Feature-relative Real-time Obstacle Avoidance and Mapping

by

Jacques Chadwick Leedekerken

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 2005 [February 2005]

Author .....
Department of Electrical Engineering and Computer Science
January 18, 2005

Certified by .....
John J. Leonard
Associate Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Feature-relative Real-time Obstacle Avoidance and Mapping

by

Jacques Chadwick Leedekerken

Submitted to the Department of Electrical Engineering and Computer Science
on January 18, 2005, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

A substantial challenge in robotics is integration of complex software systems for real-time performance. This thesis integrates the robust and generic mapping framework *Atlas*, a feature-based local Simultaneous Localization and Mapping (SLAM) module, and obstacle avoidance using information from mapped features. The resulting system performs autonomous feature-relative Real-time Obstacle Avoidance and Mapping (ROAM) with laser or sonar range sensors, and results are shown for wide-beam sonar. This system will allow high-speed feature-relative obstacle and avoidance and navigation on mobile robots with wide-beam sonar and/or laser sensors.

Thesis Supervisor: John J. Leonard
Title: Associate Professor

Acknowledgments

I'd like to thank my advisor John Leonard for introducing me to robotics and guiding me through this work. John has helped me learn much about robots and sonar, topics I knew nearly nothing about a year ago. I would also like to thank Andrew Patrikalakis for all his help with both hardware and software problems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis demonstrates a system for real-time performance of simultaneous localization, mapping, and obstacle avoidance using a feature-based representation of the enviroment for a robot with laser and/or sonar range sensors. A Real-time Obstacle Avoidance and Mapping (ROAM) implementation demonstrates the ability to use obstacle avoidance algorithms for laser sensors on a robot using wide-beam sonar sensors by simulation of a range scanning sensor with high angular resolution from the mapped environment. Results using sonar for real-time feature-based obstacle avoidance demonstrate real-time feature-based sonar navigation and mapping is possible.

1.1 Motivations

The primary questions motivating this thesis research are 1) Can an autonomous vehicle perform SLAM and navigate relative to mapped features? and 2) Can feature-relative obstacle avoidance and navigation be done in real-time at high speeds using sonar sensing?

The first question involves making SLAM into a blueprint for action, and provides a basis for control feedback for closer inspection of uncertain features. The second question involves addressing high-speed safety with cheap wide-beam sonars, which have significant angular uncertainty and many outlier measurements from sen-

sor crosstalk, mutliple reflections, and echos reflected away from the robot.

Compared to grid-based mapping, another popular mapping approach, feature-based mapping avoids *smearing* data. Grid-based approaches often use spread range measurements across regions of the map to accomodate uncertainty in both the measurements themselves and the features. One should note the difference in those two types of uncertainty; measurement noise is a constant and stipulated *a priori*, whereas navigational uncertainty is *a posterior*, resulting from a history of measurement and process noises. Coupling those two uncertainty sources prematurely leads to information loss. Extraction of local features may still be acheived with accuracy despite navigational uncertainty, and action relative to local features is still possible.

Another motivation for feature-based mapping is for interpretation. As stated in [16], a feature-based representation of the environment can determine *what* measurements come from rather than just *where* the measurements come from. For many applications, goals are formulated around features. The robot is observing on behalf of the human researcher, whose reasoning can relate more to features than a raw statistical formulation. This allows continued development for more intelligent navigation strategies. By demonstrating the simplest navigation strategy, obstacle avoidance, using only information from features, other feature-relative navigation strategies may operate cooperatively with SLAM.

1.2 Applications

Applications of the feature-based ROAM described in this thesis are numerous. In a research context, feature-based ROAM provides a basis of support for more complex path-planning algorithms and topological navigation. The accomdation of wide-beam sonar permits operation in many environments.

For land-based robots this research provides a platform for further feature-relative autonomous path-planning or missions in large environments, and fusion of laser and sonar sensing permits functionality in a wide-range of environments. While laser sensors are very accurate, they are costly, only reliably detect opaque objects, are

sensitive lighting conditions, and work only in air. For marine robotics where laser sensors are not applicable, this research provides the infrastructure for low-cost wide-beam sonar based autonomous inspection of piers, ship hulls, etc.

1.3 Related Research

A review of research relevant to feature-based ROAM reveals issues that must be tackled. ROAM inherents issues from both SLAM and obstacle avoidance in addition to issues from integration constraints. The integration constraints narrow the choice of valid representations for acheiving SLAM and obstacle avoidance.

1.3.1 Sonar SLAM

Previous work on mapping features with wide-beam sonar shows the difficulty of this task. In [7] measurement models for several sonar feature types are presented, and localization using sonar is demonstrated. The sonar measurement models presented in [7] provide the basis for the models in this thesis and other works, such as [16] and [9]. A submap-based sonar SLAM with line and point features with impressive results was shown in [16]. [16] presented robust data association for sonar returns using Random Sampling and Consenus (RANSAC). This thesis adopts a similar approach.

1.3.2 Vector Field Histogram

Previous work on real-time obstacle avoidance with sonar has been deomonstrated sucessfully with the Vector Field Histogram in [1]. The Vector Field Histogram (VFH) extended the familiar virtual force field approach to obstacle avoidance with a representation for accomdating sonar sensors. The VFH probabalistically smeared sonar returns across cells in a global confidence grid for mapping. The grid is reduced to a polar histogram, whose bins are finally reduced to a single value scoring the occupancy for each sector.

The drawbacks of the VFH are it is not feature-based, limited to small environ-

ments, and subject to various local minima behaviors. Addressing the limitation to small environments requires localization for correction of accumulating odometry errors. Concurrent localization would also allow increasing the scale of the map, which would provide information that could be used to plan and avoid many of the local minima behaviors.

1.4 System Overview

The feature-relative ROAM system may be described as the integration of real-time SLAM and obstacle avoidance systems. The core technology for real-time mapping derives from the framework *Atlas* [3], an efficient framework for localization and mapping in large-scale, cyclic environments. The *Atlas* framework manages a graph of local submaps for the environment, and each submap performs localized SLAM. The localized SLAM implementations for submaps are feature based supporting sonar sensing for map building and localization. Obstacle avoidance and navigation modules interface the mapping framework, and iteration of mapping and navigation occur synchronously.

Chapter 2

Simultaneous Localization and Mapping

The goal of Simultaneous Localization and Mapping (SLAM) may be stated as estimate both the position of a robot and its environment by processing measurements from its sensors. Real-time implementations must be causal, where estimates can only use measurements already observed.

The basic tool used for SLAM is the extended Kalman Filter (EKF). This thesis uses an EKF to estimate the dynamic state parameters of the mobile robot's state as well as the static state parameters of the enviroment's features.

The SLAM implementation used in this thesis may be described from the models of the robot and the environment. The models dictate which states the EKF must have, how those states are updated with new data, and how they are predicted. Additionally, the enviroment model determines when to add additional states to the EKF. The EKF states related to the robot paramterize the robot pose, and the EKF states for the enviroment parameterize the features. In this implementation synchronization of odometry measurements and range sensor readings occur prior to updates, where observations are structured as views containing the timestamp, observed odometry, range data, and features extracted in the view of raw data.

2.1 Robot Model

The B21 mobile robot used for experiments uses a four wheel synchro-drive system. The robot allows independent control of translational and rotational motion; the dynamics are more like a tank than a car. Odometry measurements from the driver are simple integrations of linear and angular path lengths converted into poses. Measurement error increases with time as the wheels or turning base slip or skid. Any concurrent translational and angular velocity also introduce some error into the reported pose since the driver does not account for the position and heading dependencies from path curvature. Modelling the B21's four wheel synchro-drive system is not necessary, and a two wheeled odometry is used for reasons of simplicity and relatively high sensor frequency. Odometry measurements may be observed with periods as small as 30ms and sonar measurements as small as 180ms. The details of the software driver written are discussed in section ??.

The robot pose has three parameters, $\mathbf{x_r} = [x \ y \ \theta]^T$, for the position and heading of the robot. Propagation of the robot state follows a two-wheeled odometry model with differential motion updates. The change in observed poses is converted into translational and rotational velocities (ν, ω) to compute the differential update and noise, which is velocity dependent.

$$\Delta x = \nu \Delta t \, (-\sin \omega \Delta t) \tag{2.1}$$

$$\Delta y = \nu \Delta t \, (\cos \omega \Delta t) \tag{2.2}$$

$$\Delta \theta = \omega \Delta t \tag{2.3}$$

The noise model for odometry is velocity dependent. Noise source parameters for wheel slippage (σ_{slip}) , wheel scale (σ_{scale}), and wheel base (σ_B).

$$\sigma_T = \frac{\sigma_{scale}}{2}\sqrt{\nu^2+\omega^2} \tag{2.4}$$

$$\sigma_R = \sqrt{\left(\nu^2+\omega^2\right)\sigma_{scale}^2 + \left(\omega\sigma_B\right)^2} \tag{2.5}$$

$$\mathbf{J_{robot}} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

$$\mathbf{Q} = \mathbf{J_{robot}} \begin{bmatrix} \sigma_T^2 & 0 & 0 \\ 0 & \sigma_R^2 & 0 \\ 0 & 0 & \sigma_{slip}^2 \end{bmatrix} \mathbf{J_{robot}}^T \tag{2.7}$$

2.2 Measurement Models

The measurement model depends on the sensor type. The implementation in this thesis supports both laser and wide-beam sonar range sensors. The measurement model for laser range sensors uses line features, and the sonar model uses both lines and point features.

