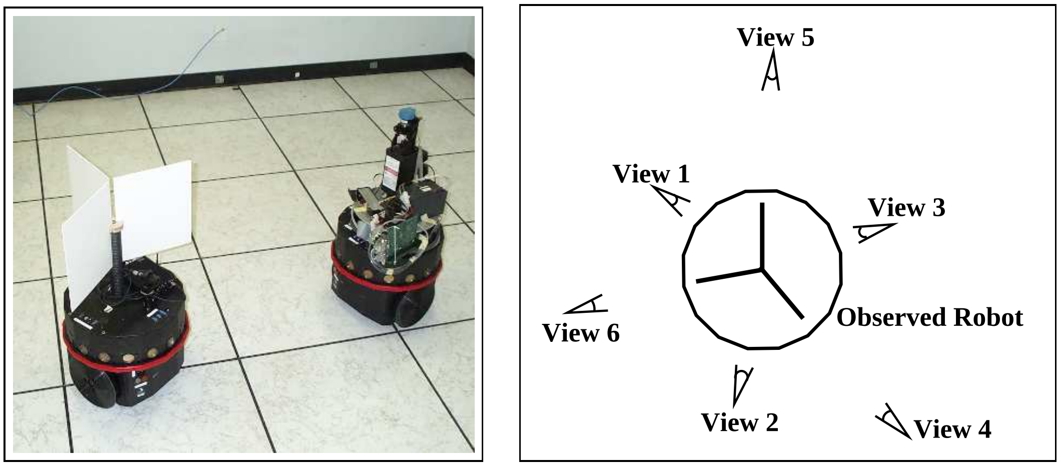
from our ideal camera model, we use experimental calibration data to relate observed target positions with actual ranges. Calibration data relating the projected image and the distance estimates is shown in Figure 6.2b. It is possible to estimate distances between roughly 180 and 450 cm with the camera configuration used in this experiment, although accuracy degrades with increasing distance (due to decreased image resolution) [**137**].

Distance Accuracy	1.5cm/pixel
Orientation Accuracy	1.3°

TABLE 6.1. Accuracy of simple visual tracker

3. Laser Tracker

The current implementation uses a laser range finder on the observing robot and a target with a distinct three dimensional pattern mounted on the observed robot. We implemented a robot tracker using a time of flight laser range finder (AccuRange 4000 from Acuity Research Inc.) mounted on the observing robot and a three plane target mounted on the observable robot (see Figure 6.4). The target was developed for this work in order



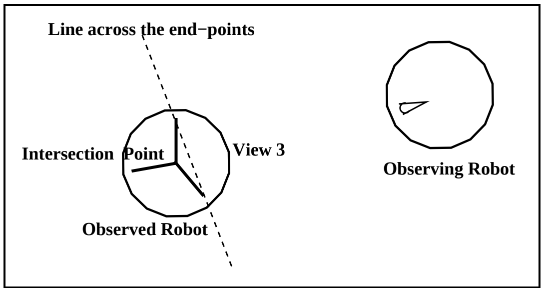
(a) (b)

FIGURE 6.3. (a)The two robots: the observed robot with white target on the left and the observing robot with the laser range finder on the right. (b) Top view of the observed robot 6 canonical views that are qualitative different depending on the position of the observing robot.

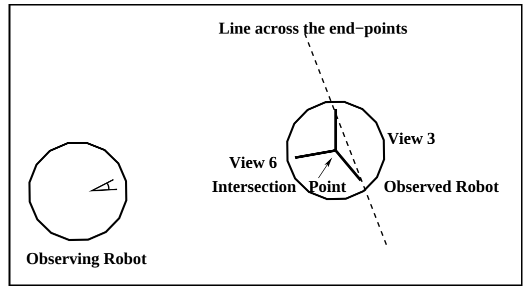
to permit accurate and robust estimation of position and orientation using the AccuRange line scanner. The target can be seen in Figure 6.4a (the robot on the left) while the robot on the right carries the laser sensor.

The target used consists of three vertical planes that fan out of the center at $100°, 120°$ and $140°$ angles. Figure 6.4b shows a top view of the target: as can be seen, from any point around the robot at least two planes are always visible. The different combinations (in Figure 6.4b) are numbered as View 1 to View 6. The intersection of the laser stripe with the target gives rise to at least two line segments in 3D. The laser range finder takes a dense laser scan (4096 points), and the laser points are filtered and only those near the last known location of the observed robot are kept. During the next step lines are fitted using the MacKenzie-Dudek algorithm [**52, 107**] and lines segments that do not match the length and/or the appropriate angles of the planes are discarded. From the remaining lines, at most three, the two longest are selected and their intersection point and their orientation is calculated [2].

[2]The probability of two straight lines of length 25cm meeting at an angle of $140°$ plus or minus two degrees is extremely low.



(a)



(b)

FIGURE 6.4. Top view of the observing/observed robots. For illustration purposes we examine the case where the two planes meet at a 140°, view 3 and view 6. (a) *View 3* 140 degrees angle between the two target lines; the observing robot and the intersection point are on opposite sides of the line that passes between the two end points of the target. (b) *View 6* Same angle (140 degrees) but the observing robot and the intersection point are in the same side.

The view numbering is done to efficiently estimate the view observed, first the angle between the two lines is calculated (one of $100°, 120°$ or $140°$) and the view is given a number 1-3 accordingly. Then the side of robot from where the two lines are seen is determined (if they form a convex or concave corner relative to the observing robot). This is achieved by calculating whether the intersection point of the two lines falls on the same side (see Figure 6.4b) or the opposite sides (see Figure 6.4a) of the line that passes through the end-points of the two target lines. If a convex corner is detected (observing robot and intersection point at the same side) then the view number is increased by three (see the numbering at Figure 6.3b). The detected target data consist of a quadruple $T = [x_t, y_t, \theta_{t_1}, \theta_{t_2}]$ containing the center of the target ($[x_t, y_t]$) and the orientations of the two planes ($[\theta_{t_1}, \theta_{t_2}]$). In case of failure (i.e. large odometry error) then all the laser points are considered, lines are fitted again and the above described process is repeated.

3.1. Laser Sensor Accuracy. As noted earlier the laser sensor returns a dense scan of 4096 points over 360 degrees (one point per 0.0879 degree) up to 15m distance. Table 6.2 shows the uncertainty in cm and degrees for different size landmarks observed 100 times at a distance of 300cm. The standard deviation of the measurements is reported together with the maximum difference in the measurements. Two walls were used as the landmark providing similar measurements with the target; the intersection point of the two walls and the average orientation of the two walls was measured. The distance of the landmark from the observing robot is reported together with the angle at which the landmark is seen (the polar coordinates of the landmark to the robot) and also the orientation of the two walls. It can be seen that the larger the landmark the more consistent are the observations, but even for a landmark as small as the target the accuracy is high.

Length of Landmark		25cm	30cm	50cm	100cm	150cm	200cm
Distance (cm)	Max Diff.	0.4175	0.3416	0.3765	0.1946	0.2057	0.1602
Distance (cm)	STD	0.1042	0.0751	0.0784	0.0521	0.0432	0.0333
Angle	Max Diff.	0.0529°	0.0399°	0.0245°	0.0213°	0.0289°	0.0290°
Angle	STD	0.0084°	0.0075°	0.0046°	0.0037°	0.0066°	0.0055°
Orientation	Max Diff.	1.0420°	1.0240°	0.3530°	0.1450°	0.1140°	0.0940°
Orientation	STD	0.2124°	0.1582°	0.0768°	0.0316°	0.0269°	0.0205°

TABLE 6.2. Maximum Difference and standard deviation of observing different sized landmarks. Distance is measured in centimeters and the angles in degrees.

3.2. Laser Sensor Calibration. The laser sensor returns data points (P_l) in coordinates in the sensor's frame of reference (LC). Naturally, the robot-mounted sensor returns readings with respect to the robot base. We proceed by transforming these measurements into a common global reference frame (WC). The transformation from the laser frame of reference to the global frame of reference consists of two stages: first from the laser sensor coordinate system to a coordinate system located at the robot and then from the robot's coordinate system to the world coordinate system. More formally, $P_w = M_{wr}M_{rl}P_l$, where P_w is a point in WC, P_l is a point in LC, both points are expressed in 2D homogeneous coordinates, M_{wr} is the transformation matrix from the robot coordinate frame to the world coordinate frame, and M_{rl} is the transformation matrix from the laser coordinate frame to the robot-centered coordinate frame (see Equation 6.1). Matrix M_{wr} expresses the transformation based on the pose of the robot ($P_{robot} = [x_r, y_r, \theta_r]^T$) in world coordinates and varies with the motion of the robot. The matrix M_{rl} expresses the transformation based on the distance and the orientation of the laser sensor coordinate system relative to the coordinate system of the robot (x, y, θ). As the laser sensor is securely mounted on the robot the values x, y, θ are constant. In order to calibrate the laser sensor to return values in the world coordinate frame, the values of x, y, θ should be estimated. We use an environment with known geometry (see Figure 6.5) and we estimate these values in two stages: first we estimate the difference in the orientation θ, and then the displacement x, y. The numerical results are obtained for a Superscout robot from Nomadic Technologies, Inc (see Figure 6.5).

FIGURE 6.5. An environment with four walls.

$$\underbrace{\begin{pmatrix} x_w \\ y_w \\ 1 \end{pmatrix}}_{P_w} = \underbrace{\begin{bmatrix} \cos(\theta_r) & -\sin(\theta_r) & x_r \\ \sin(\theta_r) & \cos(\theta_r) & y_r \\ 0 & 0 & 1 \end{bmatrix}}_{M_{wr}} \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix}}_{M_{rl}} \underbrace{\begin{pmatrix} x_l \\ y_l \\ 1 \end{pmatrix}}_{P_l} \tag{6.1}$$

Orientation (θ) calibration: The robot detects the same landmarks from two different positions. For ease of calculation we set the pose of the robot to be the same as the origin of the world coordinate system ($P1 = [x_r, y_r, \theta_r]^T = [0,0,0]^T$). The intersection of two walls (in laser coordinates $A_l = [A_{x_l}, A_{y_l}]^T$) is calculated, then the robot translates by a distance D along the x-axis moving to the pose $P2$ ($P2 = [D,0,0]^T$) [3], and the same landmark (intersection of the same two walls) is detected ($B_l = [B_{x_l}, B_{y_l}]^T$). [4] The world coordinates of the observed landmark should be the same before and after the motion ($A_w = B_w$) and from Equation 6.1 we could estimate the orientation (see Equation 6.2) [5].

$$
\begin{aligned}
A_w = \begin{bmatrix} A_{x_w} \\ A_{y_w} \\ 1 \end{bmatrix} = \begin{bmatrix} B_{x_w} \\ B_{y_w} \\ 1 \end{bmatrix} = B_w & \quad \Longleftrightarrow \\
\begin{bmatrix} \cos(0) & -\sin(0) & 0 \\ \sin(0) & \cos(0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A_{x_l} \\ A_{y_l} \\ 1 \end{bmatrix} & \quad = \\
\begin{bmatrix} \cos(0) & -\sin(0) & D \\ \sin(0) & \cos(0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} B_{x_l} \\ B_{y_l} \\ 1 \end{bmatrix} & \quad \Longleftrightarrow \\
\begin{bmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A_{x_l} \\ A_{y_l} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & D \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} B_{x_l} \\ B_{y_l} \\ 1 \end{bmatrix} & \quad \Longleftrightarrow \\
\begin{bmatrix} (A_{x_l} - B_{x_l}) & (-A_{y_l} + B_{y_l}) \\ (A_{y_l} - B_{y_l}) & (A_{x_l} - B_{x_l}) \end{bmatrix} \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} = \begin{bmatrix} D \\ 0 \end{bmatrix} &
\end{aligned}
\tag{6.2}
$$

After repeating the above described experiment n times we collect a set of landmarks and distances traveled $[A^i_{x_l}, A^i_{y_l}, B^i_{x_l}, B^i_{y_l}, D^i] : \ i = 1 \ldots n$. From the data and Equation 6.2 we use a least squares fit for Equation 6.3. A different approach is to analytically calculate

[3] The poses $P1, P2$ of the robot are expressed as a triplet (x, y, θ) while the detected landmarks are in 2d homogeneous coordinates.

[4] During the translation the robot speed is kept low in order to achieve reliable odometry measures.

[5] As can be seen the unknowns x, y are canceled out and we have two equation with two unknowns (although they are not independent) $[\cos(\theta), \sin(\theta)]$.

$[\cos(\theta), \sin(\theta)]^T$ for each data point (see Equation 6.4) and then take the mean values. Both methods were used for robustness.

$$\underbrace{\begin{bmatrix} (A^1_{x_l} - B^1_{x_l}) & (-A^1_{y_l} + B^1_{y_l}) \\ (A^1_{y_l} - B^1_{y_l}) & (A^1_{x_l} - B^1_{x_l}) \\ (A^2_{x_l} - B^1_{x_l}) & (-A^2_{y_l} + B^2_{y_l}) \\ (A^2_{y_l} - B^1_{y_l}) & (A^2_{x_l} - B^2_{x_l}) \\ \vdots & \vdots \\ (A^n_{x_l} - B^n_{x_l}) & (-A^n_{y_l} + B^n_{y_l}) \\ (A^n_{y_l} - B^n_{y_l}) & (A^n_{x_l} - B^n_{x_l}) \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}}_{\mathrm{X}} = \underbrace{\begin{bmatrix} D^1 \\ 0 \\ D^2 \\ 0 \\ \vdots \\ D^n \\ 0 \end{bmatrix}}_{\mathrm{b}} \tag{6.3}$$

$$\begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} = \begin{bmatrix} \frac{D(A_{x_l} - B_{x_l})}{(A_{x_l} - B_{x_l})^2 + (A_{y_l} - B_{y_l})^2} \\ \frac{D(-A_{y_l} + B_{y_l})}{(A_{x_l} - B_{x_l})^2 + (A_{y_l} - B_{y_l})^2} \end{bmatrix} \tag{6.4}$$

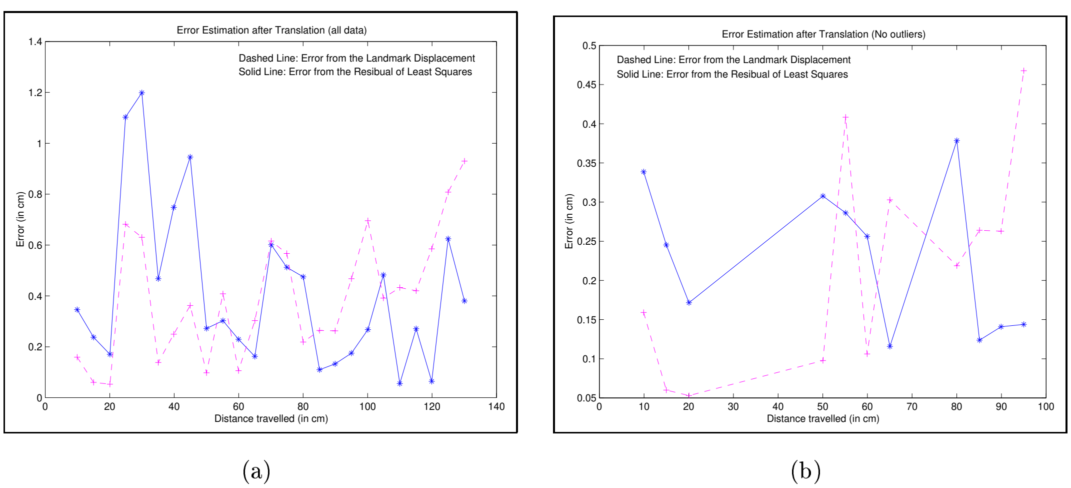


FIGURE 6.6. Error estimation after translation (a) Using all the data. (b) Without any outliers.

Two different ways to measure the quality of the data are used. First, from the solution of the inverse problem ($\hat{X} = \mathbf{A}^{-1}\mathrm{b}$) we can estimate the residual ($r = \mathbf{A} \cdot \hat{X} - \mathrm{b}$ see Equation 6.3) which provides a measure of the quality of the data (the residual can be seen in Figure 6.6 as a solid line). Second, the difference between the distance traveled and the displacement of the observed landmark (in LC) provides a measure of consistency in the data (as can be seen in Figure 6.6 as a dashed line). It is worth noting that the two measures eliminate the same outliers from the data. Both methods were used in order to estimate the orientation (θ) of the Laser Coordinate system relative to the Robot Coordinate system: analytical calculations and a least square estimation. Table 6.3 presents the values of $[\cos(\theta), \sin(\theta)]$ calculated with the two methods using all data (no outliers removed). Table 6.4 presents the results after the removal of outliers.

	$cf = \cos(\theta)$	$sf = \sin(\theta)$	$F = arctan(sf/cf)$	$\cos(F)$	$\sin(F)$
	Analytical Calculations (With outliers)				
Mean	-0.7428	-0.6710	-137.9019	-0.7419	-0.6704
STD	0.0149	0.0044	0.7208	0.0084	0.0093
	Least Square Calculation (With outliers)				
	-0.7471	-0.6712	-138.0637	-0.7439	-0.6683

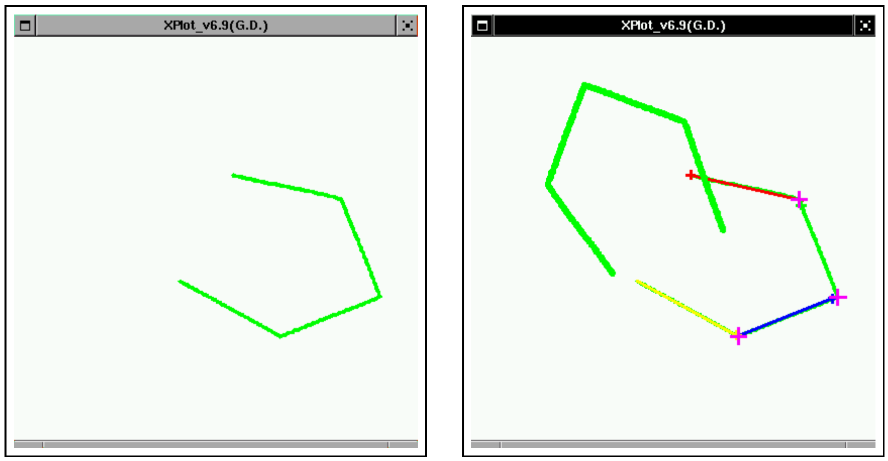
TABLE 6.3. Orientation estimation with two different methods (analytical and least squares estimation). For the analytical calculation the mean value and the standard deviation is reported. The estimates contain all data collected (no outliers removed).

	$cf = \cos(\theta)$	$sf = \sin(\theta)$	$F = arctan(sf/cf)$	$\cos(F)$	$\sin(F)$
	Analytical Calculations (No outliers)				
Mean	-0.7478	-0.6692	-138.1733	-0.7451	-0.6668
STD	0.0116	0.0056	0.6656	0.0077	0.0087
	Least Square Calculation (No outliers)				
	-0.7473	-0.6699	-138.1240	-0.7446	-0.6675

TABLE 6.4. The same as Table 6.3 but with the outliers removed.

With an error margin of two tenths of a degree the orientation of the Laser Coordinate system with respect to the robot is $\theta = -138.1240$.

Displacement (x, y): Following the same methodology as in the orientation estimation we use the rotation of the robot in order to estimate the other two unknowns x, y in Equation 6.1. The pose of the robot again is set at the origin. The robot detects three landmarks (in laser coordinates A_l, C_l, E_l, see Figure 6.7a and Figure 6.5), then the robot is rotated by an angle θ_r, and the same landmarks are detected again (B_l, D_l, F_l, see Figure 6.7b). Given a pair of landmarks $\mathbf{A} = A_l - B_l$ and $\mathbf{C} = C_l - D_l$ (each landmark observed from two different angles) we can calculate the robot orientation θ_r (see Equation 6.6). Consequently, given an estimate for $[\cos(\theta), \sin(\theta)] = [cf, sf] = [-0.7446, -0.6675]$ and Equation 6.1, we get Equation 6.5 for the first landmark $\mathbf{A} = A_l - B_l$ (same Equation for the other two landmarks $\mathbf{C} = C_l - D_l$ and $\mathbf{E} = E_l - F_l$).



(a) (b)

FIGURE 6.7. The four walls providing three landmarks. (a) Before the rotation. (b) After the rotation.

$$\begin{bmatrix} cf & -sf & x \\ sf & cf & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A_{x_l} \\ A_{y_l} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_r) & -\sin(\theta_r) & x_r \\ \sin(\theta_r) & \cos(\theta_r) & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cf & -sf & x \\ sf & cf & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} B_{x_l} \\ B_{y_l} \\ 1 \end{bmatrix} \tag{6.5}$$

From the cosine law [6] we can calculate the rotation of the robot θ_r using a pair of landmarks. For example, using the landmarks **A** and **C** we get Equation 6.6.

$$\begin{aligned} \cos\theta_r &= \frac{(Ax - Cx)(Bx - Dx) + (Ay - Cy)(By - Dy)}{(Bx - Dx)^2 + (By - Dy)^2} \\ \sin\theta_r &= \frac{(Ay - Cy)(Bx - Dx) + (Ax - Cx)(By - Dy)}{(Bx - Dx)^2 + (By - Dy)^2} \end{aligned} \tag{6.6}$$

[6] Mathematica was used for solving and simplifying the equations.

Using the two of the three landmarks at a time ([**A**, **C**], [**A**,**E**], [**C**,**E**]), we get a robust estimate of the rotation of the robot. Figure 6.8 presents the distribution of the estimates (top sub-plot), as well as the difference between the estimates of the rotation (using landmarks) and the odometry estimate (bottom sub-plot). More precisely, the top sub-plot presents the standard deviation of the orientation estimates from the landmarks (in "+") and the difference between the maximum and the minimum values (in "*"). The lower sub-plot presents the difference of the odometry estimate of the orientation from the mean of the orientation estimates from the landmarks (in "x"). The odometry-based orientation differs from the landmark-based estimates as expected due to odometry error.

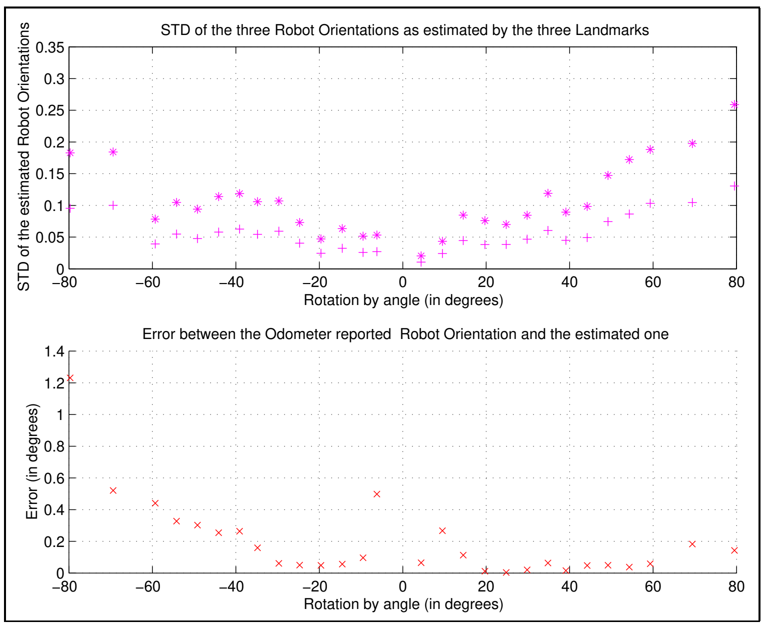


FIGURE 6.8. Error in the odometry based orientation estimate versus landmark based orientation estimates. Top sub-plot presents the standard deviation of the three estimates of the orientation using three pairs of landmarks("+") and the maximum difference ("*"). Bottom sub-plot presents the difference between the mean estimate of orientation based on the landmarks and the orientation reported from odometry.

Using the mean orientation estimate of the three landmarks we proceed to estimate the displacement $[x, y]$ of the origin of the Laser Coordinate system with respect to the center of the robot. An analytical solution and a least squares estimation of the empirical data provide the same results up to millimeter accuracy. For the least squares formulation, let $\cos(\theta_r) = cfr$ and $\sin(\theta_r) = sfr$, using Equation 6.5 we get Equation 6.7:

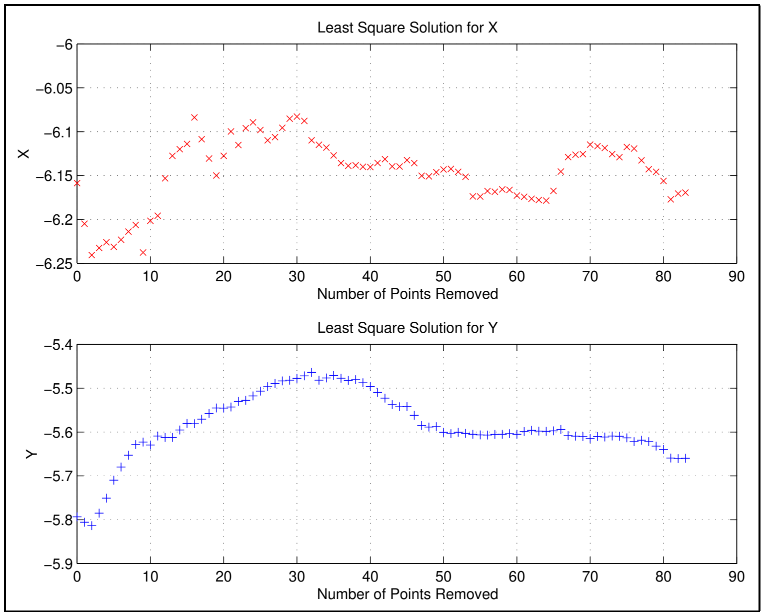


FIGURE 6.9. Iterative least squares calculation of the X (top sub-plot) and Y (bottom sub-plot) by removing one outlier at the time. The accuracy in both cases is better than one millimeter.

$$\begin{bmatrix} cf\cdot A_{x_l} - sf\cdot A_{y_l} + x \\ sf\cdot A_{x_l} + cf\cdot A_{y_l} + y \\ 1 \end{bmatrix} = \begin{bmatrix} cfr & -sfr & x_r \\ sfr & cfr & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cf\cdot B_{x_l} - sf\cdot B_{y_l} + x \\ sf\cdot B_{x_l} + cf\cdot B_{y_l} + y \\ 1 \end{bmatrix} =$$
$$\begin{bmatrix} cfr\cdot(cf\cdot B_{x_l} - sf\cdot B_{y_l} + x) - sfr\cdot(sf\cdot B_{x_l} + cf\cdot B_{y_l} + y) + x_r \\ sfr\cdot(cf\cdot B_{x_l} - sf\cdot B_{y_l} + x) + cfr\cdot(sf\cdot B_{x_l} + cf\cdot B_{y_l} + y) + y_r \\ 1 \end{bmatrix} \iff$$

$$\begin{bmatrix} cf\cdot A_{x_l} - sf\cdot A_{y_l} + x - cfr\cdot cf\cdot B_{x_l} + cfr\cdot sf\cdot B_{y_l} - cfr\cdot x + sfr\cdot sf\cdot B_{x_l} + sfr\cdot cf\cdot B_{y_l} + sfr\cdot y - x_r \\ sf\cdot A_{x_l} + cf\cdot A_{y_l} + y - sfr\cdot cf\cdot B_{x_l} + sfr\cdot sf\cdot B_{y_l} - sfr\cdot x - cfr\cdot sf\cdot B_{x_l} - cfr\cdot cf\cdot B_{y_l} - cfr\cdot y - y_r \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$\iff$$
$$\begin{bmatrix} 1-cfr & sfr \\ -sfr & 1-cfr \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} cf(-A_{x_l} + cfr\cdot B_{x_l} - sfr\cdot B_{y_l}) + sf(A_{y_l} - cfr\cdot B_{y_l} - sfr\cdot B_{x_l}) + x_r \\ cf(-A_{y_l} + sfr\cdot B_{x_l} + cfr\cdot B_{y_l}) + sf(-A_{x_l} - sfr\cdot B_{y_l} + cfr\cdot B_{x_l}) + y_r \end{bmatrix} \tag{6.7}$$

An analytical solution is given in Equation 6.8.

$$x = \frac{1}{2}(X_r - Y_r \cot(\frac{\theta_r}{2}) + \csc(\frac{\theta_r}{2})(A_{y_l}\cos(\theta - \frac{\theta_r}{2}) - B_{y_l}\cos(\theta + \frac{\theta_r}{2}) + A_{x_l}\sin(\theta - \frac{\theta_r}{2}) - B_{x_l}\sin(\theta + \frac{\theta_r}{2})))$$
$$y = \frac{1}{2}(Y_r + X_r \cot(\frac{\theta_r}{2}) + \csc(\frac{\theta_r}{2})(-A_{x_l}\cos(\theta - \frac{\theta_r}{2}) + B_{x_l}\cos(\theta + \frac{\theta_r}{2}) + A_{y_l}\sin(\theta - \frac{\theta_r}{2}) - B_{y_l}\sin(\theta + \frac{\theta_r}{2}))) \quad (6.8)$$

	X (cm)	Y (cm)
Analytical Calculations (All data)		
Mean	-6.1631	-5.7632
Std	0.6304	0.8801
Least Square Calculation (All data)		
	-6.1586	-5.7934

TABLE 6.5. Estimated values for x and y (the position of the laser in robot coordinates) using an analytical solution and a least squares solution.

Table 6.5 shows both the analytical results and the least squares solution using all the data. Figure 6.9 shows how the least squares solution changes after the removal of the worst fit each time. The top sub-plot presents the solution for x versus the number of points removed and the bottom sub-plot presents the solution for x. As can be seen from Table 6.5 and Figure 6.9, the results are stable with an accuracy of a millimeter.

For the current position of the laser mounted on top of the observing robot the final estimates are: $x = -6.15\text{cm},\ y = -5.65\text{cm},\ \theta = -138.1240°$.

3.3. Target Calibration. The next step in the calibration of the robot tracker sensor is the estimation of the relation between the target location and the observed robot reference. As we saw in the beginning of Section 3 the laser detects (at least) two planes from any generic direction around the robot. The intersection of the two planes provides a fixed point (the center of the target $[x_t, y_t]$); the angle between the two planes identifies which view (see Figure 6.3b); and the orientation of each plane is fixed relative to the orientation of the robot. For example, Figure 6.12a presents a top view of the target as can be seen from a position such that the two planes that are visible present a non-reflex corner, and the angle is 140° (View 3). The raw robot tracker sensor measurement is a quadruple $T = [x_t, y_t, \theta_{t_1}, \theta_{t_2}]$ containing the center of the target ($[x_t, y_t]$) and the orientation of the two planes ($[\theta_{t_1}, \theta_{t_2}]$). The pose of the robot ($P_r = [x_r, y_r, \theta_r]$) is given by Equation 6.9, where d is the distance between the center of the target and the center of the robot. The angles ϕ_1 and ϕ_2 are the angles between the planes of the target and the line (d) that connects the center of the target to the center of the robot. Finally, the angles $\delta\theta_1$ and $\delta\theta_2$ are the angles between the planes of the target and the orientation of the robot. In conclusion, the calibration of the target requires that we estimate the above mentioned five parameters $[d, \phi_1, \phi_2, \delta\theta_1, \delta\theta_2]$ for every view of the target. We divide the estimation into two independent steps, the estimation of the orientation parameters ($[\delta\theta_1, \delta\theta_2]$), and the estimation of the robot-center parameters ($[d, \phi_1, \phi_2]$).

$$P_r = \begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix} = \begin{bmatrix} x_t + d\frac{\cos(\theta_{t_1}+\phi_1)+\cos(\theta_{t_2}+\phi_2)}{2} \\ y_t + d\frac{\cos(\theta_{t_1}+\phi_1)+\sin(\theta_{t_2}+\phi_2)}{2} \\ \frac{(\theta_{t_1}+\delta\theta_1)+(\theta_{t_2}+\delta\theta_2)}{2} \end{bmatrix} \tag{6.9}$$

It is worth noting the following about the pose of the robot: by moving the robot forward it is possible to establish the orientation of the robot θ_r unambiguously because the forward motion is performed along the orientation of the robot. The center of the robot though ($[x_r, y_r]$) can be defined in many different ways. The geometrical center of the robot is one option, the center of rotation is another, and the point where the sonar sensors intersect could also be an option. Unfortunately, in order to properly correct the odometry error, the pose of the robot should use the center of rotation, which in turn is not a fixed point as the robot is unable to perform pure rotations. The problem we encountered was that we had to define a center for the robot that was changing for different amounts of rotations. In other words, during any rotation the center of the robot is displaced by a small amount. We tried to reduce the lateral motion of the robot by maintaining small speed/acceleration and by keeping the rotation small during the calibration. Nonetheless, this method was used only for calibrating one view of the target.

3.3.1. ***Estimation of relative orientation (*$[\delta\theta_1, \delta\theta_2]$*):*** As can be seen in Figure 6.11 an ideal forward translation of the robot has the following properties: the orientation of the robot remains constant [7] $\theta_r = \theta_r'$ (see Figure 6.11), the orientation of the line defined by the centers of the target ($[x_t, y_t], [x_t', y_t']$) before and after the translation and the

[7]For translations of small distances and for small speed the drift is negligible.

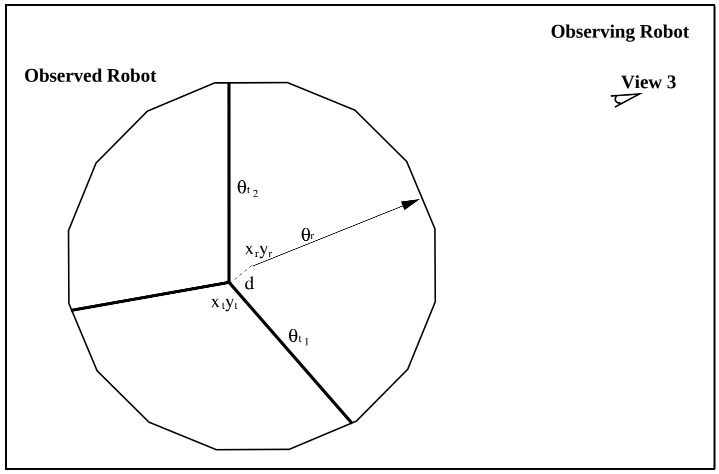


FIGURE 6.10. The robot is observed from *View 3*: relation between the detected target ($T = [x_t, y_t, \theta_{t_1}, \theta_{t_2}]$) and the pose of the robot ($P_r = [x_r, y_r, \theta_r]$).

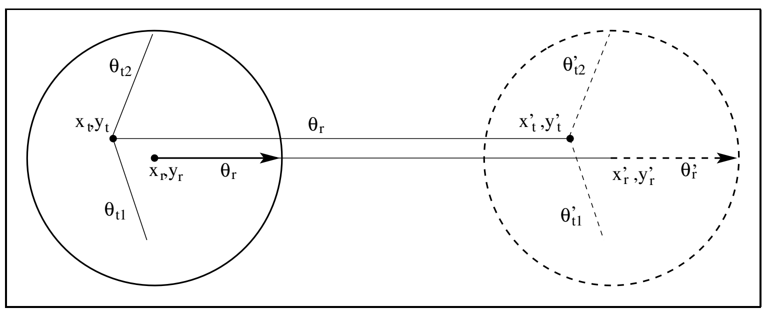
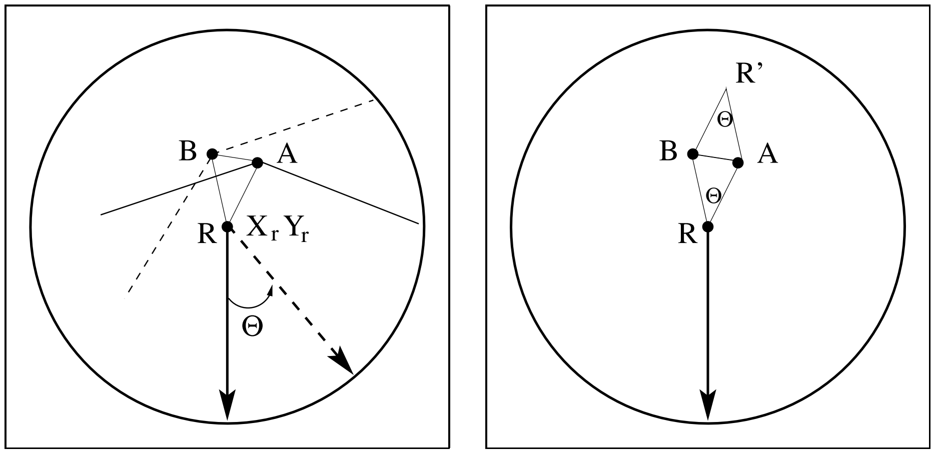


FIGURE 6.11. The observed robot translates along its heading (specified by θ_r) by a distance d, the center of the target translates also along the same heading (θ_r) by the same distance d.

orientation of the line defined by the centers of the robot (($[x_r, y_r], [x_r^{'}, y_r^{'}]$) before and after the translation are equal with the orientation of the robot θ_r (in WC), and the orientations of the two planes of the target are unchanged ($\theta_{t_1} = \theta_{t_1}^{'}, \theta_{t_2} = \theta_{t_2}^{'}$). Therefore, even though the center of the robot ($[x_t, y_t]$) is still unknown, the orientation θ_r can be estimated directly by the two centers of the target ($[x_t, y_t], [x_t^{'}, y_t^{'}]$) (see Equation 6.10). The parameters $[\delta\theta_1, \delta\theta_2]$ are given by Equation 6.11.

$$\theta_r = arctan(y_t^{'} - y_t / x_t^{'} - x_t) \tag{6.10}$$

$$\begin{aligned} \delta\theta_1 &= \theta_r - \theta_{t_1} \\ \delta\theta_2 &= \theta_r - \theta_{t_2} \end{aligned} \tag{6.11}$$



(a) (b)

FIGURE 6.12. (a) The robot rotates around its center (R) by an angle Θ, the fixed target rotates accordingly (from A to B). The solid lines starting from A represent the target before the rotation and the dashed lines starting from B represent the target after the rotation. (b) Two possible centers (R,R') on the opposite sides of AB line.

3.3.2. ***Estimation of displacement (*** $[d, \phi_1, \phi_2]$ ***):*** As mentioned above the center of the robot is not uniquely defined. In some cases the estimated parameters are of the same order of magnitude as the error in the measurements. The following procedure was followed: the parameters for the calculation of the center of the robot were estimated for one view, then the observed robot was stationary and the observing robot was moved in two consecutive positions in order to sense two different views of the target (one view for which the robot center is estimated and a new view). Then the parameters for the second view are estimated.

Figure 6.12a shows the robot and the target before a rotation ($A = [A_x, A_y]$: center of target, the two planes drawn in solid lines) and after a rotation by Θ ($B = [B_x, B_y]$: center of target, the two planes drawn in dashed lines); a second rotation would introduce a third point ($C = [C_x, C_y]$: center of target). The robot rotates around a point $R = [X_r, Y_r]$ and the target is detected after two rotations of the robot that carries the target, therefore the target is sensed three times. The three points A,B,C define one circle (see Equation 6.12 for the center of the circle); moreover, every two points and the rotation angle (as estimated by the change in orientation of the two target planes) give two symmetrical solutions for a different circle (see Figure 6.12b and Equations 6.13,6.14 for the center of the circle). The two points and one angle define two centers (R,R' in Figure 6.12b), one on each side of line that connects the two points (A,B in Figure 6.12b). From the two solutions (Equations 6.13,6.14) we select the one that is consistent with the three point circle.

$$R_x = \frac{B_y C_x^2 - (B_x^2 + B_y^2)C_y + B_y C_y^2 + (A_x^2 + A_y^2)(-B_y + C_y) + A_y(B_x^2 + B_y^2 - C_x^2 - C_y^2)}{2(A_y B_x - A_x B_y - A_y C_x + B_y C_x + A_x C_y - B_x C_y)}$$
$$R_y = \frac{(A_x^2 + A_y^2)(B_x - C_x) + C_x(B_x^2 + B_y^2 - B_x C_x) - B_x C_y^2 + A_x(-B_x^2 - B_y^2 + C_x^2 + C_y^2)}{2(A_y B_x - A_x B_y - A_y C_x + B_y C_x + A_x C_y - B_x C_y)} \quad (6.12)$$

$$\begin{aligned} R_x &= \frac{1}{2}\left(A_x + B_x + \frac{(A_y - B_y)(1 - \cos(\Theta))\sqrt{(A_x - B_x)^2 \csc(\Theta)^2}}{A_x - B_x}\right) \\ R_y &= \frac{1}{2}\left(A_y + B_y - (1 + \cos(\Theta))\sqrt{(A_x - B_x)^2 \csc(\Theta)^2}\right) \end{aligned} \quad (6.13)$$

$$\begin{aligned} R_x^{'} &= \frac{1}{2}\left(A_x + B_x - \frac{(A_y - B_y)(1 - \cos(\Theta))\sqrt{(A_x - B_x)^2 \csc(\Theta)^2}}{A_x - B_x}\right) \\ R_y^{'} &= \frac{1}{2}\left(A_y + B_y + (1 + \cos(\Theta))\sqrt{(A_x - B_x)^2 \csc(\Theta)^2}\right) \end{aligned} \quad (6.14)$$

We repeat the double rotation experiment several times and keep only the measurements where the radius of the four circles (one from the three points and three circle by combining every two points) are consistent. From each experiment the center of the robot is estimated as $[X_r, Y_r]$ and the parameters $[d, \phi_1, \phi_2]$ are calculated using Equation 6.15.

$$\begin{aligned} d &= \sqrt{(X_r - x_t)^2 + (Y_r - y_t)^2} \\ \phi_1 &= arctan(Y_r - y_t / X_r - x_t) - \theta_{t_1} \\ \phi_2 &= arctan(Y_r - y_t / X_r - x_t) - \theta_{t_2} \end{aligned} \quad (6.15)$$

The orientation estimation of the target is always done using the mean value of the two planes for better accuracy. Table 6.6 presents the estimates of the $[d, \phi_1, \phi_2, \delta\theta_1, \delta\theta_2]$ parameters by moving the observed robot.

View	d	ϕ_1	ϕ_2	$\delta\theta_1$	$\delta\theta_2$
1	N/M	N/M	N/M	176.9034	77.9354
2	N/M	N/M	N/M	296.6401	176.3852
3	1.8252	81.6194	301.8621	77.3893	297.6944
4	N/M	N/M	N/M	N/M	N/M
5	N/M	N/M	N/M	N/M	N/M
6	N/M	N/M	N/M	77.3648	298.5089

TABLE 6.6. The estimated five parameters for the target calibration. Moving the observed robot (N/M: Not Measured).

In order to calibrate all faces of the target the two robots were placed in an enclosure formed by a set of nine walls larger than one meter each. The robot with the target was kept stationary while the robot equipped with the laser range finder was moved in a circular trajectory around the target robot and at a distance of one meter. After each motion the walls were used in order to localize (see Section 4). The moving robot took 36 steps around the stationary one, each step approximately 17.3cm. After each step the pose of the moving robot was corrected and then the target was detected ten times (in order to measure the repeatability of the experiment). Table 6.7 presents the values for the five parameters for every view of the target. The laser data and the trajectory of the moving observing robot can be seen in Figure 6.13. Figure 6.14 presents the mean estimate of the intersection point from each position. Figure 6.15 presents a closer view of the mean value together with an uncertainty ellipse drawn at one standard deviation of the data.

View	d	ϕ_1	ϕ_2	$\delta\theta_1$	$\delta\theta_2$
1	5.9520	187.2810	89.2976	175.7135	77.7302
2	5.5150	277.0460	157.3268	296.5156	176.7964
3	1.8224	80.7237	301.5761	76.7238	297.5762
4	3.2858	130.7391	32.5019	176.2607	78.0235
5	3.9434	333.8887	213.9336	296.7751	176.8200
6	6.7692	81.5267	300.4969	77.0238	295.9940

TABLE 6.7. The estimated five parameters for the target calibration. The calculations are all relative to face 3 and present average values.

The previous experiment was repeated one more time with the observing robot moving on a circle of radius 120cm from the stationary robot, in intervals of ten degrees (around the circle), and 10 measurements were collected from each position. The intersection points

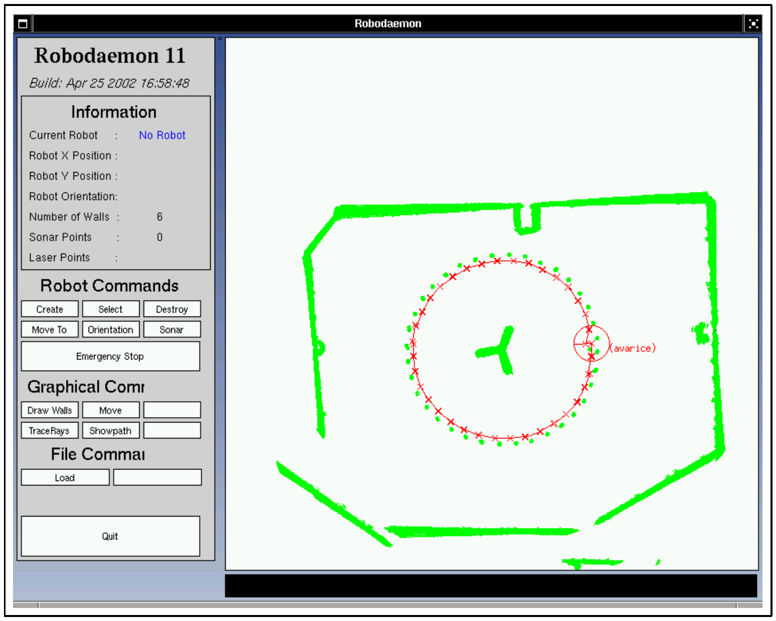


FIGURE 6.13. The trajectory of the observing robot with the laser range finder and the detected laser points. In the center stands the observed robot which is visible only in the form of the three lines of the target. Enough walls existed around the two robots in order to provide adequate landmarks during the calibration process.

of the targets are shown in Figure 6.16 together with the mean estimated position of the robot. Figure 6.17 presents a closer view.

View	d	ϕ_1	ϕ_2	$\delta\theta_1$	$\delta\theta_2$
1	3.1218	187.9969	88.8799	177.0988	77.9818
2	3.3148	270.5552	150.9222	297.4960	177.8630
3	1.8634	77.0882	295.7289	78.4453	297.0860
4	2.4131	146.8717	49.4735	177.1362	79.7380
5	2.2933	311.8284	192.5956	296.6677	177.4349
6	2.5814	75.4404	293.8753	79.7157	298.1506

TABLE 6.8. The estimated five parameters for the target calibration. Measuring the geometry of the body of the observed robot.

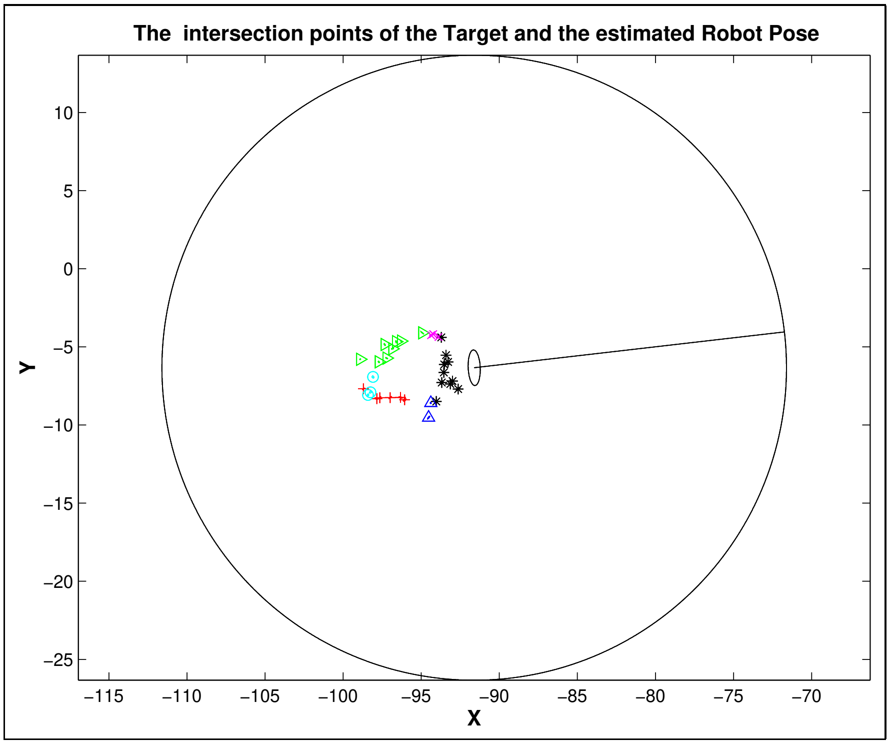


FIGURE 6.14. The Robot Pose is estimated based on data from the face 3 (the ellipse indicates the uncertainty of the robot pose estimation. The intersection points for the different faces are displayed as follows: face 1 as "+", face 2 as "▷", face 3 as "*", face 4 as "x", face 5 as "△", face 6 as "o".

Finally, one more calibration experiment is described, where the five parameters were estimated based on the frame of the robot. Table 6.8 contains the values of the five parameters estimated using the chassis of the robot. In this case the geometric center of the frame of the robot and the orientation of the front plate of the robot were used.

4. Laser Sensor based Localization

Every time the laser robot tracker is used a large number of laser points are collected (between 3000-4000). Since the same features are often observed repeatedly, the robot's motion as well as the environment layout can be estimated. The problem of localizing from laser data has been investigated by several researchers (see [**68, 101, 100, 103, 102**]). In order to assist us in the calibration of the target the following method was used for localizing the observing moving robot using the laser data. First the lines are fitted using the MacKenzie-Dudek algorithm [**52, 107**]; the first time laser data are collected every line

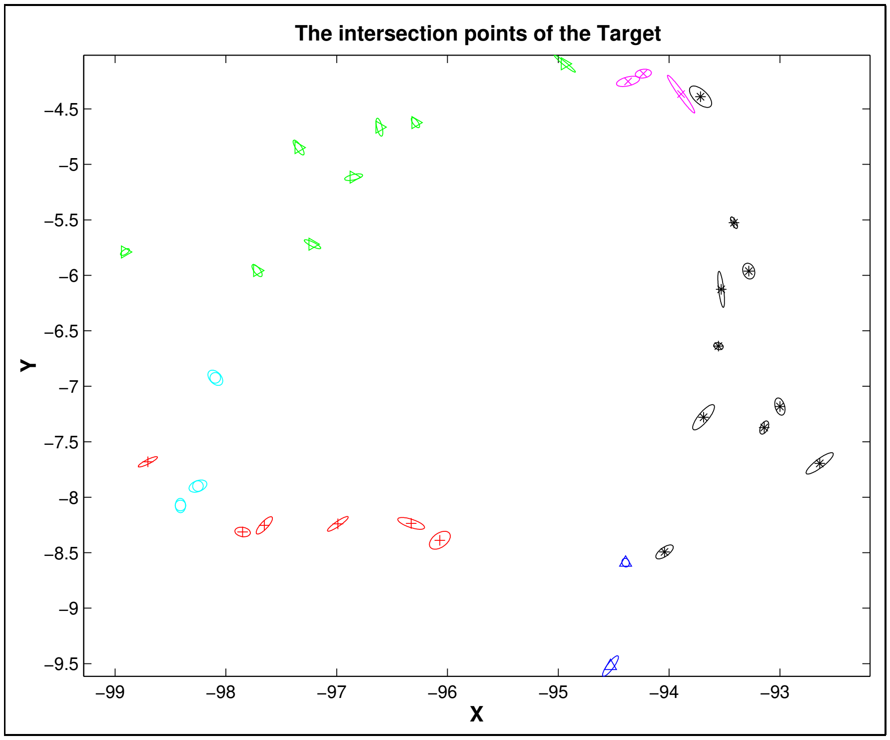


FIGURE 6.15. Zoom at the intersection points of the Target (from Figure 6.14); the ellipse around each point represents the uncertainty over ten measurements.

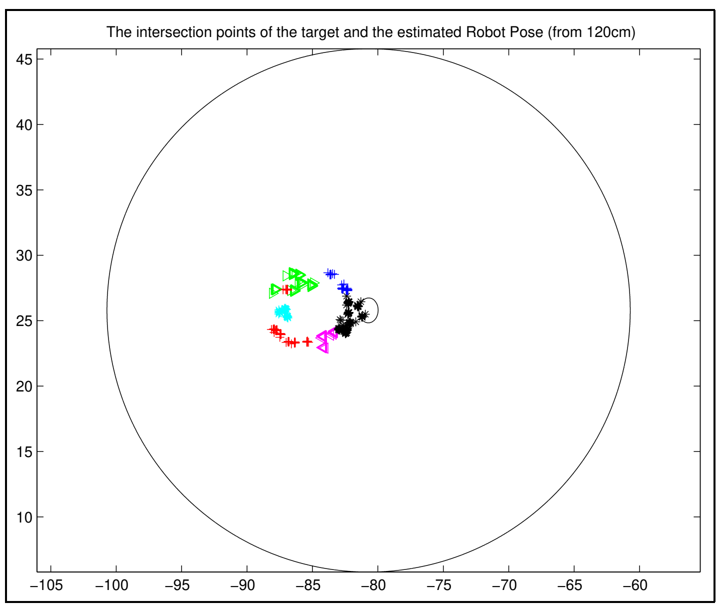


FIGURE 6.16. The Robot Pose is estimated based on the calibration table of Table 6.7 (the ellipse indicates the uncertainty of the robot pose estimation. The intersection points for the different faces are displayed as follows: face 1 as "+", face 2 as "▹", face 3 as "*", face 4 as "x", face 5 as "△", face 6 as "o".

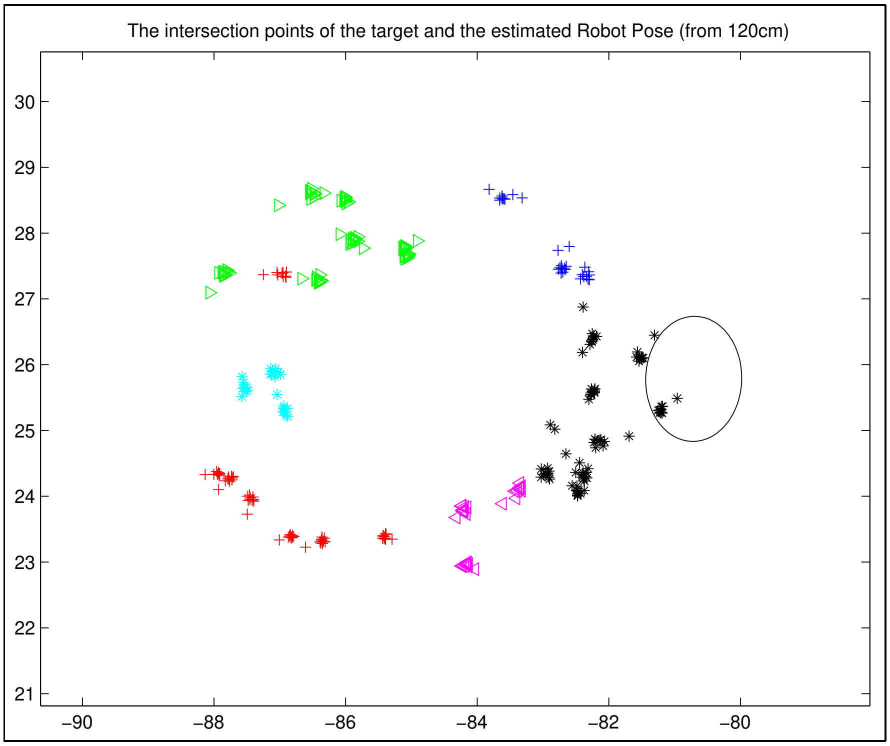
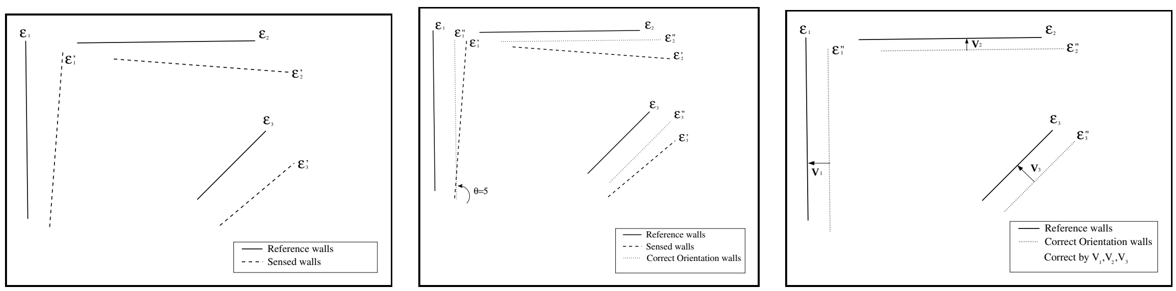


FIGURE 6.17. Zoom at the intersection points of the Target (from Figure 6.16). Ten measurements were collected from each position. .

that is more than one meter is saved as a landmark. After every motion a new set of laser data is collected and new lines are fitted then the new lines are matched with the saved landmarks; in particular only lines that differ less than 10 degrees and intersect or their end points are closer than 20cm from each other are selected see Figure 6.18a. When we finish selecting the matching lines the error is calculated. First the angular error is calculated and a weighted mean estimate is used (θ) (the length of the lines is used as a weight where longer lines contribute more). The new lines are rotated around the position of the observing robot by the calculated mean angle as in Figure 6.18b. Finally the displacement for each pair of lines (dx_i, dy_i) is computed and the weighted mean $[dx, dy]$ is used to correct the robot pose. The mean is given by $dx = \sum dx_i w_i, dy = \sum dy_i w_i$, where w_i is the weight proportional to the length of the i^{th} line pair. The pose of the moving robot is corrected as in Equation 6.16.

$$\mathbf{x}'_m = \begin{bmatrix} x'_m \\ y'_m \\ \theta'_m \end{bmatrix} = \begin{bmatrix} x_m + dx \\ y_m + dy \\ \theta_m + \theta \end{bmatrix} \tag{6.16}$$



(a) (b) (c)

FIGURE 6.18. Localizing from the observed lines.

Two alternative robot tracker sensors based on different sensing modalities were described in this chapter. The first robot tracker is vision-based and is more robust over uneven terrain. The accuracy of the pose estimation deteriorates over distance. The second robot tracker is based on a laser range finder, it has higher accuracy than the vision-based one and the accuracy of pose estimation is constant over a longer range. The laser based tracker requires flat floors in order to function. In the next chapter we present experimental results from the application of these robot tracker sensors in exploration and mapping.

CHAPTER 7

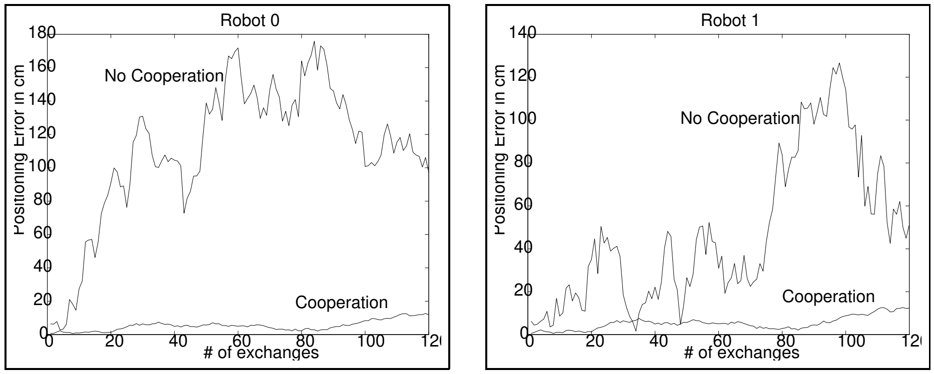
Experimental Results

> Don't believe the results of experiments until they're confirmed by theory.
>
> –Attributed to Arthur Eddington.

In the Chapters 3 and 4 we discussed two exploration strategies for mapping the free space of an unknown environment with the use of a novel sensor: the *robot tracker*. Furthermore, in Chapter 5 we presented a probabilistic framework for reducing the uncertainty that accumulates due to sensor noise during the exploration. Finally in the previous chapter (Chapter 6) we presented the construction of two different robot tracker sensors and discussed their operational parameters (range of sensing, accuracy, etc.). In this chapter we present the results from a series of experiments using the two different robot tracker sensors and also using different robots. Experiments with the real robots allow us to validate our exploration strategies but also to examine the strength and weaknesses of different sensor/robot combinations. Simulated experiments provide results from more complicated environments and also for the effect of different noise parameters and different robot configurations that were not available in our laboratory.

1. Trapezoidation Algorithm

As we saw in Chapter 4 the trapezoidation algorithm is used to map areas that extend beyond the range of the robot tracker sensor. In the available space for experiments it was not possible to find an area that extended to a few times the sensor range (5-6m for the visual robot tracker and 10-14m for the laser robot tracker sensor). Therefore the



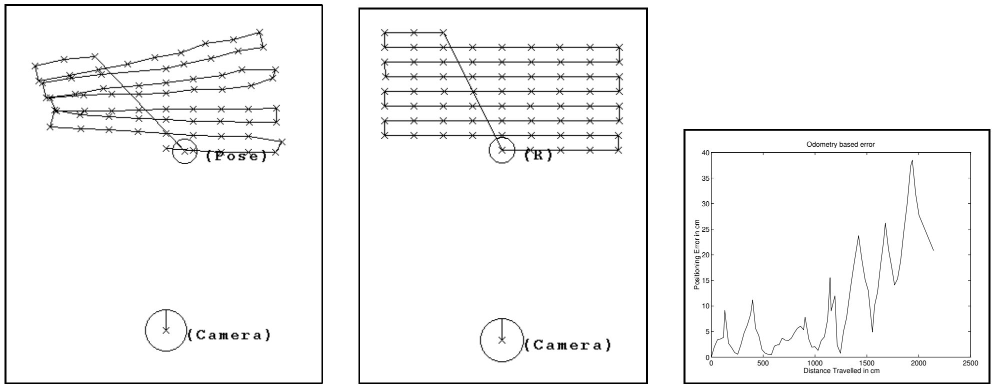
(a) (b)

FIGURE 7.1. Exploration of one stripe (120 exchanges). The results are from a single run.

majority of the experiments were done in simulation. In addition the motion strategies of trapezoidation algorithm were used in simulated multi-robot experiments for determining the trade-offs between the number of robots and the type of robot tracker sensor used.

1.1. Simulation Results. In order to estimate the improvement in position estimation when the two robots collaborate, a series of experiments were performed in simulation. The two robots explored a single stripe of the environment, exchanging roles 120 times. Odometry error estimates gathered during experiments with an RWI robot were used to parameterize the error model in dead reckoning. The accuracy of the helix vision tracker was used to model the accuracy and the range of the robot tracker. The same path was traveled twice, and the error in positioning was measured. The first time no cooperation took place between the two robots, while the second time every time the two robots exchanged roles they corrected their position estimates. In the case of no cooperation the two robots are following the same trajectory as before but without correcting their position estimate.

From the results shown in figures 7.1(a,b), it is clear that the cooperative exploration strategy improves performance substantially over the non-cooperative strategy. It is worth pointing out that in this experiment no systematic error was included in the model, such as would occur on an inclined floor where with every translation a small amount of slippage



(a) (b) (c)

FIGURE 7.2. (a) Path of the moving robot as estimated by the visual tracker. Measurements taken in different positions validate the accuracy with precision of roughly 2cm. (b) The desired path of the moving robot. Although the robot can be driven along this path using open-loop control, dead reckoning error leads to a substantial discrepancy. (c) The error in positioning from the odometry estimations.

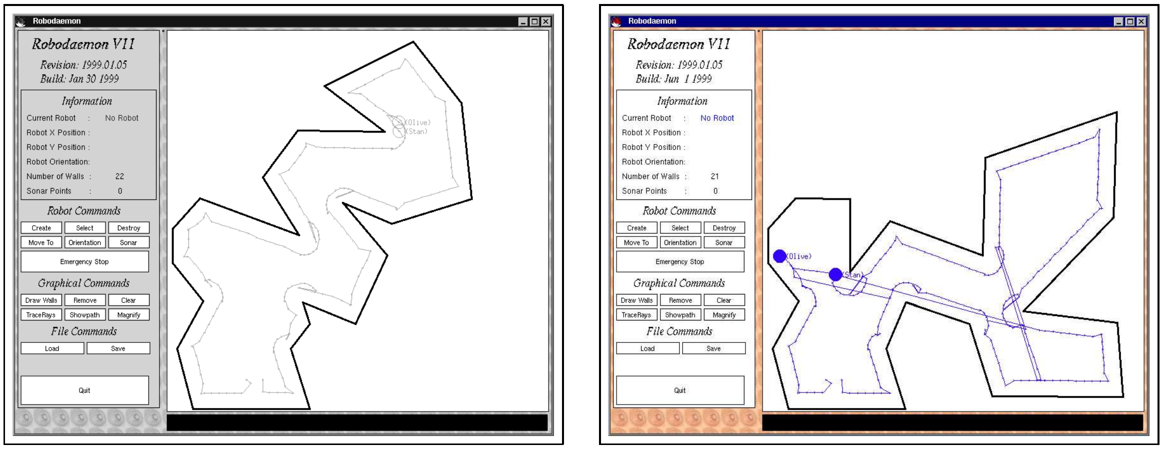
would occur. It is clear that the cooperation of the two robots helps to maintain reduced localization error and improves mapping robustness.

1.2. Laboratory. Due to space constrains we performed an experiment in our laboratory using a motion pattern similar to the one traveled by the two robots using only one robot while the other one was stationary.

In order to measure the accuracy of the map, a few locations along the path were selected and the position of the robot was estimated relative to the stationary camera. The accuracy of the positions estimated by the camera-based tracker was between 1.0 and 2.3 cm. As can be seen in Figure 7.2a,b the inaccuracy is largely due to rotational error and thus it is more evident near the sides of the rectangle. Figure 7.2c presents the absolute odometry error as it accumulates over the distance traveled by the robot.

2. Triangulation Algorithm

The triangulation algorithm presented in Chapter 3 is used to map areas that are bounded by the range limit of the robot tracker sensor. Experiments conducted first in simulation and then in different locales in our building allowed us to verify the performance



(a) (b)

FIGURE 7.3. The paths of the two robots after the completion of the exploration.

of our approach. In the next section we present experimental results from two typical environments.

2.1. Simulation. Extensive experiments have been conducted using the robotic simulation package RoboDaemon. The simulations allowed us to specify different parameters such as odometry error, robot-tracker uncertainty and the complexity of the explored environments. Figure 7.3 presents two typical environments used in the simulations (approximate area 144 m^2) and the path the two robots followed. Figures 7.4 and 7.5 present the exploration of two model environments; these examples illustrate different aspects of the triangulation algorithm. Figures 7.4 (a-i) present snapshots of the exploration as perceived by *Robot 0* and *Robot 1*, and the resulting map at different instances of the exploration. The two robots exchange roles when the line of visual contact is interrupted. In the first row an early phase of the exploration is presented. The two robots have exchanged roles twice and *Robot 0* explores five new triangles. Consequently, in the second row *Robot 1* is exploring again (Figure 7.4d), then the two robots exchange roles and *Robot 0* explores three additional triangles. The third row illustrates the final stages of the exploration where *Robot 1* explores the final parts of the environment using *Robot 0* as a reference.

In Figure 7.4, in the last row, the early phase of the exploration is presented, using pure odometry for positioning. The dashed line depicts the real path of the robot and the solid

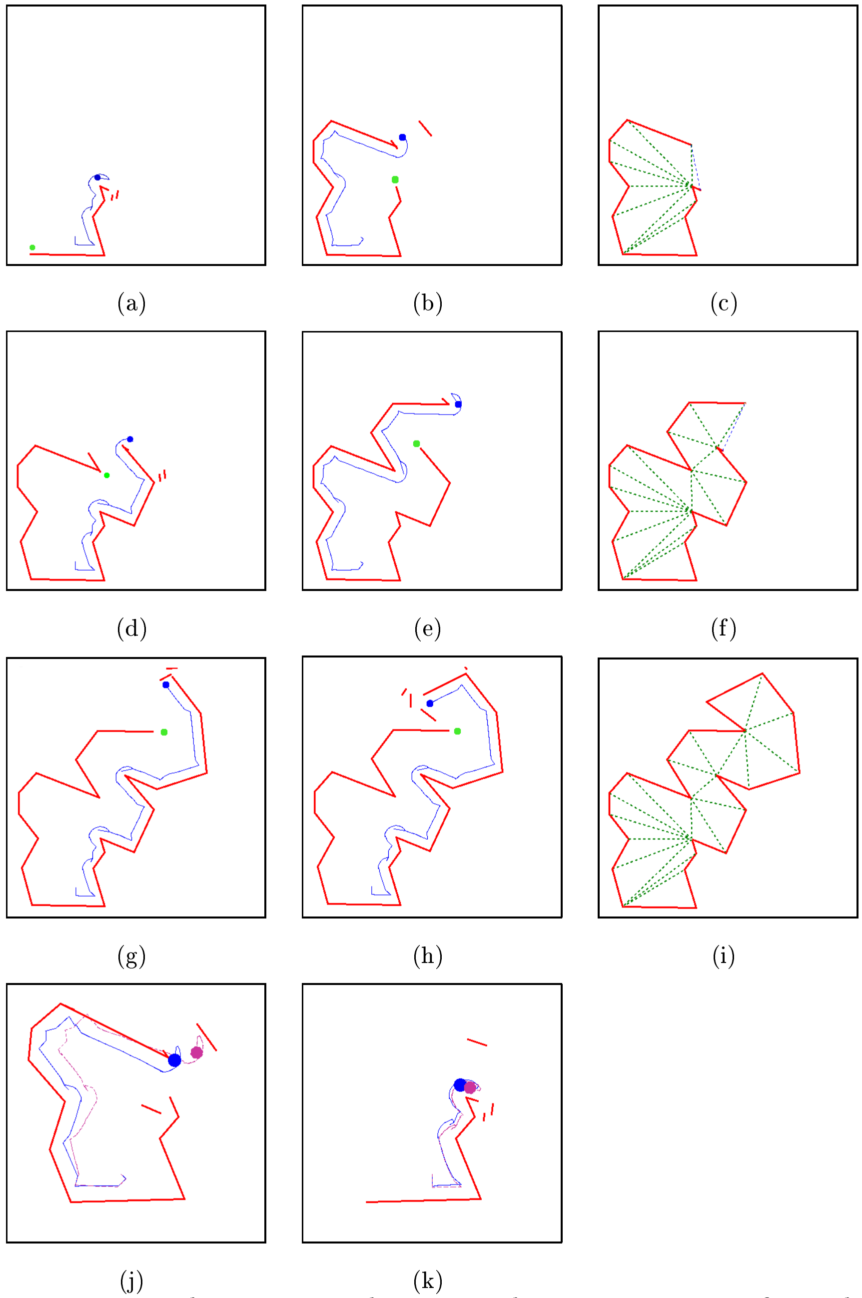


FIGURE 7.4. First three rows: Exploring an unknown environment, figures b and e illustrate the trajectory of *Robot 0*. Figures a,c,d,f,g,h illustrate the trajectory of *Robot 1*. Finally the third column (Figures c,f,i) presents the map up to that point. Last row: Close-up on the build up of the uncertainty when only odometry was used. The solid line is the odometry based estimation of the robots while the dashed line is the real position of the robots (see text Section 7.1.1).

line the odometry based paths. In small worlds and/or cluttered environments multiple observations of the same object could be used in order to correct the positioning of the moving robot. As can be seen in Figure 7.4(j,k), the accumulation of uncertainty causes the map to be distorted although local consistency is maintained. These distortions could lead over time to a map that is not even topologically sound.