2.2.1 Laser Feature Model

Feature Extraction

Feature extraction with laser range sensors is limited to line segments. Line segments are estimated with a split-and-fit algorithm for each laser scan. Lines are paramterized by ρ, the perpendicular distance of the line from the map origin, and ϕ, the angle of the line's normal. Endpoints do not have corresponding Kalman states, and thus not used in updating the robot's pose. However, endpoints are tracked and used for data association with observed lines since overlap and coverage is used in matching and extent of mapped lines may increase.

Line parameters ρ and ϕ are estimated from a set of N points in a laser scan

according to the following equations:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.8}$$

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{2.9}$$

$$c_{xx} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{2.10}$$

$$c_{yy} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2 \tag{2.11}$$

$$c_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}) \tag{2.12}$$

$$\phi = \frac{1}{2}\arctan\left(\frac{2c_{xy}}{c_{xx} - c_{yy}}\right) - \frac{\pi}{2} \tag{2.13}$$

$$\rho = \bar{x}\cos\phi + \bar{y}\sin\phi \tag{2.14}$$

Extracted line segments in the SLAM module require a minimum number of observations prior to becoming a feature. Due to the low noise of the SICK laser scanner used and its angular precision of about .5 degrees, this number is very low at 2 observations. With a peroid of about .4s, extracted features are quickly incorporated into the map.

Feature Prediction and Matching

Mapped features, which are in the local map coordinate frame, are transformed into predicted features, which are in a robot centered coordinate frame, with a transformation derived from the robot's pose relative to the map origin (see appendix A).

After transforming mapped features into the robot's view, possible matches are found using nearest neighbor gating based upon a Mahalanobis distance using the feature parameters and covariances. Multiple matches between observed features a mapped feature is allowed. For example separate portions of a line may be observed due to a partial occlusion. However, mapping observed features to more than one mapped feature is forbidden, because the measurements may not be used twice.

For a predicted mapped feature L_m and an observed feature L_z, the Mahalanobis distance d is computed from the difference in feature paramaters and covariances from both features:

$$L_{zm} = \begin{bmatrix} \rho_z - \rho_m \\ \phi_z - \phi_m \end{bmatrix} \tag{2.15}$$

$$d_z = L_{zm}^T (\Sigma_{L_z} + \Sigma_{L_m})^{-1} L_{zm} \tag{2.16}$$

The distance d_z is compared with two thresholds. A near threshold determines valid matches. A far threshold is used to prune out gross mismatches and for preliminary tracking of lines observed with laser, but not sonar. Observed features that have distance exceeding the far threshold for all mapped features are considered new, and begin the initialization process to be described shortly. For line features, an additional two thresholds are used to determing near and far lines based upon percentage overlap. Percentage overlap for nearly parallel lines in this implementation is defined to be the length of the intersect divided by the length of the union.

Feature Updates

The parameter residuals of the matched features are used in the EKF update step. For line features the endpoints do not have Kalman states, but the line feature object's endpoints may be extended when the observed lines project further than the extent of the mapped line.

Feature Initialization

When using a laser sensor, the map feature initialization process requires a feature to be tracked for a brief period before insertion into the map. Preliminary features are those that are judged to be far from all mapped features using the thresholds mentioned in the section 2.2.1. In subsequent iterations, observed features unmatched with mapped features are matched the prelimary feature set, and newer preliminary lines are found from the remaining unmatched far observations. The prelimary fea-

tures are propagated with the robot's change in odometry, but there is no update step to preliminary features. Preliminary features must be observed a minimum number of times during this stage to be mapped and are otherwise deleted. Once a preliminary feature is observed the minimum number of times, the latest observation for that preliminary feature is inserted into the map.

2.2.2 Sonar Feature Model

Feature extraction with wide-beam sonar presents several challenges not seen with laser range sensors. Unlike laser sensors, wide-beam sonar typically produces many spurious readings and individual range measurements with wide-beam sonar have significant angular uncertainty. This requires special processing of the raw sonar data for feature extraction and multiple sonar echos for echolocation of objects. This implementation assumes a 2D representation to dramatically simplify the model and reduce computation time.

Multiple Vantage Points

The principle driving the measurement model of sonar features in this implementation is mapping features from multiple vantage points due to partial observability [8]. This principle provides the means to accomodate the angular uncertainty inherent to wide-beam sonar sensors. Observation of an object from multiple vantage points provides the information necessary to reduce angular uncertainty and differentiate between feature types.

The EKF state maintains several previous robot poses sampled using a minimum motion baseline to provide the multiple vantage points. These vantage points are bounded in number, and once the maximum number of poses is reached, adding new poses causes deletion of the oldest pose. The time of the oldest pose limits the history of sonar returns to use in the feature extraction module that preprocesses the raw sonar data for adjacencies.

The Kalman states for these saved poses are not propagated like the current robot

pose state, but the saved poses are subject to change during the update step. The changes from the update state are propagated into the feature extraction module as well.

Two configurable parameters for the vantage points in the Kalman state affect how the vantage points are chosen, which affects how new features are initalized into the map. The maximum number of vantage points should be small enough to maintain some locality around the current robot pose since measurements taken from distant poses are less likely to have been from the same feature, especially in cluttered enviroments where object occlusion restricts visibility. The distance between vantage points should be large enough for sonar parallelax so observation of feature between two vantage points can reasonably conclude that it is in fact a feature and not a phantom feature due to multiple reflections or sensor crosstalk.

Feature Extraction

The basic model for wide-beam sonar range processing in this implementation uses pair-wise consistency tests since sonar range measurements give the range to the nearest surface in the beam but not the exact bearing. Each sonar range measurement represents arc of width equal to the beam width. Pairwise consistency occurs when the arcs of range measurements taken from multiple poses intersect and denote that the measurements are likely to be reflections of the same object. A Random Sampling and Consenus (RANSAC) approach is used to extract the best estimates of features from sets of consistent range measurements.

The sonar RANSAC module keeps a short history of the robot's position, and the range and bearing of the sonar echo (x, y, θ, r). This history is limited by the time of the oldest saved pose, or vantage point, in the SLAM module's state vector. New sonar data is added when the robot has traveled a minimum distance (a movement threshold) from the previous pose where sonar data was added. When adding new sonar range data, each measurement is tested for compatability with existing measurements. The compatability tests are binary constraints for adjacency of measurements from point and line features. A feature may be extracted from a set of

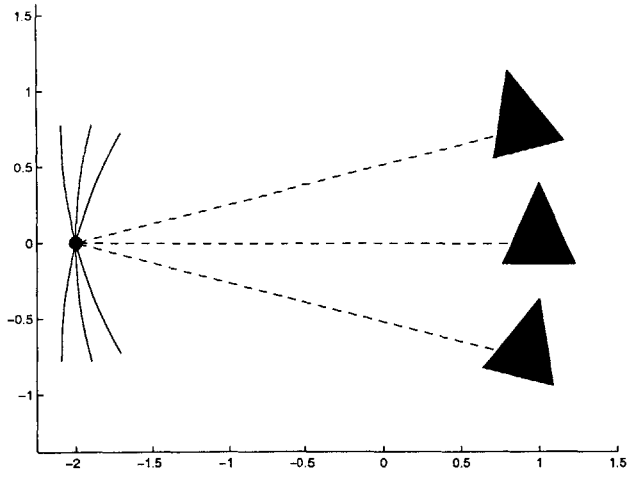

Figure 2-1: Example of multiple vantage points of a sonar point feature.

compatible measurements when that set reaches minimum count.

For point features, two sonar echos are adjacent only if the distance d between the centers of their arcs is less than the average width of the arc. Also the arcs must intersect, and a circle intersection test, where *det* as defined below is real, is used when the previous test passes as well. The arc intersect is computed as follows given two measurements $(x_1, y_1, \theta_1, r_1), (x_2, y_2, \theta_2, r_2)$:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.17}$$

$$det = \sqrt{((r_1 + r_2)^2 - d^2)(d^2 - (r_2 - r_1)^2)} \tag{2.18}$$

$$x = \frac{1}{2}\left(\frac{-(y_1 - y_2)(\pm det) - (r_2^2 - r_1^2)(x_2 - x_1)^2}{d^2} + (x_1 + x_2)\right) \tag{2.19}$$

$$y = \frac{1}{2}\left(\frac{+(x_1 - x_2)(\pm det) - (r_2^2 - r_1^2)(y_2 - y_1)^2}{d^2} + (y_1 + y_2)\right) \tag{2.20}$$

Extraction of line features requires the arcs of two echos to intersect and to be contangent to a line. When the circles of the arcs have possible cotangent lines, the value of *det* computed as shown below will be real.

$$det = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 - (r_2 - r_1)^2} \tag{2.21}$$

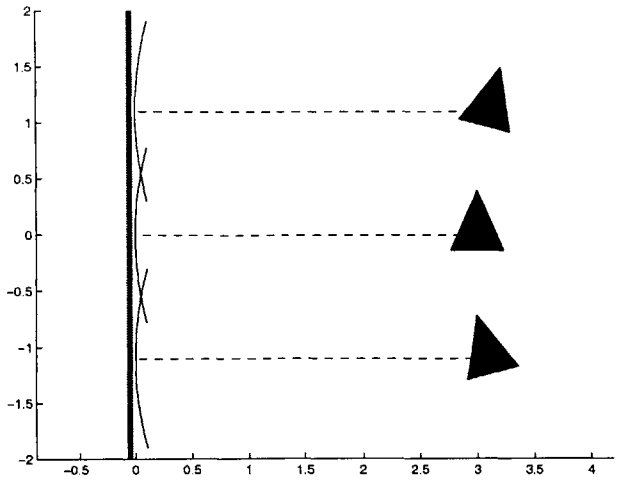

Figure 2-2: Example of multiple vantage points of a sonar line feature.