The results from the exploration of a different environment are presented in Figure 7.5 depicted as a sequence of snapshots at successive times. The presence of reflex vertices that interrupt the line of visual contact introduce internal triangles and therefore branches in the dual graph making the exploration more complicated. The results are presented in three columns. The first column depicts the trajectory of *Robot 0* and the environment as perceived by that robot. The second column depicts the trajectory of *Robot 1* and its perception of the world. Finally, the third column presents the constructed map up to that point in the exploration. The light grey disk represents the position of the stationary robot and the black disk the location of the moving robot in the figures of the moving robot.

Over the interval spanned by the images in the first row, *Robot 1* is stationary (after mapping two triangle), while *Robot 0* is mapping the right branch of the first bifurcation. The line of visual contact was broken by a reflex vertex; thus, a internal triangle was built (node with degree 3), and two branches were started. Each branch consists of one open triangle with a gateway to unexplored space. In the second row, *Robot 0* is stationary (after adding two more nodes in the embedded graph and *Robot 1* is mapping the second occluding vertex. Again an internal triangle is created (node with degree 3), and *Robot 1* is mapping the left branch of the bifurcation. In row three the environment is mapped for the area that corresponds to the branch being explored, with the last triangle having two walls and one internal diagonal (node with degree 1). Figure 7.5i presents the map up to that point where the last wall (not fully explored yet) is marked with a thiner line. Then the two robots proceed to the closest gate (following a depth first traversal of the embedded graph). Row four demonstrates the exploration of *Robot 0* of the final branch (right) of the second bifurcation, while *Robot 1* is stationary at the second occluding vertex. Finally, the fifth row illustrates the final step of the exploration. *Robot 0* is stationary at the first occluding vertex encountered, while *Robot 1* maps the final triangle. In Figure 7.5o the completed map is shown. The dual graph is presented in the figures of the third row superimposed on the metric map.

2.2. Exploration with the two Superscout robots. A series of experiments were performed in different locations in our building (McConnell Engineering Building, McGill University) with a pair of Superscout robots from Nomadic Technologies, Inc. The robots use a differential drive and are equipped with a ring of 16 sonar transducers. In addition one of them (*Robot 0* in our experiments) has a laser range finder from Accurange (see Chapter 6 Section 3 for details) mounted on top, while the second robot (*Robot 1* in our experiments) has a three plane target. The combination laser range finder/target implements the robot tracker sensor (please refer to Chapter 6 for more information on the sensor). The control software was run remotely on a 1GHz pentium 4 with 512Mb of RAM and the communication was done over radio-ethernet. The two robots are powered by two 12V/20Ah and can operate autonomously for 3-4 hours. The software controlling the robots consisted of two components, the RoboDaemon RD11 control software that provided an interface with the robots and my exploration program developed in C++ that is guiding the exploration and interpreting the data.

2.2.1. *Exploration of two laboratories.* A complex environment was created inside two adjacent rooms in the Centre for Intelligent Machines; Figure 7.6 presents pictures of this environment from different views during the exploration. The two robots started at the lower left side, inside the mobile robotics laboratory (as can be seen in Figure 7.6a); after mapping the inside of one laboratory they passed through a narrow corridor (see Figure 7.6b,c) to the outer laboratory (see Figure 7.6d).

Figure 7.7 presents the progress of the triangulation algorithm as the two robots proceeded to explore the free space. Initially the two robots move to the closest wall and proceed to opposite end-points of the wall (see Figure 7.7a). The path of *Robot 0* is marked in red and the path of *Robot 1* is marked black. *Robot 1* explores a single triangle (Figure 7.7b), after which it encounters a reflex vertex that interrupts the line of visual contact and exchanges roles with *Robot 0* (see Figure 7.7c). *Robot 0* starts exploring and maps the inside of the lab (see Figures 7.7d,e,f,g,h and 7.6a). During the exploration *Robot 0* encounters two reflex corners (see Figures 7.7e,h) but the line of visual contact is maintained and *Robot 0* continues the exploration. In Figure 7.6b *Robot 0* can be seen entering the corridor that

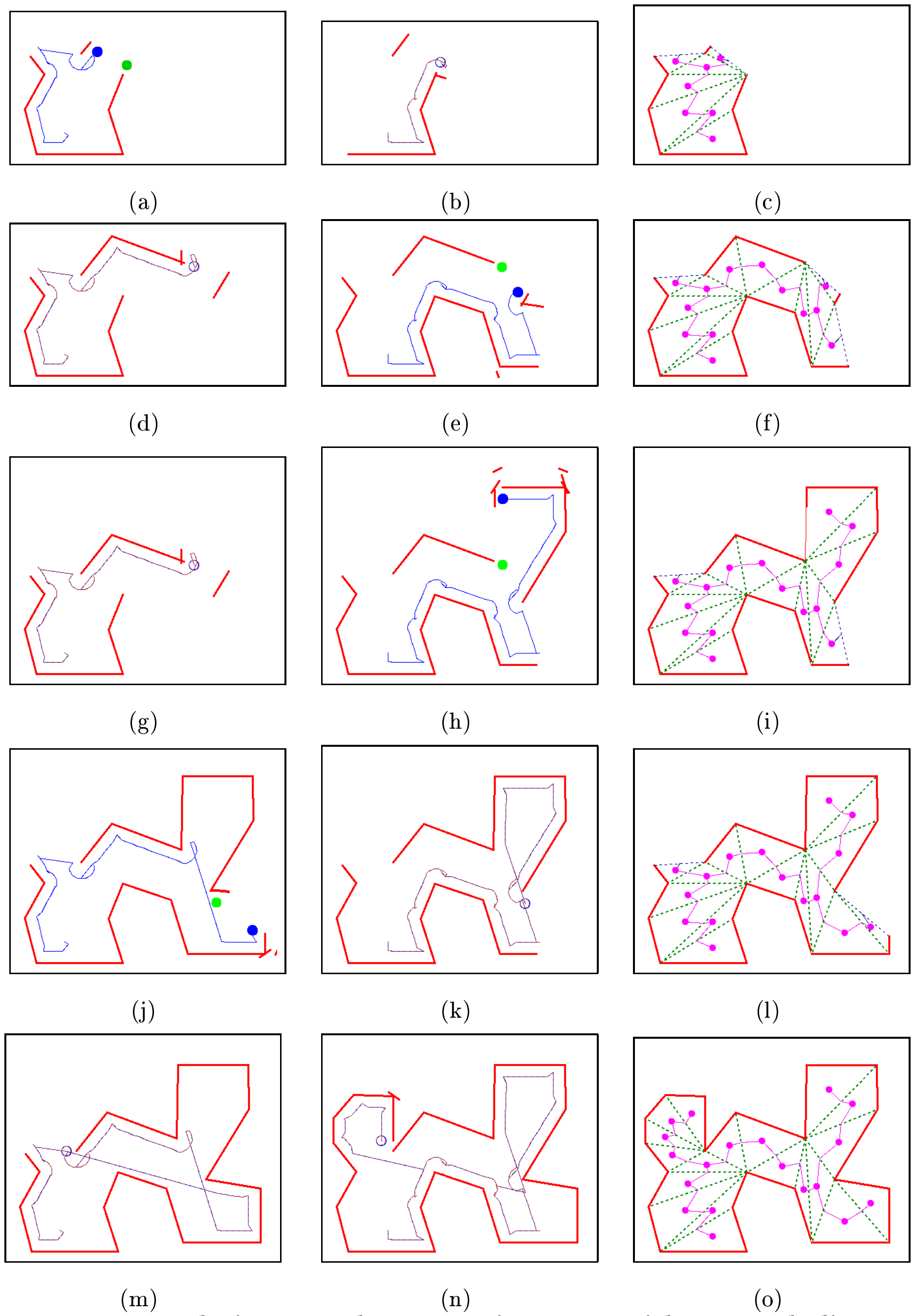
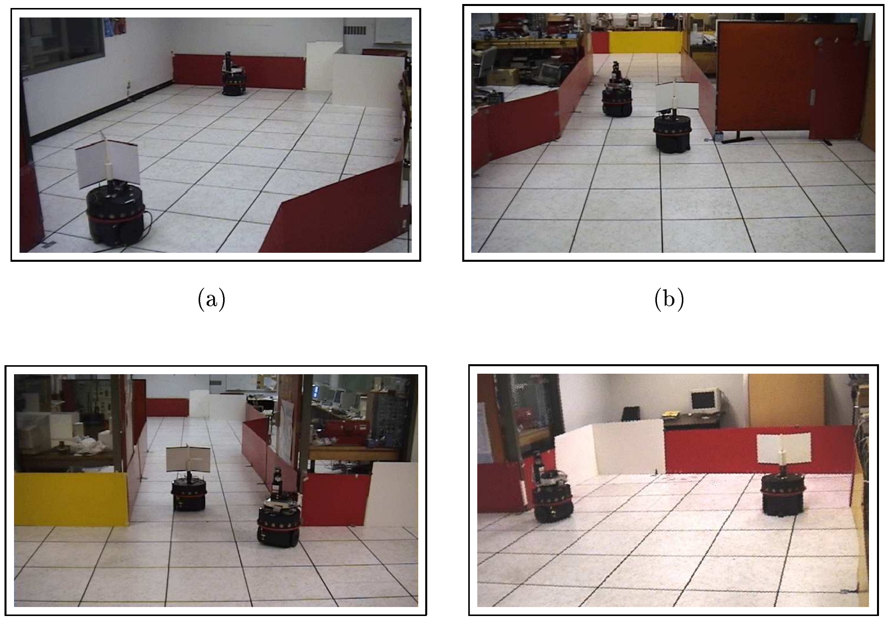


FIGURE 7.5. **Exploring an unknown environment with two occluding vertices:** The first column illustrates the trajectory of *Robot 0*. (a,d,g,j,m). The second column illustrates the trajectory of *Robot 1* (b,e,h,k,n). Finally the third column presents the **map** up to that point (c,f,i,l,o). (See text Section 7.1.1).



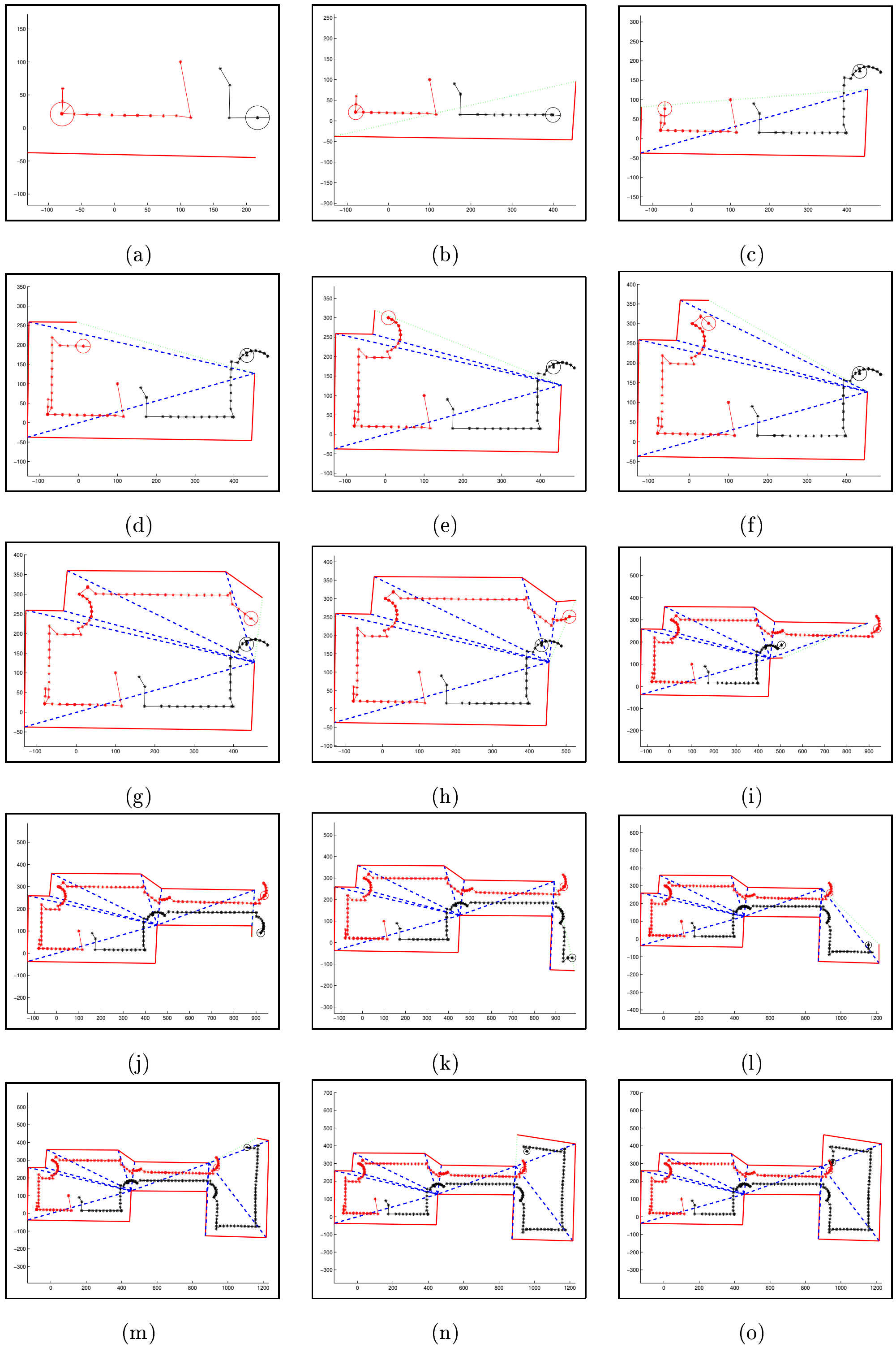
(a) (b)

(c) (d)

FIGURE 7.6. Exploration inside two adjacent labs at the fourth floor of our building (a) Beginning of the exploration *Robot 1* is stationary at a reflex corner and *Robot 0* explores a wall across from it. (b) *Robot 0* has mapped inside of the lab and moves along the corridor towards the exit. (c) *Robot 1* starts exploring coming through the corridor. *Robot 0* is positioned at a reflex corner. (d) The last part of the exploration, *Robot 1* (on the right) is mapping the wall across from *Robot 0* that is placed at a reflex corner and provides corrections for *Robot's 1* pose estimation.

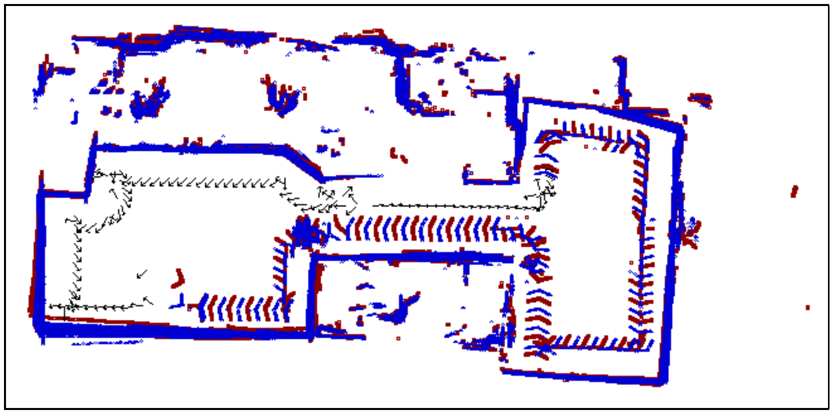
connects the two labs; the resulting map is presented in Figures 7.7(h,i). A reflex corner is encountered and *Robot 0* stops the exploration because the line of visual contact is interrupted. The two robots exchange roles; *Robot 1* passes through the corridor (see Figure 7.6c and 7.7j) and comes out to the second lab. Finally with *Robot 0* position at the reflex corner *Robot 1* maps the outer lab (see Figures 7.7(k,l,m,n) and 7.6d). Figure 7.7o presents the final map with the complete triangulation of free space; the two robots are positioned next to each other.

The laser sensor data were recorded during the exploration of *Robot 0*. In order to demonstrate the performance of cooperative localization the laser range finder was used only as part of the robot tracker sensor; thus the map produced from the triangulation algorithm is constructed solely by the sonar sensor data calculated using the corrected poses of the two robots. To further validate our approach we fused the recorded laser data

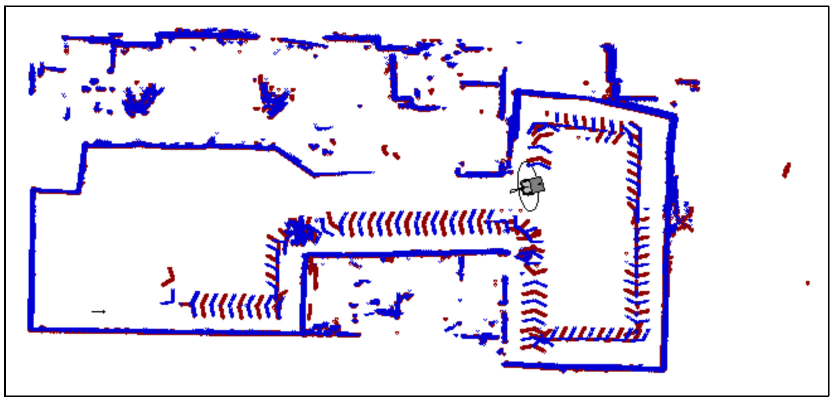


(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l) (m) (n) (o)

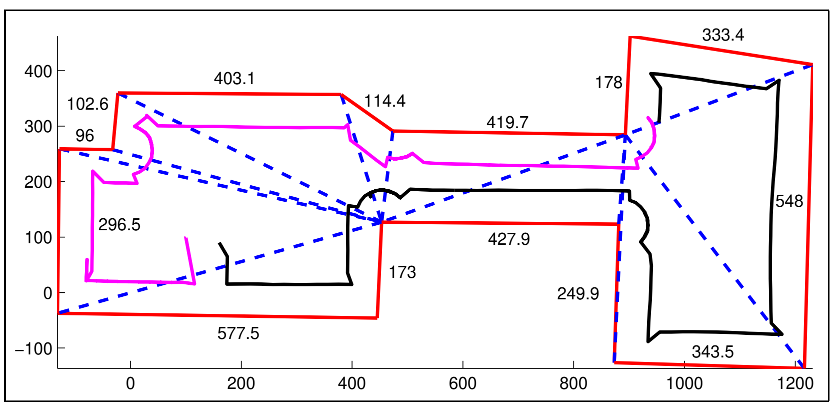
FIGURE 7.7. The exploration of the two laboratories. Red lines are the walls, blue dashed lines are diagonals, green dotted lines are gates to unexplored space.



(a)



(b)



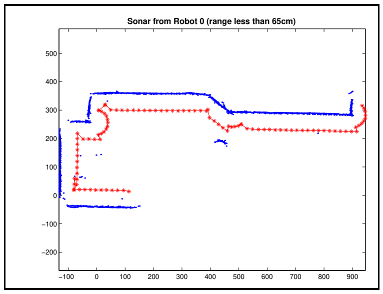
(c)

FIGURE 7.8. (a) The laser data collected during the exploration. (b) Scan match applied to the laser data and their position corrected using Stephen Gutmann's scanstudio. (c) The triangulation map produced using only the sonar for mapping. The trajectory of *Robot 0* is marked in magenta and the trajectory of *Robot 1* is marked black. The walls are displayed in red and their lengths (in cm) is marked next to them. The internal diagonals that define the triangulation are marked as blue dashed lines.

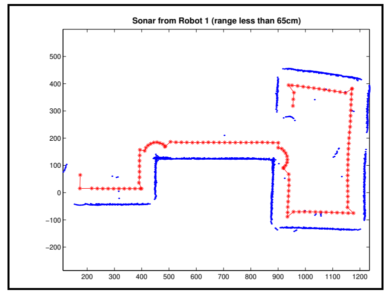
using scan matching with Stephen Guttman's scanstudio software that performs localization of a single robot using the scan matching algorithm by Lu and Milios [**68, 101**]. Figure 7.8a presents the laser points collected and Figure 7.8b presents the same data after the scan matching. Note the detected target that marks the trajectory of *Robot 1*. Because the laser range finder senses on the horizontal plane (when the robot is not tilted) in flat terrains would map all the obstacles at the height it senses. As can be seen from the laser data in Figure 7.8a,b the sensor from different positions scans at different planes (because the floor is uneven and the robot tilts). Thus the top wall of the corridor is not fully mapped (see section 4 for further discussion). Finally, Figure 7.8c presents the map created by the triangulation algorithm. As can be seen the maps of Figures 7.8b and c are very similar.

The sonar data used for wall following were collected during the exploration. Figure 7.9 presents the sonar points drawn in blue; the left column presents the data gathered by *Robot 0* while the right column presents the data gathered by *Robot 1*. The trajectories of the two robots are marked in red, and the positions from where the sonar scans were taken are marked with "*". During the exploration the robots follow the walls at distance of 60cm (Figure 7.9a,b presents the only the sonar points detected in less than 65cm). In order to filter out most of the noisy data any sonar point further than 120cm was rejected. Figure 7.9c,d presents the data actually used during the exploration. As we discussed earlier (Chapter 3 section 2) the sonar points are fit into line segments and then the lines are merged together; as can be seen in Figure 7.9c,d the sonar data filtered at 120cm are aligned with the walls. Figures 7.9e,f and 7.9g,h present the sonar scans filtered at 250cm and 400cm. The data are very noisy and if the robots believed the occupancy of space from them navigation would be impossible. It is worth noting some straight lines formed inside open space (see Figure 7.9g). They correspond to small anomalies on the floor at the borders of the tiles in our laboratory (see Figure 7.6 for the appearance of the floors). As can be seen in Figures 7.9a-d there is virtually no distortion of the data; this is due to our cooperative localization approach which maintains an accurate position.

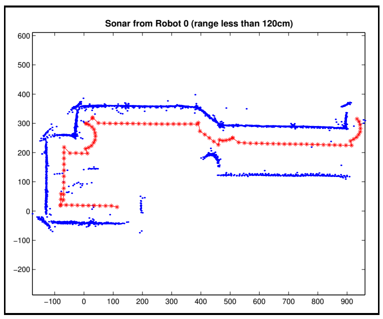
The positional error is maintained low throughout the exploration by the use of cooperative localization. The motion commands given to the robot were recorded and used to guide the prediction phase of a particle filter with the noise parameters recorded in our laboratory using the two robots. Figure 7.10 presents the positional uncertainty growth



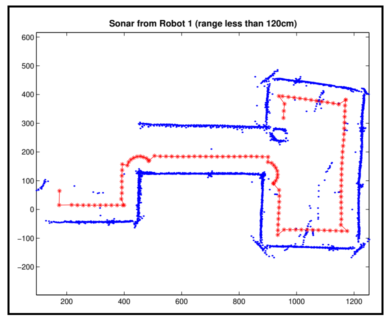
(a)



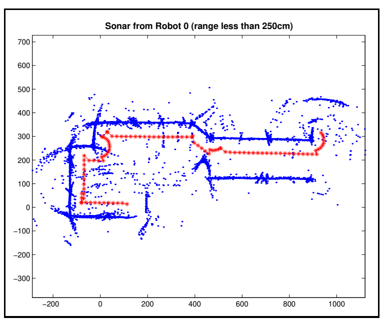
(b)



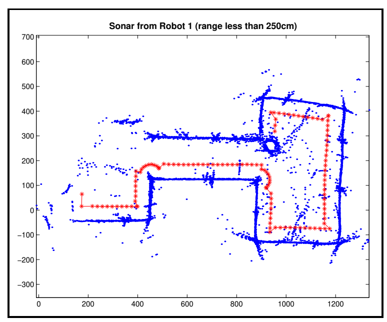
(c)



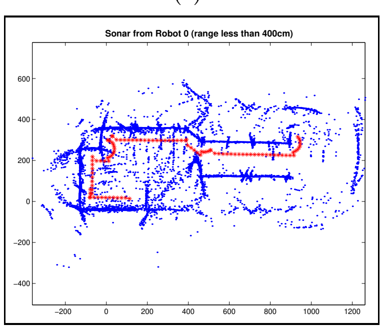
(d)



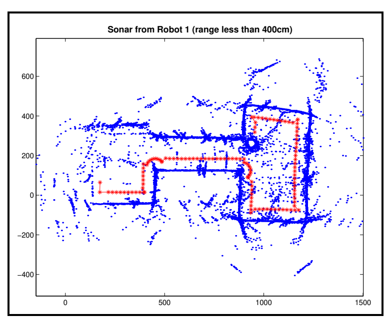
(e)



(f)

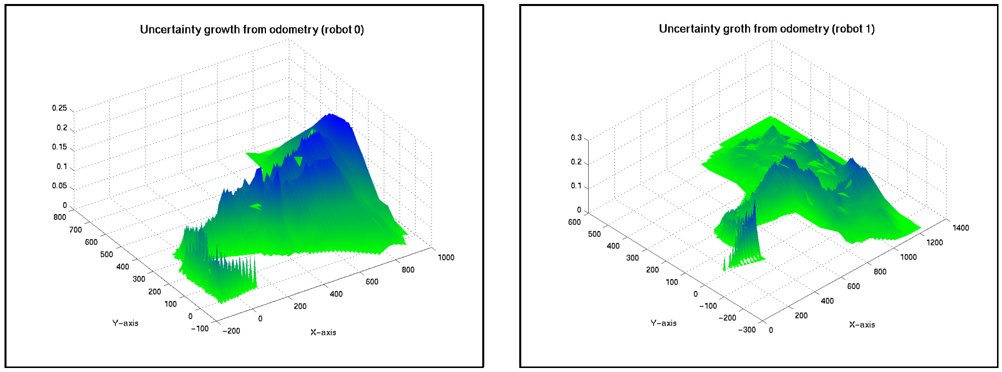


(g)



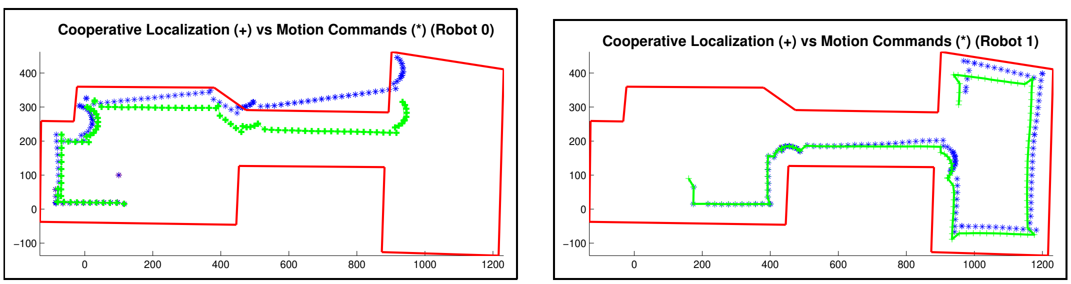
(h)

FIGURE 7.9. Sonar data collected from *Robot 0* (on the left) and *Robot 1* (on the right), filtered at different ranges: (a,b) 65cm; (c,d) 120cm; (e,f) 250cm; (g,h) 400cm. The sonar points are marked blue and the path of the robot is marked red.



(a) (b)

FIGURE 7.10. (a)The odometry uncertainty growth during the trajectory of *Robot 0*. (b) The odometry uncertainty growth during the trajectory of *Robot 1*.

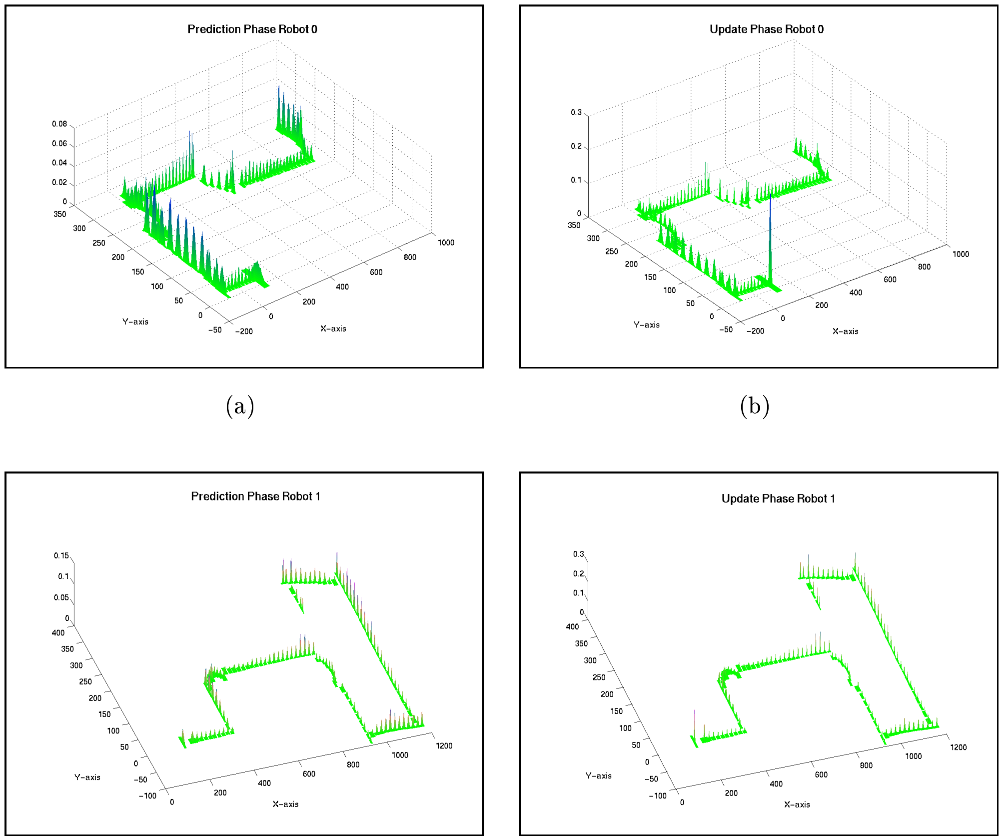


(a) (b)

FIGURE 7.11. (a)The cooperative localization estimates (*) versus the trajectory resulting from the motion commands (+) given to *Robot 0* during the exploration. (b) The same for *Robot 1*.

over the whole trajectory - the left figure is for *Robot 0* and the right for *Robot 1*. As can be seen the highest regions follow the intended path but over time the uncertainty spreads out.

Figure 7.11 presents the pose estimates during the exploration when cooperative localization was used (marked as green "+") together with the position of the robot estimated using the recorded motion commands (marked as blue "*"); the map of the environment is drawn in red. The left figure presents the trajectory of *Robot 0* and the right figure presents the trajectory of *Robot 1*. Even though the actual trajectory of each robot was kept in a



(a) (b) (c) (d)

FIGURE 7.12. (a) The prediction phase of the particles for the trajectory of *Robot 0*. (b) The update phase of the particles for the trajectory of *Robot 0*. (c),(d) Prediction and update phases for *Robot 1*.

straight line and closely corresponds with the cooperative localization estimates, the motion commands show a systematic drift (masked as blue "*" in Figure 7.11). The observed drift corresponds to the odometry error during the exploration. In other words, the motion commands drove the robot on the blue trajectory but because of odometric error the actual trajectory is marked in green.

The robot tracker sensor estimates were combined with the odometer estimates using a particle filter. As we saw in Chapter 5 the particle filter operates in two phases, prediction and update:, first the particles are moved in order to predict what the pose of the moving robot would be and then their weights are updated using the current sensor measurements. Figure 7.12 presents the spatial distribution of the particles for the prediction phase (Figure

Manual	Map	Absolute Error
588cm	577.5cm	0.5cm
305cm	296.5cm	8.5cm
99cm	96.0cm	3.0cm
102cm	102.6cm	0.6cm
410cm	403.1cm	6.9cm
99cm	114.4cm	15.4cm
419cm	419.7cm	0.7cm
179cm	178.0cm	1.0cm
341cm	333.4cm	7.6cm
545cm	548.0cm	3.0cm
343cm	343.5cm	0.5cm
241cm	249.9cm	8.9cm
432cm	427.9cm	4.1cm
168cm	173.0cm	5.0cm

TABLE 7.1. The length of the walls measured with tape and from the triangulation map (first two columns) together with the error. See Figure 2.2.1c for the correspondence between walls and lengths.

7.12a,c) and for the update phase (Figure 7.12b,d). The top row corresponds to *Robot 0* and the bottom row to *Robot 1*. The spread of the distribution in maintained narrow by the frequent updates.

The resulting map of the exploration can be seen in Figure 7.8c. The trajectory of *Robot 0* is marked in magenta and the trajectory of *Robot 1* is marked black. The walls are displayed in red and their lengths (in cm) is marked next to them. The internal diagonals that define the triangulation are marked as blue dashed lines. Moreover the lengths of the walls were measured manually (by measuring tape) and the results are presented in table 7.1 together with the estimated length from the triangulation map and the difference

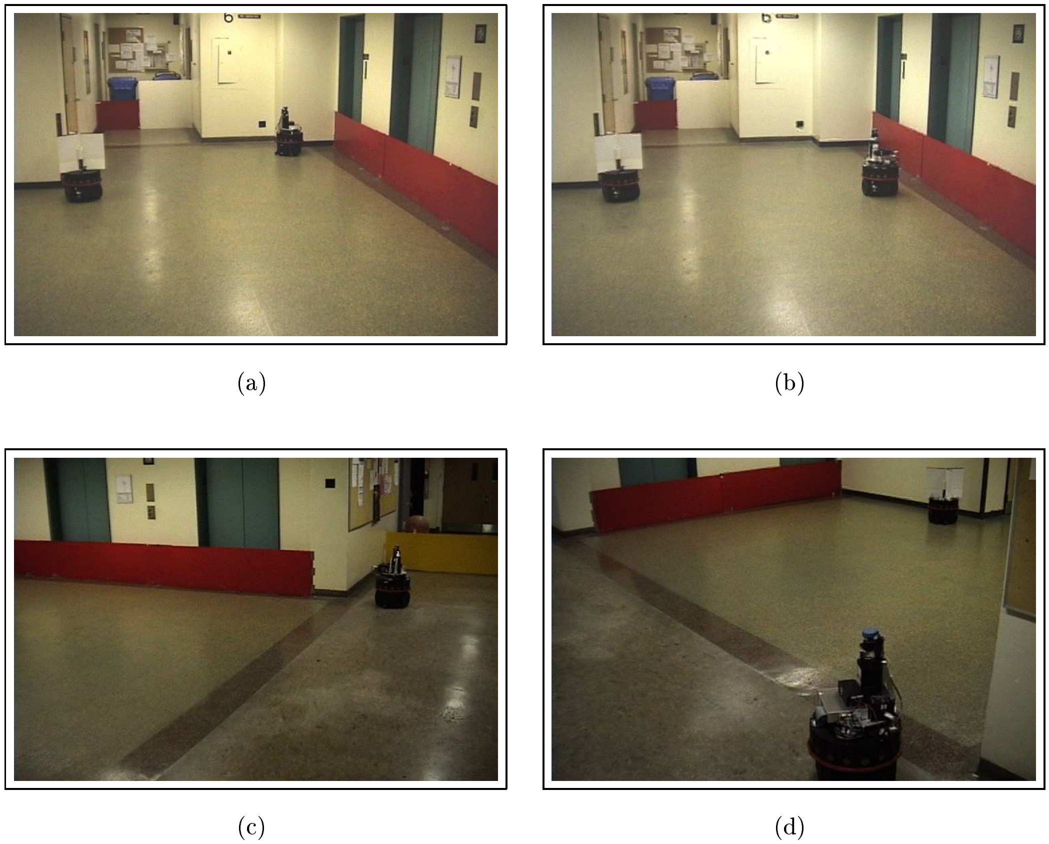
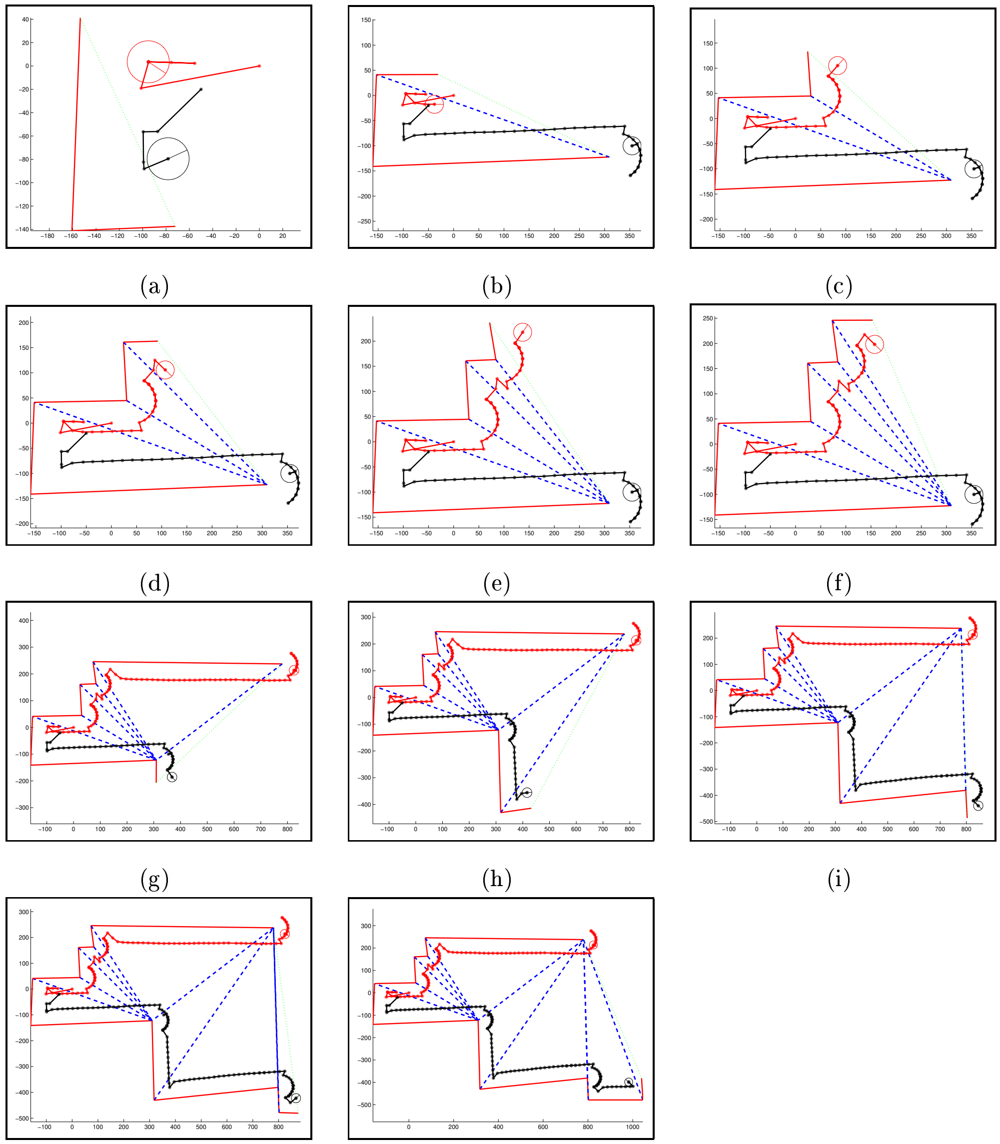


FIGURE 7.13. Exploration at the sixth floor of our building (a) *Robot 0* maps a reflex corner. (b) *Robot 0* follows a wall (c) *Robot 0* reached a reflex corner that interrupts line of sight. (d) *Robot 1* starts exploring after a roles exchange.

between the two measurements. The mean error was 4.6cm per wall, the perimeter of the environment mapped was measured to 42.71m while the perimeter of the resulting map was 42.63m. The angle of the two walls of the upper right corner was measured to 98° and the map estimate is 97°.

2.2.2. *Exploration of the 6th floor McConnell Eng. Building.* We explored a different environment in the hallways of the 6th floor McConnell Eng. Building in McGill University. Figure 7.13 presents pictures of this area from different views. The area explored was larger and the only modifications done were to cover the openings of the elevators. The two robots started at the back of the corridor facing the camera in Figure 7.13a. It was very narrow as can be seen in Figure 7.14a where *Robot 0* (in red) positioned itself on the to left corner and *Robot 1* (in black) started exploring clockwise from the lower left corner.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k)

FIGURE 7.14. The exploration hallways in the 6th floor. Red lines are the walls, blue dashed lines are diagonals, green dotted lines are gates to unexplored space.

Figure 7.14 presents the exploration process through the resulting triangulation. *Robot 1* explores first and maps one triangle before it encounters a reflex vertex and thus exchange roles with *Robot 0* (Figure 7.14b). *Robot 0* start exploring and two reflex vertices are mapped (Figures 7.14c,f, see also 7.13a for a picture of *Robot 0* at the second reflex corner). As these vertices do not interrupt the line of visual contact *Robot 0* continues the exploration (Figure

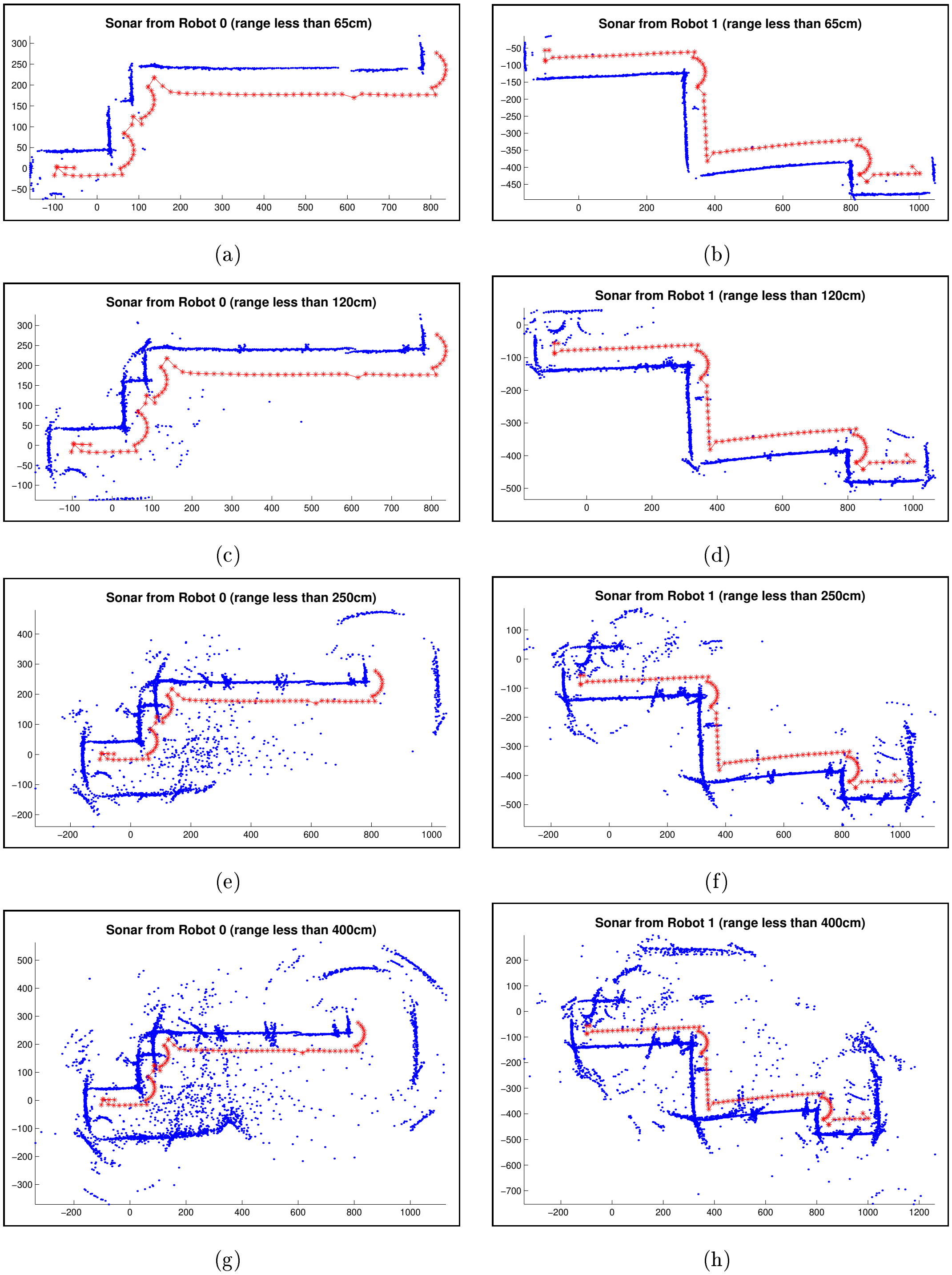


FIGURE 7.15. Sonar data collected from *Robot 0* (on the left) and *Robot 1* (on the right), filtered at different ranges: (a,b) 65cm; (c,d) 120cm; (e,f) 250cm; (g,h) 400cm. The sonar points are marked blue and the trajectory for the robot is marked red.

(a) (b)

FIGURE 7.16. (a) The laser data collected during the exploration. (b) Scan match applied to the laser data and their position corrected using Stephen Gutmann's scanstudio.

7.13b) mapping in total five triangles (see Figures 7.14b-f for the sequence of exploration. As can be seen in Figure 7.13c *Robot 0* reached a reflex corner that interrupts the line of visual contact and the two robots exchange roles. *Robot 1* proceeds to map the opposite side of the open area in front of the elevators (see Figure 7.13d). The distance across the open area was approximately 8m and the laser barely detected the target by scanning the lower part of it. *Robot 1* then proceeded to map the rest of the space (see Figure 7.14g-k).

The sonar data again were recorded and are presented in Figure 7.15 (in the same format as in Figure 7.9). It is worth noting that there is less noise, the main reason for this being the smoothness of the floor (lack of tiles). But in general the sonar sensor is not reliable for more than 2m. Part of the limitation comes from the height at which the sonar transducers are located on the robot. The laser data were also recorded and are presented using the Scanstudio: Figure 7.16a shows the raw measurements and Figure 7.16b shows the data after successful scan matching. The results are similar to the triangulation map obtained which is presented in Figure 7.17. As can be seen there is not much difference between the raw and the scan matched laser data because the pose of the robot is maintained through cooperative localization.

The motion commands were used again to estimate the trajectory of the robots (marked as blue "*") and compared with the pose estimates from cooperative localization (marked as

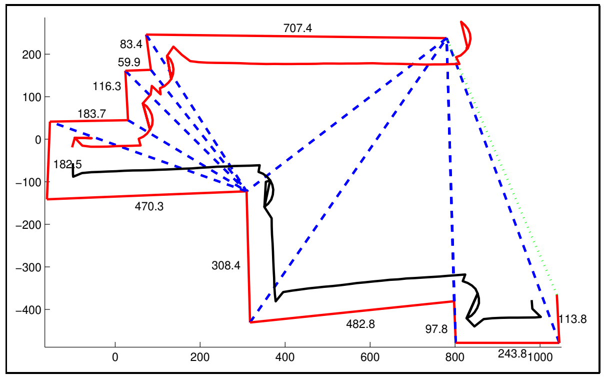
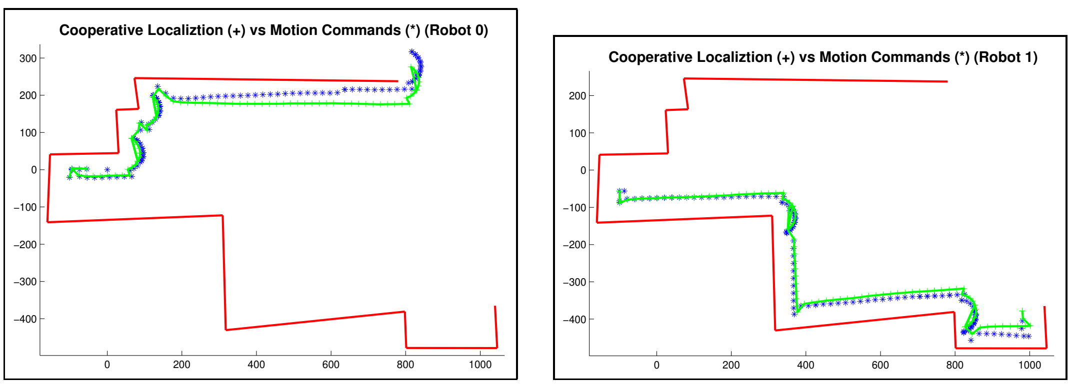


FIGURE 7.17. The triangulation map produced using only the sonar for mapping. The trajectory of *Robot 0* is marked in magenta and the trajectory of *Robot 1* is marked black. The walls are displayed in red and their lengths (in cm) is marked next to them. The internal diagonals that define the triangulation are marked as blue dashed lines.

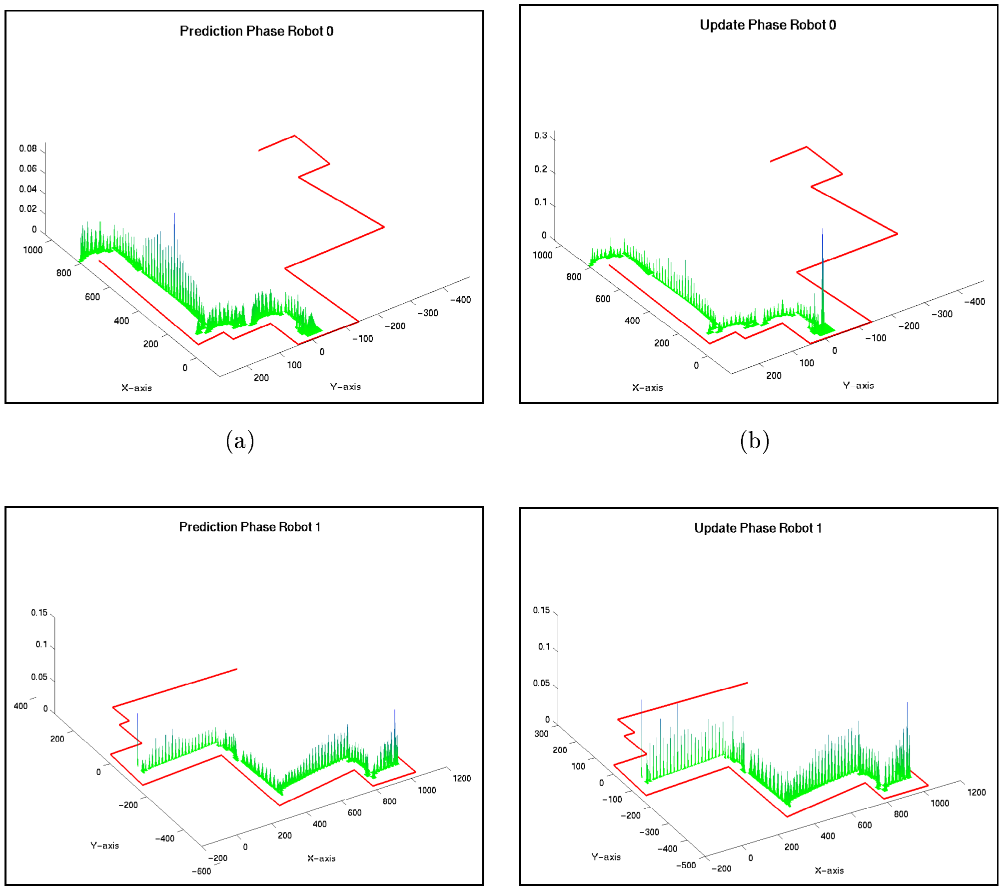


(a) (b)

FIGURE 7.18. (a) The cooperative localization estimates (*) versus the trajectory resulting from the motion commands (+) given to *Robot 0* during the exploration. (b) The same for *Robot 1*.

green "+") in Figure 7.18a,b, for *Robot 0* and *Robot 1* respectively. The walls are presented in red. Again the drift due to odometry error is clearly detectable.

Figure 7.19 present the distribution of particles during the prediction phase 7.19a,c and the update phase 7.19b,d. *Robot 0* is presented on the top row and *Robot 1* on the bottom; the walls are drawn in red. As can be seen during the prediction phase the particles spread and then during the update phase they create a peak; this also can be seen at the z axes of the graphs, for prediction the highest value is 0.08 and for update is 0.3; these numbers



(a) (b) (c) (d)

FIGURE 7.19. (a) The prediction phase of the particles for the trajectory of *Robot 0*. (b) The update phase of the particles for the trajectory of *Robot 0*. (c),(d) Prediction and update phases for *Robot 1*. The walls of the environment are indicated in red.

represent the confidence of the estimation. In general though the particle distribution is kept concentrated by the frequent updates with the estimates of the robot tracker sensor.

A second experiment was recorded on the 6th floor with a small modification in the environment: two additional walls were used forming a reflex corner in front of the elevators. Figure 7.20 presents the exploration sequence. The new corner is mapped in Figure 7.20h. The two robots started in similar positions as in the previous experiments and they proceed to map the environment, *Robot 1* moving first (Figures 7.20a,b). Then *Robot 0* maps the top part of the environments (Figure 7.20c-i). *Robot 1* then proceeds counter clockwise and completes the exploration (see Figures 7.20j-o).

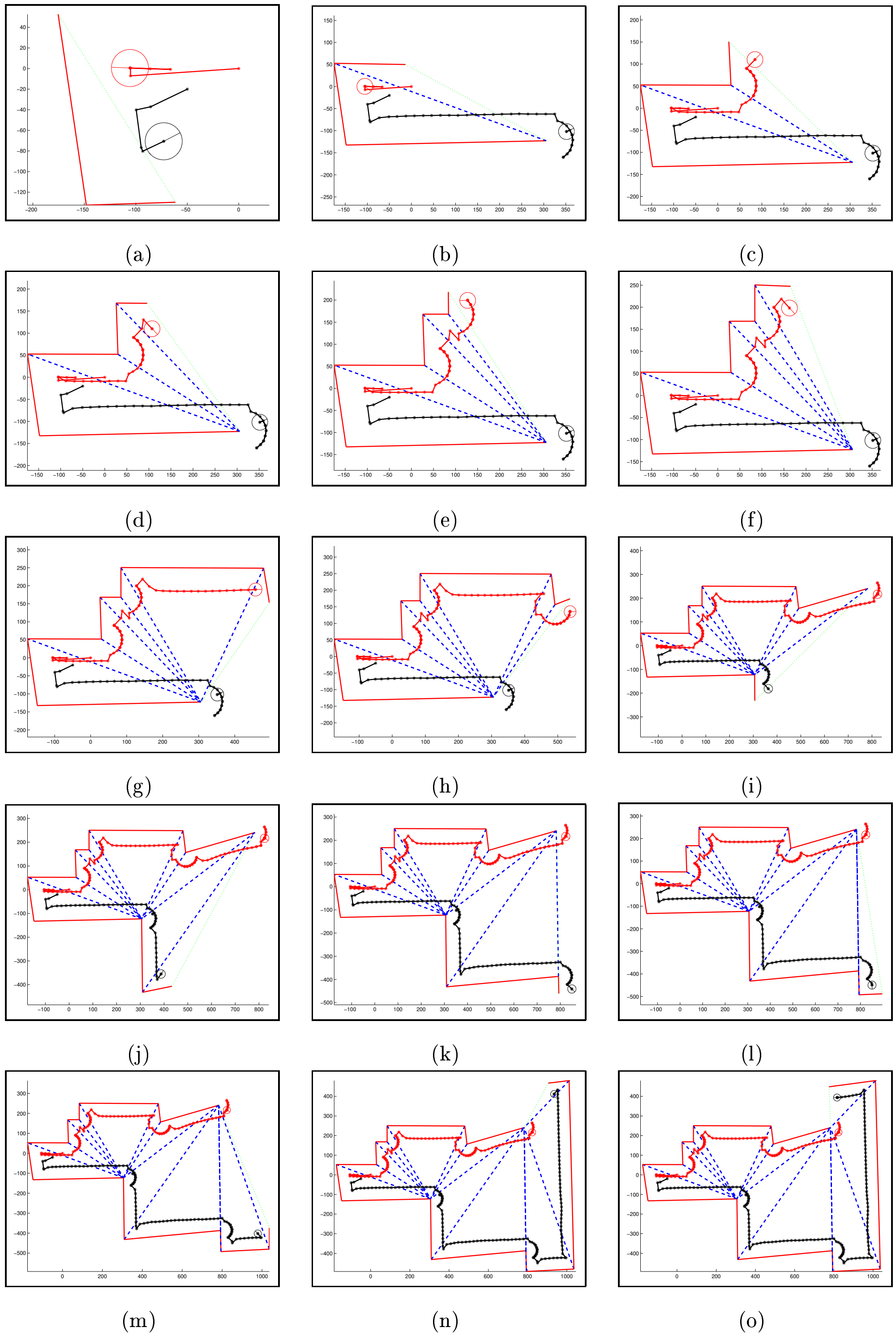
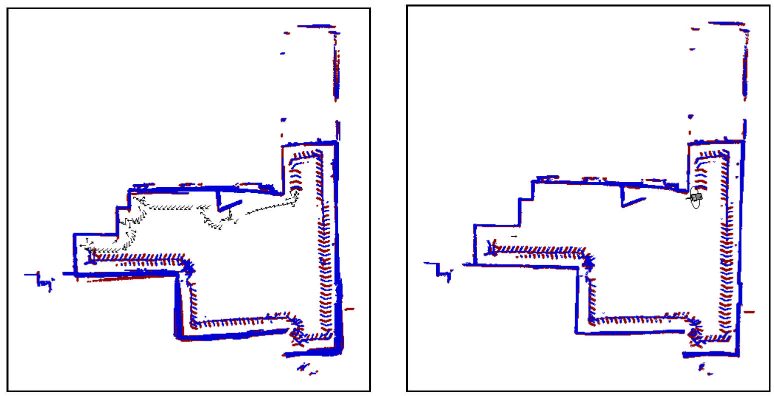


FIGURE 7.20. The exploration of the 6th floor with a small modification. Red lines are the walls, blue dashed lines are diagonals, green dotted lines are gates to unexplored space.



(a) (b)

FIGURE 7.21. (a) The laser data collected during the exploration. (b) Scan match applied to the laser data and their position corrected using Stephen Gutmann's scanstudio. (c) The triangulation map produced using only the sonar for mapping.

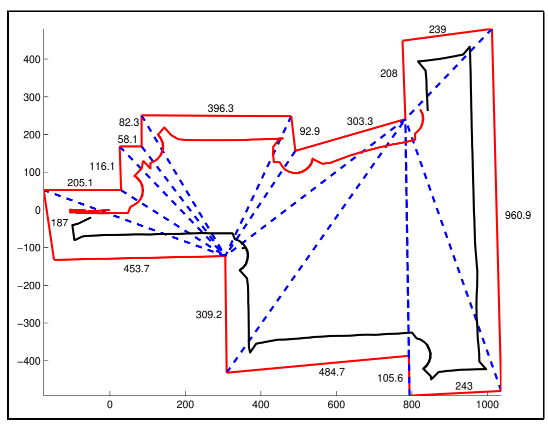


FIGURE 7.22. The triangulation map produced using only the sonar for mapping. The trajectory of *Robot 0* is marked in magenta and the trajectory of *Robot 1* is marked black. The walls are displayed in red and their lengths (in cm) is marked next to them. The internal diagonals that define the triangulation are marked as blue dashed lines.

The recorded laser data can be seen in Figure 7.21a,b; note the target detected along the trajectory of *Robot 1*. The resulting map is presented in Figure 7.22, the numbers represent the estimated length of the walls and they agree again up to a few centimeters with the measured length of the walls (as in the map of the two laboratories).

2.2.3. *Experiments with the Visual Robot Tracker.* In addition to the experiments with the laser robot tracker sensor the visual robot tracker was used in a small environment.

Preliminary exploration tests were carried out in our laboratory in workspaces of area roughly 16 m^2. This comparatively small test-bed allowed us to control various factors such as inhomogeneities in the terrain as a function of trajectory and obtain ground truth data. Using this test-bed we compared the time, accuracy, and robustness of different exploration strategies. In our experimental arrangement the role of the stationary robot is played by a tripod mounted camera at the same height as Nomad 200 from Nomadics Technologies, Inc. The camera was placed next to the first wall. This allowed us to more reliably and repeatably verify ground truth. It is worth noting that our strategy works equally well with homogeneous robots and with heterogeneous robots, eg. one robot has a camera the other robot has the pattern.

A laser pointer pointing straight down to the floor has been placed on top of the moving robot in order to accurately mark its current position on the floor. This setup allowed us to measure the displacement from the initial position after the completion of the tour.

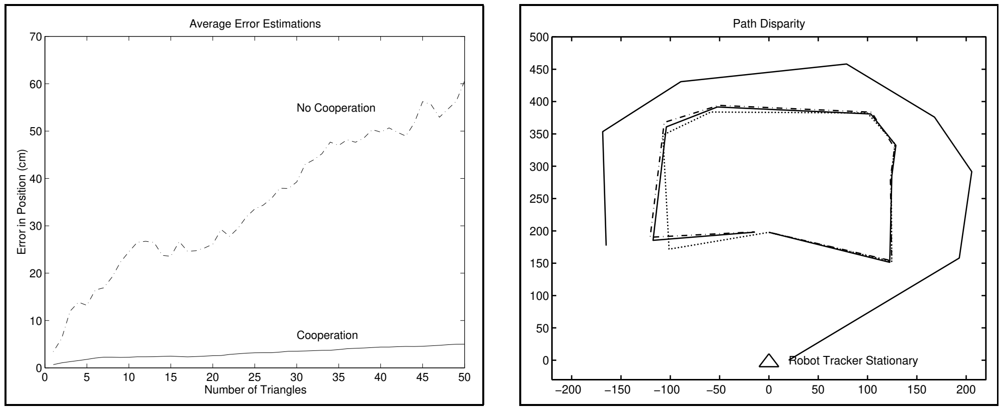
Figure 7.23b shows the actual path of the moving robot, the odometry-based estimate of position, and the tracker-based estimate. The final displacement from pure odometry estimates is approximately 15cm with an orientation error of 15°. The tracker estimate has approximately 1.3cm error. This corroborates our assumption that joint exploration and localization using a "tracker" can lead to much more robust modeling than odometry alone.

3. More than two robots

In this section we discuss the benefits of *cooperative localization* for a team of mobile robots. Furthermore, we consider the effects of different robot tracker sensors on the accuracy of localization for a moving robot *using only* the information from the rest of the robots (as opposed to observations of the environment). This approach results in an open loop estimate (with respect to the entire team) of the moving robot's pose without dependence on information from the environment. The experimental results allows us to examine the effectiveness of cooperative localization and estimate upper bounds on the error accumulation for different sensing modalities.

3.1. Cooperative Localization.

Several different sensors have been employed for the estimation of the pose of one robot with respect to another robot. We restrict our attention to robot tracker sensors which return information in the frame of reference of the



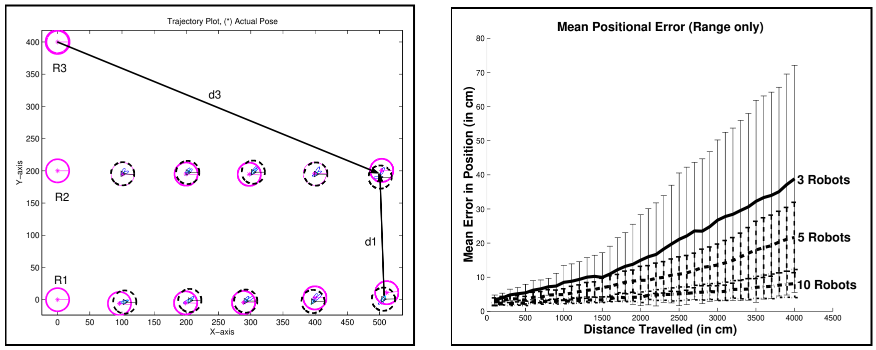
(a) (b)

FIGURE 7.23. (a) The average error during the exploration of 50 triangles (over 100 experiments) without and with cooperative localization. (b)The path of the robot after the completion of the exploration. The outside solid line marks the position of the walls the moving robot followed. The actual path of the robot is the solid line, the odometry based estimate of position is the dotted line, while the tracker estimate is the dashed-dotted line.

observing robot (i.e they estimate pose parameters relative to the robot making the observation). Consequently, for “two-dimensional robots” in a two dimensional environment, or for robots whose pose can be approximated as a combination of 2D position and an orientation, we can express the pose using three measurements; for ease of reference we represent them by the triplet $T = [\rho \ \phi \ \theta]$, where ρ is the distance between the two robots, ϕ is the angle at which the observing robot sees the observed robot relative to the heading of the observing robot, and θ is the heading of the observed robot as measured by the observing robot relative to the heading of the observing robot. If the stationary robot is equipped with the Robot Tracker, where $\mathbf{X}_m = [x_m, y_m, \theta_m]^T$ is the pose of the moving robot and $\mathbf{X}_s = [x_s, y_s, \theta_s]^T$ is the pose of the stationary robot then equation 7.1 returns the sensor output T:

$$\begin{bmatrix} \rho \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{dx^2 + dy^2} \\ atan2(dy, dx) - \theta_s \\ atan2(-dy, -dx) - \theta_m \end{bmatrix} \begin{array}{l} \text{Where :} \\ , dx = x_m - x_s \\ dy = y_m - y_s \end{array} \tag{7.1}$$

In order to estimate the probability distribution function (*pdf*) of the pose of the moving robot i at time t ($P(\mathbf{X}_i^t)$) we employ a particle filter (Monte Carlo simulation approach: see



(a) (b)

FIGURE 7.24. (a) Estimation of the pose of robot R2 using only the distance from robot R1 (d1) and from robot R3 (d3). (b) Average error in position estimation using the distance between the robots only (3,4 and 10 robots; bars indicate standard deviation).

[**80, 37, 98**]). The weights of the particles (W_i^t) at time t are updated using a Gaussian distribution (see equation 7.2 where $[\rho_i, \theta_i, \phi_i]^T$ has been calculated as in equation 7.1 but using the pose of particle "i" ($\mathbf{X}_{m_i}$) instead of the moving robot pose ($\mathbf{X}_m$)).

$$W_i^t = W_i^{t-1} \frac{1}{\sqrt{2\pi}\sigma_\rho} e^{\frac{-(\rho-\rho_i)^2}{2\sigma_\rho^2}} \frac{1}{\sqrt{2\pi}\sigma_\theta} e^{\frac{-(\theta-\theta_i)^2}{2\sigma_\theta^2}} \frac{1}{\sqrt{2\pi}\sigma_\phi} e^{\frac{-(\phi-\phi_i)^2}{2\sigma_\phi^2}} \tag{7.2}$$

3.2. Sensing Modalities. As noted above, several simple sensing configurations for a robot tracker are available. For example, simple schemes using a camera allow one robot to observe the other and provide different kinds of positional constraint such as the distance between two robots and the relative orientations. Moreover the group size affects the accuracy of the localization.

In the next part we present the effect the group size has on the accuracy of the localization for different sensors. The experimental arrangement of the robots is simulated and is consistent across all the sensing configurations. The robots start in a single line and they move abreast one at a time, first in ascending order and then in descending order for a set number of exchanges. The selected robot moves for 5 steps and after each step cooperative localization is employed and the pose of the moving robot is estimated. Each step is a forward translation by 100cm. Figure 7.24a presents a group of three robots, after the first robot has finished the five steps and the second robot performs the fifth step.

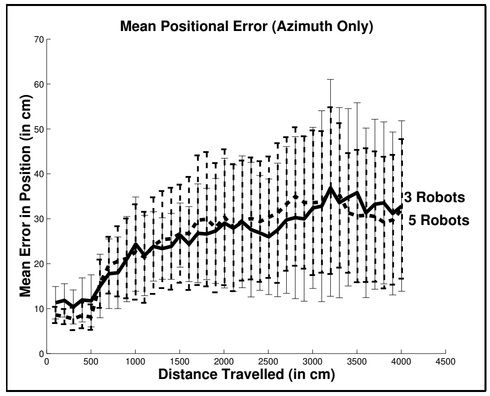


FIGURE 7.25. Average error in position estimation using the orientation of the moving robot is seen by the stationary ones.