Of the possible four cotangent lines of two circles, only the ones on the same side of the two circles are considered, and one of those two will be on the portion of the circle within the beam arc. The line parameters (ρ, ϕ) are then computed as:

$$\rho = \frac{(x_1y_2 - x_2y_1)(\pm det) - (y_2 - y_1)(r_1y_2 - r_2y_1) - (x_2 - x_1)(r_1x_2 - r_2x_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (2.22)$$

$$\phi = \arctan\left(\frac{-(x_2 - x_1)(\pm det) - (y_2 - y_1)(r_2 - r_1)}{(y_2 - y_1)(\pm det) - (x_2 - x_1)(r_2 - r_1)}\right) \quad (2.23)$$

Feature Prediction and Matching

The data association step for sonar features is very similar to the procedure used with laser features, and the key differences are discussed here. Unlike laser, data association of sonar features is done in local map coordinate frame since multiple vantage points are required to observe features. Matching observed sonar line features with mapped line features is done with the procedure as laser line features, but a separate set of thresholds are used. For sonar point features, the same matching procedure is used for the point parameters (ρ, ϕ), but there are no overlap conditions.

Far thresholds for sonar features simply prune out gross mismatches and determine candidates for new features. Unlike laser lines, there is no tracking of preliminary

sonar features because extraction of sonar features requires multiple vantage points that is similar to tracking of laser lines through several poses although minimum baseline between the vantage points is required with sonar.

Feature Updates

When a valid match between an observed sonar feature and mapped sonar feature is found, updates to the Kalman state of the feature are not based upon the feature parameter residuals. Instead the residuals of the sonar echos from the mapped feature are used since the noise is more easily and accurately modelled. The data association step for features simply determines which the set of consistent echos associated with the mapped feature to perform the updates. The residuals for the echos of the observed feature from the predicted echos for the associated mapped feature update the feature parameters according to the following computations.

The robot position in the map's frame is (x_r, y_r, θ_r), the sensor location in the robot's frame is (x_s, y_s). Point prediction and updates are performed according to:

$$\Delta x = (x_r + x_s \cos\theta_r - y_s \sin\theta_r) - x_p \tag{2.24}$$

$$\Delta y = (y_r + x_s \sin\theta_r + y_s \cos\theta_r) - y_p \tag{2.25}$$

$$d_p = \sqrt{\Delta x^2 + \Delta y^2} \tag{2.26}$$

$$\nabla_{\mathrm{p}} \mathrm{d_p} = \begin{bmatrix} \frac{-\Delta x}{d} & \frac{-\Delta y}{d} \end{bmatrix} \tag{2.27}$$

$$\nabla_{\mathrm{r}} \mathrm{d_p} = \begin{bmatrix} \frac{\Delta x}{d} & \frac{\Delta y}{d} & 0 \end{bmatrix} \cdot \mathrm{J_1(x_r, x_s)} \tag{2.28}$$

Line prediction and updates are performed according to:

$$x_{rs} = x_r + (x_s \cos\theta_r - y_s \sin\theta_r) \tag{2.29}$$

$$y_{rs} = y_r + (x_s \sin\theta_r + y_s \cos\theta_r) \tag{2.30}$$

$$d_l = \rho_l - (x_{rs} \cos\phi_l + y_{rs} \sin\phi_l) \tag{2.31}$$

$$\nabla_{\mathrm{l}} \mathrm{d_l} = [1 \ (x_{rs} \sin\phi_l - y_{rs} \cos\phi_l)] \tag{2.32}$$

$$\nabla_{\mathrm{r}} \mathrm{d_l} = [-\cos\phi_l \ -\sin\phi_l \ 0] \cdot \mathrm{J_1(x_r, x_s)} \tag{2.33}$$

Feature Initialization

Candidate features for initialization into the map require visibility from more than one saved vantage point in the Kalman state, so the set of sonar echos for that feature must contain echos from more than one saved robot pose in the Kalman state. At least three echos are required for initialization, and a configuration file parameter allows this number to be increased. The echos from the oldest and newest vantage points are used to initialize the feature. The remaining echos that occur at are used to update the newly initialized feature in the same manner as echos from an observed feature and an already mapped feature.

2.3 Large Scale Mapping

In large-scale environments a single EKF implementations may not be sufficient. As the number of features grows, complexity grows $\Omega(n^2)$, and real-time performance may be impossible with the computation resources. Furthermore, the system is not fault-tolerant and EKF divergence from bad data association, spurious sensor readings, or error propagation from distant an uncertain objects may cause system collapse. To achieve a more fault-tolerant design with bounded complexity, this thesis uses a graph local submaps with hypotheses and matching techniques to manage traversal in large, complex environments. This design choice benefits from spatial locality of features for localization and mapping within submaps, but sacrifices global map accuracy for stability, bounded complexity, and scalability.

This thesis incorporates the mapping framework *Atlas* [3] for efficient SLAM in large-scale envrionments to address several engineering issues with ROAM. *Atlas* builds a graph of nodes consisting of local submaps performing SLAM and edges between these submaps. *Atlas* performs uncertainty projection, submap genesis, map-matching, cycle-verification, and traversal with competing hypotheses in a robust and modular fashion. The advatanges of *Atlas* for a robust ROAM system are numerous. *Atlas*'s use of submaps allows for substantial global error without failure by capturing uncertainty in the graph edges. In a large-scale environment this property allows for

continued operation with accurate local maps for obstacle avoidance without influence of errors from distance submaps or loop closure failures. In essence this fault-tolerant operation is possible at the expense of global error. *Atlas* bounds the complexity of local maps for constant-time performance, allowing real-time obstacle avoidance and path-planning control methods time to compute regardless of the environment. Another advantage of submaps for ROAM is locality of features for obstacle avoidance. Since distant features are not relavant to obstacle avoidance, the locality and limited complexity of the local submap bounds the computation required obstacle avoidance, allowing real-time high-speed performance. However, features from all submaps may be accessed should one want such information for global path-planning.

Atlas manages local submaps, or *map-frames*, through operations for genesis, uncertainty projection, map-matching, traversal with competing hypotheses, and cycle verification. *Atlas* abstracts the local maps' SLAM implementation from the mapping framework. For the purposes of this thesis, only a feature-based SLAM implementation accomodating laser and/or sonar range sensors is used, but other local SLAM methods are possible, and [3] demonstrated *Atlas* with laser scan-matching and vision. For use with *Atlas*, valid SLAM implementations provide functions for a performance metric and map matching alignment error. The basic iteration steps of *Atlas* are:

1. Iterate the active local SLAM modules.
2. Update state of hypotheses.
3. Update graph using uncertainty projection and map matching .

2.3.1 Genesis

To acheive constant-time performance, *Atlas* limits the complexity of submaps, and the genesis operation creates new submaps when a submap reaches capacity and the robot leaves the current submap's region. Each submap will have a performance metric, which is high when the robot is near the current submap features, and low when the robot is not in proximity of the mapped features or the submap is constrained from adding new features. The performance metric quantifies how well a

map can explain the robot's sensor measurements. When the performance metric is too low with any existing submap, *Atlas* creates a new submap. Thus, submap creation is not directly dictated by spatial extent, but environment complexity and density of features. Other than at startup, submaps created through genesis are not empty. Recently observed features from the previous submap are propagated into new submap, so robot localization in the new submap will begin with lower initial uncertainty growth.

During genesis of a new submap, *Atlas* adds the submap as a vertex in its graph and an edge from the current submap to the new submap. The edge represents an approximate coordinate transformation between the two submap origins and the uncertainty in that transformation. The coordinate transform and uncertainty captured by the edge is used with map matching in cycle verification and loop-closure.

2.3.2 Traversal with Competing Hypotheses

As the robot leaves the area of the current submap, either a new submap must be created to explain the sensor measurements or the sensor measurements must be matched to an existing submap. *Atlas* conservatively defers this decision until is has enough information to form a conlusion. To do so, it allows instantiation of multiple hypotheses. Hypotheses have one of four states: *juvenile,mature,dominant,retired.* Retired hypothesis simply mark inactive submaps, but may be reactivated later as a juvenile. The dominant hypothesis represents the best submap based upon its performance metric. Juvenile hypothesis are instantiated from adjacent submaps to check for feasible traversal of edges. A juvenile hypothesis may become mature if its performance metric exceeds all other mature hypotheses after a short probationary period, 5-10 measurement steps is typically adequate. Otherwise a juvenile hypothesis is deleted. Of the mature hypotheses, the one with the highest performance metric becomes the dominant hypothesis.

When the robot leaves a submap with an adjacent submap, two hypotheses are instantiated. One hypothesis represents the genesis of a new submap, and the other a return the adjacent submap. The adjecent submap has a juvenile hypothesis. The

genesis hypothesis is mature, since it must begin mapping new features immediately. The submap that becomes the dominant hypothesis depends on the performance metrics of the current (dominant) submap, the genesis submap, and the adjacent submap. If the genesis submap's performance metric exceeds the dominant map's performance metric after the probationary period, it will become dominant and the previously dominant submap will remain active until it's metric falls low enough to cause retirement. Otherwise the genesis map will be pruned, and the juvenile hypothesis for the adjacent submap will mature and become dominant once it's metric exceeds that of the current submap.