3.2.1. *Range Only.* One simple tracking method is to return the relative distance between the robots. Such a method has been employed by [**67**] in the millibots project where an ultra-sound wave was used in order to recover the relative distance. In order to recover the position of one moving robot in the frame of reference of another at least two stationary robots (not collinear with the moving one) are needed thus the minimum size of the group using this scheme is three robots.

Estimating the distance between two robots is very robust and relatively easy. In experimental simulations, the distance between every pair of robots was estimated and Gaussian, zero mean, noise was added with $\sigma_\rho = 2cm$ regardless the distance between the two robots. Figure 7.24b presents the mean error per unit distance traveled for all robots, averaged over 20 trials. As can be seen in Figure 7.24b with five robots, the positional accuracy is acceptable with an error of 20cm after 40m traveled; for ten robots the accuracy of the localization is very good.

3.2.2. *Azimuth (Angle) Only.* Several robotic systems employ an omnidirectional vision sensor that reports the angle at which another robot is seen. This is also consistent with information available from several types of observing systems based on pan-tilt units. In such cases orientation at which the moving robot is seen can be recovered with high accuracy. We performed a series of trials using only the angle at which one robot is observed, with groups of robots of different sizes. As can be seen in Figure 7.25 the accuracy of the localization does not improve as the group size increases. This is not surprising because small errors in the estimated orientation of the stationary robots scale non-linearly with the

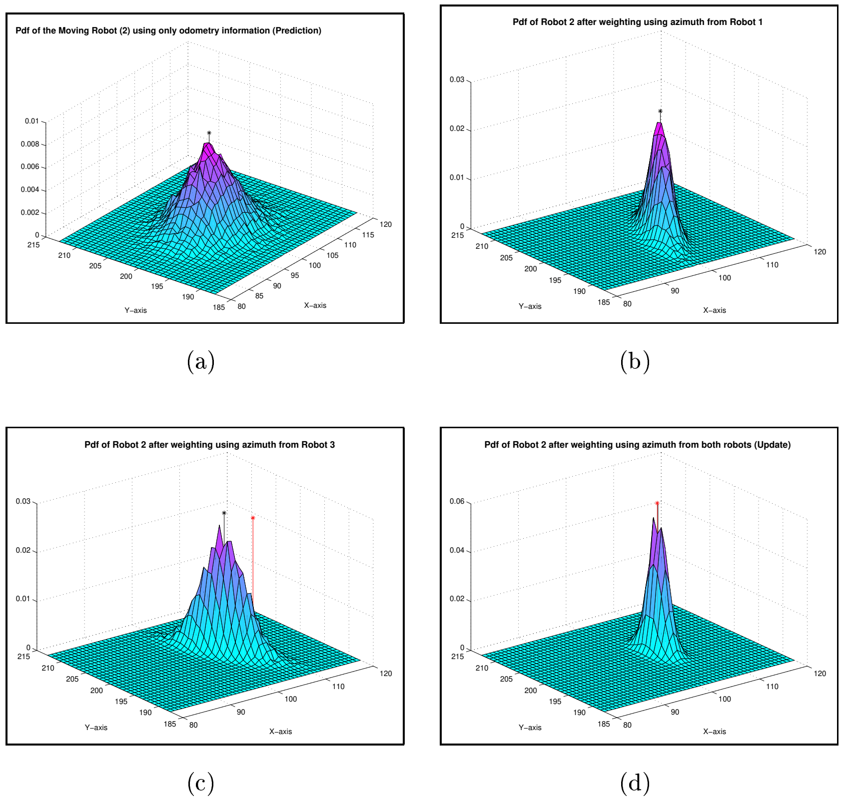
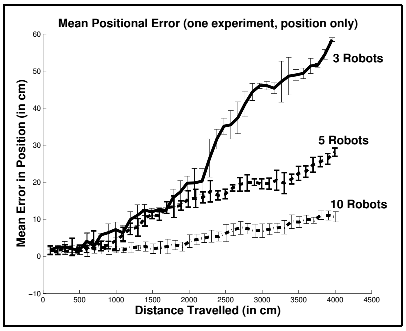


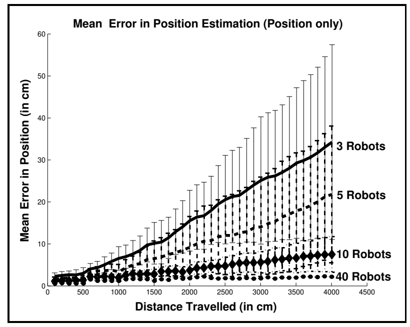
FIGURE 7.26. The *pdf* of the moving robot (R2) at different phases of its estimation: (a) prediction using odometry only; (b) using the orientation from stationary robot R1; (c) using the orientation from stationary robot R3; (d) final *pdf*.

distance. Thus after a few exchanges the error in the pose estimation is dominated by the error in the orientation of the stationary robots.

To illustrate the implementation of the particle filter, we present the probability distribution function (*pdf*) of the pose of the moving robot after one step (see Figure 7.26). The robot group size is three and it is the middle robot R2 that moves. The predicted *pdf* after a forward step (using odometry information only) can be seen in the Figure 7.26a the next two Figures 7.26 and 7.26c present the *pdf* updated using the orientation at which the moving robot is seen by a stationary one (first by robot R1 then by robot R3); finally, the Figure 7.26d presents the final *pdf* which combines the information from odometry and the observations from the two stationary robots. Clearly the uncertainty of the robot's position is reduced with additional observations.



(a)



(b)

FIGURE 7.27. Average error in position estimation using both the distance between the robots and the orientation the moving robot is seen by the stationary ones. (a) Average error in positioning of the team of robots one trial (3,5 and 10 robots). (b) Average error in position estimation over twenty trials (3,5, 10 and 40 robots).

3.2.3. *Position Only.* Another common approach is to use the position of one robot computed in the frame of reference of another (relative position). This scheme has been employed with two robots (see [**24**]) in order to reduce the uncertainty. The range and azimuth information ($[\rho, \theta]$) is combined in order to improve the pose estimation. As can be seen in Figure 7.27a even with three robots the error in pose estimation is relatively small (average error 30cm for 40m distance traveled per robot, or 0.75%). In our experiments the distance between the two robots was estimated and, as above, zero-mean Gaussian noise was added both to distance and to orientation with $\sigma_\rho = 2cm$ and $\sigma_\theta = 0.5°$ respectively. The experiment was repeated twenty times and the average error in position is shown in Figure 7.27b for groups of robots of size 3,5,10 and 40.

3.2.4. *Full Pose.* Some robot tracker sensors provide accurate information for all three parameters $[\rho, \theta, \phi]$ and they can be used to accurate estimate the full pose of the moving robots (see [**87, 137**]). In the experimental setup the robot tracker sensor was characterized by Gaussian, zero mean, noise with $\sigma = [2cm, 0.5°, 1°]$. By using the full equation 7.2 we weighted the *pdf* of the pose of the moving robot and performed a series of experiments for 3, 5 and 10 robots; very low positional error was observed (see Figure 7.28).

3.2.5. *Summarizing.* In the previous sections we examined the effect of the size of the team of robots and the sensing paradigm on cooperative localization; a synopsis

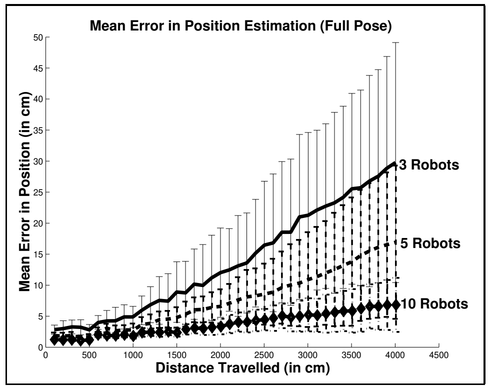


FIGURE 7.28. Average error in position estimation using full pose $[\rho, \theta, \phi]$.

# of Robots	3	5	10
Range only (ρ)	38.80cm	21.63cm	8.13cm
Azimuth only (θ)	27.06cm	32.20cm	33.72cm
Position only (ρ, θ)	34.25cm	21.79cm	7.50cm
Full Pose (ρ, θ, ϕ)	28.73cm	16.71cm	6.05cm

TABLE 7.2. The mean error in position estimation after 40m travel over 20 trials.

of the results can be seen in Table 7.2. Also, preliminary results from experiments with varying odometry error have shown that cooperative localization is robust even with 10-20% odometry errors.

4. Discussion

The experiments performed with the real robots made clear the fact that different sensors have different strengths and weaknesses. The sonar sensor is very noisy even at the range of two meters, it was able however to robustly detect obstacles of different heights in close range. Therefore, the sonar sensor is very useful for obstacle avoidance and wall following in close range. On the contrary, the laser range finder has higher accuracy but is limited on one plane only; small variations on the floor inclination changed the height at which the laser was scanning by as much as 25cm. The laser sensor was valuable as part of the robot tracker sensor but it was not possible to utilize it at its full range because of the floor inclination. During an experiment on the 7th floor of our building the target mounted on the robot was not detected even at a distance of 7m. Moreover, the laser range finder would miss any obstacle above or below the scanning plane and thus it can not be used for navigation. Extending the robot tracker in such a way that the scanning plane would be controlled (e.g. by using a pan and tilt unit) is necessary for any use in real world applications. One other limitation of the current implementation of the triangulation algorithm is the mapping of small walls, in particular when the robot goes around a reflex corner; the length of the adjacent walls should be minimum 50cm.

The experimental results verify the improvement in the map accuracy. Areas of 13m by 5m and 1200m by 9m were mapped completely, with mean error less than 5cm. The perimeters of the environments mapped were of the order of 42-44 meters. These results demonstrate the practical feasibility of the algorithms used, and illustrate some of the performance characteristics we predicted. In the next chapter we present the use of cooperative localization in order to assist in the mapping of the spatial distribution of a property of interest over an unknown environment, a process akin to coverage. A heterogeneous team of robots is used.

CHAPTER 8

Collaborative Exploration for Visual Map Construction

There is nothing more practical than a good theory.
–Author unknown.

1. Introduction

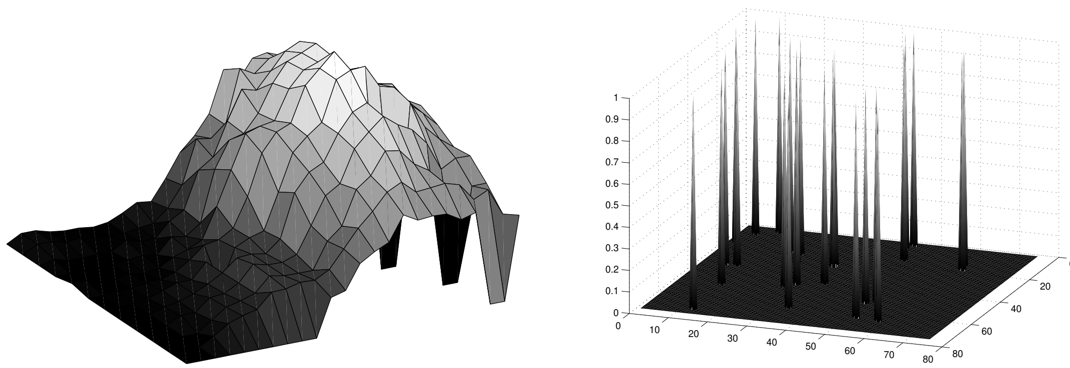
In the previous chapters we examine the use of cooperative localization together with mapping by sweeping the line of visual contact. In this chapter we are going to apply cooperative localization in a different mapping application. In particular, we discuss how to map a property of interest over an unknown environment. A significant issue faced by many map-building schemes is the management and estimation of positional (or pose) errors as the robot collects observations from the environment. That is, as the robots collect successive measurements from different poses, the certainty of their pose estimates decreases with each new measurement. In some cases where the observation lies on a high-dimensional manifold, correlation between dimensions allows for globally consistent alignment of the observations via an expectation-maximization or iterative optimization approach to correcting the observation poses [**103, 169**]. However, it is often the case that either there is insufficient geometric constraint in the observations to produce confident pose estimates even *post hoc*, or that the computational cost of making the appropriate inferences is infeasible. Uncertainty modeling methods such as Kalman filtering can reduce the severity of the problem, but certainly do not eliminate it.

FIGURE 8.1. Collaborative explorers

This chapter addresses the problem of establishing accurate pose estimates in the context of robotic mapping. The pose estimates can be used to collect accurately localized measurements in their own right, or as a precursor to a system that builds a map. The robot collecting measurements for the map operates in concert with a second robot that acts as an active observer. In our *cooperative localization* scheme, this second robot tracks the motions of the first as it collects data and provides it with the information required to prevent odometric error from accumulating. We can view the robots are being "connected" by a *virtual tether* which is established between the two robots and which enables the task of mapping to be accomplished without significant error and independent of the ground surface conditions and the quality of the odometry estimate. In particular, the property of interest we map is the visual appearance and the utility of visual landmarks as developed by Robert Sim [**154**]. In principle, more than one of these active observers could be used simultaneously, although this is not elaborated in this thesis. Beyond presenting the details of the approach and its implementation, this chapter provides a quantitative evaluation validating the effectiveness of this methodology.

The remainder of this chapter is structured as follows: Section 2 discusses the general framework in which our approach applies. We then discuss a particular application of our approach to the task of visual landmark learning in Section 3 and experimental results are presented in Section 4.

2. Motivation



(a) (b)

FIGURE 8.2. Mapping: **(a)** Continuous function such as: Radiation, Visual appearance, Elevation, Magnetic field, Temperature, etc. **(b)** Discrete function such as: Mine detection, Lost objects, Holes, Electrical outlets, etc.

The work presented here is motivated by the need to use a mobile robot in order to accurately map a spatially varying property of an unknown (possibly hazardous) environment. Such a property could be a continuous function (see Figure 8.2a) over the accessible area such as radiation, temperature, magnetic field variation, elevation or visual appearance, or the property could be a discrete function (see Figure 8.2b) such as presence of mines, lost objects, holes/anomalies on the ground, or electrical outlets. In most cases the sensor used to map arbitrary properties such as those noted above is not suitable for the accurate localization of the exploring robot – for example, a radiation meter cannot readily be used to accurately recover the pose of the exploring robot. Therefore, the self-localization ability of a single robot on the basis of the measurement of the continuous function of interest is poor without the assistance of additional sensory apparatus. Furthermore, the ground surface quality may be uneven, resulting in wheel slippage, and rendering the odometry sensors unreliable. Our approach employs cooperative localization as described in the previous chapters in order to recover the pose of the exploring robot with high accuracy, independent of the ground surface properties and the reliability of the odometry sensors.

Another motivation for using more than one mobile robot is that several applications require the exploration or inspection of hazardous environments with an attendant risk to the robot doing the work. Such applications include but are not limited to: de-mining rural areas, inspecting nuclear facilities or marking/mapping of chemical spills. In order to improve robustness or reduce the potential cost in such a scenario we can deploy a team of heterogeneous robots consisting of a "base" robot which is equipped with the main computer, a communication module and the robot tracker sensor, and a team of lower-cost "exploring" robots that are equipped with only the mapping sensor (and the target for the robot tracker). In particular, our scheme obviates the need for accurate odometry on the exploring robots. The base robot is always located at a safe area keeping visual and radio contact with the exploring robots. If any of the exploring robots are destroyed the expense is limited, and the mission can continue with the surviving robots.

3. Application: Landmark Learning

In this section we demonstrate the effectiveness of our approach as it applies to the problem of learning visual landmarks which are useful for the task of pose estimation of a single robot equipped with a camera based on visual observations of the environment. The trackers described in the Chapter 6 Sections 2,3 can be employed to properly register the landmark observations on the map, i.e. to provide "ground truth" positions while the robot explores the visual environment. We employ the landmark learning framework developed by Sim and Dudek and described in [**53, 151**] and [**149**], which tracks the set of points output by an arbitrary model of visual attention and attempts to construct a representation of the landmark as a function of the pose of the robot. Such a representation can then be later exploited for the task of estimating the pose of the robot from detected visual landmarks in the absence of a second robot or a tracker.[1].

[1]The work on visual landmarks is used as a test-bed in order to validate the *cooperative localization* approach introduced in this thesis. A brief overview has been included here for completeness sake. For a detailed description of the visual landmarks based localization please refer to [**154, 155, 152, 153, 151, 149, 150, 157, 156**] and to http://www.cim.mcgill.ca/~simra.

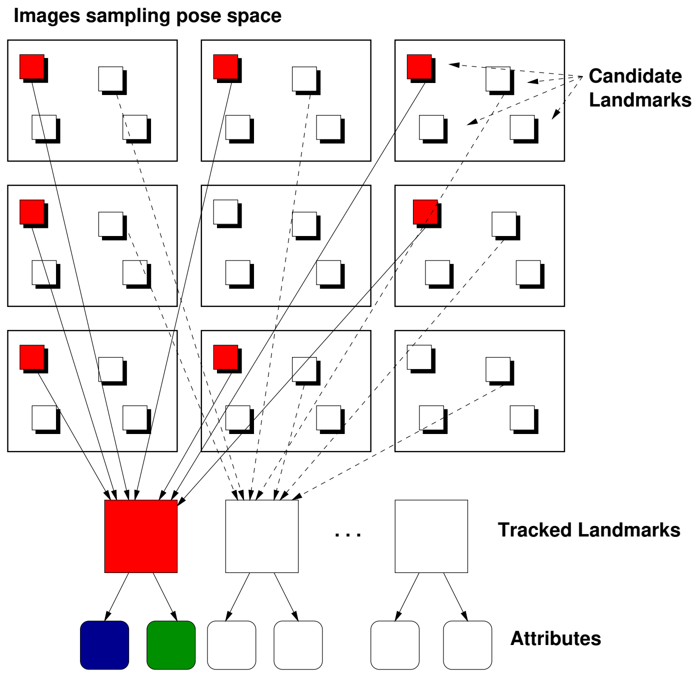


FIGURE 8.3. The off-line training method. Images (large rectangles) are collected sampling the pose space. Landmarks are extracted from the images and matched across the samples. The *tracked landmarks* are parameterized as a function of pose and saved for future pose estimation. Figure courtesy of R. Sim.

The learning method is depicted in Figure 8.3 and operates as follows (refer to the cited work for further details):

(i) **Exploration:** One robot tracks the other as it collects images sampling a range of poses in the environment. The pose at which each image is taken is recorded as the estimate given by the tracker.

(ii) **Detection:** *Landmark candidates* are extracted from each image using a model of visual attention. Landmark candidates are rectangular image regions that satisfy a visual attention criterion.

(iii) **Matching:** *Tracked landmarks* are extracted by tracking visually similar candidate landmarks over the configuration space.

(iv) **Parameterization:** The tracked landmarks are parameterized on the basis of a set of computed landmark attributes (for example, position in the image, intensity distribution, edge distribution, etc), and then measured in terms of their *a priori utility* for pose estimation.

(v) The set of sufficiently useful tracked landmarks is stored for future retrieval.

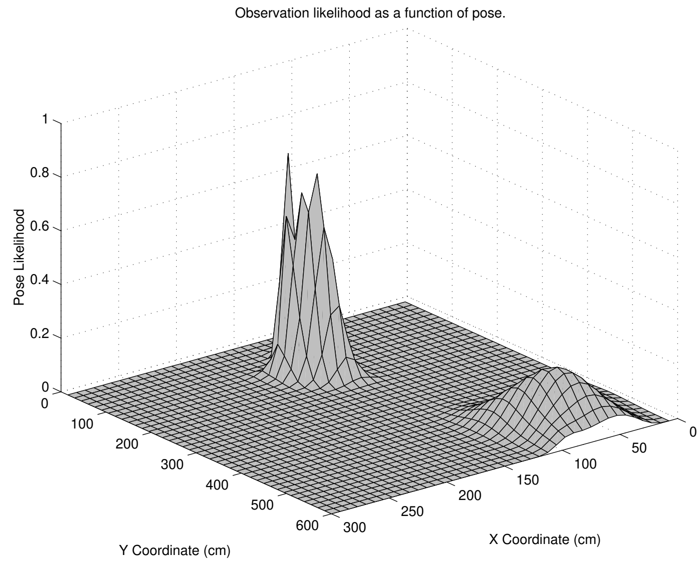


FIGURE 8.4. The likelihood of an observation as a function of pose. Figure courtesy of R. Sim.

For the purposes of our experiments, the visual landmarks are initially selected from a subset of the training images using an attention operator that responds to local maxima of edge density in the image. The selected landmark candidates are then tracked over the remaining images along the robot's trajectory by maximizing correlation with the local appearance of the initially detected landmark. The set of matches to a given candidate constitute a *tracked landmark*, and is stored for parameterization and evaluation.

The parameterization of each landmark feature f_i is accomplished by employing a radial basis function regularization framework to model the observation generating function

$$\mathbf{z} = F_i(\mathbf{q}), \tag{8.1}$$

where $\mathbf{z}$ is a low dimensional vector-valued representation of the landmark attributes and $\mathbf{q}$ is the pose of the robot. In other words, $F_i(\cdot)$ is the function that predicts the attributes of the landmark as a function of the pose of the robot. Furthermore, the landmark is evaluated for its utility by computing the covariance C of a randomly sampled subset of leave-one-out cross-validation residuals over the training set.

The parameterization of each landmark affords a maximum likelihood prediction of an observation, given an *a priori* pose estimate $\mathbf{q}$, as well as a measure of the uncertainty (C) of that prediction. As such, the landmark models are useful for the task of *probabilistic robot localization.* That is, we can construct a likelihood function $p(\mathbf{z}|\mathbf{q})$ which allows us to measure the likelihood of an observation $\mathbf{z}$, assuming knowledge of the robot's pose $\mathbf{q}$. Such a likelihood function can be employed in a *Bayesian framework* to infer the probability distribution of $\mathbf{q}$ given the observation $\mathbf{z}$:

$$p(\mathbf{q}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{q})p(\mathbf{q})}{p(\mathbf{z})} \tag{8.2}$$

where $p(\mathbf{q})$ represents the prior information about $\mathbf{q}$ and $p(\mathbf{z})$ is a constant relative to the independent variable $\mathbf{q}$. Several such probability distributions can be generated– one for each observed landmark– and can be combined to obtain a full posterior pose distribution. Note that this framework is more generic than a Kalman filter in that it allows for a multi-modal representation of the pose likelihood.

When the robot requires a pose estimate without the aid of the tracker, it can obtain a camera image and locate the learned landmarks in the image using the predictive model. The differences in appearance and position between the prediction and the observation of each landmark are combined to compute the likelihood of the observation in the Bayesian framework. This process is illustrated in Figure 8.5. The maximum *a priori* pose estimate can be recovered by gradient ascent over the observation likelihood as a function of pose. An example likelihood function is plotted at a coarse scale in Figure 8.4. Note that the pose likelihood is a useful measure of confidence in the final estimate allowing for the rejection of outlier pose estimates on the basis of a user-defined threshold.

As noted, the pattern of landmarks observed and computed over the environment during the mapping stage can be used for accurate single-robot pose estimation.

4. Experimental Results: Construction of Landmark-based Visual Maps

In this section we present the results of deploying the tracking method for the task of landmark learning. Our environment consisted of a laboratory partitioned into two "rooms" by room dividers, with an open doorway connecting them. The first two pictures

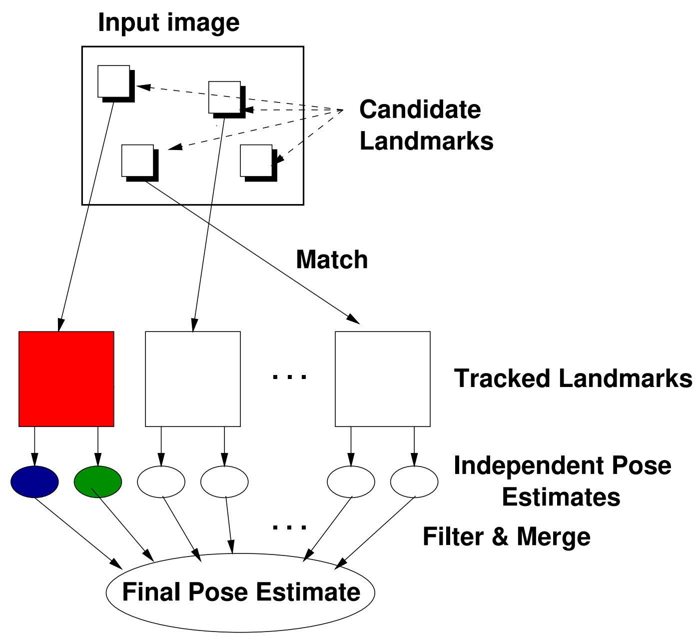


FIGURE 8.5. Pose estimation based on learned visual landmarks. Landmarks (small squares) are extracted from the current camera observation and matched to the previously learned tracked landmarks. Each match generates a pose estimate, which are filtered and combined to generate a final pose estimate. Figure courtesy of R. Sim.

in Figure 8.6 are the robot's-eye-view of the two rooms, and the third picture presents the top view of the floor plan. At the outset, one robot remained stationary while the other used a seed-spreader exploration procedure [**104**] across the floor, taking image samples at 40cm intervals. When the robot had completed the first room, it moved beyond the door and the stationary robot followed it to the threshold, where it remained stationary while tracking the exploratory robot as it continued its exploration of the second room.

4.1. Experiment 1: a) Odometry versus tracking: The trajectory of the exploratory robot was defined at the outset by a user. However, as the robot explored, accumulated error in odometry resulted in the robot straying from the desired path. The tracking estimate of the stationary robot was provided to the moving robot in order to correct this accumulated error. During the exploration the pose of the robot was corrected based on the observations of the robot tracker. During the experiment the pure odometry estimates were kept for comparisons. Figure 8.7 plots the uncorrected odometric trajectory (plotted as 'x') and the actual trajectory of the robot, as measured by the tracker (plotted

FIGURE 8.6. Views of the two "rooms" as seen by the robot, and the floor plan of the two "rooms".

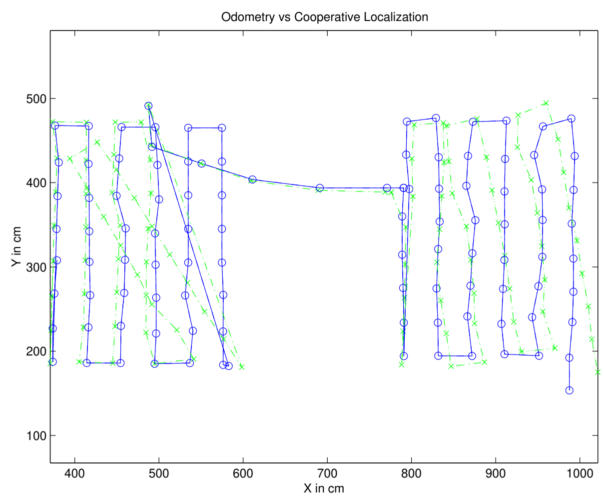


FIGURE 8.7. Odometric (x) vs Tracker-corrected (o) trajectories of the robot.

as 'o'). For the sake of clarity, the odometric error was reset to zero between the first and second rooms. Figure 8.8 presents the accumulated odometric error in the second room versus total distance traveled (after it was reset to zero)

b) Tracking versus vision-based pose estimation: Once image samples were obtained using the tracker estimates as ground truth position estimates, it was possible to apply our landmark learning framework to the image samples in order to learn a mapping between appearance-based landmarks and the pose of the robot. Figure 8.9 shows the discrepancies between the pose estimates from the tracker (marked as circles) and the landmark-based vision pose estimator (marked as x's) in Room 2. At each position, the two 2-D projections

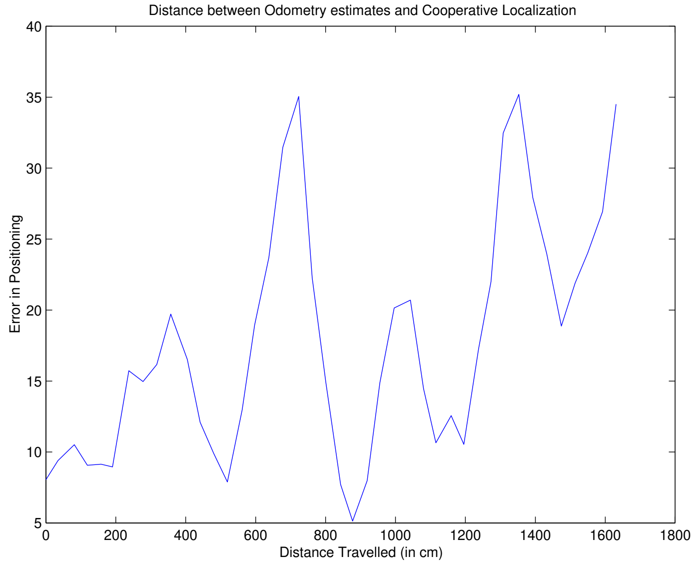


FIGURE 8.8. Odometric error versus distance traveled.

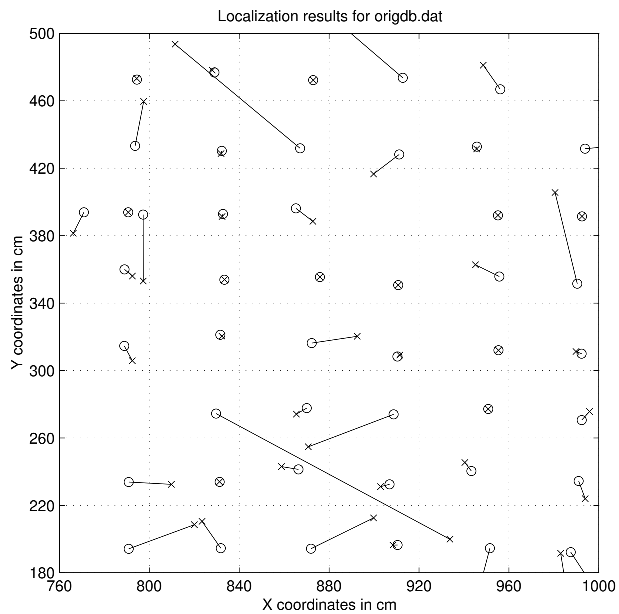


FIGURE 8.9. Tracker estimates (o) vs Vision-based estimates (x) for training images.

of the alternative pose estimates are joined by a line. While the tracker is clearly more accurate, the quality of the landmark-based pose estimates is sufficient for situations where only one robot is present.

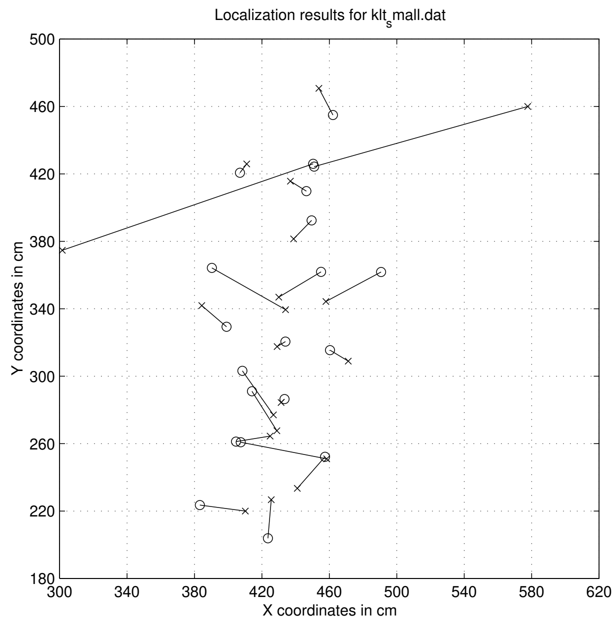


FIGURE 8.10. Tracker estimates (o) vs Image-based estimates (x) for a set of 21 random positions.

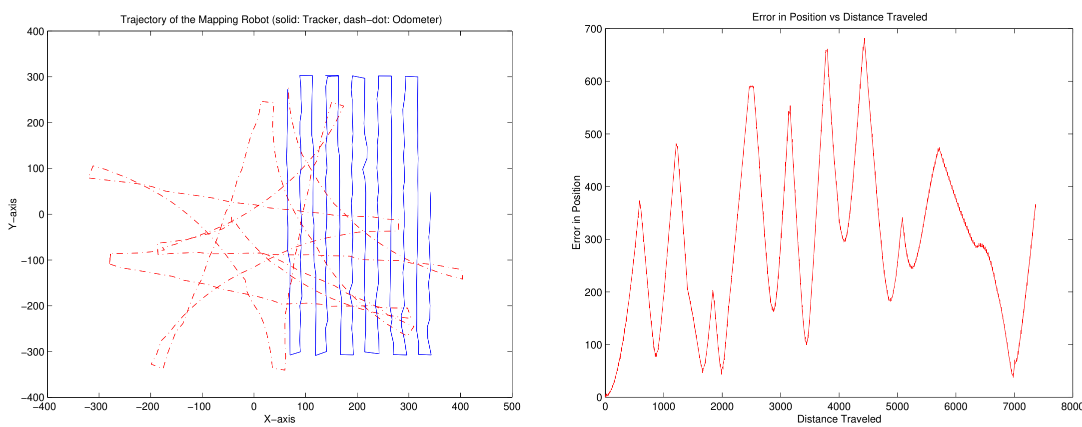
Our final stage of this experiment involved navigating the robot to a series of random positions and acquiring image- and tracker-based pose estimates, which are plotted together in Figure 8.10. This final experiment illustrates the use of a multi-sensor estimator in removing outliers. Assuming that the tracker-based position is correct, the mean error in the image-based estimate was 33cm, a large part of which can be attributed to the two significant outliers from nearly the same position.

4.2. Experiment 2. A second experiment was performed where the two robots explored a single large room. At the outset, one robot remained stationary while the other used a seed-spreader exploration procedure [**104**] across the floor, taking image samples at 25cm intervals, and in four orthogonal viewing directions, two of which are illustrated in Figure 8.11.

As before, the trajectory of the exploratory robot was defined at the outset by a user. However, as the robot explored, accumulated error in odometry resulted in the robot straying from the desired path. The differential drive configuration of the exploratory robot, coupled with frequent rotations to capture the four viewing directions led to a rapid, and somewhat systematic degradation in dead reckoning, as illustrated in Figure 8.12a, where

FIGURE 8.11. Opposing views of the lab as seen by the exploring robot.

the uncorrected odometric trajectory is plotted as a dash-dotted line, and the actual trajectory of the robot, as observed by the tracker, is plotted as a solid line. The accumulated odometric error is plotted versus total distance traveled in Figure 8.12b.



(a) (b)

FIGURE 8.12. In this experiment the robot took pictures in four orientation; the higher number of rotations increased non-linearly the odometric error. (a) Odometric (denoted by dash-dotted line) vs Tracker-corrected (denoted by a solid line) trajectories of the robot. (b) Odometric error versus distance traveled.