2.3.3 Map Matching and Uncertainty Projection

Atlas finds the best estimate of relative coordinate transformations between submaps based upon a quantitative measure of alignment for a given transformation provided by the submap's SLAM implementation. Specifically for this thesis, this measure describes how well features of two maps align. Recall that submap genesis includes propagation of some features, and adjacent maps will share some features and cover overlapping regions.

The network of edges with their uncertainties in the *Atlas* graph is used in uncertainty projection for global optimization and loop detection and closure. The composite transformation and uncertainty of a sequence of edges determines candidates for map matching. When multiple paths exist, the path used will be the one with minimum uncertainty. When a possible loop is found through uncertainty projection, and the submaps at the loop ends sufficiently match, *Atlas* adds a new edge to the graph. Since the new edge is likely to have a large uncertainty, *Atlas* attempts to distribute some of the residual error among edges along the loop's path.

The uncertainty projection and map matching steps are rooted at the current submap. When a loop is detected and closed by a new edge, the effect on the graph is not considered until later. Other loop possiblities might be created by the new edge, but they won't be detected until the robot enters a submap within the new cycle. Deferring such computation keeps complexity bounded to maintain real-time

performance.

Uncertainty projection is also used when spawning a juvenile hypothesis for the robot moving into an adjacent submap. Although the submaps are independent, the initial uncertainty of the robot's pose is seeded from the composite projected uncertainty of the robot in the current submap with the edge between the two maps. The projected uncertainty provides a better initial guess for robot relocalization than using infinite uncertainty due to statistically independent submaps.

Chapter 3

Feature-relative Obstacle Avoidance and Navigation

In the previous chapter, the problems of localizing and building a map of the environment were addressed, and this chapter focuses on how to use that knowledge to command the robot. The first concern with control is the safety of the robot, to be described next. The next concern of this chapter is the design for incorporating feature-relative navigation.

3.1 Basic Obstacle Avoidance

3.1.1 Sensor Simulation

The basic principle for obstacle avoidance in this thesis is to use mapped and/or observed features to provide a richer image of the environment than could be currently observed. This idea takes some inspiration from the Vector Field Histogram (VFH) [1] for fast obstacle avoidance using sonar. In [1] the VFH reduced the occupancy grid-map to a score for each of n angular sectors around the robot. The VFH in [1] used an occupancy grid-map accumulating sonar returns over a period of time to account for the difficult nature of sonar. This thesis shows that a feature-based representation is permissible for navigation in a static environment. In this thesis,

the feature based map and observations are reduced to a minimum range for each sector, effectively producing a ray-tracer scan for the environment like a laser scanner that could *see* glass obstacles, be invariant of lighting conditions, etc. like sonar. The generated range image is then used for basic obstacle avoidance and exploration.

For robots with laser alone, this method is not very useful, unless an obstacle avoidance routine relied upon ranges to obstacles not within the angular bounds of the laser sensor. For robots with laser and sonar, this method can merge information about obstacles detected by the two sensors. For robots with sonar alone, this method is crucial to overcome the difficulties of sonar: partial observability of objects, angular uncertainty, and echos reflected away from the robot by objects. Addressing the first two difficulties allow the robot to react more appropriate to the environment with knowledge of obstacles that are not currently seen by the sonar. Sonar reflections many times are not returned to the sensor, making unsafe regions appear safe until the robot finally gets a normal reflection.

The choice of features to use for simulating scans is very important. The implementation used in this thesis is configurable to use observed features and/or mapped features for sonar and/or laser features. Observed laser features provide a limited view of the enviroment, but are not subject to errors from the SLAM module. For sonar sensing, the mapped features provide the required ranging information with least uncertainty. Observed sonar features may be spurious and slopes of observed lines from the same object have significant variation due estimation model for a wide beam sonar.

3.1.2 Scan Generation

A scan generator module provides a laser-like range scan image for use with basic obstacle avoidance and exploration. Features are transformed into the robot's coordinate frame prior to calculation of the range values for each sector. For line features, each sector is checked if a ray from the robot cast with the sector's angle intersects the line. Of the intersections found, the one with minimum range determines the sector's range value. Point features do not require an intersection calculation, since

their angular extent is theoretically zero. Since these features typically denote an object with volume, they are treated as arcs with a maximum length of 20cm and maximum span of 15 degrees.

3.1.3 Shield

Collision avoidance is implemented with the *Shield* module. This module processes range scan images, such as the generated scans, to determine to determine whether the robot is in danger of an impending collision, and will either slow or stop the robot if it is too close to an obstacle, and will issue steering controls to avoid the obstacle. A benefit of this obstacle avoidance method is that it allows for easy integration of feature-relative navigation modules because the module is passive when the robot is not close enough to an obstacle to require avoidance. By using a scan generated from features, the task of coordinating control between basic obstacle avoidance and feature-relative control is much simpler, since the same features are being used.

Two elliptical shields protect the robot and are configured with half-lengths for their principle axes. The major semiaxis of each ellipse is centered on the robot's current heading. The boundaries of two ellipses govern the classification of range scan points into three zones: danger, caution, and safe. The safe zone includes anything outside both ellipses, and the module ignores this zone. The danger zone includes all points within the danger ellipse, and the caution zone include includes all points within the caution ellipse. Although not a requirement, the caution ellipse is typically configured to fully enclose the danger ellipse. Figure 3-1 shows a typical configuration for the two zones.

After processing a range scan to find the points in the caution and danger zones, the *Shield* module calculates controls as follows. When no caution or danger points are found, no action is taken. Priority is given to the danger zone points, and if any danger points are found, controls are calculated based upon those points. When danger points are found, the robot's translational velocity is set to zero. When caution points but no danger points are found, the robot's translational velocity is slowed to a configurable creeping velocity.

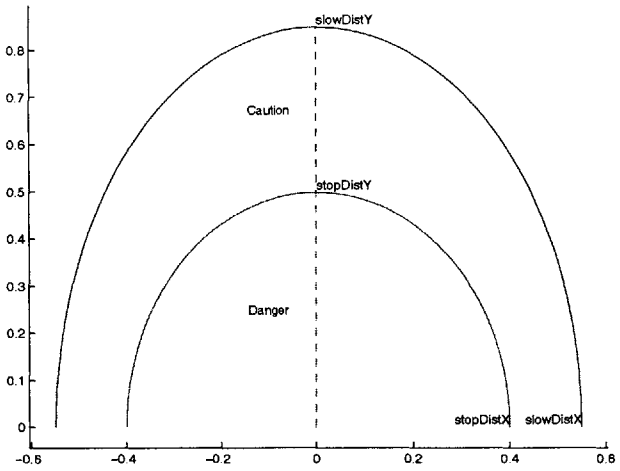

Figure 3-1: The caution and danger zones used for obstacle avoidance.

To steer the robot away from any obstacle, *Shield* uses the minimum range measurement to determine the direction it should steer. It also compares the number of points on its left and right sides and if this difference is larger than 2 and conflicts with the direction of the minimum range measurement, the robot is steered in the other direction. The magnitude of rotational velocity is typically configured to be larger for avoiding points in the danger zone and smaller when avoiding caution points.

3.1.4 Simple Exploration

In the absence of more intelligent navigation control, a default exploration method is used to keep the robot moving in the environment when the *Shield* is inactive. This exploration method attempts to move the robot up to a maximum speed and steering slightly to avoid aiming the robot directly at obstacles. The method uses a range scan to find points within a small view angle ahead of the robot that are not too far away. When no points are found in the region, then the robot's rotational velocity is set to zero. The steer controls are calculated from these points by taking the ratio of mean distances between the points on the left and right. When no points are found in one of the sides, the robot steers in that direction.

The relevant configuration parameters for this method are for the robot's maxi-

mum translational velocity, the distance it should look ahead, and angular span of the view ahead it should look.

The controls from the simple exploration process are smoothed to prevent jerking. Otherwise the robot may appear jerky when restarting exploration after a manual (remote control) override or when the robot's prior velocity was small because the *Shield* module just relinquished control.

3.2 Feature-based control

This section describes a design for executing feature-relative navigation. Coordination of feature-relative control with basic obstacle avoidance follows a subsumption architecture [5]. Two basic behavior types were designed for use with the two feature types. A *Tether* controller operates relative to point features, and the goal behavior is to approach and orbit the point feature. A *Follow* controller operates relative to line features, and implements wall following.

The algorithm used for control would follow these basic steps:

1. Score features.

2. Initialize an appropriate controller for feature with maximum valid score.

3. Run controller until done, unsafe, or better scoring feature exists.

Scoring the features effectively determines the feature path the robot will follow. Scores are updated each iteration. The scoring function ranks nearby and visible features higher than features that are distant or not visible. The scoring function uses a large coefficient for visibility and a smaller coefficient for distance squared.

Transistions between features occur for three reasons. If the robot encounters an obstacle in its path, then the controller is aborted by setting its score to an invalid value. The second reason for finding a new feature is when the controller for the current feature has reached a *done* status. The *Tether* controller is considered done when the robot has orbited the feature The *Follow* controller is done when the line

feature can no longer be followed. The third reason for a transition to another feature is when a better scoring feature is found. The *Tether* and *Follow* controllers both decay the score of their current feature. Once the controller has approached the feature, the score for that feature will decay linearly to zero based upon the percent coverage.

Chapter 4

Infrastructure

This chapter describes the software platform used for development and experimentation. The software driver written for the B21 robot is also described here.

4.1 Software Platform

The software platform for implementing this research is **MOOS** : Mission Oriented Operating Suite [11]. This system defines the structure and interface for implementation all the real-time software used in this research. This system allows for modular development of software in C++ for multiple types of robots, data logging, wireless communication and remote control (manual or automated) via TCP/IP, and simulation. Currently this system provides the support and interface to Autonomous Underwater Vehicles AUVs, autonomous kayaks, ER1 land robots, and B21 land-robots. This system has been used in the experiments in [10], [4], and [2]. .

The **MOOS** system consists of a distrubuted set of processes, and communication follows a *star* network model. The central process acting as the short-term memory database is the **MOOSDB**. All inter-process communication must occur via the **MOOSDB**. Drivers and processes publish data under a name, or key, to the **MOOSDB** where other processes may access the data via a subscription to that key. The **MOOSDB** is the server for all communications and other processes are clients. Unlike the client processes, the **MOOSDB** does all processing in its com-

munications thread asynchronously via the `select` command. Each **MOOS** process uses at least two threads: one for communication and one for processing. The communication thread manages incoming and outgoing messages. The communication thread passes new incoming messages to the processing thread, and the processing thread passes outgoing messages the communication thread. Each thread is given a fundamental frequency. This frequency should be chosen carefully since **MOOS** uses a data polling model. The frequencies should be set a rate high enough to prevent latency in real-time control.

A logger process recorded data for the experiments. The logger **pLogger2** registered for the keys of relevant data, and wrote their values to a log file. **pLogger2** polled the **MOOSDB** for data at 40Hz, a rate much higher than the frequency of any sensor or process producing data to log. This logger is capable of producing synchronous logs and asynchrounous logs.

4.2 B21 Software Driver

This thesis used a B21 robot for all experiments with sonar, and a new driver **irFlex** was written to interface the hardware. During startup of the robot, **irFlex** reads a configuration file for parameter settings. The relevant parameters discussed here are for the robot's radius and the minimum sonar period.

As a **MOOS** application process, **irFlex** has both a processing frequency and communications frequency. Since **irFlex** simply reformats messages between the hardware and other processes, the communications and processing frequencies are identical. The frequency chosen is chosen to at least prevent buffering of outgoing messages, so other processes see little latency between the timestamp and receipt of data. With a minimum sonar period configured to .180s and odometry data period of roughly .035s, the frequency was set to 30Hz. This frequency is higher than the frequency for the process performing SLAM and obstacle avoidance **pROAM**.

4.2.1 Odometry

The B21 robot's hardware provided odometry is reported as a cumulative distance traversed for translational and angular motion. Translational motion is simply the net distance the wheels have traveled forward, and rotational distance is an unwrapped angular value. To put the robot's odometry into a coordinate frame such that the translational movement will depend on the heading state, the linear and angular values the B21 provides are processed in the following manner.

When the system starts or when the driver receives a request to reset the odometry, the odometry state is reset to zero. The linear and angular values reported by the hardware are corrected to be in units of meters and radians and used to determine differential motion for the robot's odometry. The difference between the linear and angular values reported and the previously reported values $(\Delta\rho, \Delta\phi)$ are converted to the state change $[\Delta x \ \ \Delta y \ \ \Delta\theta]$ using the current heading state and assuming that a heading of zero has the robot facing the $+y$ direction.

$$\Delta x = \Delta\rho\,(-\sin\theta) \tag{4.1}$$

$$\Delta y = \Delta\rho\,(\cos\theta) \tag{4.2}$$

$$\Delta\theta = \Delta\phi \tag{4.3}$$

The odometry is updated with the differential change and the heading wrapped before being published.

4.2.2 Sonar Data

The B21 robot has a ring of 24 polaroid ultrasonic sensors, each having a beam width of 30 degrees. Updates of sonar range data occur asynchronously for each of the B21's six panels of four sensors. The typical period between updates of two of the panels was measured at 33ms, and occasional checksum errors for a panel resulted in doubling this period. Sensors that do not detect a return report a maximum range

value, roughly 22.664 meters.

The interface for sonar range data provided by the **irFlex** driver provides sonar data for all 24 sensors and does not report data for each of the panels independently. The range data for each sensor is held in an array, and another array holds the timestamps for that range reading. When the newest timestamp exceeds the timestamp of the last published set of ranges, all ranges are published. The ranges not updated within this period are set to the maximum range value prior to being published to prevent subscribing processes from double counting measurements.

irFlex also publishes the geometry of the sonar sensors on the B21 robot at startup and whenever it receives a request from another process. The geometry of the sensors is based upon the robot radius and the mean angular difference between sensors. Although the name of the B21 robot refers to a diameter of 21 inches, this is the diameter of the robot's base. The top portion of the robot with the ring of 24 sonars has a slightly smaller diameter, measured at 19.5 inches. Each sensor is given an index from 0 to 23, beginning at the sensor just to right of the robot's heading and going clockwise. The angular separation of sensors is approximated at $2\pi/24$ with the initial angular offset being half the separation. Figure 4-1 shows the robot's sensor geometry and its six panels.

On the particular B21 robot used during experimentation, the sonars were not all working. Several of sonar sensors would fail on occasion and only report a maximum range value. The sonar at index 11 as shown in 4-1 was definitely broken. These problems were not critical; the other sonars provided more than sufficient data to conduct the experiments.

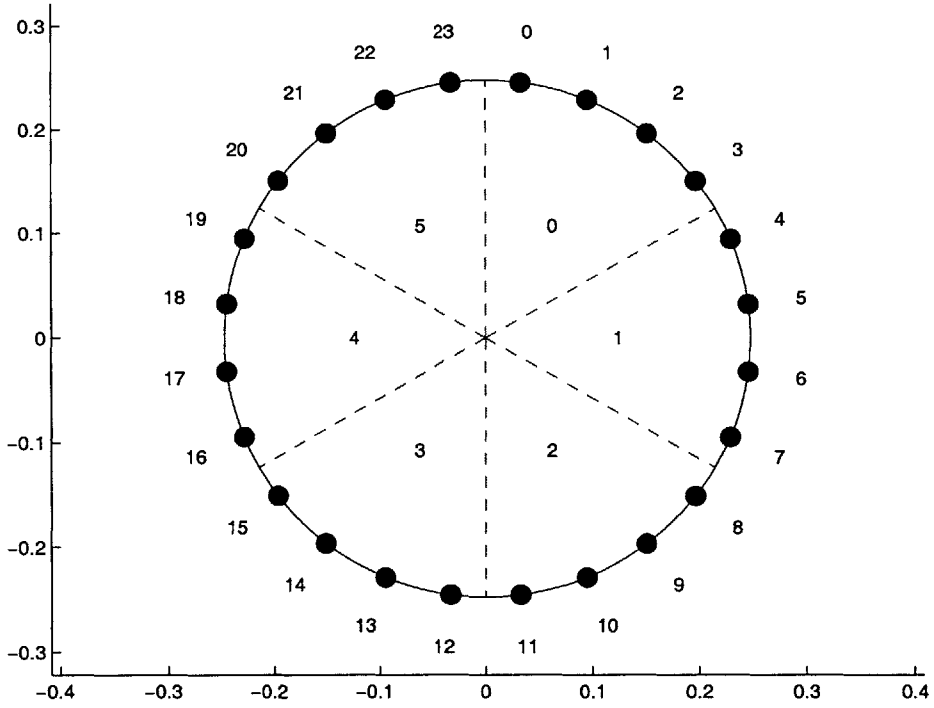

Figure 4-1: The sonar sensor geometry for the B21 robot when facing in the $+y$ direction. The inner numbers and red lines denote the six panels, and the outer numbers and black circles represent the sonar sensors.

Chapter 5

Results

5.1 Tank Experiments

A series of experiments using a B21 mobile robot were conducted in a controlled environment testing the mapping and obstacle avoidance using only sonar sensors. The robot was placed in large drained tank, approximately 13 x 30 feet. To provide point features for obstacle avoidance, cylindrical PVC pipes of varying diameter, an elliptical metallic pipe (semiaxis radii are approximately 10cm and 5cm), and broomsticks were placed around the tank. The placement of objects for point features varied between experiments to provide slightly different environments for testing. Figure 5-1 shows the experimental setup with one configuration of obstacles.

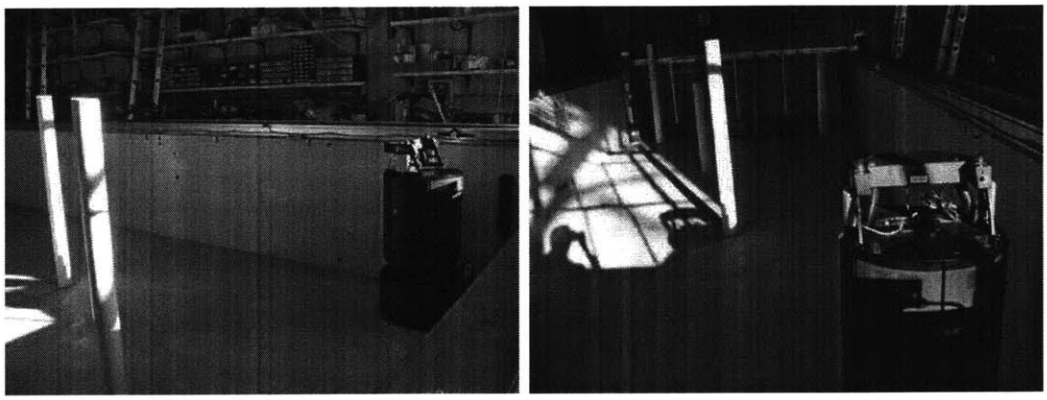

Figure 5-1: View of a sample experimental setup.