Once image samples were obtained using the tracker estimates as ground truth position estimates, it was possible to apply our landmark learning framework to the image samples in order to learn a mapping between appearance-based landmarks and the pose of the robot. Training was applied separately to each of the four viewing directions, developing a set of tracked landmark observations. Again the final stage of our experiment involved navigating the robot to a series of 93 random positions and acquiring images along the four orthogonal viewing directions. Image- and tracker-based maximum likelihood pose estimates were then generated for one of the viewing directions, and outliers removed on the basis of a likelihood threshold. Of the 93 observations, 4 estimates were rejected. In general, these outliers corresponded to observations where the robot was very close to the wall it was facing. One would expect that an observation from a different viewing direction would return an estimate with higher confidence. We have omitted this application for the sake of brevity.

The remaining 89 image-based estimates of high confidence are plotted with their associated tracker-based estimates in Figure 8.13. Assuming that the tracker-based position is correct, the mean error in the image-based estimate was 17cm, (7.7cm in the x direction vs 15cm in the y direction). The larger error in the y direction corresponds to the fact that the camera was pointed parallel to the positive y axis, and changes in observations due to forward motion are not as pronounced as changes due to side-to-side motion. The smallest absolute error was 0.49cm, which is insignificant compared to the "ground truth" error, and the largest error was 76cm. Note that most of the larger errors occur for large values of y. This is due to the fact that the camera was closest to the wall it was facing at these positions y, and as has been mentioned, tracking scene features over 25cm pose intervals became difficult.

4.2.1. *Random Walk.* Figure 8.14 presents a random walk of the exploring robot through the mapped environment. The robot starts at an random location (marked as a "*"), initially the odometry estimate is set to the value of the robot tracker estimate at that starting position, the pose estimate from the vision based system is approximately 30cm to the right of the robot tracker estimate. The robot took seven random steps and the three estimated trajectories are presented in Figure 8.14. First the odometer estimate (marked as triangles connected with a dashed line) is plotted; second, the robot tracker estimate (marked as "+" connected by a solid line), and third the visual pose estimator results (marked as "o" connected with a dash-dotted line). The robot tracker estimate provides a close approximation to ground truth at the end of the random walk the disparity between the robot tracker and the visual pose estimator is 17.5cm and between the robot

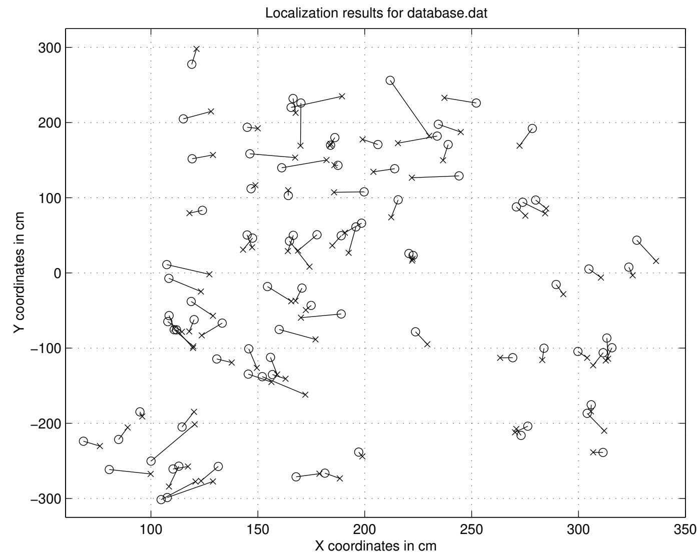


FIGURE 8.13. Tracker estimates (o) vs Image-based estimates (x) for a set of 93 random positions.

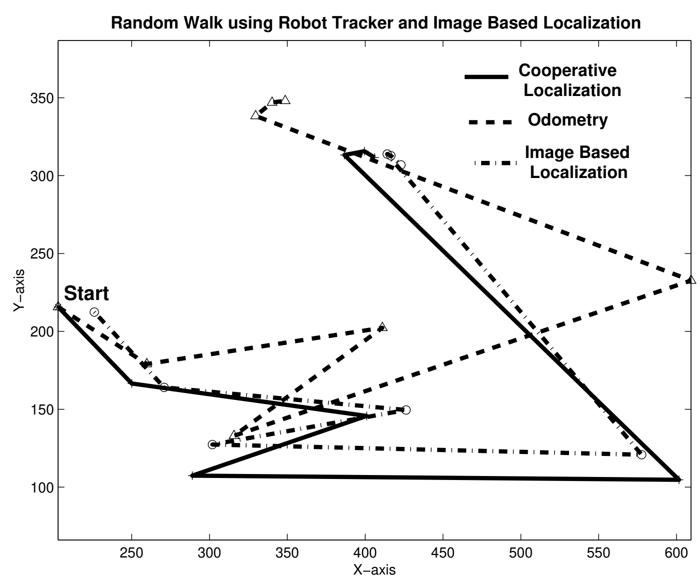


FIGURE 8.14. The trajectory of the moving robot based on odometry estimates (triangles connected with a dashed line), the robot tracker cooperative localization ('+' connected with a solid line) and the image based localization ('o' connected with a dash-dotted line).

tracker and the odometer is 68cm. The much higher disparity is a result of an increase in the accumulated error in orientation.

In this chapter we presented a method for the automatic mapping of an arbitrary environment which utilizes *cooperative localization* in order to maintain a *virtual tether* between two robots as one explores the environment and the other tracks its pose. Furthermore, we validated the utility of a set of learned landmarks for localization when the second robot cannot be deployed. This demonstrates conclusively that the cooperative localization approach provides more accurate pose estimates, and hence a more accurate appearance-based map, than could be achieved with the robots operating independently.

The particular map we produce, an appearance based representation of the environment, allows a single robot to accurately estimate its position on subsequent visits to the same area. While such single-robot pose estimates are not as accurate as when two robots are used, their accuracy is substantially ameliorated by the fact that two robots were used in the initial mapping stage. The use of an appearance-based model obviates most dependences on the particular geometry of the local environment. Further conclusions and extensions to the work presented in this and the previous chapters are discussed in the next chapter.

CHAPTER 9

Conclusions

> ... Ithaca has given you the beautiful voyage.
> Without her you would have never set out on the road.
> ... Wise as you have become, with so much experience,
> you must already have understood what Ithacas mean.
> –From the poem Ithaca by C. Cavafis 1911

In this thesis, we have described a new solution to the exploration and navigation problem based on the use of cooperating robots. We have described alternative approaches depending on the size of the environment relative to the range of the robot tracker sensor we employ.

Our approach is particularly suited in environments where robot positioning and obstacle detection might be difficult using traditional methods. In fact, such difficulties are likely to arise in many real-world environments. Our approach is based on exploiting a line-of-sight constraint between two (or more) robots to achieve exploration with reduced odometric error. This approach can also cope with obstacles with hard-to-sense reflectance characteristics. Different algorithms were proposed depending on the scale of the environment. Where the environment is small enough so that the robots can see each other from any two points on its boundary that have clear line of sight between them (i.e. they are never unable to see one another simply because they are too far away), then the triangulation algorithm is applied. If the environment is larger than the range of the robot tracker sensor then the trapezoidation algorithm is used [1]. Moreover, a robot tracker sensor was employed to monitor the moving robot and judiciously correct its pose when uncertainty became to big.

A probabilistic framework was developed in order to estimate the uncertainty build up during the exploration. Monte Carlo simulation in the form of particle filtering was used to model the complex odometry error behaviour and the impact of the robot tracker sensor. The robot tracker was used in an "as needed" basis when the uncertainty was above a certain bound.

The most important contribution of this thesis is dual. First the ability of the robots to see each other is used in a systematic manner to explore/map the free space in the environment independently from the reflectance properties of the obstacles. It is the first time when the ability of two (or more) robots to observe each other is exploited in order to infer about the occupancy of the space between them. Second, the use of *cooperative localization* decouples the odometry error from the environmental conditions such as the quality of the floor and the visibility of the objects (which in an unknown environment cannot be predicted).

A series of experiments were conducted in order to validate our approach. An accurate model for odometry error accumulation was developed and validated through experiments using different robots (the Nomad 200 and the Superscout II from Nomadic Technologies Inc, and the RWI B12 from the Real World Interface) navigating over different surfaces (tiles, carpet, concrete, etc.) Finally, a large number of experiments were conducted in simulation using different world models. During the simulation experiments different types of odometry error were used and the number of robots was varied from two to forty.

In the multi-robot paradigm we proposed a new methodology for estimating the bounds of the accumulated uncertainty based on the statistical properties of the robot tracker sensor used and the number of robots.

Finally the methodology of *cooperative localization* was applied in a different multi-robot exploration application. More specifically, the problem of accurately mapping a spatially varying property of an unknown (possibly hazardous) environment, and in particular creating a map of visual appearance (as defined by a set of visual landmarks), was addressed.

[1] An open issue is how to automatically detect such situations *efficiently* during exploration and switch strategies, or switch back-and-forth between strategies based on local properties of the environment.

Furthermore, we validate the utility of a set of learned landmarks for localization when the second robot cannot be deployed. This demonstrates conclusively that the cooperative localization approach provides more accurate pose estimates than single robot, and hence a more accurate appearance-based map, than could be achieved with the robots operating independently.

1. Future Work

The problem of autonomous exploration and mapping of unknown environments appears in numerous applications and covers a wide variety of environments. In this thesis, we presented two algorithms for the complete exploration of unknown environments by a team of mobile robots. We looked at large environments which were assumed to be flat, the mobile robots were wheeled indoor models, and the resulting map was of the form of a spatial decomposition of a simple polygon with holes. Extensions to our work are discussed below.

- During the exploration the mobile agents collect information from different sources. This information is corrupted by noise and the greedy algorithms proposed in this thesis build only an incremental map over time. The incremental nature of the map synthesis, like other EM algorithms, does not assure a probabilistically optimal path. After every motion/sensing operation the following data are available for the robot: the motion command $(\Delta X, \Delta Y)$ executed, the new position from the odometry estimates $\{x_{odom}, y_{odom}, \theta_{odom}\}^T$, the tracker estimate $< \rho, \theta, \phi >$ relative to the stationary robot, and the sensor data $\sum L_{x,y,\theta}$. [2] These data are stored and at the completion of the exploration they construct a constraint network for the environment and the path travelled by the mobile robots team. An off-line optimization algorithm can be applied in order to construct a map that is optimal over the above set of constraints [**101**].
- Both the triangulation and the trapezoidation algorithm assume that the robots operate inside a polygonal world. Although a polygonal approximation is always possible (see Chapter 2 Section 6) it will increase the computational complexity

[2] a set of line segments relative to the robots pose.

of the algorithm. An extension of the triangulation algorithm for curved environments appears feasible from preliminary research. Future work should include the development of wall following strategies for curved environments and the derivation of a completeness proof for the exploration algorithm.

- One of the main assumptions in the current work is that the mobile robots operate on a level environment and a 2D map of the world is enough. This assuption is a reasonable approximation for most indoors environments and for wheeled robots. Two major extensions are proposed as future areas of research.
  - The collaborative exploration philosophy could be extended to explore uneven terains such as outdoors or on another planet using a team of rovers or even legged robots. A polyhedral terrain approximation of such an environment (2.5D) can be used. The pose of the robot is extended to position in 3D $\{x, y, z\}$ and orientation of roll, pitch, yaw $\{\theta, \phi, \psi\}$. In this case a more elaborate tracking device is needed.
  - Furthermore, the extension to aerial mapping from flying robots or underwater exploration should be addressed, together with cooperating teams of grounded and flying vehicles.
- In certain environments the odometry estimates are so poor that they are useless. In such situations cooperative localization could be used in order to update the pose estimate of a mobile robot after an action. For example legged robots moving over debris are unable to maintain any sort of reliable dead reckoning.

  Preliminary experiments performed in an open area where the robots were manually moved to different position showed that cooperative localization could provide a basic localization scheme in existing systems.
- The motion strategies proposed in our work are deterministic. Preliminary experiments showed that a randomized motion strategy can sometimes outperform a deterministic one. While this bears further examination it seems likely that for teams of more than two or three robots randomized formation control may provide an appealing alternative to deterministic methods.

- In Chapter 8 we presented a method for the automatic mapping of an arbitrary environment which utilizes *cooperative localization* in order to maintain a *virtual tether* between two robots as one explores the environment and the other tracks its pose. The experimental results collected demonstrated conclusively that the virtual tether provides more accurate pose estimates, and hence a more accurate appearance-based map, than could be achieved with the robots operating independently. It would appear that these advantages become even more profound if more than two robots are used for position estimation and mapping. In the particular algorithmic scheme presented the use of many more robots would be an issue, but it seems likely that several feasible solutions can be formulated.
- In this thesis we employed a particle filter in order estimate the pose of a pair of robots during collaborative exploration. Further extensions of the particle filter approach could include more elaborate particle resampling methods that dynamically trade off efficiency for potential robustness. By estimating the parameters of the particle cloud, it seems possible to vary the model complexity on an as-needed basis. Further work involves the introduction of an additional weighting function based on other sensory input, such as the sensor used for wall following.

2. Final Words

The main problem addressed in this thesis (Simultaneous Localization and Mapping) is central to the field of Mobile Robotics. Our approach takes advantage of a multi-robot system in order to robustly explore an unknown environment. While the research of multi-robot systems is still at its early stages, many researchers are applying collaborative methods in order to overcome the limitations of single robot systems. The results produced by this thesis fit in this ongoing effort.

REFERENCES

[1] Ercan U. Acar and Howie Choset, *Complete sensor-based coverage with extended-range detectors: A hierarchical decomposition in terms of critical points and voronoi diagrams*, Proc. of IEEE IROS'01, International Conference on Intelligent Robots and Systems, (Maui, Hawaii, USA), 2001, pp. 1305–1311.

[2] ———, *Exploiting critical points to reduce positioning error for sensor-based navigation*, Proc. 2002 IEEE International Conference on Robotics & Automation (Washington DC,), 2002.

[3] Kurz Andreas, *Constructing maps for mobile robot navigation based on ultrasonic range data*, IEEE Transactions on Systems, Man, & Cybernetics, Part B: Cybernetics. **26** (1996), no. 2, 233–242.

[4] R. C. Arkin and T. Balch, *Motor schema-based formation control for multiagent robot teams*, International Conference on Multiagent Systems (San Francisco, CA), 1995, pp. 10–16.

[5] Ivan A. Bachelder and Allen M. Waxman, *A view-based neurocomputational system for relational map-making and navigation in visual environments*, Robotics and Autonomous Systems **16** (1995), 267–289.

[6] Leo Bagrow, *History of cartography*, Cambridge Harvard University Press, Cambridge, 1966.

[7] Gavin Baker and Andrew Howard, *An exploration algorithm for autonomous mobile robots"*, Australian Conference on Robotics and Automation, 2000, pp. 7–12.

[8] Tucker Balch and Ronald C. Arkin, *Communication in reactive multiagent robotic systems*, Autonomous Robots **1** (1994), no. 1, 27–52.

[9] Tucker Balch and Lynne E. Parker, *Guest editorial*, Autonomous Robotics,special issue on Heterogeneous Multi-Robot Systems **8** (2000), no. 3, 207–208.

[10] R. Bauer and W D. Rencken, *Sonar feature based exploration*, IEEE International Conference on Intelligent Robots and Systems., vol. 1, IEEE, 1995, pp. 148–153.

[11] Andrew T. D. Bennett, *Do animals have cognitive maps?*, Journal of Experimental Biology **199** (1996), 219–24.

[12] M. Betke and L. Gurvits, *Mobile robot localization using landmarks*, IEEE Trans. on Robotics and Automation **13** (1997), no. 2, 251–263.

[13] P. Bison, G. Chemello, C. Sossai, and G. Trainito, *Cooperative localization using possibilistic sensor fusion*, Proc. of 3rd IFAC Symposium on Intelligent Autonomous Vehicles (Madrid), Madrid, pp. 621–626.

[14] Michael Black and David Fleet, *Probabilistic detection and tracking of motion boundaries*, International Journal of Computer Vision **38** (2000), no. 3, 231–245.

[15] J. Borenstein, *The clapper: a dual-drive mobile robot with internal correction of dead-reckoning errors.*, Proceedings of the 1994 IEEE International Conference on Robotics and Automation (San Diego, CA,), May 8-13 1994, pp. 3085–3090.

[16] J. Borenstein, H. R. Everett, and L. Feng, *Navigating mobile robots: Systems and techniques*, no. ISBN 1-56881-058-X, A K Peters, Wellesley, MA, 1996.

[17] J. Borenstein, H R. Everett, L. Feng, and D. Wehe, *Mobile robot positioning: sensors and techniques*, Journal of Robotic Systems **14** (1997), no. 4, 231–249.

[18] Johann. Borenstein and Liqiang Feng, *Measurement and correction of systematic odometry errors in mobile robots*, IEEE Transactions on Robotics & Automation **12** (1996), no. 6, 869–880.

[19] S. M. Bozic, *Digital and kalman filtering*, second ed., Edward Arnold, 1994.

[20] R.A Brooks, *Visual map making for a mobile robot*, IEEE Trans. Robotics and Automation (1985), 824–829.

[21] ———, *A robust layered control system for a mobile robot*, IEEE Journal of Robotics and Automation **RA-2** (1986), no. 1, 14–23.

[22] Lloyd Arnold Brown, *The story of maps*, Bonanza Books, 1949.

[23] H. Bulata and M.Devy, *Incremental construction of a landmark-based and topological model of indoor environments by a mobile robot*, Proc. IEEE Conference on Robotics and Automation (April 1996), 1054–1060.

[24] Wolfram Burgard, Dieter Fox, Mark Moors, Reid Simmons, and Sebastian Thrun, *Collaborative multi-robot exploration*, Proceedings of the IEEE International Conference on Robotics and Automation (San Francisco, CA), IEEE Press, May 2000, pp. 476 –481.

[25] Zack Butler, Alfred Rizzi, and Ralph Hollis, *Cooperative coverage of rectilinear environments*, IEEE International Conference on Robotics and Automation 2000, vol. 3, April 2000, pp. 2722 – 2727.

[26] Y. U. Cao, A. S. Fukunaga, A. B. Kahng, and F. Meng, *Cooperative mobile robotics: Antecedents and directions*, Proc. IEEE/RSJ IROS (Pittsburgh, PA), vol. 1, 1995, pp. 226–234.

[27] J. Carpenter, P. Clifford, and P. Fearnhead, *An improved particle filter for non-linear problems*, IEE proceedings - Radar, Sonar and Navigation **146** (1999), 2–7.

[28] Cristiano Castelfranchi, *Modelling social action for a.i. agents*, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI), vol. 2, 1997, pp. 1567–1576.

[29] Kok Seng Chong and L. Kleeman, *Accurate odometry and error modelling for a mobile robot*, Proceedings of the 1997 IEEE International Conference on Robotics and Automation, vol. 4, 1997, pp. 2783 –2788.

[30] H. Choset and K. Nagatani, *Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization*, IEEE Transactions on Robotics and Automation **17** (2001), no. 2, 125–137.

[31] Howie Choset, *Incremental construction of the generalized voronoi diagram, the generalized voronoi graph and the hierarchical generalized voronoi graph*, First CGC Workshop on Computational Geometry (Baltimore, MD), October 1997.

[32] ———, *Coverage of known spaces: The boustrophedon cellupdar decomposition*, Autonomos Robots **9** (2000), 247–253.

[33] Howie Choset, Keiji Nagatani, and Alfred Rizzi, *Sensor based planning: Using a honing strategy and local map method to implement the generalized voronoi graph*, SPIE Mobile Robotics (Pittsburgh, PA), 1997.

[34] William W. Cohen, *Adaptive mapping and navigation by teams of simple robots*, Robotics & Autonomous Systems **18** (1996), no. 4, 411–434.

[35] A. Curran and K J. Kyriakopoulos, *Sensor-based self-localization for wheeled mobile robots*, Journal of Robotic Systems **12** (1995), no. 3, 163–176.

[36] F. Dellaert and A. Stroupe, *Linear 2d localization and mapping for single and multiple robots*, Proceedings of the IEEE International Conference on Robotics and Automation (IEEE, ed.), May 2002.

[37] Frank Dellaert, Wolfram Burgard, Dieter Fox, and Sebastian Thrun, *Using the condensation algorithm for robust, vision-based mobile robot localization*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Press, June 1999.

[38] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun, *Monte carlo localization for mobile robots*, IEEE International Conference on Robotics and Automation (ICRA99), May 1999.

[39] Frank Dellaert and Ashley Stroupe, *Linear 2d localization and mapping for single and multiple robots*, Proceedings of the IEEE International Conference on Robotics and Automation, 2002, IEEE, May 2002.

[40] Xiaotie Deng and Andranik Mirzaian, *Competitive robot mapping with homogeneous markers*, IEEE Trans. on Robotics and Automation **12** (1996), no. 4, 532–542.

[41] G. Dissanayake, S. Clark P. Newman, H. Durrant-Whyte, and M. Csorba, *A solution to the simultaneous localization and map building (slam) problem*, IEEE Transactions on Robotics and Automation **17** (2001), no. 3, 229–241.

[42] Bruce Randall Donald, James Jennings, and Daniela Rus, *Analyzing teams of cooperating mobile robots*, Proceedings - IEEE International Conference on Robotics and Automation, vol. 3, IEEE, 1994, pp. 130–135.

[43] Arnaud Doucet, Nando De Freitas, and Neil Gordon, *Sequential monte carlo methods in practice*, Springer-Verlag, Series Statistics for Engineering and Information Science, January 2001.

[44] Gregory Dudek, Paul Freedman, and Souad Hadjres, *Using local information in a non-local way for mapping graph-like worlds*, Proceedings of the Thirteenth International Conference on Artificial Intelligence (Chambery, France), Internation Joint Conf. on Artificial Intelligence Inc., August 1993, pp. 1639–1645.

[45] ———, *Mapping in unknown graph-like worlds*, Journal of Robotic Systems **13** (1996), no. 8, 539–559.

[46] Gregory Dudek, Paul Freedman, and Ioannis M. Rekleitis, *Just-in-time sensing: efficiently combining sonar and laser range data for exploring unknown worlds*, International Conference in Robotics and Automation, vol. 1, IEEE, April 1996, pp. 667–671.

[47] Gregory Dudek and Michael Jenkin, *Computational principles of mobile robotics*, no. ISBN: 0521568765, Cambridge University Press, May 2000.

[48] Gregory Dudek, Michael Jenkin, Evangelos Milios, and David Wilkes, *Using multiple markers in graph exploration*, Proc. Symposium on Advances in Intelligent Robotics Systems: Conference on Mobile Robotics (Philadelphia, PA), International Society for Optical Engineering, 1989.

[49] ———, *Robotic exploration as graph construction*, Transactions on Robotics and Automation **7** (1991), no. 6, 859–865.

[50] ———, *Experiments in sensing and communication for robot convoy navigation*, Proceedings IEEE International Conference on Intelligent Robots and Systems (IROS) (Pittsburgh, PA), vol. 2, August 1995, pp. 268–273.

[51] ———, *A taxonomy for multi-agent robotics*, Autonomous Robots **3** (1996), 375–397.

[52] Gregory Dudek and Paul MacKenzie, *Model-based map construction for robot localization*, Proceedings of Vision Interface 1993 (Toronto, ON), May 1993.

[53] Gregory Dudek and Chi Zhang, *Vision-based robot localization without explicit object models*, Proc. International Conference of Robotics and Automation (Minneapolis, MN), IEEE Press, 1996.

[54] G. D. Dunlap and H. H. Shufeldt, *Dutton's navigation and piloting*, ch. pp. 557-579, Naval Institute Press, 1974.

[55] A. Elfes, *Sonar-based real-world mapping and navigation*, IEEE Journal of Robotics and Automation **3(3)** (1987), 249–265.

[56] David Eu and Godfried Toussaint, *On approximating polygonal curves in two and three dimensions*, CVGIP: Graphical Models and Image Processing **56** (1994), no. 3, 231–246.

[57] L. Feng, J. Borenstein, and H.R. Everett, *Where am i: Sensors and methods for mobile robot positioning,*, Tech. Report UM-MEAM-94-21, University of Michigan, Ann Arbor, University of Michigan, Ann Arbor, MI, USA, December 1994.

[58] J. W. Fenwick, P. M. Newman, and J. J. Leonard, *Cooperative concurrent mapping and localization*, IEEE International Conference on Robotics and Automation, vol. 2, May 2002, pp. 1810–1817.

[59] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun, *A probabilistic approach to collaborative multi-robot localization*, Autonomous Robots **8** (2000), no. 3, 325–344.

[60] T. Fujii, H. Asama, T. von Numers, T. Fujita, H. Kaetsu, and I. Endo, *Co-evolution of a multiple autonomous robot system and its working environment via intelligent local information storage*, Robotics & Autonomous Systems **19** (1996), no. 1, 1–13.

[61] Toshio Fukuda, Shigenori Ito, Fumihito Arai, Yasunari Yokoyama, Yasunori Abe, Kouetsu Tanaka, and Yoshio Tanaka, *Navigation system based on ceiling landmark recognition for autonomous mobile robot - landmark detection based on fuzzy template matching (ftm)*, IEEE International Conference on Intelligent Robots and Systems, vol. 2, 1995, pp. 150–155.

[62] C. R. Gallistel and Audrey E. Cramer, *Computations on metric maps in mammals: getting oriented and choosing a multi-destination route*, Journal of Experimental Biology **199** (1996), 211–17.

[63] Arthur Gelb, *Applied optimal estimation*, MIT Press, Cambridge, Massachusetts, 1974.

[64] L. Girod and D. Estrin, *Robust range estimation using acoustic and multimodal sensing*, IEEE/RSJ/ International Conference on Intelligent Robots and Systems (Maui, Hawaii, USA), vol. 3, IEEE/RSJ, October 2001.

[65] F. Giuffrida, C. Massucco, P. Morasso, G. Vercelli, and R. Zaccaria, *Multi-level navigation using active localization system*, IEEE International Conference on Intelligent Robots and Systems, vol. 1, IEEE, 1995, pp. 413–418.

[66] N.J. Gordon, D.J. Salmond, and A.F.M. Smith, *Novel approach to nonlinear/non-gaussian bayesian state estimation*, IEE Proceedings For Radar and Signal Processing **140** (1993), no. 2, 107 –113.

[67] Robert Grabowski and Pradeep Khosla, *Localization techniques for a team of small robots*, Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (Maui, Hawaii, USA), no. ISBN:0-7803-6614-X, Oct. 29 - Nov. 03 2001, pp. 1067–1072.

[68] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige, *An experimental comparison of localization methods*, Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 1998.

[69] J.-S. Gutmann and K. Konolige, *Incremental mapping of large cyclic environments*, International Symposium on Computational Intelligence in Robotics and Automation (CIRA'99), (Monterey, CA), November 1999.

[70] John Hancock and Chuck. Thorpe, *Elvis: Eigenvectors for land vehicle image system*, IEEE International Conference on Intelligent Robots and Systems., vol. 1, IEEE, 1995, pp. 35–40.

[71] K. R. Harinarayan and V. Lumelsky, *Sensor-based motion planning for multiple mobile robots in an uncertain environment*, IEEE/RSJ/GI International Conference on Intelligent Robots and Systems. (IEEE Service Center, Piscataway, NJ, USA), vol. 3, IEEE, 1994, pp. 1485–1492.

[72] Susan Hert and Vlad Lumelsky, *Moving multiple tethered robots between arbitrary configurations*, Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. (Pittsburgh, PA, U), vol. 2, 1995, pp. 280–285.

[73] Susan Hert and Vladimir Lumelsky, *The ties that bind: Motion planning for multiple tethered robots*, Robotics and Autonomous Systems **17** (1996), 187–215.

[74] A. Howard and L. Kitchen, *Cooperative localisation and mapping*, International Conference on Field and Service Robotics (FSR99), 1999, pp. 92–97.

[75] H. Imai and Masao Iri, *Polygonal approximations of a curve – formulations and algorithms*, Computational Morphology (G. T. Toussaint, ed.), Elsevier Science Publishers B. V., New York, N.Y., 1988, pp. 71–86.

[76] L. Iocchi, K. Konolige, and M. Bajracharya, *Visually realistic mapping of a planar environment with stereo*, In Proc. of Seventh International Symposium on Experimental Robotics (ISER'2000) (Hawaii), 2000.

[77] Michael Isard and Andrew Blake, *Condensation-conditional density propagation for visual tracking*, International Journal of Computer Vision **29** (1998), no. 1, 2–28.

[78] ———, *Icondensation: Unifying low-level and high-level tracking in a stochastic framework*, Proc 5th European Conf. Computer Vision, vol. 1, 1998, pp. 893–908.

[79] Patric Jensfelt, Olle Wijk, David J. Austin, and Magnus Andersso, *Experiments on augmenting condensation for mobile robot localization.*, IEEE International Conference on Robotics & Automation (ICRA) (San Francisco, CA, USA), April 2000, pp. 2518–2524.

[80] ———, *Feature based condensation for mobile robot localization*, IEEE International Conference on Robotics & Automation (ICRA) (San Francisco, CA, USA), April 2000, pp. 2531–2537.

[81] Rudolph Emil Kalman, *A new approach to linear filtering and prediction problems*, Transactions of the ASME–Journal of Basic Engineering **82** (1960), no. Series D, 35–45.

[82] K. Kimoto and Charles Thorpe, *Map building with radar and motion sensors for automated highway vehicle navigation*, Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '97), vol. 3, September 1997, pp. 1721 –1728.

[83] S. Koenig and R.G. Simmons, *Unsupervised learning of probabilistic models for robot navigation*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE, 1996, pp. 2301–2308.

[84] S. Koenig, C. Tovey, and W. Halliburton, *Greedy mapping of terrain*, IEEE International Conference on Robotics and Automation, vol. 4, 2001, pp. 3594 –3599.

[85] B. Kuipers and T. Levitt, *Navigation and mapping in large-scale space*, AI Magazine (1988), 25–43.

[86] Yasuo Kuniyoshi, Jukka Riekki, Makoto Ishii, Sebastien Rougeaux, Nobuyuki Kita, Shigeyuki Sakane, and Masayoshi Kakikura, *Vision-based behaviors for multi-robot cooperation*, IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, vol. 2, IEEE, 1994, pp. 925–932.

[87] R. Kurazume and S. Hirose, *Study on cooperative positioning system - optimum moving strategies for cps-iii*, Proc. IEEE Int. Conf. on Robotics and Automation (IEEE, ed.), vol. 4, 1998, pp. 2896–2903.

[88] Ryo Kurazume, Shigeo Hirose, Shigemi Nagata, and Naoki Sashida, *Study on cooperative positioning system*, International Conference in Robotics and Automation, vol. 2, IEEE, April 1996, pp. 1421–1426.

[89] Ryo Kurazume and Shigemi Nagata, *Cooperative positioning with multiple robots*, International Conference in Robotics and Automation, vol. 2, IEEE, 1994, pp. 1250–1257.

[90] S. Lang and A. Wong, *Building geometric world models with graph synthesis for sensor fusion in mobile robots*, Computational Intelligence **6** (1990), 91–107.

[91] Latimer, Srinivasa, V. Lee-Shue, S. S. Sonne, H. Choset, and Hurst, *Toward sensor based coverage with robot teams*, Proc. 2002 IEEE International Conference on Robotics & Automation, (Washington DC) (IEEE, ed.), May 2002.

[92] Jean-Claude Latombe, *Robot motion planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[93] J. J. Leonard, R. N. Carpenter, and H. J. S. Feder, *Stochastic mapping using forward look sonar.*, Robotica **19** (2001), 341.

[94] J. J. Leonard and H. J. S. Feder, *Decoupled stochastic mapping*, IEEE Journal of Oceanic Engineering (2001), 561–571.

[95] John J. Leonard and Hugh F. Durrant-Whyte, *Mobile robot localization by tracking geometric beacons*, IEEE Transactions on Robotics and Automation **7** (1991), no. 3, 376–382.

[96] John J. Leonard and Hans Jacob S. Feder, *A computationally efficient method for large-scale concurrent mapping and localization*, Robotics Research: The Ninth International Symposium (London) (John Hollerbach and Dan Koditschek, eds.), Springer-Verlag, 2000, pp. 169–176.

[97] C. Lin and R. Tummala, *Mobile robot navigation using artificial landmarks*, Journal of Robotic Systems **14** (1997), no. 2, 93–106.

[98] Jun S. Liu, Rong Chen, and Tanya Logvinenko, *A theoretical framework for sequential importance sampling and resampling*, Sequential Monte Carlo in Practice (A. Doucet, N. de Freitas, and N.J. Gordon, eds.), Springer-Verlag, January 2001.

[99] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun, *Using em to learn 3d models with mobile robots*, In Proceedings of the International Conference on Machine Learning (ICML),, 2001.

[100] F. Lu and E. Milios, *Globally consistent range scan alignment for environment mapping*, Autonomous Robots **4** (1997), 333–349.

[101] Feng Lu and E. Milios, *Robot pose estimation in unknown environments by matching 2d range scans*, Journal of Intelligent and Robotic Systems (1998), 249–275.

[102] Feng Lu and Evangelos Milios, *Robot pose estimation in unknown environments by matching 2d range scans*, Tech. Report RBCV-TR-94-46, University of Toronto, December 1994.

[103] ______, *Optimal global pose estimation for consistent sensor data registration*, International Conference in Robotics and Automation, vol. 1, IEEE, 1995, pp. 93–100.

[104] V. Lumelsky, S. Mukhopadhyay, and Kang Sun, *Sensor-based terrain acquisition: The 'sightseer' strategy.*, Proceedings of the IEEE Conference on Decision and Control Including The Symposium on Adaptive Processes (IEEE Service Center, Piscataway, NJ, USA), vol. 2, IEEE, IEEE, 1989, pp. 1157–1161.

[105] Vladimer Lumelsky, Snehasis Mukopadhyay, and Kang Sun, *Dynamic path planning in sensor-based terrain acquisition*, IEEE Trans. on Robotics and Automation **6** (1990), no. 4, 462–472.

[106] John MacCormick and Andrew Blake, *A probabilistic exclusion principle for tracking multiple objects*, Proc. Int. Conf. Computer Vision, 1999, pp. 572–578.

[107] Paul MacKenzie and Gregory Dudek, *Precise positioning using model-based maps*, Proceedings of the International Conference on Robotics and Automation (San Diego, CA), IEEE Press, 1994.

[108] M. Di Marco, A. Garulli, and A. Vicino, *Cooperative localization and map building for multi-robot systems: a set-membership approach*, Proc. of the 9th Int. Symposium on Intelligent Robotic Systems (Toulouse (France)), July 18-20 2001, pp. 415–424.

[109] D. Marr, *Vision*, Freeman, New York, 1982.

[110] M. Mataric, *Minimizing complexity in controlling a mobile robot population*, Proc. IEEE Int. Conf. on Robotics and Automation, 1992, pp. 830–835.

[111] ———, *Minimizing complexity in controlling a mobile robot population*, Proceedings of the 1992 IEEE International Conference on Robotics and Automation (Nice, France), 1992, pp. 830 – 835.