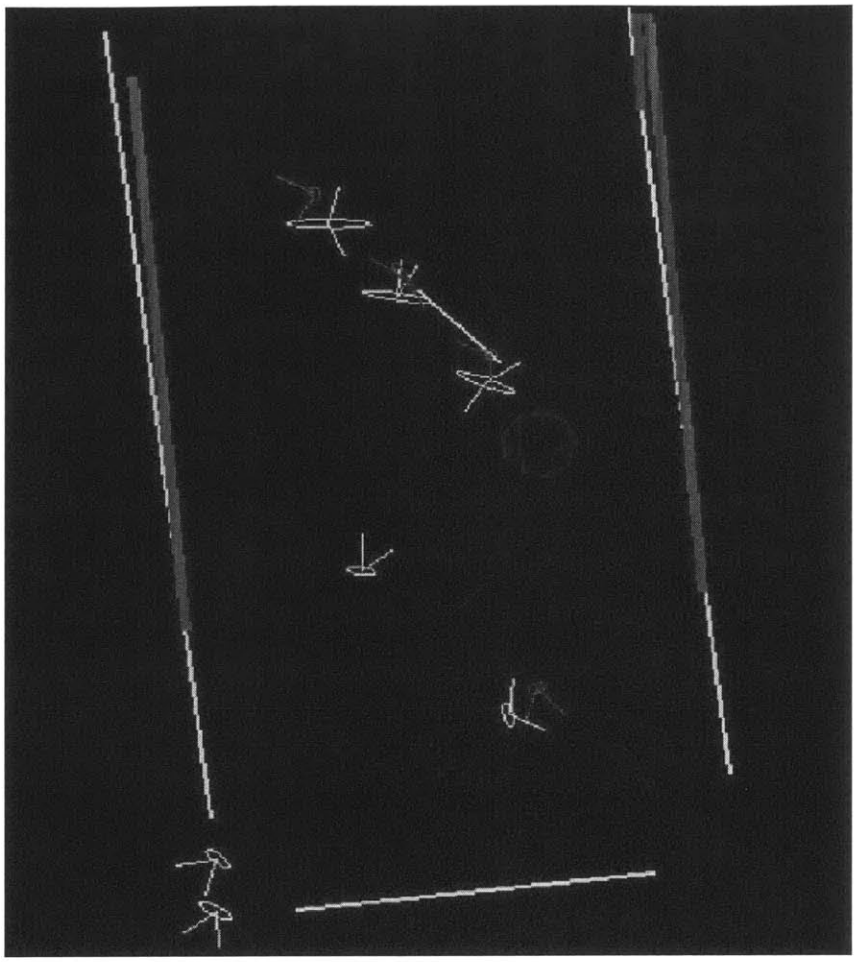

Figure 5-2: Mapped and observed features during an experiment.

Figure 5-2 shows a screenshot of the robot, the mapped features, and the observed features. In this experiment the robot demonstrated reliable obstacle avoidance based solely upon mapped sonar features. The maximum translational velocity was .3m/s, but typical velocities were around .2m/s since the environment was cluttered. The experiment was terminated after approximately 560 seconds.

In another experiment using a different configuration of obstacles, the robot exhibited successful obstacle avoidance. The duration was approximately 480 seconds. The following figures show screenshots of the mapping process working as the robot was avoiding obstacles. The robot is shown in blue, mapped features in light blue, observed features in red, and vantage points (previous poses) are shown in purple.

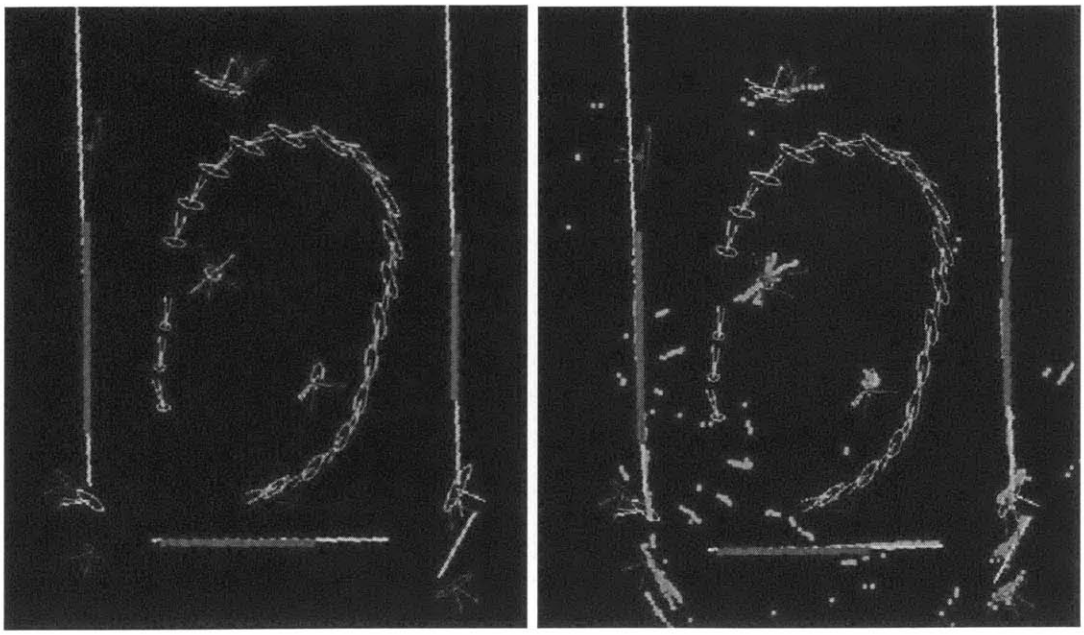

Figure 5-3: Mapped and observed features from multiple vantage points.

Green dots show raw sonar data for the last 1000 valid echos (less than the maximum range value).

Figure 5-3 shows how the robot's map compares to the currently observed features. The figure on the right shows how the raw sonar data for comparison. Figure 5-4 shows again the same instant in time. Arcs are shown in this figure for the sonar returns found in the RANSAC data association engine for sonar returns at vantage points that have at least one adjacent return. The small clusters of spurious sonar data in front of the robot with no associated features (or real object). Such clusters would not be rejected by grid-based approaches to obstacle avoidance, and would incorrectly affect behavior.

Figures 5-3 and 5-4 both show a substantial 'gap' in the robot's track, as seen from the covariance ellipses for the saved vantage points. The reason for this 'gap' is due to a temporary hardware failure. For unknown reason, the driver for the B21 robot will have temporary periods of time where the odometry values reported by

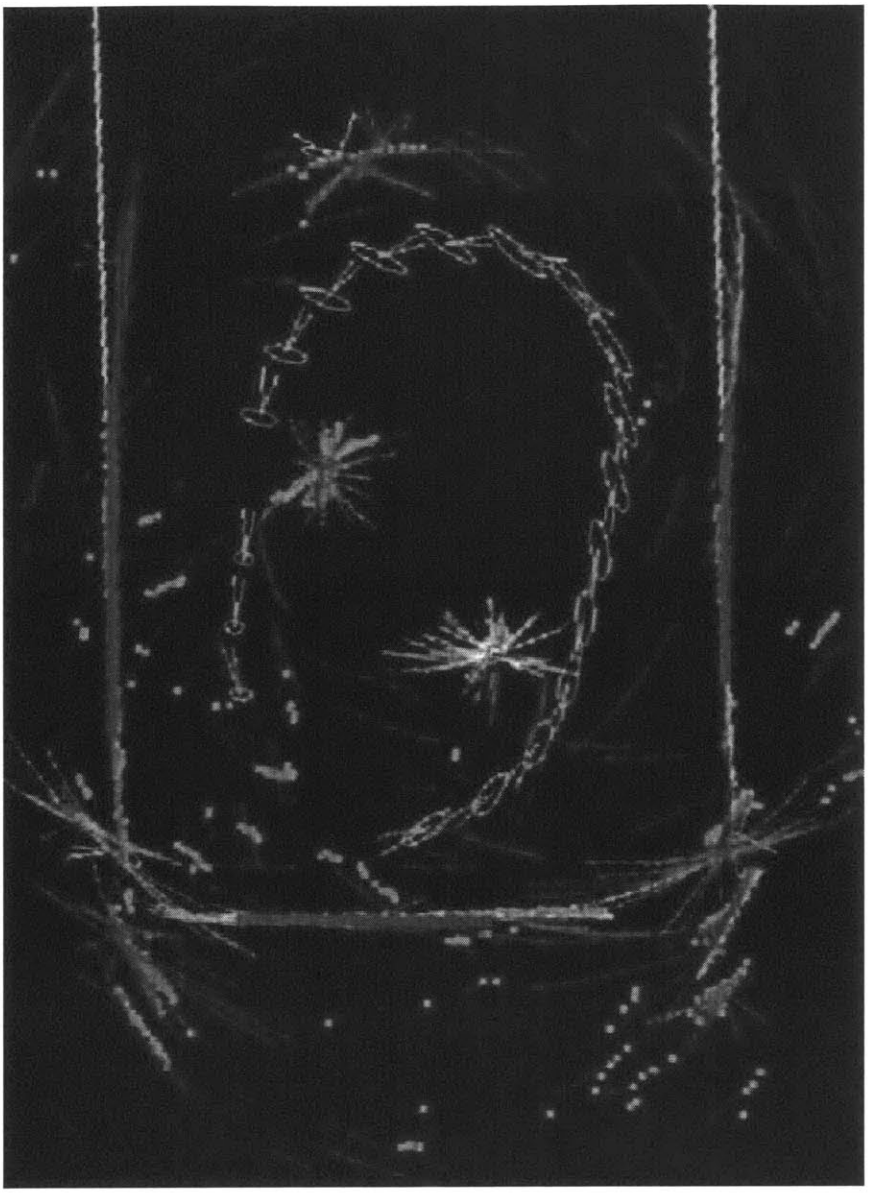

Figure 5-4: A snapshot of ROAM.

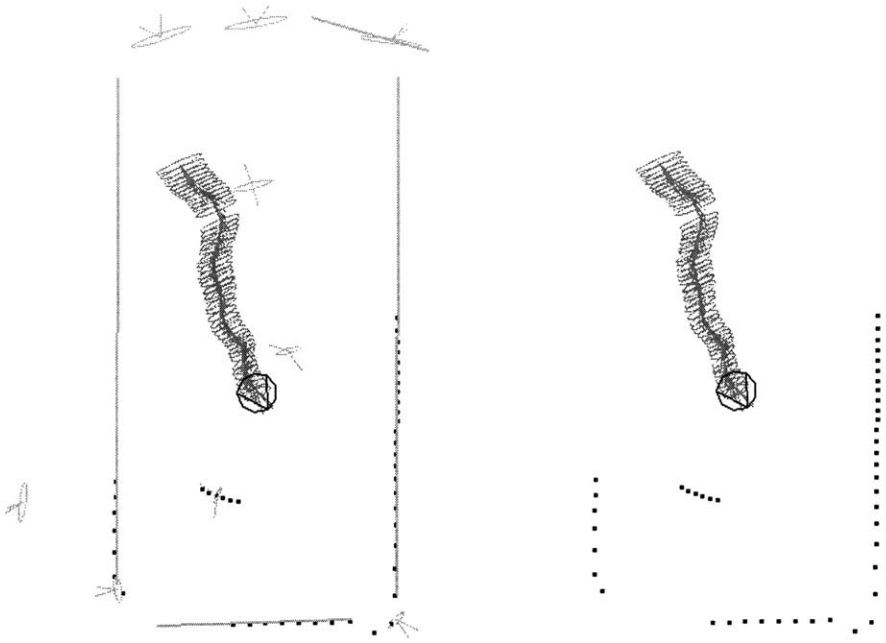

Figure 5-5: ROAM action snapshot at time 1.

the hardware fail their Cyclic Redundancy Check (CRC), so odometry data is absent for a brief period of time. The end of the gap in the figures represents the robot relocalizing once odometry data was again available.

The sequence of figures 5-5, 5-6, 5-7, and 5-8 show a sequence of screenshots taken during an experiment. The trail of covariance ellipses following the robot show both the localized uncertainty of the robot's pose as well as a short path history. The screenshots on the right hide the map to show the simulated range scan sensor used for obstacle avoidance.

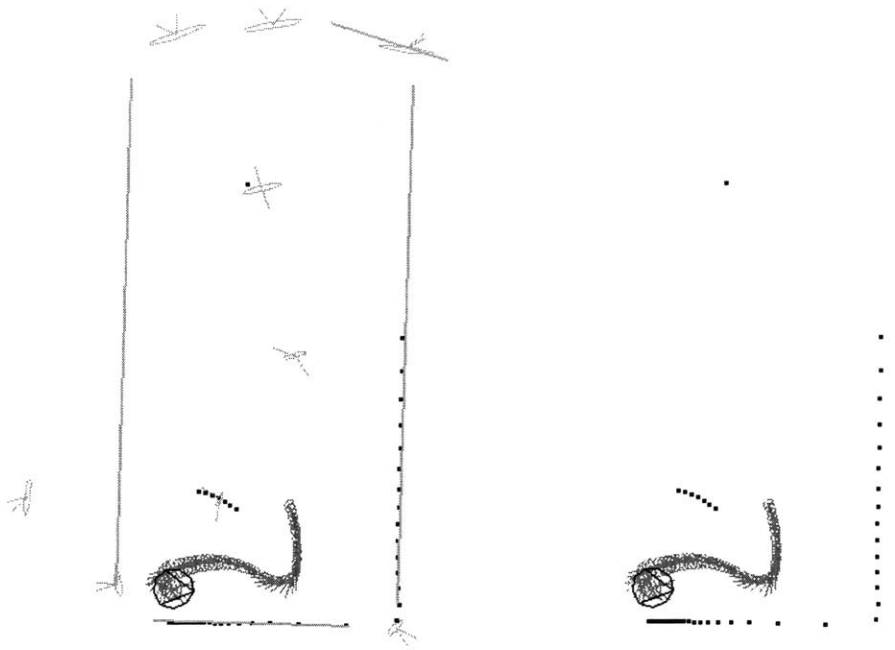

Figure 5-6: ROAM action snapshot at time 2

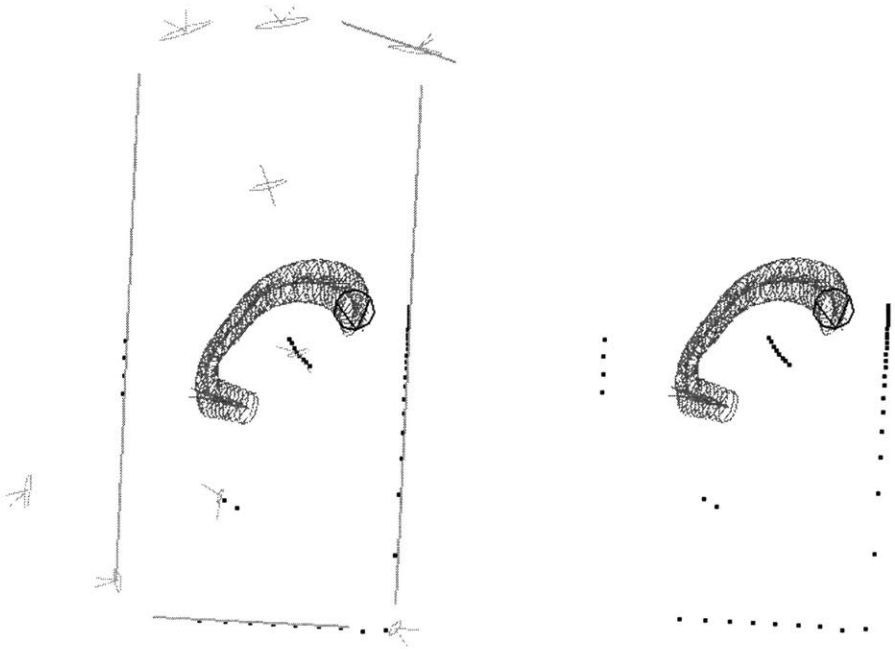

Figure 5-7: ROAM action snapshot at time 3

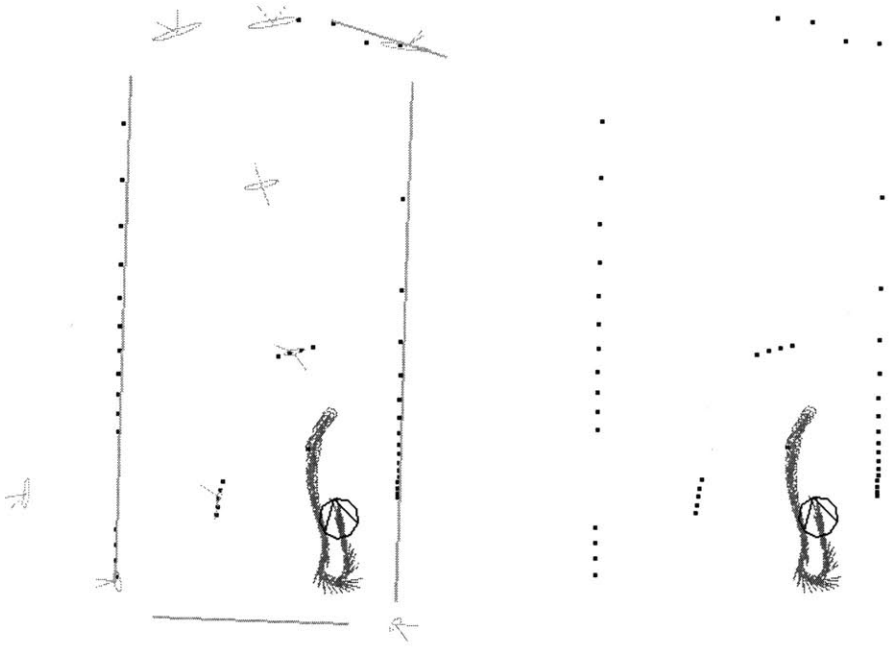

Figure 5-8: ROAM action snapshot at time 4

Chapter 6

Conclusion

6.1 Summary

This thesis presented an implementation of feature-relative Real-time Obstacle Avoidance and Mapping and demonstrated operation using wide-beam sonar sensors. This thesis integerated previous research by Bosse [3] for mapping in large scale, cyclic environments and a method for obstacle avoidance that simulates a high angular resolution range sensor using mapped features.

6.2 Future Work

This thesis presented a simple implementation of ROAM. Several key issues are described in this section that could improve performance.

6.2.1 Environment Model

A substantial limitation of the implementation in this thesis is the representation of the environment with only static lines and points. A richer set of feature types could model more environments with better accuracy, and feature-based obstacle avoidance could be applied reliably in more environments.

Expansion of the current sonar feature set to include lines, cylinders, edges, and

corners could be accomplished with the models described in [7]. The measurement models are similar in form to those used in this thesis, only slightly more complex. Using these features could help optimize a robot for indoor enviroments.

One possibility for further research would be to incorporate a feature extractor using Trajectory Sonar Perception [13],[14]. Trajectory Sonar Perception allows representation of objects with any constant radius of curvature. Line objects would have an infinite radius of curvature, and point objects would have zero radisu of curvature, but most 2-D point objects are actually cylinders with a small radius. The curvature parameter would become another feature parameter requiring estimation and a Kalman state.

Another limitating factor of the implementation in this thesis is that the obstacle avoidance assumes a static environment, since the sonar feature-based obstacle avoidance relies upon mapped features. One extension for this implementation is to add mobile object tracking, and add those objects to the mapped features prior to generating scans.

6.2.2 Path Planning

The basic obstacle avoidance and exploration used in the experiments was rather crude. Incorporation of other algorithms for more intelligent exploration need to be explored. At the time of this thesis, several partial implementations for feature relative navigation behaviors remained to be finished and integrated.

One possible extension for smoother control of the robot would be to incorporate a Curvature-Velocity Method (CVM) [15],[6], [12]. Using CVM could replace the existing obstacle avoidance and exploration by controlling the robot's trajectory based upon time to collision with obstacles.

6.2.3 Data Latency

The **MOOS** platform provides a very convenient and modular environment for developing and testing robot software, as **pROAM** and **irFlex** were developed inde-

pendently. For most purposes, **MOOS**'s limitations do not affect performance. The most relevant limitation for the research in this thesis is the communication latency between the hardware and the obstacle avoidance and mapping process. At higher velocities it may become important to have combine **irFlex** and **pROAM** into a single process so odometry and sensor data are processed immediately. This single process should still have a **MOOS** interface for purposes of remote control/manual override and data logging.

Appendix A

2-D Coordinate Transforms

Object poses in two dimensions may be represented with a vector of three dimensions containing the Cartesion coordinates and the heading relative to some base.

$$\mathbf{x}_{\mathbf{base}}^{\mathbf{object}} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{A.1}$$

The operators for inversion $\ominus$ and composition $\oplus$ provide the basis the transformations for conversion between coordinate frames. For an object $\mathbf{A}$ in the coordinate base $\mathbf{B}$, $\mathbf{x}_{\mathbf{B}}^{\mathbf{A}}$ defines the location. The inversion operator provides the relative location of the base from the object $\mathbf{x}_{\mathbf{A}}^{\mathbf{B}}$:

$$\mathbf{x}_{\mathbf{A}}^{\mathbf{B}} = \ominus \mathbf{x}_{\mathbf{B}}^{\mathbf{A}} \tag{A.2}$$

$$= \begin{bmatrix} -\cos\theta_{ba} & -\sin\theta_{ba} & 0 \\ \sin\theta_{ba} & -\cos\theta_{ba} & 0 \\ 0 & 0 & -1 \end{bmatrix} \mathbf{x}_{\mathbf{B}}^{\mathbf{A}} \tag{A.3}$$

$$= \begin{bmatrix} -x_{ba}\cos\theta_{ba} - y_{ba}\sin\theta_{ba} \\ x_{ba}\sin\theta_{ba} + y_{ba}\cos\theta_{ba} \\ -\theta_{ba} \end{bmatrix} \tag{A.4}$$

Given two objects A and C, the relative pose of A in base B, and the relative pose of C to A, composition provides the relative pose of C to A :

$$\mathbf{x}_{\mathbf{B}}^{\mathbf{C}} = \mathbf{x}_{\mathbf{B}}^{\mathbf{A}} \oplus \mathbf{x}_{\mathbf{A}}^{\mathbf{C}} \tag{A.5}$$

$$= \mathbf{x}_{\mathbf{B}}^{\mathbf{A}} + \begin{bmatrix} \cos\theta_{ba} & -\sin\theta_{ba} & 0 \\ \sin\theta_{ba} & \cos\theta_{ba} & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{\mathbf{A}}^{\mathbf{C}} \tag{A.6}$$

$$= \begin{bmatrix} x_{ba} + x_{ac}\cos\theta_{ba} - y_{ac}\sin\theta_{ba} \\ y_{ba} + x_{ac}\sin\theta_{ba} + y_{ac}\cos\theta_{ba} \\ \theta_{ba} + \theta_{ac} \end{bmatrix} \tag{A.7}$$

These two basic operators provide the basis for other transformations of relative poses between bases. Given the relative pose of A from B and the pose of A from C, the pose of C from B is as follows:

$$\mathbf{x}_{\mathbf{B}}^{\mathbf{C}} = \mathbf{x}_{\mathbf{B}}^{\mathbf{A}} \oplus \left(\ominus \mathbf{x}_{\mathbf{C}}^{\mathbf{A}}\right) \tag{A.8}$$

For uncertain poses, the operators have associated Jacobians for transforming the covariances. There are two Jacobians for composition, the first with respect to the first argument pose and the second with respect to the second argument pose.

$$\mathbf{J}_{\ominus}(\mathbf{x}_{\mathbf{B}}^{\mathbf{A}}) = \begin{bmatrix} -\cos\theta_{ba} & -\sin\theta_{ba} & (x_{ba}\sin\theta_{ba} - y_{ba}\cos\theta_{ba}) \\ \sin\theta_{ba} & -\cos\theta_{ba} & (x_{ba}\cos\theta_{ba} + y_{ba}\sin\theta_{ba}) \\ 0 & 0 & -1 \end{bmatrix} \tag{A.9}$$

$$\mathbf{J}_{1\oplus}(\mathbf{x}_{\mathbf{B}}^{\mathbf{A}}, \mathbf{x}_{\mathbf{A}}^{\mathbf{C}}) = \begin{bmatrix} 1 & 0 & -(x_{ac}\sin\theta_{ba} + y_{ac}\cos\theta_{ba}) \\ 0 & 1 & (x_{ac}\cos\theta_{ba} - y_{ac}\sin\theta_{ba}) \\ 0 & 0 & 1 \end{bmatrix} \tag{A.10}$$

$$\mathbf{J}_{2\oplus}(\mathbf{x}_{\mathbf{B}}^{\mathbf{A}}, \mathbf{x}_{\mathbf{A}}^{\mathbf{C}}) = \begin{bmatrix} \cos\theta_{ba} & -\sin\theta_{ba} & 0 \\ \sin\theta_{ba} & \cos\theta_{ba} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.11}$$

Bibliography

[1] J. Borenstein and Y. Koren. The vector field histogram – fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, June 1991.

[2] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[3] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. *Int. J. of Robot. Res.*, 2004.

[4] M. Bosse, P. Newman, J. Leonard, and S. Teller. An atlas framework for scalable mapping. Technical Report Marine Robotics Laboratory Technical Memorandum 02-04, Massachusetts Institute of Technology, 2002.

[5] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, March 1986.

[6] D. Castro, U. Nunes, and A. Ruano. Reactive local navigation. In *Proceedings of the 28th Annual Conference of the IEEE Industrial Electronics Society*, Sevilla, Spain, Nov 2002.

[7] J. J. Leonard and H. F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation.* Boston: Kluwer Academic Publishers, 1992.

[8] J. J. Leonard, R. J. Rikoski, P. M. Newman, and M. Bosse. Mapping partially observable features from multiple uncertain vantage points. *Int. J. Robotics Research*, 2002. To Appear.

[9] J. H. Lim and J. J. Leonard. Mobile robot relocation from echolocation constraints. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(9):1035–1041, September 2000.

[10] P. Newman, , M. Bosse, and J. Leonard. Autonomous feature-based exploration. In *Proc. IEEE Int. Conf. Robotics and Automation*, 2003.

[11] P. Newman. *MOOS - Mission Oriented Operating Suite.* MIT, 2003.

[12] B. Quasny, T. Pyeatt, and J. Moore. Curvature-velocity method for differentially steered robots. In *Proceedings of the IASTED International Conference on Modelling, Identification, and Control*, Innsbruck, Austria, Feb 2003.

[13] R. Rikoski and J. Leonard. Sonar trajectory perception. Technical Report Marine Robotics Laboratory Technical Memorandum 02-05, Massachusetts Institute of Technology, 2002.

[14] R. Rikoski and J. Leonard. Sonar trajectory perception. In *Proc. IEEE Int. Conf. Robotics and Automation*, Taiwan, 2003.

[15] Reid Simmons. The curvature-velocity method for local obstacle avoidance. In *International Conference on Robotics and Automation*, April 1996.

[16] J.D. Tardós, J. Neira, P.M. Newman, and J.J. Leonard. Robust mapping and localization in indoor environments using sonar data. *Int. J. Robotics Research*, 21(4):311–330, April 2002.