[112] Maja J. Mataric, Martin Nilsson, and Kristian T. Simsarian, *Cooperative multi-robot box-pushing*, IEEE International Conference on Intelligent Robots and Systems., vol. 3, IEEE, 1995, pp. 556–561.

[113] M.J Mataric, *Environmental learning using a distributed representation*, IEEE International Conference on Robotics and Automation **1** (1990), 402–406.

[114] P. Maybeck, *Stochastic models, estimation ond control*, vol. 1, Academic, New York, 1979.

[115] Bruce McNaughton, *Cognitive cartography*, Nature **381** (1996), 368–369.

[116] Jong-Woo Moon, Chong-Kug Park, and Fumio Harashima, *Kinematic correction of a differential drive mobile robot and a design for velocity trajectory with acceleration constraints on motor controllers*, IEEE/RSJ International Conference on Intelligent Robots and Systems (Piscataway, NJ, USA), vol. 2, 1999, pp. 930–935.

[117] F. Nashashibi and M. Devy, *Combining terrain maps and polyhedral models for robot navigation*, 1993 International Conference on Intelligent Robots and Systems., July 1993, pp. 685–691.

[118] John O'Keefe, *Brain and space*, ch. The hippocampal cognitive map and navigational strategies, pp. 273–295, Oxford University Press, 1991.

[119] ———, *Spatial representation: Problems in philosophy and psychology*, ch. Kant and the sea-horse: An essay in the neurophilosophy of space, pp. 43–64, Blackwell Publishers, Inc, 1993.

[120] John O'Keefe and Nadel L., *The hippocampus as a cognitive map.*, Clarendon Press., Oxford, 1978.

[121] B. Oommen, S. Iyegar, S. Nageswara, S. Rao, and R. Kashyap, *Robot navigation in unknown terrains using learned visibility graphs, part i: The disjoint convex obstacle case*, IEEE J. of Robotics and Automation **3** (1987), no. 6, 672–681.

[122] J. O'Rourke, *Computational geometry in C*, Cambridge University Press, 1994, ISBN 0-521-44592-2/Pb.

[123] Lynne E. Parker, Kingsley Fregene, Yi Guo, and Raj Madhavan, *Multi-robot systems: From swarms to intelligent automata*, ch. Distributed Heterogeneous Sensing for Outdoor Multi-Robot Localization, Mapping, and Path Planning, Kluwer, 2002.

[124] B.L. Partridge, *The structure and function of fish schools*, Scientific American (1982), 114–123.

[125] David Pierce and Benjamin. Kuipers, *Learning to explore and build maps*, Proceedings of the National Conference on Artificial Intelligence. (Menlo Park, CA, USA), vol. 2, AAAI, 1994, pp. 1264–1271.

[126] Jonathan Potter, *Country life book of antique maps*, Country Life Boo, 1989.

[127] F. P. Preparata and M. I. Shamos, *Computational geometry: An introduction*, Springer-Verlag, New York, NY, 1985.

[128] Claudius Ptolemy, *Geographia*, Lipsiae, 1845.

[129] ______, *Almagest*, Chicago : Encyclopedia Britannica, 1952.

[130] N. S. V. Rao, V. Protopopescu, and N. Manickam, *Cooperative terrain model acquisition by a team of two or three point-robots*, International Conference in Robotics and Automation, vol. 2, IEEE, April 1996, pp. 1427–1433.

[131] Nageswara S. V. Rao, *Robot navigation in unknown generalized polygonal terrains using vision sensors*, IEEE Transactions on Systems, Man, and Cybernetics **25** (1995), no. 6, 947–962.

[132] Ioannis Rekleitis, Gregory Dudek, and Evangelos Milios, *Multi-robot exploration of an unknown environment, efficiently reducing the odometry error*, International Joint Conference on Artificial Intelligence (Nagoya, Japan) (IJCAI, ed.), vol. 2, Morgan Kaufmann Publishers, Inc., August 1997, http://www.cim.mcgill.ca/˜yiannis/Publications/IJCAI97.pdf, pp. 1340–1345.

[133] Ioannis Rekleitis, Robert Sim, Gregory Dudek, and Evangelos Milios, *Collaborative exploration for map construction*, 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Banff, Alberta, Canada), no.

ISBN 0-7803-7203-4, IEEE, IEEE, July 2001, http://www.cim.mcgill.ca/˜yiannis/Publications/cira01.pdf, pp. 296–301.

[134] ______, *Collaborative exploration for the construction of visual maps*, IEEE/RSJ/ International Conference on Intelligent Robots and Systems (Maui, Hawaii, USA), vol. 3, IEEE/RSJ, IEEE,ISBN 0-7803-6614-X, October 2001, http://www.cim.mcgill.ca/˜yiannis/Publications/iros01.pdf, pp. 1269–1274.

[135] Ioannis M. Rekleitis, Gregory Dudek, and Evangelos Milios, *Graph-based exploration using multiple robots*, 5th International Symposium on Distributed Autonomous Robotic Systems (DARS) (Knoxville, Tennessee, USA), Springer, October 4-6 2000, http://www.cim.mcgill.ca/˜yiannis/Publications/darsU131.pdf, pp. 241–250.

[136] ______, *Multi-robot collaboration for robust exploration*, Proceedings of International Conference in Robotics and Automation (San Francisco, USA), April 2000, http://www.cim.mcgill.ca/˜yiannis/Publications/icra00.pdf, pp. 3164–3169.

[137] ______, *Multi-robot collaboration for robust exploration*, Annals of Mathematics and Artificial Intelligence **31** (2001), no. 1-4, 7–40, http://www.cim.mcgill.ca/˜yiannis/Publications/journal.pdf.

[138] ______, *Multi-robot cooperative localization: A study of trade-offs between efficiency and accuracy*, IEEE/RSJ/ International Conference on Intelligent Robots and Systems (Lausanne, Switzerland), IEEE/RSJ, October 2002, http://www.cim.mcgill.ca/˜yiannis/Publications/iros02pdf.pdf.

[139] Ioannis M. Rekleitis, Gregory Dudek, and Evangelos E. Milios, *Accurate mapping of an unknown world and online landmark positioning*, Proceedings of Vision Interface 1998 (Vancouver, Canada), June 1998, http://www.cim.mcgill.ca/˜yiannis/Publications/VI98.pdf, pp. 455–461.

[140] Ioannis M. Rekleitis, Gregory Dudek, and Evangelos E. Milios, *On the positional uncertainty of multi-robot cooperative localization*, Multi-Robot Systems Workshop, Naval Research Laboratory, Washington, DC, USA, March 18-20 2002, http://www.cim.mcgill.ca/
~yiannis/Publications/mrworkshop.pdf.

[141] Ioannis M. Rekleitis, Vida Dujmović, and Gregory Dudek, *Efficient topological exploration*, Proceedings of International Conference in Robotics and Automation (Detroit, USA), May 1999, http://www.cim.mcgill.ca/
~yiannis/Publications/icra99.pdf, pp. 676–681.

[142] Stergios I. Roumeliotis and George A. Bekey, *Bayesian estimation and kalman filtering: A unified framework for mobile robot localization*, Proc. 2000 IEEE International Conference on Robotics and Automation (San Francisco, California), IEEE, April 22-28 2000, pp. 2985–2992.

[143] ______, *Collective localization: A distributed kalman filter approach to localization of groups of mobile robots*, Proc. 2000 IEEE International Conference on Robotics and Automation (San Francisco, California), IEEE, April 22-28 2000, pp. 2958–2965.

[144] ______, *Distributed multi-robot localization*, 5th International Symposium on Distributed Autonomous Robotic Systems (DARS) (Knoxville, Tennessee, USA), Springer, October 4-6 2000, pp. 179–188.

[145] N. Roy and S. Thrun, *Online self-calibration for mobile robots*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 1999.

[146] Bernhard Scholkopf and Hanspeter A. Mallot, *View-based cognitive mapping and path planning.*, Adaptive Behavior **3** (1995), no. 3, 311–348.

[147] M. Seiz, P. Jensfelt, and H. I. Christensen, *Active exploration for feature based global localization*, Proceedings IEEE International Conference on Intelligent Robots and Systems (IROS) (Takamatshu), October 2000.

[148] Hagit Shatkay and Leslie Pack Kaelbling, *Learning topological maps with weak local odometric information*, Proceedings of the 15th International Joint Conference on

Artificial Intelligence (IJCAI-97) (San Francisco), Morgan Kaufmann Publishers, August 23–29 1997, pp. 920–929.

[149] R. Sim and G. Dudek, *Learning and evaluating visual features for pose estimation*, Proceedings of the Seventh IEEE International Conference on Computer Vision(ICCV) (Kerkyra, Greece), IEEE Press, sept 1999.

[150] ———, *Learning environmental features for pose estimation*, Proceedings of the 2nd IEEE Workshop on Perception for Mobile Agents (Ft. Collins, CO), IEEE Press, June 1999.

[151] ———, *Learning visual landmarks for pose estimation*, Proceedings of the IEEE International Conference on Robotics and Automation(ICRA) (Detroit, MI), IEEE Press, May 1999.

[152] ———, *Visual landmarks for pose estimation*, Canadian Artificial Intelligence (1999), no. 43, pp13–17.

[153] ———, *Learning landmarks for robot localization*, SIGART/AAAI 2000 Doctoral Consortium (Austin, TX), SIGART/AAAI, AAAI Press, July 2000, p. 2.

[154] ———, *Learning environmental features for pose estimation*, Image and Vision Computing, Elsevier Press **19** (2001), no. 11, 733–739.

[155] ———, *Learning generative models of scene features*, IEEE Conference on Computer Vision and Pattern Recognition (Hawaii), IEEE Press, December 2001.

[156] Robert Sim, *Mobile robot localization using learned landmarks*, Master's thesis, McGill University, 1998.

[157] Robert Sim and Gregory Dudek, *Mobile robot localization from learned landmarks*, Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS) (Victoria, Canada), vol. 2, October 1998, pp. 1060–1065.

[158] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, *Coordination for multi-robot exploration and mapping*, Proceedings National Conference on Artificial Intelligence (Austin TX), 2000.

[159] R. Smith and P. Cheeseman, *On the representation and estimation of spatial uncertainty*, International Journal of Robotics Research **5(4)** (1986), 56–68.

[160] R. Smith, M. Self, and P. Cheeseman, *Estimating uncertain spatial relationships in robotics*, Autonomous Robot Vehicles (I.J. Cox and G. T. Wilfong, eds.), Springer-Verlag, 1990, pp. 167–193.

[161] J. Spletzer, A.K. Das, R. Fierro, C.J. Taylor, V. Kumar, and J.P. Ostrowski, *Cooperative localization and control for multi-robot manipulation*, IEEE/RSJ/ International Conference on Intelligent Robots and Systems (Maui, Hawaii, USA), vol. 3, IEEE/RSJ, October 2001.

[162] Kazuo Sugihara and Ichiro Suzuki, *Distributed algorithms for formation of geometric patterns with many mobile robots*, Journal of Robotics Systems **13** (1996), no. 3, 127–139.

[163] J. Sullivan, A. Blake, M. Isard, and J. MacCormick, *Object localization by Bayesian correlation*, Proc. Int. Conf. Computer Vision, 1999, pp. 1068–1075.

[164] Scott Thayer, *Four generations of robotic mapping and exploration in extreme environments*, International Symposium on Artificial Intelligence, Robotics and Human Centered Technology for Nuclear Applications (AIR'02), January 2002, pp. 85–92.

[165] S. Thrun, W. Burgard, and D. Fox, *A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping*, In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (San Francisco, CA) (IEEE, ed.), 2000.

[166] Sebastian Thrun, *Learning metric-topological maps for indoor mobile robot navigation*, AI Journal **99:1** (1998), 21–71.

[167] Sebastian Thrun, Dieter Fox, and Wolfram Burgard., *A probabilistic approach to concurrent mapping and localization for mobile robots*, Machine Learning and Autonomous Robots **31,5** (1998), 29–53,253–271.

[168] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert, *Robust monte carlo localization for mobile robots*, Artificial Intelligence Journal **101** (2001), 99–141.

[169] Sebastian Thrun, Dieter Fox, and Wolfram Burghard, *A probabilistic approach to concurrent mapping and localization for mobile robots*, Autonomous Robots **5** (1998), 253–271.

[170] Niko Tinbergen, *The animal in its world; explorations of an ethologist, 1932-1972*, Harvard University Press, 1972.

[171] Unknown, *NATO MAS Reference list*, November 21 2001.

[172] R. T. Vaughan, G. S. Sukhatme, F. J. Mesa-Martinez, and J. F. Montgomery, *Fly spy: lightweight localization and target tracking for cooperating air and ground robots*, 5th International Symposium on Distributed Autonomous Robotic Systems (DARS) (Knoxville, Tennessee, USA), Springer, October 4-6 2000, pp. 315–324.

[173] Nikos Vlasis, Bas Terwijn, and Ben Krose, *Auxiliary particle filter robot localization from high-dimensional sensor observations*, IEEE International Conference in Robotics and Automation (IEEE, ed.), vol. 1, May 2002, pp. 7–12.

[174] Thomas von Numers, Hajime Asama, Takanoru Fujita, Shin'ya Kotosaka, Sakae Muyao, Hayato Kaetsu, and Isao Endo, *An intelligent data carrier system for local communication between cooperative multiple mobile robots and environment.*, 2nd IFAC Conf. on Inteligent Autonomous Vehicles 95, 1995, pp. 366–370.

[175] Ashely Walker, Hihn Hallman, and David Willshaw, *Bee-havior in a mobile robot: The construction of a self-organized cognitive map and its use in robot navigation within a complex, natural environment*, IEEE Conf. on Neural Networks, 1993, pp. 1451–1456.

[176] Chieh-Chih Wang and Charles Thorpe, *Simultaneous localization and mapping with detection and tracking of moving objects*, IEEE International Conference on Robotics and Automation, May 2002.

[177] P. Weckesser, R. Dillmann, M. Elbs, S. Hampel, and U. Rembold, *Multiple sensor-processing for high-precision navigation and environmental modeling with a mobile robot*, IEEE International Conference on Intelligent Robots and Systems., vol. 1, IEEE, 1995, pp. 453–458.

[178] Gerhard Weiss, Christopher Wetzler, and Ewald. von Puttkamer, *Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans*, IEEE/RSJ/GI International Conference on Intelligent Robots and Systems., vol. 1, IEEE, 1994, pp. 595–601.

[179] S.B. Williams, P. Newman, G. Dissanayake, and H.F. Durrant-Whyte, *Autonomous underwater simultaneous localisation and map building*, IEEE International Conference on Robotics and Automation (San Francisco CA), 2000, pp. 1793–1798.

[180] W. Yeap, *Towards a computational theory of cognitive maps*, Artificial Intelligence **32** (1988), 297–360.

[181] Xiaoping Yun, Gokhan Alptekin, and Okay Albayrak, *Line and circle formation of distributed physical mobile robots*, Journal of Robotic Systems **14** (1997), no. 2, 63–76.

[182] Ludek Zalud, Luks Kopecn, and Toms Neuzil, *Laser proximity scanner correlation based method for cooperative localization and map building*, 7th International Workshop on Advanced Motion Control AMC'02 (Maribor, Slovenia), July 2002.

[183] G. Zunino and H.I. Christensen, *Simultaneous localization and mapping in domestic environments*, International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), 2001, pp. 67 –72.

APPENDIX A

Proof of Optimal Coverage

In this Appendix we present the optimality proof of the trapezoidation algorithm with respect to the distance traveled. Every time one robot is moving it covers a small area of free space, by alternating roles the two robots cover all of free space. Therefore, the total area covered is the sum of the areas covered in each move every single exchange minus the overlapping covered area. This is equivalent with a tiling problem where every small area covered by the motion of one robot represents a tile, and the objective is to cover the free space with tiles without leaving any space uncovered. We present an optimal tiling of the space, under certain assumptions.

Assumptions Every tile can be contained in a wedge of radius R and angle θ.

We consider only tilings defined by the line of sight, between the two robots, sweeping across the space. One end of the line is fixed (the position of the stationary robot) and the other follows the trajectory of the second robot. The length of the line of sight is bounded by the sensing range R. The sweep of the line and the bounding circle define a wedge inside which a tile is placed.

1. Proof

Lemma 1: A tiling in which no tile with an angle more than $180°$ is used can replace any arbitrary tiling, without increasing the complexity (number of tiles).

Proof: If a tile with an angle more than $180°$ is used then at least another tile with an angle less than $180°$ is necessary to cover the remaining free area. The combination of the two (or more) tiles is equivalent (in terms of complexity) to two (or more) tiles of angle

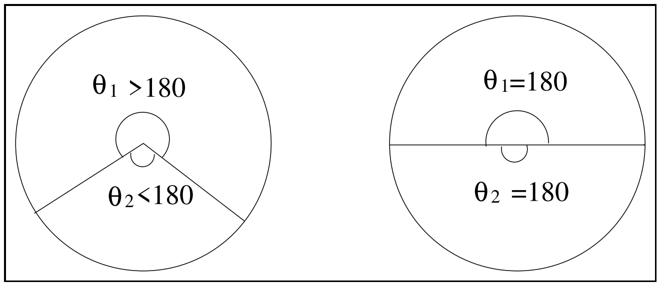


FIGURE A.1. Equivalence among two pairs of tiles.

180° or less (as in Figure A.1). The number of exchanges stays the same and the total path traveled is $2\pi R$ for a πR^2 area.

Lemma 2: When two tiles are connected along the curved portion of their wedges, the most efficient curve (in terms of path length) is the common chord they share.

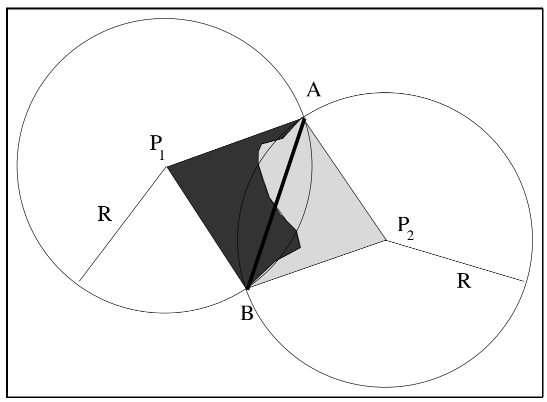


FIGURE A.2. Shortest path along the arc connections.

Proof: Given two tiles (eg. P_1AB and P_2AB, see Figure A.2), the shortest distance between A and B is given by the straight line connecting A and B. If the two robots travel in a a different path (other than a straight line) then the length of the path traveled would be larger than AB.

Lemma 3: When the robots exchange roles they produce tiles (areas of free space) that are connected along the rays that specify the boundaries of the wedges (see Figure A.3).

Proof: Let two robots explore a series of areas of free space by exchanging roles. Without loss of generality, let robot number one move inside the corresponding wedges as it covers the paths, AC, CE, EG, GI while robot number two moves across BD, DF, FH.

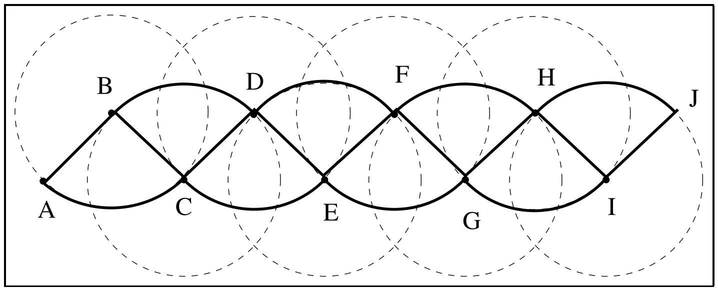


FIGURE A.3. Sequence of adjacent tiles. Dark lines define the wedges and each wedge is defined by three letters. Dashed lines define the sensor range boundary.

That happens because after the motion of one robot and the mapping of free space the two robots have to exchange roles and the other robot would continue mapping the free space starting from the common line of sight. Therefore each two neighboring tiles (such as ACB, BDC) are going to be connected along the common ray / line_of_sight (BC) that connects them when they exchange roles.

Lemma 4: Assume a sequence of N tiles (a stripe) connected as in **Lemma 3** with angles $\theta_i (1 \leq i \leq N)$ for each wedge. There exist an angle θ' such that: a sequence of N tiles all with the same angle θ' will cover an equal or larger area for the same length or less of the path traveled.

Proof: The total length of the path traveled for the two stripes is a sum of the sub-paths P^{θ_i} and $P^{\theta'}$ and is given in Equation A.1:

$$\begin{gathered}\sum_{i=1}^{N} P^{\theta_i} = \sum_{i=1}^{N} P^{\theta'} \\ \text{which is equivalent to} \\ \sum_{i=1}^{N} 4R \sin\frac{\theta_i}{4} = \sum_{i=1}^{N} 4R \sin\frac{\theta'}{4}\end{gathered} \tag{A.1}$$

We are going to prove that the sum of the areas covered with different angles is smaller or equal to the area covered by the same angle (see Equation A.2).

$$\sum_{i=1}^{N} A^{\theta_i} \leq \sum_{i=1}^{N} A^{\theta'} \Leftrightarrow$$
$$\sum_{i=1}^{N} R^2 \sin\frac{\theta_i}{2} \leq \sum_{i=1}^{N} R^2 \sin\frac{\theta'}{2} \tag{A.2}$$

Removing the constant terms from both sides Equation A.1 and Equation A.2 became:

Given N angles $\theta_i \ : \ i = 1, N$. **If**

$$\sum_{i=1}^{N} \sin\frac{\theta_i}{4} = N \sin\frac{\theta'}{4} \tag{A.3}$$

Then

$$\sum_{i=1}^{N} \sin\frac{\theta_i}{2} \leq \sum_{i=1}^{N} \sin\frac{\theta'}{2} \tag{A.4}$$

- For N=2, we solve equation A.3 for θ' (N=2):

$$\sin\frac{\theta_1}{4} + \sin\frac{\theta_2}{4} = 2\sin\frac{\theta'}{4} \Leftrightarrow$$
$$\sin\frac{\theta'}{4} = \frac{\sin\frac{\theta_1}{4} + \sin\frac{\theta_2}{4}}{2} \tag{A.5}$$

  For any pair (θ_1, θ_2) where $\theta_i \in [0, \pi]$ then $\Delta A = 2\sin\frac{\theta'}{2} - (\sin\frac{\theta_1}{2} + \sin\frac{\theta_2}{2}) \geq 0$ (see the graph of ΔA in Figure A.4).

  As $\Delta A \geq 0$ then

$$\sin\frac{\theta_1}{2} + \sin\frac{\theta_2}{2} \leq 2\sin\frac{\theta'}{2} \tag{A.6}$$

  Q.E.D. for N=2.

- For $N > 2$ we examine two cases, first $N = 2^K$ and then the general case.
  - For $N = 2^K$:

    Let the angles be in pairs of $(\theta_{2i-1}, \theta_{2i} : 1 \leq i \leq (N/2) = 2^{K-1})$. Then for every pair $(\theta_{2i-1}, \theta_{2i})$ calculate the angle $\theta'_i : 1 \leq i \leq 2^{K-1}$ as in the case for N=2 (Equation A.7).

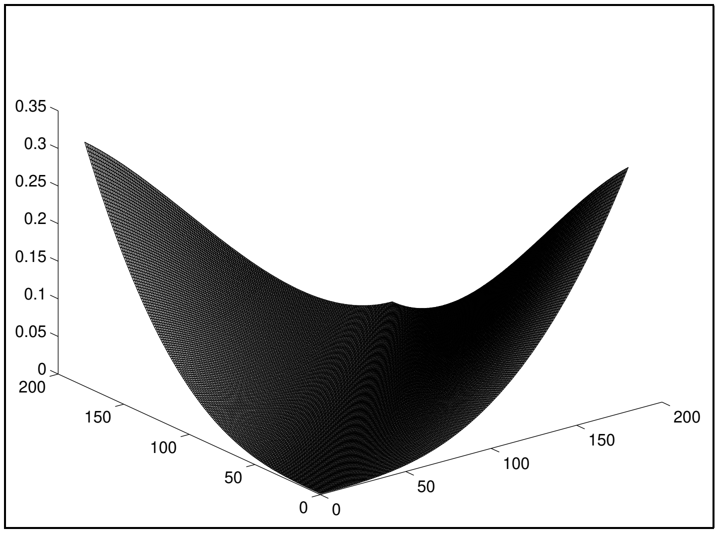


FIGURE A.4. Graph of ΔA for $0 \leq \theta_i \leq 180°$, $\Delta A \geq 0$ for any pair of angles

$$\sin\frac{\theta'_i}{4} = \frac{\sin\frac{\theta_{2i-1}}{4} + \sin\frac{\theta_{2i}}{4}}{2} \tag{A.7}$$

From Equation A.6:

$$\begin{gathered}\sum_{i=1}^{2^K} \sin\frac{\theta_i}{2} = \sum_{i=1}^{2^{K-1}} (\sin\frac{\theta_{2i-1}}{2} + \sin\frac{\theta_{2i}}{2}) \leq \\ \sum_{i=1}^{2^{K-1}} 2\sin\frac{\theta'_i}{2} = 2\sum_{i=1}^{2^{K-1}} \sin\frac{\theta'_i}{2}\end{gathered} \tag{A.8}$$

Therefore:

$$\sum_{i=1}^{2^K} \sin\frac{\theta_i}{2} \leq 2\sum_{i=1}^{2^{K-1}} \sin\frac{\theta'_i}{2} \tag{A.9}$$

Now we have $M = 2^{K-1}$ angles (θ'_i). Repeat the calculations for the θ'_i angles finding $M' = 2^{K-2}$ angles (θ''_i). Solving for pairs of angles $(\theta'_{2i-1}, \theta'_{2i} : 1 \leq i \leq (N/4) = 2^{K-2})$. Calculate from Equation A.10 all the angles $\theta''_i : 1 \leq i \leq 2^{K-2}$

$$\sin\frac{\theta''_i}{4} = \frac{\sin\frac{\theta'_{2i-1}}{4} + \sin\frac{\theta'_{2i}}{4}}{2} \tag{A.10}$$

From Equation A.6:

$$\sum_{i=1}^{2^{K-1}} \sin\frac{\theta'_i}{2} = \sum_{i=1}^{2^{K-2}} (\sin\frac{\theta'_{2i-1}}{2} + \sin\frac{\theta'_{2i}}{2}) \leq$$
$$\sum_{i=1}^{2^{K-2}} 2\sin\frac{\theta''_i}{2} = 2\sum_{i=1}^{2^{K-2}} \sin\frac{\theta''_i}{2} \tag{A.11}$$

From Equation A.8 and Equation A.11

$$\sum_{i=1}^{2^K} \sin\frac{\theta_i}{2} \leq 2\sum_{i=1}^{2^{K-1}} \sin\frac{\theta'_i}{2} \leq 2 \times 2 \sum_{i=1}^{2^{K-2}} \sin\frac{\theta''_i}{2} \tag{A.12}$$

Repeating the previous steps K times. For $\theta^{(K)}$ that satisfies Equation A.12 Equation A.14 is true.

$$\sum_{i=1}^{2^K} \sin\frac{\theta_i}{4} = 2\sum_{i=1}^{2^{K-1}} \sin\frac{\theta'}{4} = 4\sum_{i=1}^{2^{K-2}} \sin\frac{\theta''}{4} = N\sin\frac{\theta^{(K)}}{4} \tag{A.13}$$

$$\sum_{i=1}^{2^K} \sin\frac{\theta_i}{2} \leq 2\sum_{i=1}^{2^{K-1}} \sin\frac{\theta'_i}{2} \leq 4\sum_{i=1}^{2^{K-2}} \sin\frac{\theta''_i}{2} \leq \ldots \leq N\sin\frac{\theta^{(K)}}{2} \tag{A.14}$$

Q.E.D. For any N in the form: $N = 2^K$

– For $N \neq 2^K$:

Let $K \in \mathcal{N}$ such that $2^K \leq N \leq 2^{K+1}$. Then for the $\theta_i : 1 \leq i \leq N$ solve Equation A.15 for the θ'. And we want to prove that the inequality in Equation A.16 is true.

$$\sum_{i=1}^{N} \sin\frac{\theta_i}{4} = N\sin\frac{\theta'}{4} \tag{A.15}$$

$$\sum_{i=1}^{N} \sin\frac{\theta_i}{2} \leq \sum_{i=1}^{N} \sin\frac{\theta'}{2} \tag{A.16}$$

Consequently, by adding in both sides of Equation A.15 $(M\sin\frac{\theta'}{4})$ where $M = (2^{K+1} - N)$, Equation A.15 becomes Equation A.17.

$$\sum_{i=1}^{N} \sin\frac{\theta_i}{4} + M\sin\frac{\theta'}{4} = N\sin\frac{\theta'}{4} + M\sin\frac{\theta'}{4} = 2^{K+1}\sin\frac{\theta'}{4} \tag{A.17}$$

Now the number of terms in each side of Equation A.15, is 2^{K+1} and from Equation A.14 we get:

$$\begin{gathered}\sum_{i=1}^{N} \sin\frac{\theta_i}{2} + M\sin\frac{\theta'}{2} \leq (N+M)\sin\frac{\theta'}{2} \Leftrightarrow \\ \sum_{i=1}^{N} \sin\frac{\theta_i}{2} \leq N\sin\frac{\theta'}{2}\end{gathered} \tag{A.18}$$

Q.E.D.

Theorem 1: For any tiling, and with optimality criterion the shortest path for a given number of exchanges, the optimal tile is the union of two isosceles triangles (equal edges are R), with the angle of the two equal edges equal to $\theta/2$, where θ the wedge angle (see Figure A.5).

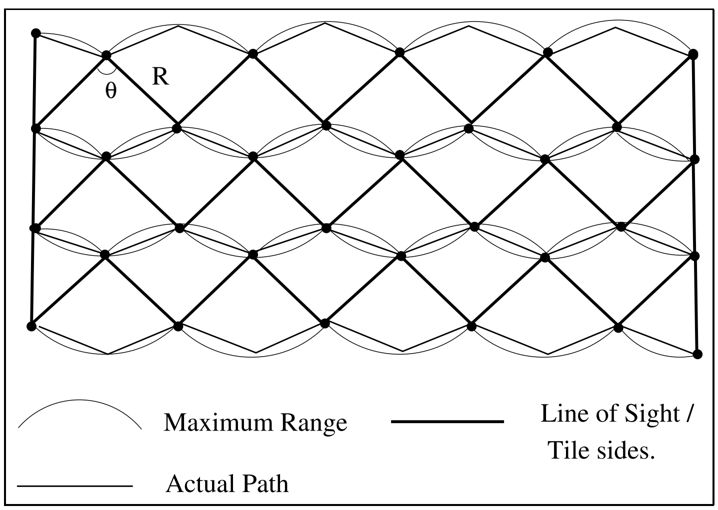


FIGURE A.5. Positioning of the wedge stripes.

Proof: From lemma 3 and for the same angle θ for every wedge, the wedges are going to be arranged into stripes as in Figure A.3. Moreover given a constant number of tiles the angle θ is set. When one stripe is positioned next to its neighbor, there must be complete coverage, with minimum overlap. The optimal positioning of the wedges is displayed in

Figure A.6, such that the curves are complement each other. From lemma 2, the optimal path is that given in Figure A.5, by connecting with straight lines the overlapped areas.

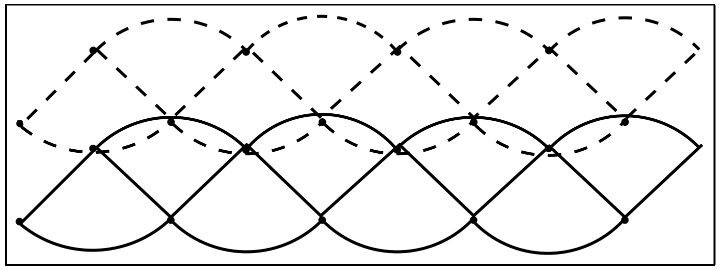


FIGURE A.6. Optimal tiling.

Q.E.D.

APPENDIX B

Odometry Error Study

1. Introduction

In this Appendix we consider the measurement of odometric uncertainty for a mobile robot. The primary emphasis is to experimentally estimate the rate of odometry error buildup in a small differential-drive research robot, and to model its behavior probabilistically. Although the use of Kalman filters and related techniques are common place for robotic systems, it is not uncommon for mobile robotics practitioners to merely make educated guesses not only for the rate of error accumulation for their robots, but also for the error model itself. While there are a few notable papers that rigorously consider error measurement for mobile robots [**16, 18, 116**], the most common error model used in practice is an unrealistic univariate two-dimensional Gaussian. Furthermore, in simulated environments very crude odometry error models are used, if the error is modeled at all.

Our goal was to develop a more realistic odometry error model that would reflect (at least partially) the complexity of the robot's locomotion. Such a model is used to describe faithfully the probability distribution function of the robot's pose after an arbitrary motion. The odometry error study presented here in combination with the proposed model provides a practical framework for the implementation of realistic odometry error in different simulation packages. Our primary experimental data is obtained from a differential-drive robot, although we believe the proposed probabilistic model applies to other types of drive mechanism and we have tested it informally on synchro-drive systems as well. The odometry error is detected using a calibrated laser range finder.

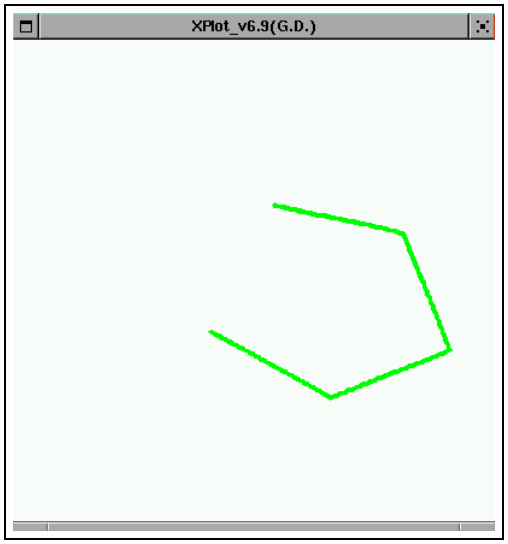
FIGURE B.1. Measuring the odometry error on carpet.

2. Odometry Study of a Differential Drive Robot

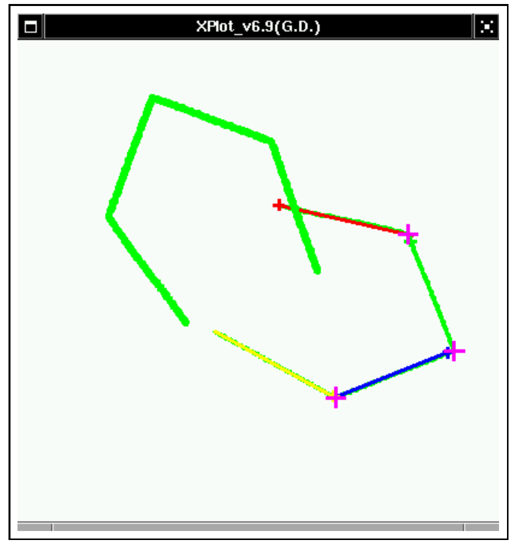
As a baseline we consider the odometry error accrued under various conditions by a commercial differential-drive robot, the Nomadic Technologies Superscout II. This robot uses two wheels to provide differential drive and odometry feedback with a third rear-mounted castor wheel for balance. Without loss of generality any arbitrary motion by $\Delta X, \Delta Y$ can be achieved by combining a rotation that points the robot towards the target location, followed by a translation that moves the robot to the target location. Therefore we divided the observations into rotational errors and translational errors.

2.1. Rotation. Empirical knowledge suggests that the largest factor in odometry error is the rotational error[1]. In order to measure the rotational error we placed the robot inside a "C"-shaped enclosure consisting of four walls (see Figure B.1,B.2a). The intersections of the four walls provide three geometric landmarks detectable both in world

[1]While we make this observation empirically, it follows naturally from the kinematics of the robot and a simple model for uncertainty in wheel velocity.



(a)



(b)

FIGURE B.2. The four walls providing three landmarks. (a) Before the rotation. (b) After the rotation.

coordinates and "raw" laser coordinates (see Chapter 6 section 3.2). Moreover, the orientations of the four walls in world coordinates should change by the amount of the robot's rotation. To estimate the error, the three landmarks are detected then the robot rotates and the three landmarks are detected again (see Figure B.2b). The three landmarks in laser coordinates provide three estimates for the rotation and the orientations of the four walls provide four more estimates. The seven estimates are kept only if they all agree up to 0.2 degree. We proceed to measure the rotational error for different motion parameters (rotation angle, speed, acceleration) and on different surfaces.

First, we measured the error in rotation for different rotation and translation speeds and for different angles. Figures B.3,B.4 present the error measurements relative to the odometer estimate (Figure B.3) and relative to the intended pose (Figure B.4); for every speed/angle we gather twenty samples. It is worth noting that they are concentrated (small standard deviation) around a non zero mean value. Moreover from Figures B.3,B.4 it is clear that a systematic error occurs that biases the error by the direction of the rotation (negative rotation have positive mean error). As it was expected the small rotations provide

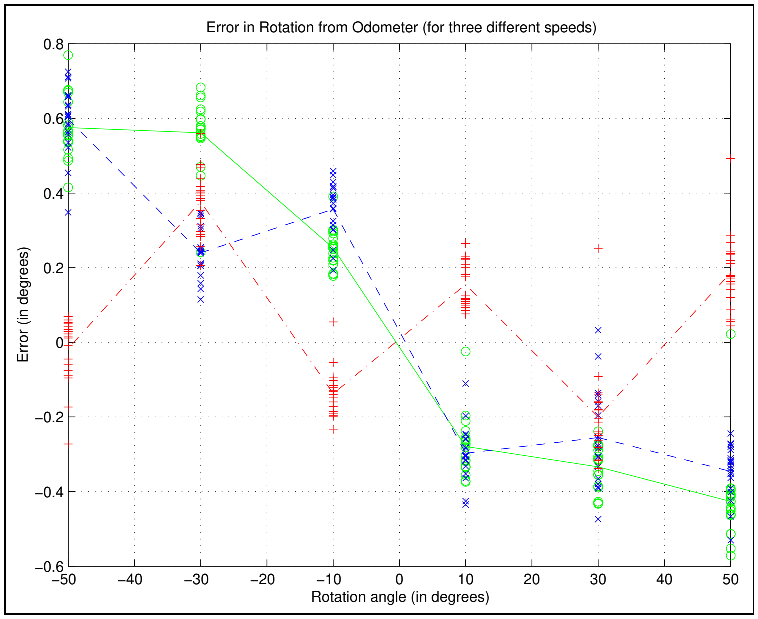


FIGURE B.3. Error in rotation relative to the odometer for different angles and for different speeds ("o" speed 10, "x" speed 50, "+" speed 90, lines connect the mean values).

negligible error. Surprisingly though, the higher speed produced less odometry error ("+" in the figures).

The effect of different surfaces on the rotational error was studied next. Four different surfaces were tested for a rotation of $-90°$ and forty samples were collected each time. The two types of carpets follow more closely a normal distribution than the other two surfaces. This is due to the fact that the surface is smooth contrary to the tile floor that contains bumps. The friction between the wheels of the robot and the floor (or the carpet) was relatively similar. On the contrary the plastic surface provided less friction thus significantly increasing the rotational error.

From Figure B.6 we see that the error from the intended rotation is much larger. Even though the odometer reported a pose different than the intended one, the control software of the robot stopped the rotation. For applications that require precise positioning, this extra error should be taken into account.

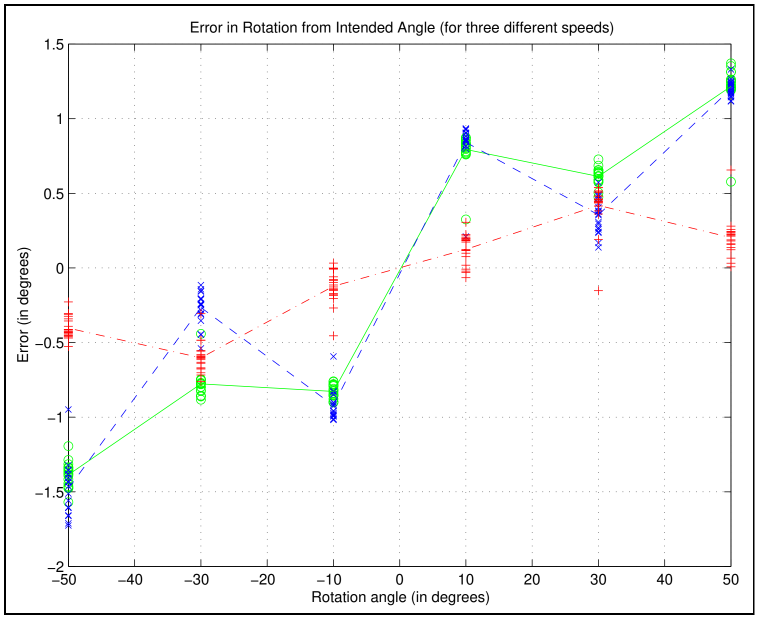


FIGURE B.4. Error in rotation relative to the intended pose for different angles and for different speeds (as in Figure B.3)

From the results described above we can deduce that a study of the odometry error of the particular mobile robot used is essential in order to model the systematic error that occurs during rotations. A zero mean Gaussian representation would require an unnecessarily large standard deviation forcing us to consider poses of the robot that are in fact unlikely.

3. Translation

The same setup used for the estimation of the rotational error is used also for the translation. The same enclosure was used (see Figure B.1). The robot was moved forward by a distance D over different surfaces and with different speeds. After every translation the robot was translated back (by -D) and the pose of the robot was reset to the origin ($P_r = [x_r, y_r, \theta_r]^T = [0, 0, 0]^T$). Figure B.7 presents the error accumulated after an intended translation of 100cm. The robot was moved 165 times over different areas of our lab (tiled floor). The first three sub-plots present a histogram of the error along the X and Y axis and for the orientation Θ. The fourth sub-plot present the spatial distribution of the robot poses for all the motions.

Table B.1 illustrates the effect of speed in the accumulation of odometry error for three different speeds (20, 60, 100) during the translation of 100cm along the x-axis. There is a

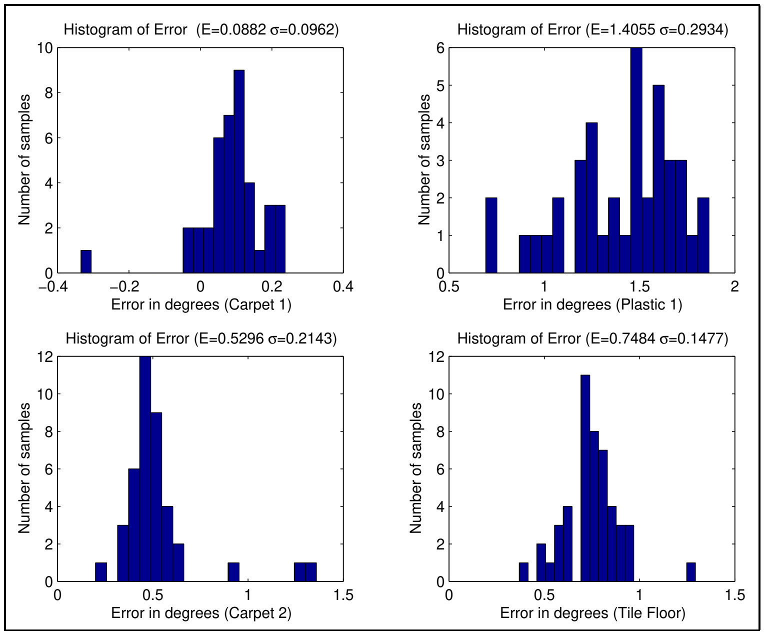


FIGURE B.5. Error distribution from the odometry measurement for different surfaces (rotation of 90°).

Speed	20		60		100	
	M	σ	**M**	σ	**M**	σ
X	-1.843	0.372	-1.850	0.363	-2.266	0.526
Y	-0.863	0.317	-0.977	0.491	-1.041	0.491
Θ	0.587	0.215	0.760	0.366	0.107	0.314

TABLE B.1. Mean error and Standard Deviation along the X,Y-axis (in cm) and orientation Θ (in degrees) after the translation of 100cm for three different speeds.

significant increase when the higher speed was used, especially in the systematic error as it manifests in the mean error along the axis of translation. The observations are consistent with the work presented by Moon *et al.* [**116**] where higher acceleration gives reduced orientation error. As can be seen in Table B.1 the high acceleration results in small orientation error but higher error along the direction of the translation.

The measurement of odometry error over different surfaces is presented next. Figure B.8 presents the results for a translation of 120cm on a plastic surface. The same behavior as with the rotation manifests during the translation, with the error in the distance traveled (X-axis) much higher than the error on carpet or tile floor. Figure B.9 presents the results for the translation on carpet.

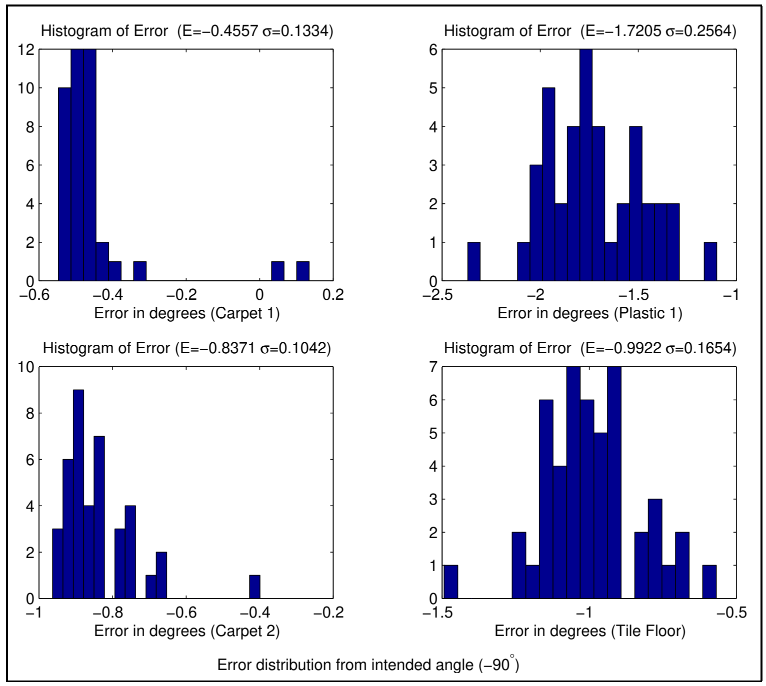


FIGURE B.6. Error distribution from the intended pose for different surfaces (rotation of 90°).

The statistical properties of the odometry error collected above enable us to create a realistic error model for the type of robot used. Furthermore, the odometry error measurements could be utilized in the construction of realistic simulation experiments.

4. Odometry Error Modeling

In the past little attention has been paid to the modeling of odometry error. The computing power was not enough to permit a precise modeling forcing early researchers to a simple Gaussian *pdf* around the final position of the moving robot as the most general error model. With the computing power currently available, even on board autonomous robots, more elaborate techniques such as condensation (a Monte-Carlo simulation method) and multiple Gaussians are used in order to track the accumulation of uncertainty during motion. In many cases, however, the error model is still based on a single random variable drawn from a normal distribution.

There are many sources of error that contribute to the accumulation of uncertainty during motion such as wheel slippage, difference in the diameters of the wheels and anomalies

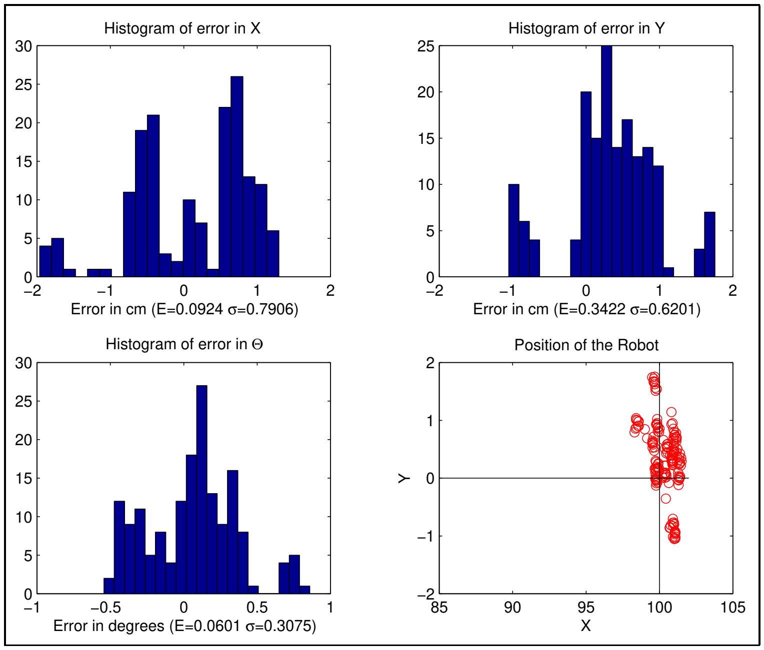


FIGURE B.7. Error distribution after translation of 100cm. Tile floor, 165 samples.

of the floor [2]. Without loss of generality any arbitrary motion by $\Delta X, \Delta Y$ can be achieved by combining a rotation that points the robot towards the target location, followed by a translation that moves the robot to the target location.

For modeling purposes the odometry error could be divided into rotational error [3] and translational error. These errors can be modeled statistically by random variables drawn from three Gaussians with zero mean and $\sigma_{\text{rot}}, \sigma_{\text{trans}}, \sigma_{\text{drift}}$ standard deviations. The first Gaussian models the error accumulated during pure rotations of the robot. The other two Gaussians model the error that occurs during a forward translation of the robot and affects the complete pose of the moving robot. It is worth noting that an additional source of error could be added that would represent bumps on the floor and small collisions by adding some "salt and pepper" noise.

4.1. Rotation. As we saw in Chapter 5 section 2.1.1 the noise model for rotational is straight forward described by the general equation B.1.

$$\theta_{k+1} = \theta_k + \Delta\theta + N(M_{rot}, \sigma_{rot}\frac{\Delta\theta}{360}) \tag{B.1}$$

[2]For a more detailed study please refer to Borenstein [**16, 57**].
[3]For simplicity's sake it is assumed that only the orientation of the moving robot is affected.

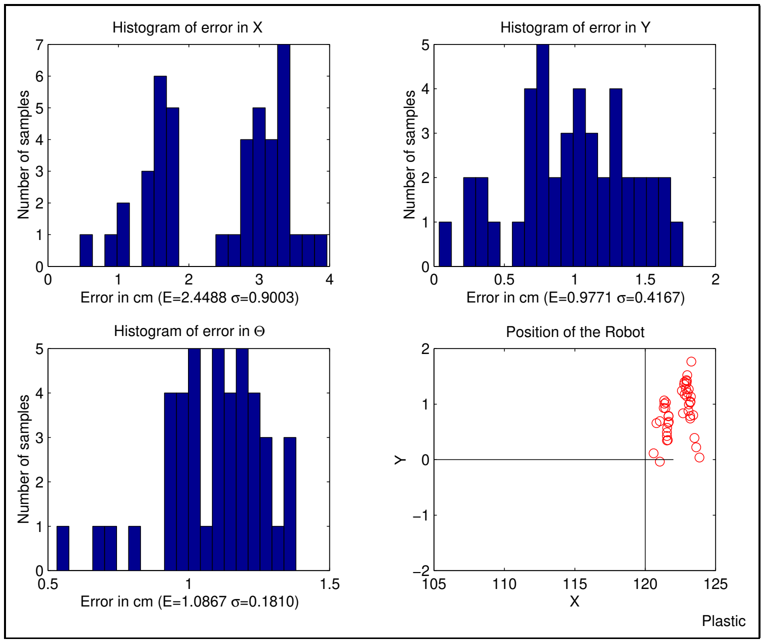


FIGURE B.8. Error distribution after translation of 120cm. Plastic surface, 43 samples.

4.2. Translation. Modeling the translation of the robot is more difficult because the noise model is more complex. During a translation by a distance R towards the orientation the robot two kinds of uncertainty accumulate: first, the distance the robot traveled is given by R plus an error. Second, the orientation of the robot constantly changes adding the equivalent of a Brownian type of noise to the final position. While for the real robot the drift is a continuous process that affects the complete trajectory of the translation during simulations, but more important during the modeling of uncertainty, a discretization of the process is required. The simplest approximation [4] of the above process is to model the translation as a partial rotation followed by a translation followed by a second rotation (Fig. B.10). The reason for this is that the robot would deviate from the trajectory, hence the initial rotation by a small angle, and also the final orientation of the robot is corrupted by some noise, hence the second rotation.

Orientation: For a single translation modeled as one step the orientation of the robot at the beginning would be θ_i and at the end the orientation is $\theta_{i+1} = \theta_i + \mathcal{E}_{\theta_1} + \mathcal{E}_{\theta_2}$, where $\mathcal{E}_{\theta_1}$ and $\mathcal{E}_{\theta_2}$ are the amount of the two rotations that occur before and after the translation (see Fig. B.10). From experimental data we could have an estimate about the standard

[4]It is the most commonly used.

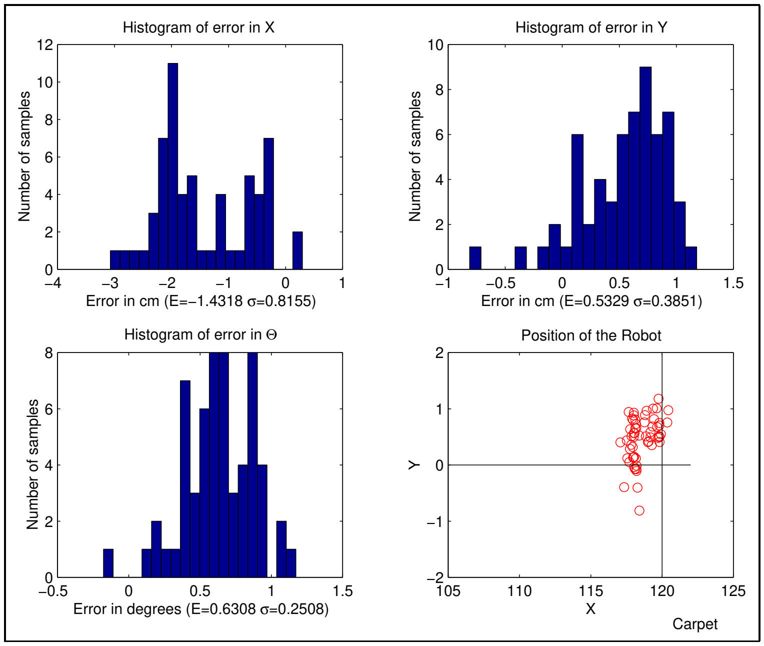


FIGURE B.9. Error distribution after translation of 120cm. Carpet surface, 43 samples.

deviation of the orientation as a function of the distance traveled (σ_{drift} in degrees per meter traveled). The standard deviation of the orientation after one translation can be calculated in terms of the characteristics of the noise $\mathcal{E}_{\theta_j}, j = 1, 2$, and if $\mathcal{E}_{\theta_1} = \mathcal{E}_{\theta_2} = \mathcal{E}_{\theta_j}$ then the standard deviation of the noise $\mathcal{E}_{\theta_j}$ is calculated in the equation B.2.

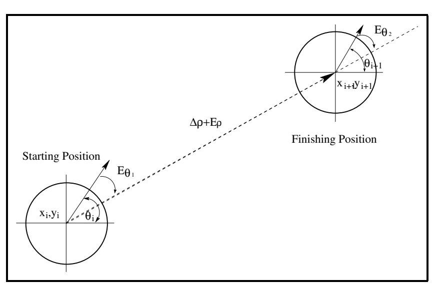


FIGURE B.10. One step in translation.

$$
\begin{aligned}
\sigma^2_{\theta_{i+1}} &= E\{\hat{\theta}_{i+1}\hat{\theta}^T_{i+1}\} \text{ where} \\
\hat{\theta}_{i+1} &= \theta_{i+1} - E\{\theta_{i+1}\} = \theta_i + \mathcal{E}_{\theta_1} + \mathcal{E}_{\theta_2} - \theta_i \\
&= \mathcal{E}_{\theta_1} + \mathcal{E}_{\theta_2} \text{ Therefore} \\
\sigma^2_{\theta_{i+1}} &= E\{(\mathcal{E}_{\theta_1} + \mathcal{E}_{\theta_2})(\mathcal{E}_{\theta_1} + \mathcal{E}_{\theta_2})^T\} \\
&= E\{(\mathcal{E}_{\theta_1})^2\} + E\{(\mathcal{E}_{\theta_1})^2\} + \\
& \quad 2E\{(\mathcal{E}_{\theta_1}\mathcal{E}_{\theta_2})\} \text{ where} \\
& \quad E\{(\mathcal{E}_{\theta_1}\mathcal{E}_{\theta_2})\} = 0 \text{ Uncorrelated, and} \\
& \quad E\{(\mathcal{E}_{\theta_1})^2\} = E\{(\mathcal{E}_{\theta_1})^2\} = \sigma^2_j \text{ Therefore} \\
\sigma^2_{\theta_{i+1}} &= 2\sigma^2_j \Leftrightarrow
\end{aligned}
$$

$$
\sigma_j = \frac{\sigma_{\theta_{i+1}}}{\sqrt{2}} \tag{B.2}
$$

More realistically, the translation could be modeled as a series of N equal steps of R/N length each, then the pose of the robot after step i would be: $\overline{\mathbf{x}}_{\mathbf{i}} = (\mathbf{x}_{\mathbf{i}}, \mathbf{y}_{\mathbf{i}}, \theta_{\mathbf{i}})^{\mathbf{T}}$ and the trajectory could be modeled as: $\overline{\mathbf{x}}_{\mathbf{0}}, \overline{\mathbf{x}}_{\mathbf{1}} \ldots \overline{\mathbf{x}}_{\mathbf{n}}$. Figure B.10 illustrates one step from $\overline{\mathbf{x}}_{\mathbf{i}}$ to $\overline{\mathbf{x}}_{\mathbf{i+1}}$. If the translation was performed in one step only then the drift could be modeled as a small rotation before the translation and a small rotation after the translation. Equation B.3 expresses the above described process, $\mathcal{E}_{\Delta\rho}$ is the noise added in the distance traveled and $\mathcal{E}_{\theta_1}, \mathcal{E}_{\theta_2}$ is the noise added in the orientation of the robot due to drift. The number of steps N used to model the uncertainty should not change the resulting distribution of the robot position. The statistical properties of the distribution of the robot Pose are established experimentally.

$$
\overline{\mathbf{x}}_{i+1} = \begin{pmatrix} x_{i+1} \\ y_{i+1} \\ \theta_{i+1} \end{pmatrix} = \begin{pmatrix} x_i + (\Delta\rho + \mathcal{E}_{\Delta\rho}) \cos(\theta_i + \mathcal{E}_{\theta_1}) \\ y_i + (\Delta\rho + \mathcal{E}_{\Delta\rho}) \sin(\theta_i + \mathcal{E}_{\theta_1}) \\ \theta_i + \mathcal{E}_{\theta_1} + \mathcal{E}_{\theta_2} \end{pmatrix} \tag{B.3}
$$

Orientation: At the end of step i the orientation of the robot is $\theta_i = \theta_{i-1} + n_i$ where n_i is the noise accumulated during that step. We assume the noise n_i to be zero mean

Gaussian and as we saw earlier the result of the addition of two Gaussians (see equation B.2). Therefore after the Nth step the orientation of the robot is $\theta_N = \theta_0 + \sum_{i=1}^{N} n_i$. And the statistical properties of the distribution are :

$$
\begin{aligned}
E\{\theta_N\} &= E\{\theta_0\} + \sum_{i=1}^{N} E\{n_i\} \text{ where} && \text{(B.4)}\\
& \sum_{i=1}^{N} E\{n_i\} = 0 \text{ zero mean noise} \Leftrightarrow && \text{(B.5)}\\
E\{\theta_N\} &= \theta_0 && \text{(B.6)}
\end{aligned}
$$

$$
\begin{aligned}
\sigma_{\mathrm{N}}^2 &= E\{\hat{\theta}_N \hat{\theta}_N^T\} \text{ where } \hat{\theta}_N = \theta_N - E\{\theta_N\} = \theta_N - \theta_1 \Leftrightarrow && \text{(B.7)}\\
\sigma_{\mathrm{N}}^2 &= E\{(\theta_N - \theta_1)(\theta_N - \theta_1)^T\} = E\{(\sum_{i=1}^{N} n_i)(\sum_{i=1}^{N} n_i)^T\} && \text{(B.8)}\\
&= E\{(n_1 + \ldots + n_N)(n_1 + \ldots + n_{\mathrm{N}})^T\} && \text{(B.9)}\\
&= \sum_{i=1}^{N} E\{n_i^2\} + E\{n_1(n_2 + \ldots n_N)^T\} + E\{n_2(n_1 + n_3 \ldots + n_N)^T\} + \ldots && \text{(B.10)}\\
&= \sum_{i=1}^{N} E\{n_i^2\} \text{ because} E\{n_i(\sum_{j=1:N}^{j<>i}(n_j))^T\} = 0 \text{ Uncorrelated } \Leftrightarrow && \text{(B.11)}\\
\sigma_{\mathrm{N}}^2 &= N\sigma_i^2 \Leftrightarrow \sigma_{\mathrm{N}} = \sqrt{N}\sigma_i \Leftrightarrow \sigma_{\mathrm{drift}}\rho = \sqrt{N}\sigma_{\mathrm{step}}\frac{\rho}{N} \Leftrightarrow && \text{(B.12)}\\
\sigma_{\mathrm{step}} &= \sigma_{\mathrm{drift}} * \sqrt{N} && \text{(B.13)}
\end{aligned}
$$

In conclusion, for a given set of uncertainty parameters, defined as $< \sigma_{trans}, \sigma_{drift} >$, the noise $(\mathcal{E}_{\Delta\rho}, \mathcal{E}_{\theta_1}, \mathcal{E}_{\theta_2})$ that should be added during the modeling of odometry error is given in equation B.14, where $\mathbf{N}(0,1)$ is a random number drawn from a Gaussian distribution with zero mean and sigma equal to one.

$$
\begin{pmatrix} \mathcal{E}_{\Delta\rho} \\ \mathcal{E}_{\theta_1} \\ \mathcal{E}_{\theta_2} \end{pmatrix} = \begin{pmatrix} \mathbf{N}(0,1)\sigma_{trans}\sqrt{N}\Delta\rho \\ \mathbf{N}(0,1)\frac{\sigma_{drift}\sqrt{N}\Delta\rho}{\sqrt{2}} \\ \mathbf{N}(0,1)\frac{\sigma_{drift}\sqrt{N}\Delta\rho}{\sqrt{2}} \end{pmatrix} \tag{B.14}
$$

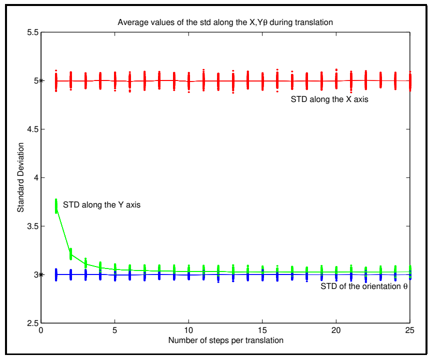


FIGURE B.11. The standard deviations of 30000 particles as they move along the x axis for 100cm using different number of steps each time. The experiment is repeated 100 times.

Using the above model we run experiments for different number of steps using multiple samples. It is worth noting that a change in the number of steps affects only the distribution of the points along the direction normal to the direction of the translation and only for small number of steps. As the number of steps increases the standard deviation of the samples along the direction perpendicular to the direction of the translation converges. Figure B.11 presents the standard deviation of 10000 particles along the X-axis, Y-axis and the orientation after they moved along the X-axis for 300cm, for different number of steps. The standard deviations along the axis of motion and for the orientation is constant for all practical purposes.

APPENDIX C

Resampling Methods

In this Appendix three methods of resampling are described together with some variations that help improve the performance. In every case the input is an array [1] of the weights of the particles (normalized to sum up to one) and the output is an array of indices that indicate which particles are going to propagate forward. The premise of all algorithm is that particles with high weights are going to be duplicated while the particles with small (or zero) weights are going to be eliminated.

1. Select with Replacement

The simplest method of resampling is to select each particle with a probability equal to its weight. In order to do that efficiently, first the cumulative sum of the particle weights are calculated, and then N sorted random numbers (sorting is $O(n\log(n))$ uniformly distributed in $[0, 1]$ are selected. Finally, the number of the sorted random numbers that appear in each interval of the cumulative sum represents the number of copies of this particular particle which are going to be propagated forward to the next stage. Intuitively, if a particle has a small weight, the equivalent cumulative sum interval is small and therefore, there is only a small chance that any of the random numbers would appear in it; in contrast, if the weight is large then many random numbers are going to be found in it and thus, many duplicates of that particle are going to survive. Algorithm 6 presents a formal description of the “select with replacement” algorithm.

[1] The arrays start at 1.

```
Input: double W[N]
Require: ∑_{i=1}^{N} W_i = 1
  Q = cumsum(W); { calculate the running totals Q_j = ∑_{l=0}^{j} W_l}
  t = rand(N+1); {t is an array of N+1 random numbers.}
  T = sort(t); {Sort them (O(n log n) time)}
  T(N+1) = 1; i=1; j=1; {Arrays start at 1}
  while (i ≤ N) do
    if T[i] < Q[j] then
      Index[i]=j;
      i=i+1;
    else
      j=j+1;
    end if
  end while
  Return(Index)
```

Algorithm 6: Select with Replacement Resampling Algorithm; functions are noted as underlined text, Comments are inside curly brackets "{}".

2. Linear time Resampling

```
Input: double W[N]
Require: ∑_{i=1}^{N} W_i = 1
  Q = cumsum(W); {calculate the running totals Q_j = ∑_{l=0}^{j} W_l}
  t = -log(rand(N+1));
  T = cumsum(t); {calculate the running totals T_j = ∑_{l=0}^{j} t_l}
  TN = T/T(N+1);{normalize T to TN;}
  i=1; j=1; {Arrays start at 1}
  while (i ≤ N) do
    if T[i] < Q[j] then
      Index[i]=j;
      i=i+1;
    else
      j=j+1;
    end if
  end while
  Return(Index)
```

Algorithm 7: Linear Time Resampling Algorithm; functions are noted as underlined text, Comments are inside curly brackets "{}".

Carpenter *et al.* [**27**] proposed a linear time algorithm for resampling from a set of particles. It is based on a manipulation of the random number sequence in order to achieve a new sorted random number sequence in linear time. Using the cumulative sum of the negative logarithm of N random numbers uniformly distributed in $[0, 1]$, a new sequence of N sorted random number uniformly distributed in $[0, 1]$ is created. The final step is the same as in the previous algorithm where the particles are selected with a probability proportional to their weights. Algorithm 7 presents a formal description of the "select with replacement" algorithm.

3. Resampling by Liu *et al.*

Instead of using directly the weights (w_j) of the particles in order to decide which ones are going to be propagated forward, another number a_j can be used, usually a function of the particles weights ($a_j = f(w_j)$). A generic choice is the the square root ($f(w_j) = \sqrt{w_j}$). Then the new weights (a_j) are normalized so they sum up to the number of particles N ($\sum_{i=1}^{N} a_i = N$). Then each particle is examined separately, and, if its weight (a_j) is greater or equal to one, k copies of it are propagated forward ($k = \lfloor a_j \rfloor$); otherwise, the particle "survives" with probability equal to a_j. One drawback of this approach is that the number of particles after resampling is not N anymore as the choice of how many particles survive is stochastic [2].

4. Variations on Resampling

Two main variations at the resampling stage have been proposed: corrective resampling and keeping a small percentage of particles from the old distribution.

4.1. Corrective Resampling. Jensfelt *et al.* [**79**] suggested a modification to the traditional SIR filter that "boosts" the contribution of the sensing versus the contribution of the predictive model. The particle population is "injected" during the update phase with a small number of particles created directly from the sensor data independently of where the rest of the particles are located.

4.2. Maintaining the variance of the distribution. Contrasting to the previous approach is the method of maintaining a small percentage of the particle population independently of their weights. More precisely during the resampling stage a small number of particles selected uniformly from the particle population are being propagated forward given a small weight. The intuition behind this approach is to maintain the coverage of the

[2] Stochastic is a process that is random but it follows certain distributions.

```
Input: double W[N]
for (j = 1 to N) do {Update the weights}
  a[j] = \sqrt{W[j]}
end for
sum = 0;
for (j = 1 to N) do {calculate \sum_{i=1}^{N} a_i}
  sum = sum + a[i];
end for
for (j = 1 to N) do {Normalize the weights (a) to sum up to N}
  a[j] = N * a[j]/sum
end for
i=1;
for (j = 1 to N) do {For each particle}
  if (a[j] ≥ 1) then {Accept the ones with bigger weights}
    for (l = 1 to ⌊a[j]⌋ ) do {Add ⌊a_j⌋ copies of the j^th Particle}
      Index[i]=j;
      i = i + 1;
    end for
  else
    R =rand(1);
    if a[j] ≥ R then {Accept the particle with probability a_j}
      Index[i]=j;
      i = i + 1;
    end if
  end if
end for
Return(Index)
```

Algorithm 8: Resampling Algorithm; functions are noted as underlined text, Comments are inside curly brackets "{}".

predictive model in the particle population without affecting the accuracy of the localization.

Document Log:

Manuscript Version 0

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX — 12 March 2003

Ioannis Rekleitis

Center for Intelligent Machines, McGill University, 3480 University St., Montréal (Québec) H3A 2A7, Canada, *Tel.* : (514) 398-2186

E-mail address: `yiannis@cim.mcgill.ca`

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX