2D Mapping of Expansive Unstructured Areas

Gary Shaffer

CMU-RI-TR-95-23

2D Mapping

of

Expansive Unstructured Areas

Gary Shaffer
The Robotics Institute
Carnegie Mellon University

1 Introduction

The fundamental quantities of mechanics are usually taken to be mass, length, and time [Sears82]. Given the ability to measure length, in this case the distance between a reference point on a robot and its surroundings, this thesis describes a method for combining distance measurements into a 2D model of a robot's workspace.

1.1 Motivation

There are many different kinds of maps. The term "map" would probably mean a road map to a truck driver, a treasure map to a pirate, a mine survey to a coal miner, navigation charts to a sailor or aviator, or a layout of Disneyworld to a child. A map is really just a visual representation of information. A road map provides topological and metric information about the graph of roads that connect selected locations. A treasure map encodes instructions for returning to a secret location. Mine surveys record the geometry of a developing underground mine. Navigation charts encode routes for safe travel through previously explored areas. Amusement park maps allow one to plan the day to minimize time or energy expended between rides or ice cream breaks. In all of these examples, a map provides information which enables a task or greatly increases the efficiency of performing a task.

Constructing a map requires a considerable investment of resources, whether it is equipment or man-hours or both. Efficiency is attained by many people benefitting from the arduous exploration performed by the map-builder. Later traversals of the same area can be conducted more quickly because there is knowledge of what lies ahead. There is no more need for exploration, only reaction to conditions not represented in the map. In general terms, there are three reasons for building a map:

1. so you can figure out where you are (pose estimation, navigation)
2. so you know where you've been (exploration, database)
3. so you know what lies ahead (planning)

In field robotics, there is a need for maps. In some workspaces, the appropriate maps exist or can be easily created by a person. However there are many workspaces for which maps do not exist and map creation is not humanly possible or practical. In the extreme, there are environments (for example nuclear) in which biological organisms cannot function or survive, but that pose no threat to an electromechanical device. There are also environments (such as an underground mine) which allow human presence but at the price of health and constant risk of injury. In both of these environments, a robotic mapper is appropriate.

There are more motivations for automatic map-building which are productivity based. Sometimes the efficiency gain from automatically building a map is so great that it enables a task that was not possible before. For example in underground mining, with the technology presented in this dissertation, a detailed model of the immediate workspace can be constructed automatically in less than a minute. This ability to quickly create a detailed model would enable automatic planning of min-

ing operations. And the ability to quickly estimate the pose of the mine equipment relative to the detailed model would allow autonomous execution of the automatically-generated plan. A map of nearly the same quality could be constructed by manually surveying, but it would probably cost a (highly-skilled) man-hour, making it impractical.

1.2 Problem Statement

The problem is for a robot to completely and precisely model the bounds of its workspace with infrequent or no external reference. The model must be complete, detailed, compact, and concise to support navigation and planning.

A complete map models the entire boundary of the robot's workspace, not just key features. A complete map is necessary for planning a collision-free trajectory. The map must be detailed enough so that small features in the environment are modelled, rather than fit to part of a long line segment. Compactness is desirable for practical reasons. To make a complete and detailed map of a large area, a lot of data is required. Conciseness refers to merging several measurements of the same area into one representation.

It is assumed that the robot's world can be sufficiently modelled as 2D (flat floor, vertical walls) and that external position reference is not available at all locations. The second assumption may seem overly restrictive, but there are situations where it applies. For example an underground mine is an environment where GPS is not possible and emplacing (and maintaining) numerous surveyed beacons is impractical. A completely self-contained mapping capability is necessary to enable autonomous operation between sparse or infrequent external updates.

1.3 Related Research

Automatic map-building requires the combination of two capabilities: sensor pose estimation and map update from sensor data. There are numerous methods for sensor pose estimation, but truly self-contained, automatic map-building requires self-reliant pose estimation. In other words, the robot must match data acquired from its current pose to data stored from previous sensor poses (in the form of a "map") to determine its position. Much research has been done in matching sensor data to a map to determine sensor pose, though in most cases the map is measured beforehand with a measuring tape or more accurate means. Sensors used are almost exclusively sonar range sensors, laser range sensors, or cameras. Environments are usually indoor labs or hallways. Dead reckoning is often employed as a starting point for an error minimization search or as an additional sensor input to be "fused" into the final pose estimate.

Mapping requires combining measurements acquired from many sensor poses into a useful representation. Two subproblems are 1) estimation of the sensor pose and 2) incrementally constructing a model of the world from raw sensor data. Many researchers have addressed these subproblems individually, but few address both subproblems concurrently. Depending on the characteristics of a robot's workspace, there are many possible methods for pose estimation ranging from dead reck-

oning to satellite-based positioning. We assume a worst-case scenario in which externally-referenced pose estimation is unavailable. For example if the robot is operating indoors or underground, satellite-based positioning is impossible. In an environment such as an underground mine it is impractical or impossible to emplace and maintain surveyed landmarks or beacons. The remaining alternative is relative pose estimation which can be accomplished by a variety of methods including dead reckoning, inertial systems (accelerometers), and map matching. I make no assumptions about the form of the environment beyond the 2D assumption.

Most mobile robots perform dead reckoning, usually by measuring position deltas on the locomotion actuators whether they be wheels, legs, or tracks. Dead reckoning alone is usually not accurate enough because error accumulates quickly and often unpredictably (if a wheel slips for example). An inertial system measures accelerations (both angular and linear) and integrates to determine position. Depending on the application, an inertial system can work very well, but a good inertial system is expensive. Dead reckoning is cheap and easy enough (both financially and computationally) that most mobile robotic systems utilize it in some way. Top performance inertial systems are prohibitively expensive in most cases, though the prices are dropping. Cost and performance requirements necessitate a relative pose estimation capability beyond dead reckoning and inertial systems, a method which senses the robot's surroundings to determine motion.

Comparing two views of the surroundings from a robot-mounted sensor is often referred to as "map-matching", or "motion estimation". These are general terms; the specifics depend greatly upon the sensor modality that is chosen. The choice of sensor depends on economy, safety, field of view, accuracy, speed, etc. (not necessarily in that order). Many researchers have worked with a ring of ultrasonic range sensors, because they're relatively inexpensive, safe in most environments and can provide a 360° field of view in a reasonable amount of time. The trick is to get the accuracy, which a few researchers have done really well [Leonard92] [Moravec85]. Others choose to work with stereo cameras, which are still reasonably inexpensive, are safe in all environments, and provide high speed and resolution, but a limited field of view. The most crippling problem is typically a large amount of processing is needed. Depending on the application, shortcuts can be used to make stereo practical [Kelly95]. With the ever-increasing performance of computer hardware, the computation time issue may go away soon. On the high end of sensor cost are laser range scanners. However, if the price is not a barrier, the laser modality wins in almost all the other performance measures. Lasers can quickly provide direct range measurements in a 360 degree field of view with large maximum range compared to cameras and sonars. The beam is very narrow compared to ultrasonic.

Spatial evidence grids are a means of combining overlapping range measurements into a map. Moravec [Moravec85] and others [Elfes87] [Kuc87] [Leonard91] have characterized the operation of various ultrasonic range transducers. Moravec and Blackwell have developed a method of self-calibrating by "learning" the sensor response for different reflecting materials. The training results in a model of the sensor's response to the reflecting material,-a probability distribution grid which assigns an occupancy probability to grid cells in the field of view of the sensor. Correlation algo-

rithms for occupancy grids can be used to match a set of range readings to the map to estimate the robot's pose. The map is updated using Bayesian techniques to combine measurements with the map. Grid maps make no assumptions about the structure of the world. A map of increasing accuracy results from combining many measurements which contain small amounts of information. Drumheller [Drumheller87] developed an absolute localization system that used a small number of ultrasonic range measurements to determine a robots position in a presurveyed map.

Crowley [Crowley85] [Crowley89] used a single rotating ultrasonic range sensor to map interior spaces which can be modelled with long line segments. He fit line segments to a 360° range scan to create a "sensor model". By matching the sensor model to the line segment map, the pose of the robot was calculated. The map was then updated by incorporating the information from the sensor model. This system allowed a robot to navigate in indoor spaces with flat walls.

Leonard and Durrant-Whyte [Leonard92] used ultrasonic range sensors to track point and line features in indoor environments. Using an extended Kalman filter framework, they matched the sensed features to a presurveyed set of features (assumed perfect) to estimate the robot's pose. The other subsystem of their mapping software computes feature locations from sonar data and perfect poses. The result a feature map sufficient for localization and navigation (which is their stated intention), but not a complete representation of the robot's environment.

Cox [Cox91] computed robot pose by matching laser range data to a presurveyed line segment map of a structured (clean indoor) environment. The system corrected errors of several inches while moving.

Gonzalez [Gonzalez92] [Gonzalez93] builds line segment maps (similar to Crowley's) from laser rangefinder data. Pose estimation is decoupled from model-fitting by matching range data to a presurveyed line segment map to estimate pose, although nothing in the algorithms preclude matching to the map that is under construction. New line segments are fit to the range data or existing map line segments are modified given the new information. It is unclear whether this algorithm would work in an unstructured environment that isn't composed of relatively long flat walls.

This dissertation describes a map-building system which concurrently computes the sensor pose and constructs a model of the robot's environment. The map is a complete representation of the area explored by the robot, built by fitting polylines with small segment length to overlapping laser range scans. Accurate pose estimation is enabled by matching a range scan to the large section of map that it overlaps.

1.4 Approach

The map-building system described in this thesis consists of two independent components that were designed to work well together. The first component is the pose estimator, which matches a range scan to a polyline map to determine the pose of the sensor in the map's coordinate system. The second component is the contour fitter, which fits polyline "contours" to a single range scan or several overlapping range scans to create or extend a map. Both components work by minimizing

an error of fit of contour line segments to range data points. The pose estimator adjusts a scan pose until the error is minimized; the contour fitter adjusts the polyline vertices until the error is minimized. The same error metric is used for both minimizations, allowing iterative refinement of a complete map, provided that the range scans have been stored.

An intermediate step between pose estimation and contour fitting is contour editing. Given a new range scan at a known pose, the contour editor modifies existing contours or creates new contours as appropriate. Where the new range scan conflicts with an existing contour, the contour is modified to reflect the new information provided by the range scan. New contour polylines are fit to the range scan where it doesn't overlap existing contours. These new contours often connect with the existing contours. In a dynamic environment, contour editing also includes removing parts of the map which correspond to part of the environment that has been removed.

The usual mode of operation is to mount the range sensor on a mobile base that has dead reckoning capability. At a specified spatial interval (which usually depends upon the geometry of the workspace), the robot stops and records a range scan. The robot can move to its next scan location while inserting the newly acquired scan into the map. Inserting the scan into the map consists of pose estimation, contour editing, and contour fitting. Dead reckoning provides a rough pose estimate as a starting point for the pose estimator's error minimization search.

A strength of the approach is the ability to continually refine the map, by iteratively refining sensor pose estimates and refining the fit of contours to the range scans. In this way, error in the map is continually reduced after new scans overlap old ones, or loops of overlapping scans close on themselves. Each sensor pose comprises three degrees of freedom which determine the absolute location of all the points in one scan. Every contour vertex comprises two degrees of freedom. If all of the scans are retained, all of the degrees of freedom can be adjusted to minimize the total error in the map.

1.5 Overview

This dissertation presents a progression of mapping technologies, starting with pose estimation in a static presurveyed structured and cluttered area, and ending with mapping of a dynamic expansive unstructured area. Mapping underground coal mines is an ideal application of 2D mapping technology. Accordingly, the mining example will be used throughout this dissertation, though the technology developed is readily applicable to other tasks.

Chapter 2 presents a feature-based pose estimator; the iconic pose estimators in Chapter 3 remove the intermediate feature representation and the need for a presurveyed map; Chapters 4 presents the main contribution of this thesis, the contour-based mapper; Chapter 5 contains the bulk of the experimental results; Chapter 6 describes the applicability of this technology to underground mining; and Chapter 7 is conclusion.

2 Pose Estimation by Feature Matching

This chapter describes a feature-based pose estimator that exploits the accuracy and resolution of a laser range sensor to determine the position and orientation of a mobile robot given a polygonal map of its environment. The system detects features from range sensor data and matches the features to the map to compute the sensor's pose. The features used are line segments and corners, which are the features that are found in an environment bounded by straight walls. The pose estimate is computed by minimizing the error between the map and sensed features. The approach can handle the expected uncertainty in the initial estimate, heavy clutter in the environment, and still compute an estimate quickly.

This system was originally intended to serve as a pose estimation system for an autonomous continuous mining machine, though it became clear that this system would not perform well in a typical mine. Looking at existing mine maps was misleading -- there appeared to be long straight walls and sharply defined corners. When we had a chance to test our sensor in a real mine, we learned that existing mine maps are built from a relatively sparse set of measurements, and long stretches of wall are interpolated as straight lines between the measurements. Therefore, this feature-based pose estimation system became an indoor application, and results are presented from our indoor test area.

This chapter is organized as follows: the feature-based approach is presented in Section 2.1 on page 6; a physical robot implementation and position estimation results are presented in Section 2.2 on page 14; and Section 2.3 on page 18 concludes the chapter with some discussion.

2.1 Approach

The pose estimator takes as input a line segment model of the environment and a scan from a laser range sensor and produces an estimate of the sensor's pose in the world coordinate frame. Given a a transformation between the sensor's coordinate frame and the coordinate frame of the robot it is mounted on, it is then possible to determine the robot's pose in the world coordinate frame. The laser range sensor referred to is a single line scanner. It measures distance to points in the robot's surroundings by rotating like a lighthouse and casting range measurements with its infrared beam. A scan is a dense array of range measurements, taken at equally spaced angular intervals around 360° , in a horizontal plane. A scan provides a planar profile of the sensor's environment formed by finding the distance to the nearest objects at a particular height. See Appendix B for specifications of the particular sensor used.

Mathematically, a scan is represented by a vector of range values, r, and a corresponding vector of azimuth angles, α, sensed in a horizontal plane at $z = z_s$, where z_s is the sensor height and is assumed to be constant for all scans. Each range value, r_i, is the sensed distance from the sensor position to the nearest object in the direction $(\theta+\alpha_i)$ where (x, y, θ) is the sensor pose and (r_i, α_i) is one pair of corresponding elements from the range and azimuth vectors.

Pose estimation is accomplished by predicting visible features from a map, extracting features from the scan, matching features from the scan and map, and estimating position from the match. (See Figure 1 on page 8.) Feature prediction generates a set of possibly visible map features from an approximate position of the robot. Feature extraction finds features in the range scan. Features extracted from the scan are paired with features from the map in the process of feature matching. The position estimator finds a position that minimizes the error between sensed features and map features. These processing steps are described in detail in the following subsections.

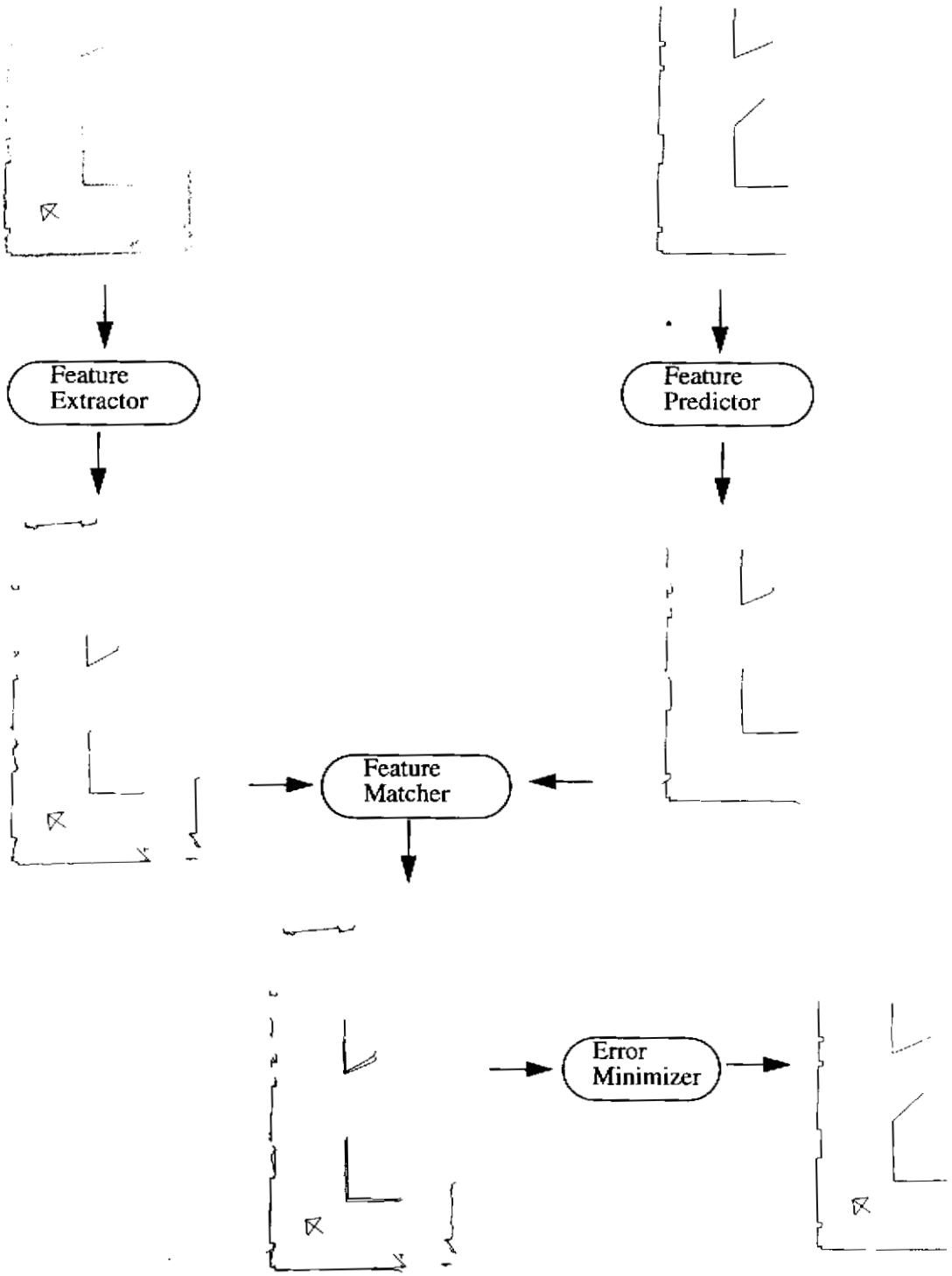


FIGURE 1. Overview of the feature-base pose estimation system.

2.1.1 World Model

For this system to work, a 2D (line segment) model must adequately represent the environment of the robot. In other words, the environment must consist of a nearly flat floor and nearly vertical walls. Furthermore, since the system relies on features, at least parts of the walls need to be relatively long and flat, allowing strong line segment features and sharp corners to be used for good pose estimation.

2.1.2 Features

A world model composed of line segments contains two categories of primitive features -- line segments and corners. The term *corner* is used loosely to mean either a regular corner (formed by two visible line segments sharing a common endpoint) or an occlusion corner (formed when a visible endpoint of a line segment obstructs the view of part of another line segment). In the case of an occlusion corner, half of the corner lies on the imaginary line which connects the occluding object to the occluded object. These features are illustrated in Figure 2 on page 9. A corner is described by four parameters: x, y, *concave angle*, and *bisector orientation*. The concave angle is the smaller of the two angles formed by a corner's two line segments. The bisector orientation is the angle that the bisector of the concave angle forms with respect to the positive x axis. Line segments are described by three parameters: *length*, perpendicular *distance* from the origin, and *orientation*, which is the angle that the perpendicular forms with respect to the positive x axis.

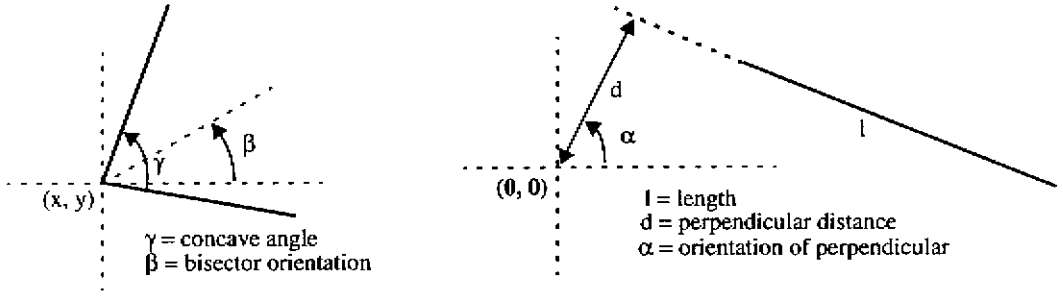


FIGURE 2. The parameterization used to define corner features and line segment features. A corner is characterized by the location of its vertex (x, y), its concave angle, and its bisector orientation. A line segment is represented by its length, the perpendicular distance from the origin, and the orientation of the perpendicular.

2.1.3 Feature Prediction

Since only a small portion of the model (in general) is visible from a particular location (x, y) of the robot, feature prediction greatly simplifies the process of feature matching (described below) by reducing the number of model features that can possibly match the sensed features. (See Figure 3 on page 10.) The feature predictor assembles an ordered list of model features visible from a given (x, y) as the scanner sweeps in a counterclockwise direction. This preprocessing

greatly constrains the possible combinations of pairings of sensed features and model features, thereby simplifying the feature matching process.

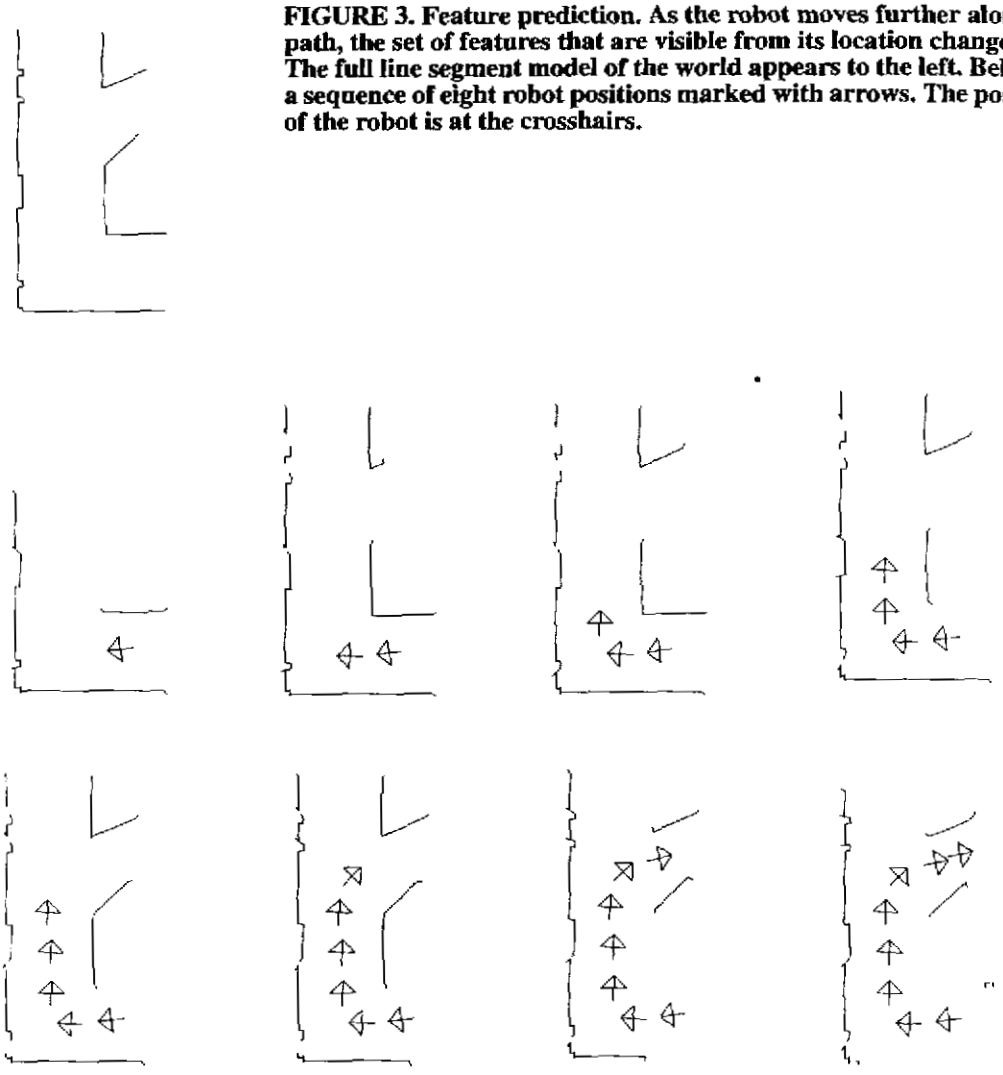


FIGURE 3. Feature prediction. As the robot moves further along its path, the set of features that are visible from its location changes. The full line segment model of the world appears to the left. Below is a sequence of eight robot positions marked with arrows. The position of the robot is at the crosshairs.

The feature prediction algorithm first creates a sorted array of the endpoints of all the line segments in the model. The endpoints are sorted on the basis of azimuth relative to the given (x, y) of the robot. The next step is to consider each line segment in the model, and making use of the sorted list of points, determine whether each point that lies in the azimuth range subtended by the current line segment either occludes or is hidden by the current line segment. The outcome of this step is that every corner in the model is labeled as either visible or invisible and, if visible, either occluding or nonoccluding. In one final pass, the array elements are stitched together into the ordered list of visible features required by the feature matcher. The worst-case complexity of the prediction

algorithm is $O(N^2)$ (where N is the number of line segments in the model) in the case where there is a large number of occluded features. However, the average-case complexity is closer to $O(NlogN)$ for typical mine maps, especially if the feature predictor uses the maximum range of the scanner to limit the number of model lines considered.

Neither Drumheller [Drumheller87] nor Krotkov [Krotkov89] address this issue of feature prediction, since they attempt to match all model features to all sensor features. Crowley [Crowley85] maintains a "composite local model" which consists primarily of the currently visible segments only.

2.1.4 Feature Extraction

Feature extraction is the process of reducing the raw range scan to an ordered list of features. The feature extraction algorithm first subdivides the scan into segments by breaking the scan at points where the distance between successive points exceeds a threshold, thereby finding occlusions. Pieces of the scan that contain only a small number of points (such as two or three) are thrown out, since they give very unreliable features. Each piece of the scan is processed using a *corner operator* to find line segments and corners. The corner operator is a template consisting of two line segments that is compared against a sequence of range points to determine whether they exhibit a corner-like shape. (See Figure 4 on page 11.) The corner operator is parameterized by size (length of the component line segments) and by the angle between them. The corner operator template is "applied" to the range sequence by fitting the left and right line segments to the range data at a particular point using an incremental line fitter. If the template matches, a corner is defined at the intersection of the two line segments which form the operator. A larger corner operator gives more reliable feature parameters. Use of an incremental line fitter is possible because the corner operator template is moved sequentially one point at a time through the range scan, and it greatly decreases the amount of processing compared to a separate line fit at every point in the scan.

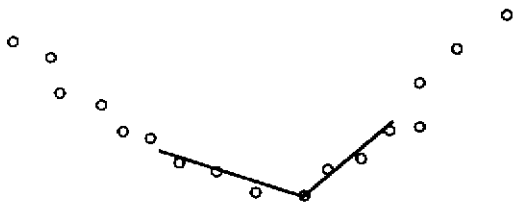


FIGURE 4. The corner operator is applied at every point in the scan. It is parameterized by length of the sides, and an angular threshold. The two sides of the corner operator are fit to the nearby scan points. If the angle between them falls within an interval around 90°, a corner is defined. Line segment features are created by fitting a line to the points between neighboring corners.

To find the parameters (length, perpendicular distance, orientation) of the line segment that lies between any two corners, a line is fit to all the data points in the scan sequence that fall between the two corners. The perpendicular distance and orientation are determined by the slope and intercept of the fit line; the length of the line segment is the distance between the two corners.

2.1.5 Feature Matching

The matcher determines the correspondence of the sensor features to the model features. A total of $N!$ sets of correspondences exist, assuming there are N model features and N sensor features. This prohibitively high complexity is further exacerbated by extra and missing sensor features corresponding to features not in the map (e.g. people, machines) and features that cannot be reliably extracted, respectively. Fortunately, the problem is usually constrained enough that only a much smaller set needs to be examined. See Wallace [Wallace88] for a survey of matching techniques. The type of matching technique used was determined by the following problem characteristics:

1. The position of the robot is approximately known.
2. The extracted features are reliable and rich in geometric information.

The first characteristic greatly constrains which sensor features can match a model feature (called unary constraints), and the second greatly reduces the number of sensor-model pairs that need to be compared to determine match consistency. Three comparisons are used, hence the constraint is labeled ternary. The algorithm begins by applying the unary constraints to construct the set of possible sensor-model pairs. Then, the ternary constraints are applied to find the largest set of consistent correspondences. Efficient cutoff techniques are employed to reduce the complexity of the search.

The matcher first converts the list of predicted model features from the world coordinate frame to the robot coordinate frame using an approximate position given by dead reckoning, so that the model features and the sensor features are both in the robot coordinate frame. The unary constraints are then applied. These constraints are subdivided into *unary bounds* and *unary properties*. The unary bounds constrain the difference in position of corresponding sensed and model features, based on the maximum position error (*poserr*) and orientation error (*angerr*) of the dead reckoning position. Specifically, the unary bounds can be stated as the following necessary conditions for correspondence between a sensor and model feature:

1. The angle to the sensed corner, relative to the positive x axis, must lie within $\pm$*angerr* of the angle to the model corner.
2. The distance from the origin to the sensed corner must lie within $\pm$*poserr* of the distance to the model corner.
3. The angle subtended by the sensed line segment must at least partially overlap, (or be partially overlapped by) that of the model line segment, allowing a rotation of $\pm$*angerr* of the sensed line segment.

The unary property constraints are similar to the unary bounds constraints, except that while the unary bounds compare locations of features, the unary properties (for the most part) compare properties of the features that are unrelated to the position of the robot. Given that a particular candidate sensor-model pair satisfies the applicable unary bound constraints, the unary properties provide additional necessary conditions to further confirm the correspondence between a sensor feature and a model feature. The unary property constraints are specified as follows:

1. The concave angle of the sensed corner must lie within ±*concerr* of the concave angle of the model corner.
2. The bisector orientation of the sensed corner must lie within ±*angerr* of the bisector orientation of the model corner.
3. The orientation of the sensed line segment must lie within ±*angerr* of the orientation of the model line segment.
4. The perpendicular distance to the sensed line segment must lie within ±(*poserr* + *disterr*) of the perpendicular distance to the model line segment.

where *concerr* is the angular tolerance of the concave angle of a corner and *disterr* is the distance tolerance of the perpendicular distance to a line segment. Each sensor feature is compared to every model feature, first with the unary bounds constraints, and then the unary properties constraints. In this way, the number of candidate sensor-model pairs (which before the unary constraints are applied is of size MN, where M is the number of model features and N is the number of sensed features) is greatly reduced. The result is a list of ordered sensor-model pairs. The unary constraints, however, are not perfect, nor do they capture all of the available constraining information. Additional pruning is needed.

In order to extract from this ordered list of sensor-model pairs a set of consistent pairings, the next step utilizes the innate ordering (features are encountered in the order that the sensor scans) of the list and a set of ternary constraints to weed out the inconsistent pairings. We assume that the largest such consistent set of pairings is most likely to be a correct match and provides the most accurate position estimate. A ternary constraint compares a sensor-model pair to two sensor-model pairs that precede it in the ordered list. These two comparisons fall into one or two of the following categories:

1. If one of the sensor-model pairs is a pair of line segments and the other is a pair of corners, then the model line segment must be positioned relative to the model corner in the same way as the sensor line segment and corner.
2. If both sensor-model pairs are pairs of corners, then the distance between the two model corners must be equal to the distance between the two sensor corners ±*terndisterr*.
3. If both sensor-model pairs are pairs of line segments, then the difference in orientation between the two model line segments must be equal to the difference in orientation between the two sensor line segments ±*ternangerr*.

where ±*terndisterr* is a tolerance on distance error and ±*ternangerr* is a tolerance on angular error. An efficient search algorithm applies these ternary constraints to the sensor-model pairs. The algorithm builds sequences of ternary-consistent sensor-model pairs using a depth-first search of the ordered sensor-model pair list. The longest sequences of consistent pairs leading into and out of each sensor model pair are tracked to enable the search algorithm to terminate fruitless branches efficiently. Thus the search process assembles progressively larger sets of pairs until it exhausts all possible combinations. The largest set of pairs is used to calculate the position estimate.

2.1.6 Error Minimization

The error minimizer finds a transformation which minimizes the total error between all transformed model features and their corresponding sensor features. Any combination of line segment feature pairs and corner feature pairs can be used in producing an estimate. The error measures are different for the two feature types -- for line segments there are two error measures: difference in orientation and difference in perpendicular distance from the origin, and for corners there are also two error measures: difference in x and difference in y. An iterative Gauss-Newton [Dahlquist74] method is used to converge on a solution. The quantity to be minimized is: $F + JD$, where F is a vector of the four error measures, J is the Jacobian of the four error measures differentiated with respect to the three transform parameters (x, y, θ) and D is the vector of differences between the transform parameters on successive iterations. For this algorithm to converge to a correct solution, it must be guaranteed that the initial orientation error is less than 90°, well within the accuracy of dead reckoning. The algorithm weights features by the number of range data points that were used to compute them.

2.2 Results

The techniques developed in this work were carried beyond simulation into implementation to navigate a real robot through a mine mockup. In this section, we describe a test of the feature-based pose estimation system using our Cyclone laser range sensor on out Locomotion Emulator robot (See Appendix D.) in an indoor mine mockup.

The perception system is implemented on a single processor (68030 20MHz, 68882 math coprocessor) real-time (VxWorks) development cage. A Sun workstation sits atop the Locomotion Emulator (LE), but is used only to boot the real-time system and then to download files. A video board in the real-time cage displays relevant information to a color monitor.

All of our navigation experiments were conducted within a mine corridor mockup which covers an area of approximately 20x30m of the highbay area of the Field Robotics Center. A photo of this area appears in Figure 5 on page 15. The walls that are part of the line model include all of the partitions and two of the concrete walls of the highbay area (only one of them is visible in the photo -- in the upper left quarter of the image). Notice that the concrete walls are not perfect -- there are a few roughly 0.4m square columns that jut out from the wall. Heavy clutter is often present (people, machines, equipment) -- none of which is represented in the line model. This clutter has never caused the perception system to fail in estimating a position; it only reduces the number of usable sensed features. We surveyed the mockup with a transit to create a line model. We generated intended paths as straight lines that direct the robot down the center of the corridor.

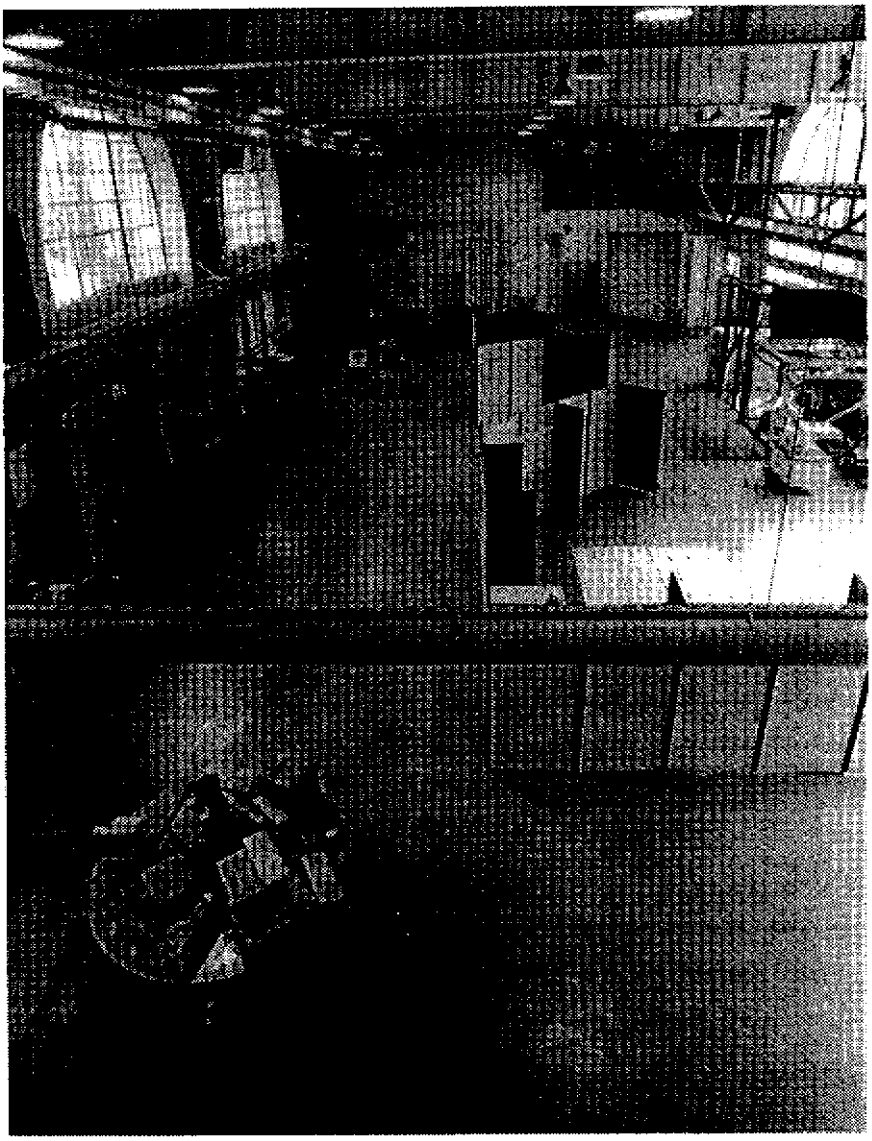


FIGURE 5. The mine corridor mockup in the highbay area of the Field Robotics Center. The corridor is bounded on the left by the concrete wall of the building and on the right by some office partition panels. The width is roughly 6m.

The tracking algorithm takes one parameter -- step size, which determines how far the robot moves between position estimates. After each estimate, the tracking algorithm moves to the point

on the desired path that is a step size away from the current position estimate. Figure 6 on page 16 shows the path of the robot during the test run.

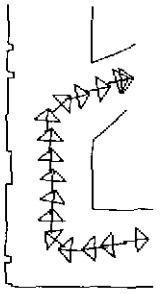


FIGURE 6. The sequence of poses during the test run. The first pose is at the bottom right and the last is at the top right.

TABLE 1. Ground truth for the run in Figure 6 on page 16

Surveyed x (m)	Surveyed y (m)	Gyro θ (deg)
9.822	3.646	n/a
8.248	3.353	187.587
6.685	3.158	181.628
5.088	3.099	177.675
3.736	3.921	143.068
3.579	5.483	89.324
3.641	7.082	81.704
3.579	8.694	87.090
3.452	10.261	88.179
3.393	11.854	87.090
3.327	13.425	86.689
4.300	14.660	45.035
5.763	15.258	15.241
7.236	15.793	12.834
8.707	16.392	18.163
9.179	16.517	10.714

TABLE 2. Estimated poses for the run in Figure 6 on page 16

Estimated x (m)	Estimated y (m)	Estimated θ (deg)
10.185	3.804	4.297
8.410	3.442	186.900
6.989	3.134	181.514

TABLE 2. Estimated poses for the run in Figure 6 on page 16

Estimated x (m)	Estimated y (m)	Estimated θ (deg)
5.155	3.033	176.300
3.885	3.824	144.099
3.610	5.320	89.210
3.684	6.869	80.845
3.636	8.520	87.147
3.521	10.112	88.637
3.441	11.697	87.663
3.210	13.292	84.798
4.307	14.614	45.722
5.527	15.236	14.840
7.110	15.802	12.433
8.547	16.416	17.934
8.892	16.489	9.568

TABLE 3. Error in the estimated poses.

Error in x (cm)	Error in y (cm)	Error in θ (deg)	Position Error (cm)
36	16	n/a	39
16	9	-0.7	18
30	-2	-0.1	30
7	-7	-1.4	10
12	-10	1.0	16
3	-16	-0.1	16
4	-21	-0.9	21
6	-17	0.1	18
7	-15	0.5	17
5	-16	0.6	17
-12	-13	-1.9	18
1	-5	0.7	5
-24	-2	-0.4	24
-13	1	-0.4	13
-16	2	-0.2	16
-29	-3	-1.1	29

Table 1 on page 16 shows the ground truth data for the run of Figure 6 on page 16. Table 2 on page 16 shows the computed pose estimates. Table 3 on page 17 summarizes the errors. The maximum position error is 39cm, though it is usually about 18cm. The maximum heading error was 1.9°, though it was usually less than 1.0°. The "ground truth" heading was provided by a gyro. The gyro drift problem was virtually eliminated by using differences in gyro readings from position to

position to update orientation, but this introduced some quantization error since the gyro reports heading to a resolution of 0.25°. Since the run consisted of 15 moves, the maximum error that this quantization produced was 1.875° by the end of the path. Another source of error in the heading was a bug in the scanner electronics which could introduce an error of as much as 1.44°. The parameter values used were: *angerr* = 5°, *poserr* = 1m, *concerr* = 10°, *disterr* = 0.5m, *terndisterr* = 0.5m, *ternangerr* = 5.0°. Approximate run times were 60 msec for feature prediction, 2 seconds for extraction, and 1 second for combined matching and estimation, giving a total cycle time of approximately 3 seconds on a 20Mhz 68030 running under VxWorks with a 68882 math coprocessor.

For both position and heading, there are three sources of error: quality of the range data, accuracy of feature extraction, and correctness of feature prediction. A raw range scan contains random error, systematic error, and outliers. The random error is about ±15cm of noise in each measurement, and there's nothing to be done about that except to take several scans and get a better range reading through a statistical technique. Systematic error was greatly reduced by calibration of the sensor. Outliers occurred mostly at occlusions, when the laser beam reflected partially from the near object and partially from the further object. Typically, this resulted in a reported range somewhere between the two objects. This could be filtered out with a median filter or by discarding scan points at occlusions, but neither of these techniques was used for the results presented here. Sometimes the feature extractor misses a real corner because it is so small that it blends into the noise. And there are several thresholds involved. The feature predictor's effectiveness depends to some degree on the quality of the dead reckoning pose estimate it is given. The feature matcher and the error minimization components are less prone to error than the other modules.

2.3 Discussion

This chapter presented a pose estimation system capable of determining the pose of a robot equipped with a laser range scanner and a line segment map of its environment. This work was inspired by the work of Crowley [Crowley89] and Drumheller [Drumheller87]. Much like their systems, features were fit to the range data and matched to a line segment map. Performance of the system was demonstrated in an indoor environment structured to approximate the layout of an underground mine. The results obtained in the mockup indicated that the performance was sufficient for the mining application. However, as it turns out, our mockup was not a very good approximation of a real mine environment. Figure 7 on page 19 shows a range scan from an underground room-and-pillar type coal mine. It is possible to tweak some of the parameters of the feature extractor such that it finds corners in a scan such as this, but in general, there won't be very many, and they won't be very accurate. This prompted us to abandon further testing and refinement of the feature-based pose estimator in order to pursue a more appropriate pose estimation technique. This new technique, which we call an "iconic" pose estimator[1], does not match features in the scan to

1. We adopt the term "iconic" from Kweon.

features from a map. Rather it treats the scan as one structureless "icon", and matches every scan point to a the map. The map is still a line segment map, but the segment size is allowed to be almost arbitrarily small to allow modelling of unstructured, rounded, lumpy environments such as a coal mine. This iconic pose estimator is the subject of the next chapter.

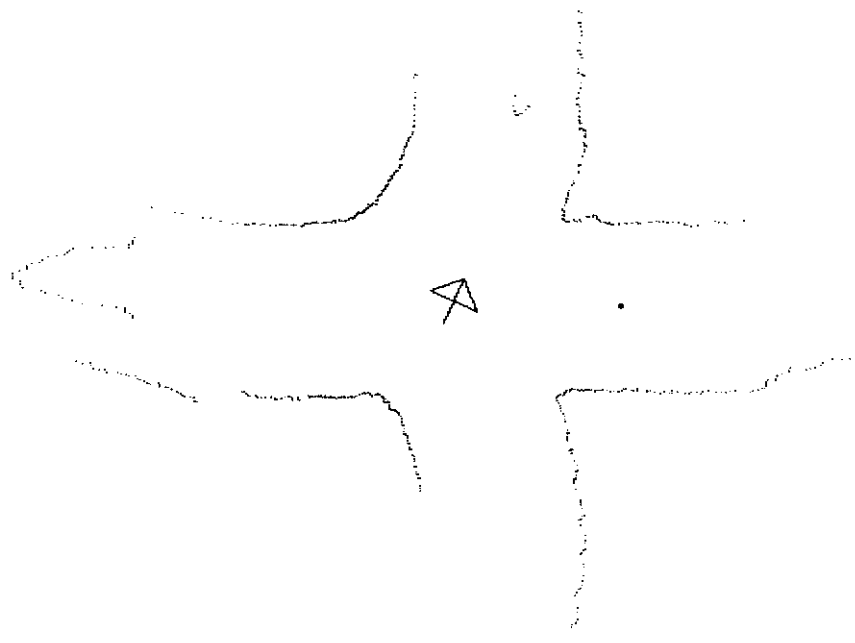


FIGURE 7. A range scan from an underground room-and-pillar type of coal mine.

3 Pose Estimation by Iconic Matching

Feature matching can work well when there are features to be matched, such as in indoor environments composed of long straight walls and right angle corners. However, not all 2D environments are so nicely structured. In featureless, unstructured environments (walls that are curved with very rounded corners) a different method of pose estimation must be employed. A map-matching technique is possible, but the map and scan can no longer be abstracted into sets of features to be matched. All of the scan points much be matched to the map.

For now, we continue to assume that the world is static. The map is still a set of line segments, though the line segments can be very short, allowing modelling of curved surfaces. The feature extraction and feature prediction modules from the feature-based system are no longer applicable. The correspondence problem remains and becomes a little more time-consuming. The error metric definition changes a little bit, but the same error minimization technique can be used.

The structure of the iconic estimator is summarized in Figure 8 on page 20. There are only two main components to the iconic matching system: correspondence determination and correspondence error minimization. The correspondence determination module pairs each scan point to a line segment in the map. The error minimization module adjusts the pose estimate until the correspondence error is minimized. The system is very similar to Gonzalez's [Gonzalez92] system. However, the correspondence determination algorithm (the meat of the system) is more efficient.

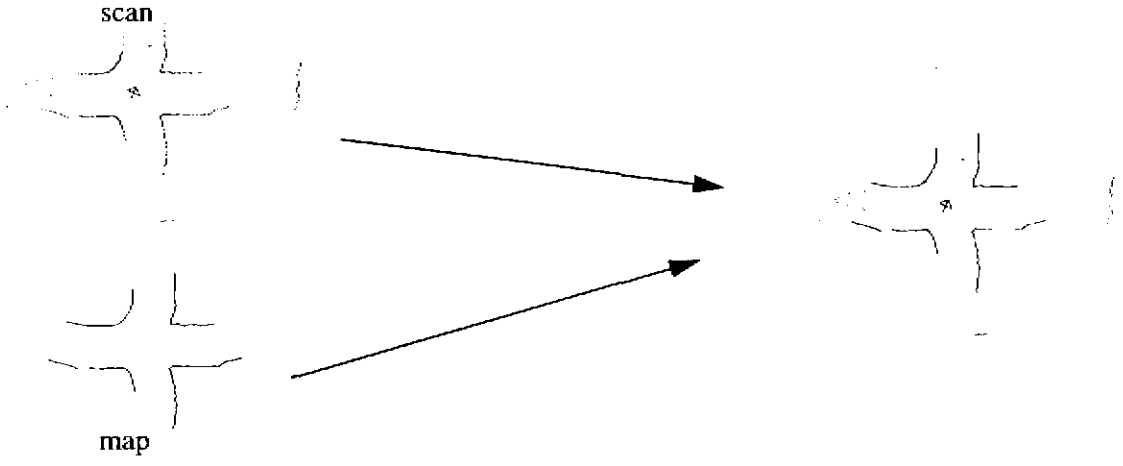


FIGURE 8. An example of iconic pose estimation. An iconic estimator matches all the points of a range scan to a map to determine the pose of the robot.

Five flavors of iconic pose estimators will be presented:

1. iconic matching of one scan to a hand measured map
2. iconic matching of one scan to a scan-generated map

The map is generated automatically by fitting polylines to a scan.

3. iconic matching of one scan to another scan

A scan from a known pose is used as the map.

4. iconic matching of one scan to many scans

All previously stored scans are used as the map.

5. iconic matching of many scans to each other

The poses of all scans are adjusted simultaneously.

3.1 Using a Hand Measured Map

The motivation for switching to an iconic method of matching was that the feature-based method could not handle a realistic unstructured environment. The map representation (list of line segments) could approximate an unstructured environment if the segment size was small enough, but the main assumption of the feature-based system (that strong corner features and long straight walls make up the environment) would be violated. In switching to an iconic method we eliminate the assumption that strong features exist. Strong features are helpful, but not required. The map representation remains as a set of line segments, but the line segments can be arbitrarily small allowing modelling of irregular surfaces. For this first incarnation of an iconic pose estimator, we assume that the map is surveyed in advance and provided to the pose estimator. We continue to assume that the world is static. In implementing this pose estimator, speed of computation was a concern. The data was not being reduced into a small set of features as with the feature-based system. The rest of this section describes the components of the iconic pose estimator. There are only two: correspondence and error minimization.

3.1.1 Correspondence Determination

The purpose of the correspondence determination module is to pair each scan point with its corresponding line segment in the map. It is assumed that each scan point corresponds to the line segment that is closest. Distance to a line segment is defined as the minimum of the distance to the line containing the segment and the distances to each of the endpoints. Therefore, there can be two varieties of correspondence: point to line and point to point. There is a limit on the correspondence distance. If this distance is exceeded, there is no pairing. There are three reasons why this distance threshold might be exceeded:

1. The initial pose estimate is poor.
2. The scan covers an area that is not represented in the map.
3. There is an error in the map.

It doesn't matter if some points aren't matched, as long as enough good pairings are made. The opposite problem occurs when one scan point is within the distance threshold of more than one map segment. To handle this case, the nearest line segment is chosen.

Each scan point presumably corresponds to a reflection of the laser beam from some part of the world, which may appear in the map as a line segment endpoint, an interior point of a line segment, or nothing at all (because an unmapped area has been scanned). The total scan-to-map correspondence consists of N point-to-segment pairings. The naive algorithm for computing these pairings would be, for all N measured points, to compute the distance to all M line segments to find the nearest. This is straightforward, but inefficient. For this reason, an efficient implementation for computing the correspondence was developed.

Efficiency is attained by drawing the map into a grid, and storing in each grid cell a list of the map line segments which intersect that cell. Then the marked grid cells are grown several levels with a grassfire transform. Grid cells that are encountered in the growth step are tagged with a pointer to the nearest grid cell that was intersected with a map line segment. After the growth step, the result is that all of the grid cells within some threshold distance (which can be as large as your initial pose estimate error requires) are tagged with a list of map line segments that are the closest to the cell. If the grid resolution is chosen appropriately, the lengths of the line segment lists will be mostly 1, 2, or 3. Then the correspondence for each of the N scan points can be determined by computing the distance to a very small number of line segments, which does not depend on M. Thus the complexity has been reduced from O(MN) to O(N). However, a preprocessing step is required, so this method is only practical if many pose estimates are going to be computed using a particular map. To determine the nearest segment to a scan point, the procedure is to find the grid box that the scan point lies in, and the associated "short" list of line segments that might be nearest. By computing the distance to each of the line segments on the short list and choosing the minium, the correspondence has been determined for a scan point.

3.1.2 Correspondence Error Minimization

The problem is to determine the sensor pose that matches a set of N measurements to a set of M line segments. This can be formulated as an optimization problem -- to be precise an overdetermined nonlinear minimization. It is overdetermined because there are three degrees of freedom and N range measurements (where N is always >> 3). It is nonlinear because one of the degrees of freedom is an angle, and therefore the error metric involves trigonometric functions. The correspondence error is defined as the sum of the distances of each scan point to the map line segment is corresponds to. If the correspondence is between scan point and map line segment endpoint (a "point-to-point" correspondence), the error is simply the distance between the two points. If the correspondence is between a scan point and the interior of a map line segment (a "point-to-line" correspondence), the error is the minimum distance from the point to the line containing the line segment. The same correspondence error minimization method from the feature-based pose estimator is used for the iconic pose estimator. The only difference is that the matrices are generally larger, because their sizes are determined by the number of scan points instead of the number of features extracted from the scan points. See Section A on page 85 for the formulas.

The minimization loop recomputes the correspondence on every iteration, and reduces the correspondence distance threshold on every iteration until a lower bound is reached. The threshold is set to the median of all the correspondence distances of the previous iteration. The lower bound is set beforehand depending on the noise level in the range data. There is one termination condition: an epsilon on the maximum change in any of the degrees of freedom. In case this epsilon is set too small, there is an iteration limit.

3.1.3 Results

An interesting test case is to use the same data set from the feature-based results. One would hope that the iconic system performs similarly, and preferably better. It will likely be slower, but the accuracy should be better because the intermediate feature representation is avoided, eliminating any error imposed with it.

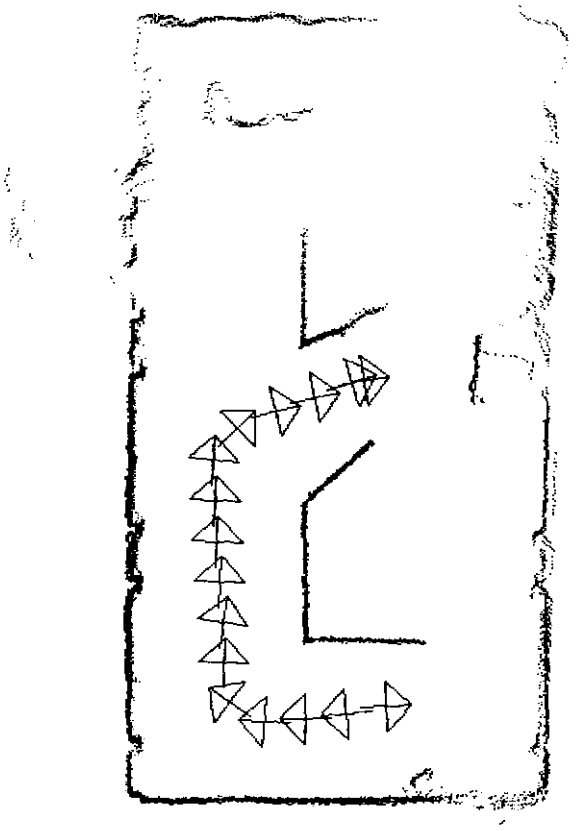


FIGURE 9. A test of the iconic scan matcher on the same data set as the feature-based example. The arrows mark sixteen estimated poses. The first pose is at the lower right. It points to the right because we always initialized the system with a heading of approximately zero (along the positive x axis).

Figure 9 on page 23 shows the result of running the iconic pose estimation system on the same data set used as an example for the feature-based system. As expected, the computation time was significantly higher. The error is summarized in Table 4 on page 24.

TABLE 4.

Error in x (cm)	Error in y (cm)	Error in theta (deg)	Error in distance (cm)
27	10	-1.1	29
29	9	-1.6	30
27	8	-1.0	28
27	6	-1.2	27
24	0	0.4	24
18	1	0.0	18
8	-2	0.0	8
1	-4	0.6	4
2	-3	-0.0	4
-1	-1	-0.7	1
-2	3	-1.8	4
-5	5	0.2	7
-3	11	-0.9	12
2	12	-1.1	12
1	11	-0.5	11
-0	5	-0.6	5

Comparing Table 4 on page 24 with Table 3 on page 17, you can see that in most cases, the iconic pose estimator was more accurate. The sources of error have been reduced to scanner calibration, error in correspondence, and experimental error in surveying the poses.

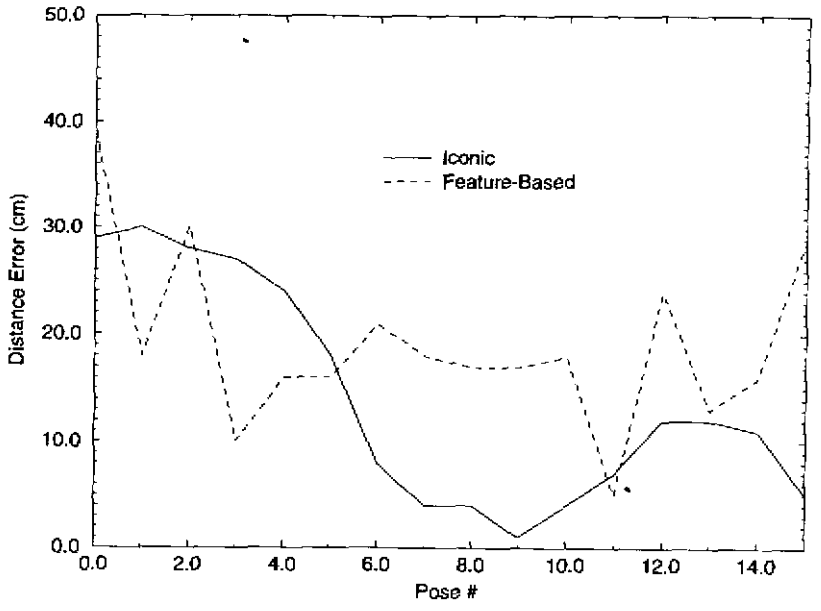


FIGURE 10. Comparison of feature-based and iconic pose estimation. The distance error for both estimators on the same dataset of 16 poses is shown.

3.2 Using a Scan-Generated Map

The previous section presented an iconic pose estimator for matching a scan to a premeasured map to compute the pose of a robot. If the robot's workspace does not exceed its range scanner's field of view, it's possible to take an initial scan from a known pose and fit polylines to the scan to form a map. If there is no need to work within a global coordinate system, an arbitrary pose can be assigned to this initial scan. Given this map, the same pose estimation system from Section 3.1 on page 21 can be used to serve as a positioning system while the robot operates within the area covered by the initial scan.

3.2.1 Results

Figure 11 on page 26 show the initial scan and the map that was generated from it. The scan is the first of a set of 10 scans from an underground room-and-pillar coal mine. The method of fitting polylines used to generate this map is actually the subject of a later section in this thesis (contour fitting) and will not be addressed here. The segment lengths in the map are roughly 0.5m. The Cyclone sensor was used to collect these scans. The position of the sensor was surveyed at each of the 10 poses, but not the orientation.

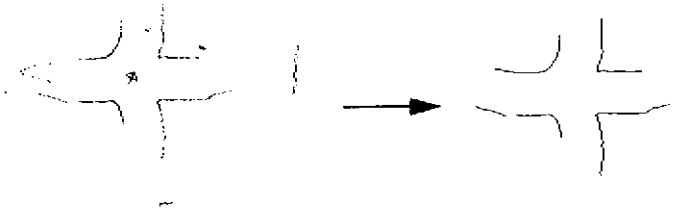


FIGURE 11. The map was generated from an initial scan by breaking the scan at occlusions and fitting polylines. Some of the clutter in the scan was removed by hand.

The results of the 10-scan run is illustrated in Figure 12 on page 26 which shows all 10 scans plotted at their dead-reckoned poses and estimated poses.

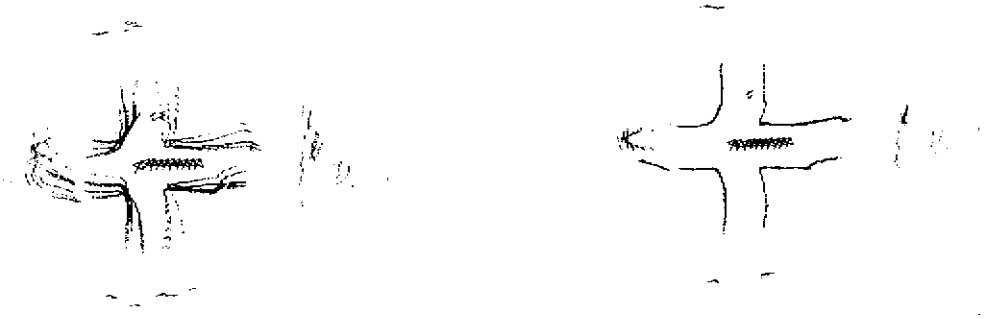


FIGURE 12. The result of matching a sequence of 10 scans to the map formed by fitting polylines to the first scan. On the left is the set of 10 scans plotted at their initial poses. On the right, the same 10 scans are plotted at their refined pose estimates.

TABLE 5.

Error in x (cm)	Error in y (cm)	Error in theta (deg)	Error in distance (cm)
0	0	-0.0	0
2	-8	0.8	8
8	-9	1.7	12
8	-9	1.1	12
9	-6	1.5	11
11	-9	1.4	14
10	-9	1.4	13

TABLE 5.

Error in x (cm)	Error in y (cm)	Error in theta (deg)	Error in distance (cm)
13	-5	1.4	14
12	-0	1.6	12
13	1	3.2	13

3.3 One to One Matching

This section presents a variation of the basic iconic pose estimator which does not require a pre-measured map. If a robot's workspace does not exceed its range scanner's field of view and there are no obstructions blocking a significant portion of the field of view, it is possible to treat a single scan as a map of the area for pose estimation. The robot initializes by acquiring a scan from the initial location and storing it. The initial location can be assigned arbitrary coordinates or can be measured by hand in a global coordinate frame. The scan is prepared for further processing by breaking it into "partial scans" at discontinuities in the scan which exceed a threshold (caused by occlusions). The threshold is set to be more than twice the accuracy of the range, so noise in the data won't effect the formation of the partial scans. The result of applying this simple algorithm to a scan is a set of partial scans which are treated as polylines to serve as a map. Using this map, the iconic pose estimation system described above can be used.

3.3.1 Results

For evaluation of this method of pose estimation system, we present the results from using the same coal mine dataset as the previous section. Figure 13 on page 28 shows the result of matching a sequence of 10 scans to the initial scan. From looking at the illustration, it's impossible to see any difference from Figure 12 on page 26.

FIGURE 13. Results for matching each scan to the initial scan. On the left is dead reckoning. On the right are estimated poses.

TABLE 6. Error resulting from matching each scan to the initial scan.

Error in x (cm)	Error in y (cm)	Error in theta (deg)	Error in distance (cm)
0	0	0.0	0
1	-8	0.8	8
7	-9	1.6	12
8	-9	1.1	11
10	-6	1.5	12
11	-10	1.4	15
11	-9	1.4	14
13	-5	1.4	14
12	1	1.7	13
14	1	3.1	14

Comparing Table 6 on page 28 to Table 5 on page 26, the difference in error is insignificant. Execution time increased because of the increased number of segments in the map. The only benefit is that the map doesn't have to be *fit* to the initial scan. This particular version of the iconic estimator might not be very useful, but it is easily extended in the next section to allow the robot to use a set of scans as a map, allowing it to operate outside the view of a single scan.

3.4 One to Many

Given the ability to match a scan to a single scan as described in Section 3.3 on page 27, it is straightforward to match a scan to several previously stored scans to determine the sensor pose. In fact the algorithm is essentially the same. The only modification required is that the new scan

points must be matched to several scans (taking the minimum distance) instead of a single scan. It is still a matter of matching a set of scan points to a set of line segments.

3.4.1 Results

Once again the coal mine dataset is used to demonstrate the operation of the one-to-many iconic pose estimator. An initial scan is acquired (the leftmost pose). The subsequent scans are matched to all previously stored scans, so for example, the 10th scan is matched to the previous 9 scans.

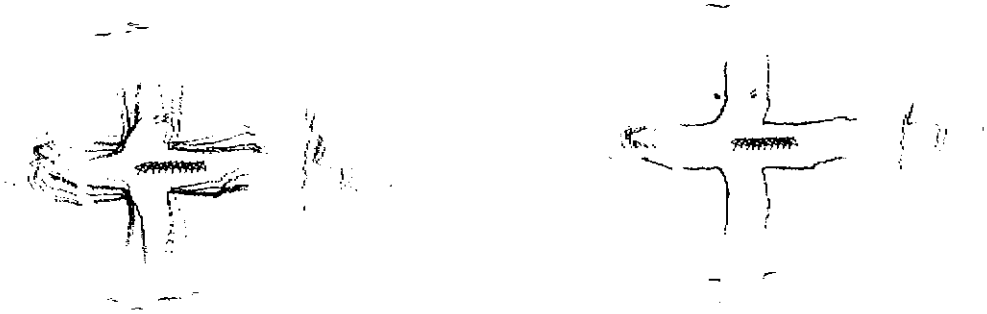


FIGURE 14. Results of matching one scan to all previous scans.

TABLE 7.

Error in x (cm)	Error in y (cm)	Error in theta (deg)	Error in distance (cm)
0	0	0.0	0
1	-8	0.8	8
7	-9	1.6	11
7	-10	1.1	12
8	-7	1.5	11
11	-11	1.3	15
9	-12	1.3	15
12	-9	1.3	15
10	-7	1.4	12
13	-10	2.6	16

3.5 Multiple Simultaneous Pose Estimation

An algorithm for matching one scan to many scans was described above. The result of this matching was an improvement in the pose estimate of one scan. A straightforward extension of this algorithm allows many scans to be simultaneously matched to each other, resulting in an improvement in the pose estimate of all of the scans involved (except at least one, which must remain fixed). However, you don't get something for nothing -- the computation time is greatly increased.

3.5.1 Results

FIGURE 15. Results of matching many to many.

TABLE 8. Error resulting from matching many to many.

Error in x (cm)	Error in y (cm)	Error in theta (deg)	Error in distance (cm)
0	0	0.0	0
3	-6	0.9	7
7	-7	1.7	10
8	-8	1.2	11
8	-4	1.6	9
11	-9	1.4	14
10	-9	1.4	13
12	-7	1.4	14
10	-4	1.5	11
13	-7	2.7	15

3.6 Comparison to Feature-Based Method

A comparison was performed using a dataset from the Cyclone sensor which formed the basis of Gonzalez's thesis [Gonzalez93]. It compared his implementation of the iconic estimator to my feature-based estimator. Though his implementation is slightly different, the comparison is still valid. The conclusions are presented here. For details, refer to [Shaffer92b]. The two pose estimation methods were compared with respect to six metrics. They are listed in no particular order, since their relative importance depends on the application.

- Accuracy

  Accuracy is important to reduce the chance of robot collisions and to position equipment or sensors for interaction with the environment. However, sometimes accuracy must be exchanged for improvement in another metric.

- Processing Time

  Processing time is measured from the time the range scan is acquired to the time that the pose estimate is available. Processing time becomes more important as the robot moves more quickly to provide rapid feedback for stable control.

- Number of Range Data Points Per Scan

  The number of range data points taken per scan mostly affects processing time, but it also affects the robustness of the estimator: if fewer points are required, the algorithms can be more selective about which points to use in the calculations.

- Robustness to Errors in the Map

  Robustness to errors in the map enables an estimator to work in less than ideal environments, where some objects which aren't in the map might move about, such as people or other pieces of mobile equipment.

- Sensitivity to Initial Pose Estimate

  Sensitivity to the initial pose estimate is important when the dead reckoning capabilities of the robot are poor.

- Environment Models

  An ideal pose estimator would be able to model any environment. Environment models which can only coarsely approximate the environment yield poor results.

Both work well for the environment used in this comparison, but each of them outperforms the other in at least one aspect of the comparison. The feature-based estimator, in general, is faster than the iconic estimator and does not require a good initial heading estimate. The iconic estimator can use fewer points than the feature-based estimator, can handle less-than-ideal environments, and is more accurate. Both estimators are robust to some error in the map. An ideal 2D pose estimation system would possess all the good qualities of both the feature-based and the iconic estimators.

3.7 Discussion

There are even more variations of the iconic estimator than were presented in this chapter. The ability to customize an iconic estimator is one of the strengths of the method. The number of points used in the calculation can be reduced to attain faster cycle time. The type of map used can be either hand-measured, automatically-generated, or simply raw scan data. Several scan poses can be refined simultaneously with respect to each other.

There are two major drawbacks. The correspondence error is calculated by summing up the contribution of each scan point. There should probably be some weighting of these error terms based on density of scans points, angle of incidence, or distance from the scanner. The second drawback is that this system doesn't provide very good maps. A set of overlapping scans is not a very easily used map representation. It could be easily converted into a binary pixel map, which might be useful. Some image processing could fit polylines to the boundaries of filled regions to generate a polyline map. This is a roundabout way of generating a line segment map. There must be a better way. That's the topic of the next chapter.

4 Contour Mapping Theory

In this chapter a map representation which we call "contours" is introduced, and a contour-based map-building system is described. Our definition of contour map should not be confused with a contour map that shows lines of constant elevation. We use the term "contour" to describe a planar curve on the surface of a robot's workspace. If the workspace is completely enclosed with no obstructions, a complete map of the robot's workspace can be represented by one closed contour. In general, a map consists of several contours. Geometrically, a contour is the intersection of a horizontal plane at sensor height with the scene.

The scan maps from Chapter 3, though they are complete and detailed, are neither compact nor concise. Because a scan map consists of the set of all scans acquired, it is forever growing. If new scans are acquired in areas that have been scanned before, they provide new information but the map continues to grow unless old scans are thrown out. The larger problem is conciseness. Most of the area mapped will be scanned at least twice, giving two scans for one section of wall for example. The two scans are not going to be identical, and might be significantly different. When a user of this map, say a path planner, needs to know where the wall is, which scan does it use? Both scans can't be correct. A conservative approach would be to assume that the wall is somewhere between the two scans and to create a "thick" representation of the wall by filling in the area between the two scans. Another approach, the one we choose, is to take an educated guess as to where the wall actually is by fitting a contour to the two scans. If a third scan is acquired that agrees with the first scan, this would add credibility to the first scan and the contour would be moved closer to the first and third scans. The estimate of the contour location is improved with every new scan. The ultimate payoff is that similar to the multiple simultaneous iconic pose estimation system, all the stored scans can be re-fit to the map, and the map can be re-fit to the scans. This iteration can continue until it converges to the desired level of accuracy. Then the scans can be discarded, leaving a complete, detailed, compact, concise map of the area that was scanned.

The contour representation has advantages over traditional line segment representations. In an unstructured environment, a line segment representation is of limited utility. Long line segments can only roughly approximate the scene. There can be gaps between neighboring line segments where there should not be. There can be several line segments overlapping each other. Binary pixel-based representations can be constructed from line segment maps or contour maps.

Contours are represented by curves, but how are curves represented? There are many possible ways. To keep it simple, we chose to use a polyline approximation for curves which represent contours. There are two key differences between this polyline contour map representation and traditional line segment maps. First, the contour line segments are short, allowing modelling of curved contours. Second, there are no gaps between neighboring contour line segments. Contours could be represented using splines or other such components. In fact, there is nothing to prevent mixing lines and arcs for example, except that the benefit probably doesn't justify the added complexity.

A polyline contour is represented as a doubly linked list of polyline vertices, in a counter-clockwise order. The ordering is imposed to indicate which side the contour was sensed from. The list can be circular, forming a polygonal contour.

Building scan maps was accomplished in the single operation of pose estimation. There was no additional processing of the scans to integrate them into a map representation such as contours. Constructing contour maps requires two additional steps: map expansion and contour refinement. For now, the environment is assumed to be static, although this assumption is lifted in later in this chapter. It is assumed that the scanner is stationary while scanning or that the scanning speed is so fast that the movement during scanning is negligible. The initial pose estimate can be obtained by a number of methods: dead reckoning of the sensor's mobile platform, rough estimate from system operator, an INS (if you've got one onboard anyway), GPS if you're outside, or just by pacing it off. In a real robotic mapping system, the method of choice would be dead reckoning off of the actuators of the mobile base, because it requires no human interaction, it's not prohibitively expensive, and it's accurate enough. The initial estimate is only required to give the pose estimation software a good enough starting point so that it gets the right match between scan and map. The pose estimator then computes a much more accurate pose estimate.

Map expansion occurs when a part of the scene is scanned for the first time. There must be some overlap between the new scan and the map to enable pose estimation, but some part of the new scan should not overlap the map, allowing expansion of the map. New contours are roughly fit to parts of the new scan that don't overlap existing contours. After this map expansion step, all new contours and all contours that were overlapped by the new scan are refined. The new contours need to be refined because the initial contour fit is rough. Contours that are overlapped with new scan data are refined to take advantage of the new information.

These steps form a map-building cycle which is:

1. Acquire scan and initial pose estimate.
2. Refine pose estimate.
3. Expand map.
4. Refine contours.
5. Move sensor.

These five steps are described in this chapter, followed by a description of map refinement, and then extensions to allow mapping of dynamic environments. Some examples are presented in this chapter, but the main results are presented in Chapter 5.

4.1 Acquiring a Scan

There is an assumption that the sensor has a sufficient maximum range, field of view, and density of data to allow matching overlapping parts of scans. The exact values for maximum range and density of data depends upon the scale of the environment. However, a 360 degree field of view is

advantageous regardless of scale. Both of the real sensors used in this thesis and all of the synthetic sensors are 360 degree sensors. The two real sensors are described in Appendix B and Appendix C. Synthetic scan generation is described in Chapter 5.

Regardless of whether the scan is read real-time from the sensor, read from disk, or generated synthetically, there is some preprocessing done before the scan is handed off to the rest of the system. First the scan is trimmed to the user-specified minimum and maximum ranges, with data points outside this interval discarded. Next the scan is run through a median filter to remove outliers. The size of the filter is selectable, and is most useful with the older sensor, Cyclone, which seems to be more susceptible to outliers caused by mixed points. With the newer sensor, Typhoon, the median filter is often disabled by setting the filter size to 1. The third and most important step in preparing a scan for further processing is segmentation into "partial scans". A partial scan is a sequential subset of points in the scan bounded by discontinuities which exceed a threshold and specified by the start index and end index within the array of scan points. The assumption is that neighboring scan points should be connected because they densely sample part of the same object or portion of the environment. However, if the spacing between neighboring scan points exceeds the threshold, then either an occlusion has occurred, the density has decreased because of a shallow angle of incidence, or the distance to the object is so great that the angular spacing between consecutive range measurements results in a large separation.

4.2 Pose Estimation

Pose estimation is the computation or refinement of the sensor's 2D pose (x, y, θ). We assume that only a rough initial estimate of the sensor's pose is available. For example, the robot is told to execute a trajectory and the robot controller reports successful completion of the trajectory. An approximate position of the robot can be determined by dead reckoning. It is further assumed that no means of pose estimation is possible except to match the new scan to the map. The process of "matching" a scan to the map is actually an optimization process. Because the sensor data is imperfect, there is no perfect match; the best that can be done is to minimize the error in the fit of the scan to the map. Pose estimation is accomplished in two major steps: first determining the correspondence between a scan and the map and second adjusting the sensor pose to minimize the error of the correspondence. These two steps are described below.

For purposes of navigation, where pose estimation speed can be traded for pose estimation accuracy, a different method of pose estimation would probably be used. The method described in this chapter is for purposes of mapping, when maximum accuracy is more important than computation time.

4.2.1 Determining the Correspondence Between a Scan and the Map

The correspondence between a scan and the map is computed by attempting to match each of the scan's component partial scans to every contour in the map. Thus the unit operation is to compute

the correspondence between a partial scan and a contour. This "correspondence engine" is described in the remainder of this subsection.

Every scan point corresponds to a reflection of the laser beam from some part of the scene, which may appear in the map as a point on a contour or as nothing at all (if an unmapped area has been scanned). We assume that a scan point corresponds to the nearest contour segment, subject to three conditions:

1. The scan point must lie between the two imaginary perpendicular lines drawn through the endpoints of the contour segment.
2. The distance between the scan point and contour segment must be below a threshold.
3. The sensor position must lie on the correct side of the contour segment (to prevent matching scans from one side of a wall to a contour that represents the other side of the wall).

Figure 16 on page 36 shows a partial scan (the set of points) and a contour (the polyline). The shaded rectangle is the region specified by conditions 1 and 2 for one of the segments of the contour. The length of the shaded rectangle is the length of the contour segment. The width of the rectangle is twice the correspondence threshold, $d_{corresp}$. The third condition is satisfied because sensor pose P is on the proper side of the contour segment. This test is possible because contour points are always stored in a counter-clockwise order. The correspondences are illustrated with lines that connect scan points to the contour segment.

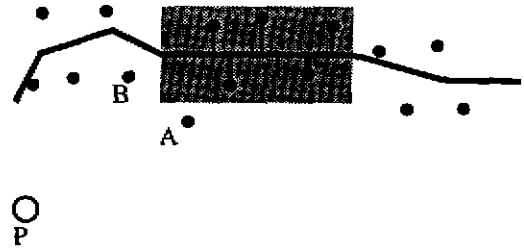


FIGURE 16. A partial scan (the set of points) and a contour (the polyline). The shaded rectangle is the region specified by conditions 1 and 2 for one of the segments of the contour. The length of the shaded rectangle is the length of the contour segment. The width of the rectangle is twice the correspondence threshold, $d_{corresp}$. The third condition is satisfied because sensor pose P is on the proper side of the contour segment. This test is possible because contour points are always stored in a counter-clockwise order. The correspondences are illustrated with lines that connect scan points to the contour segment. Point A fails condition #2 and point B fails condition #1.

To compute the correspondence between a partial scan and a contour, all partial scan points have to be checked against all of the contour segments. If all three conditions are met, and the scan point isn't already matched to a closer contour segment, the scan point and contour segment are paired.

If the scan point is already matched to a more distant contour segment, that pairing is overwritten, because we assume that a scan point corresponds to exactly one point in the scene. After all partial scan points have been checked and possibly matched to the contour, the pairings for each contour segment are sorted such that the set of scan points that correspond to a particular contour segment are ordered according to their position along a vector parallel to the contour segment. This is necessary to support computation of the error in the fit between the partial scan and the contour.

This simple algorithm becomes very inefficient when the map is much larger than a scan's maximum range. To counteract this problem, a simple speedup was implemented by maintaining bounding circles around all contours and partial scans. This allows a computationally inexpensive test to be performed between a partial scan and a contour to reject pairings before conducting the exhaustive test. If the distance between the centers of the bounding circles of a partial scan and a contour are more than ($r_{pscan} + r_{contour} + d_{corresp}$) then no point in the partial scan can correspond to any segment in the contour.

It should be noted that, even with the speedup, this correspondence determination is computationally expensive. If the map is fixed and several pose estimates are to be computed in the same area, it might be practical to build a lookup table to greatly speed up the correspondence determination. However, since we are building the map and therefore frequently modifying it, a lookup table is not practical.

4.2.2 Correspondence Error Minimization

Minimization of correspondence error is a nonlinear optimization problem with 3 degrees of freedom (x, y, θ). We use a line search technique [Dahlquist74] to minimize the correspondence error. Line search requires computation of the error and its gradient and then performing a binary search in the direction of the gradient to find a delta for all of the degrees of freedom which will decrease the total error. The parameter which represents the distance along the search line is λ and will be referred to in later sections. Line search has the desirable property that if modifying one of the degrees of freedom has little or no effect on the total error, then that degree of freedom is not modified. Some other techniques behave oppositely, which is not acceptable for this application. To illustrate this, imagine matching a range scan of an infinitely long corridor that is formed by two parallel walls. Assume that the map is perfect and consists of two parallel lines at the proper spacing. It is impossible for any map-matching system to determine the component of the position along the direction of the corridor. There is nothing to match to. The best that a matching system can do in this situation is to fail gracefully. With line search, there is graceful failure -- the initial estimate of the coordinate along the direction of the corridor is unchanged by the error minimization. The Gauss-Newton method used in Chapter 2 fails miserably in this case. The slightest nudge (or numeric inaccuracy) can cause the indeterminable coordinate to jump by large distances on each iteration of the error minimization. This is the main reason for switching to line search.

Bringing contours into the system also required a change in the error metric. This error metric needed to serve two purposes: pose estimation and contour fitting. And not only fitting one contour

to one scan, but fitting one contour to many overlapping scans. There were several "obvious" error metric definitions that didn't work well. The simple distance between point and line metric failed because it gave undue "weight" to areas where the density of scan points was higher. That screws up both pose estimation and contour fitting. To compensate, the distance was changed to area by computing the area of the trapezoid formed by two neighboring scan points and their corresponding section of contour segment. However, this quickly failed because there was no unique solution. The area squared was the next logical choice. That failed too, because there was still a density problem, though the reverse of the density problem with the distance metric. What worked was to take the integral of the squared distance. Figure 17 on page 38 illustrates these four metrics, and why the first three failed and the fourth succeeded.

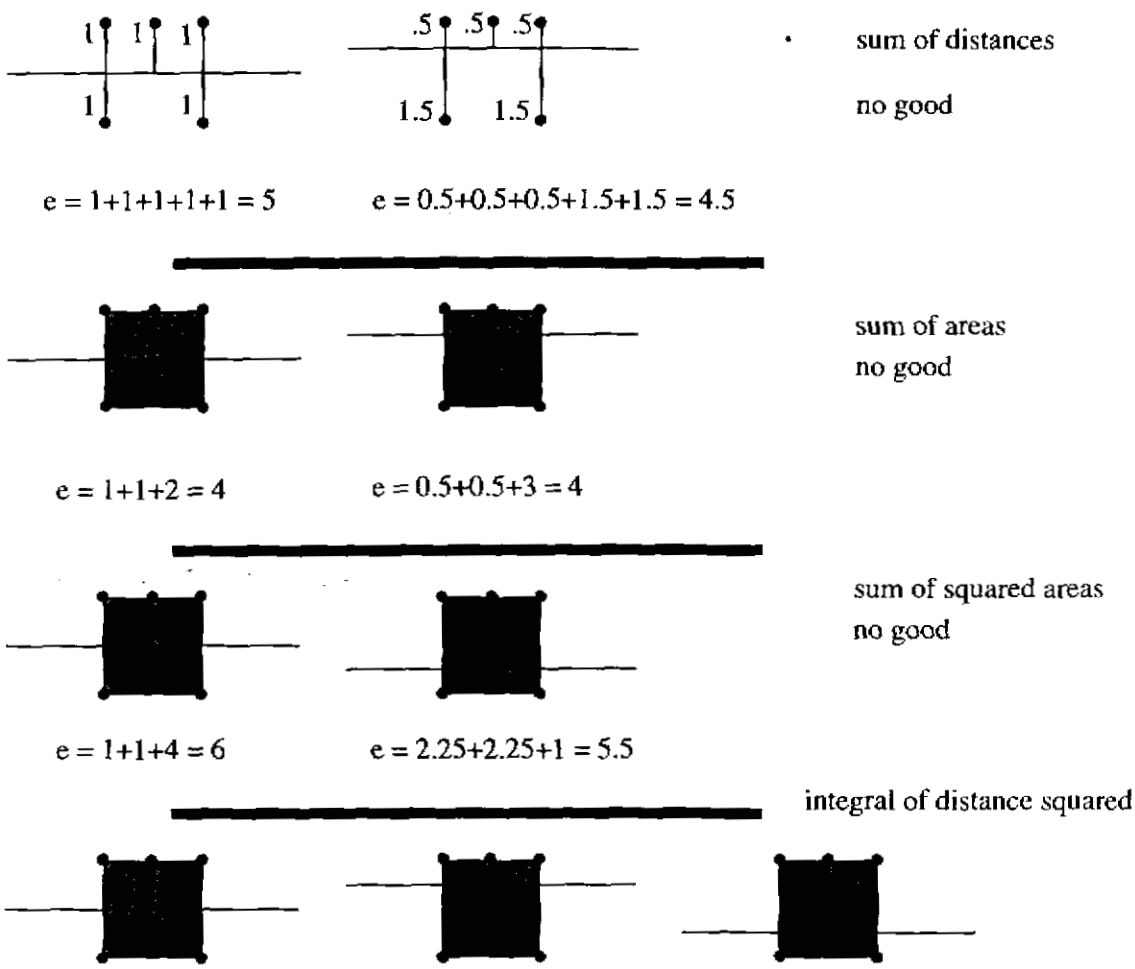


FIGURE 17. Error metrics that were evaluated. The first three failed. The examples show a case where the total error decreases instead of increasing. The fourth case succeeds, and two examples are provided.

Correspondence error is defined by treating the partial scan points as a polyline (as is the contour) and calculating the integral of the square of the perpendicular distance between the partial scan and the contour. Note that the imaginary polyline formed by connecting the partial scan points that correspond to a contour segment is generated from the sorted list of correspondences produced by the correspondence engine. Therefore the order of the scan points is not necessarily the same as their ordering within the partial scan (which is dependent upon the sensor pose). Correspondence error is computed one contour segment at a time by computing the above integral between the contour segment and the polyline formed by the sorted corresponding scan points. Figure 18 on page 39 and Equation 1 show the contribution, f, to the total error, F, due to scan points (x_i, y_i) and (x_j, y_j) and the contour segment with endpoints (x_s, y_s) and (x_e, y_e). Remember (x, y, θ) is the scan pose estimate and the vectors r and α are the range and azimuth vectors (the scan).

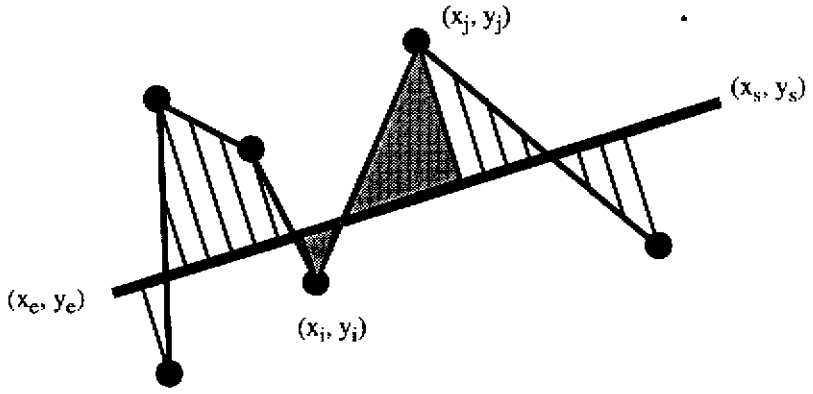


FIGURE 18. The shaded area shows the integration region for the error contribution.

(EQ 1)

$$
\begin{aligned}
x_i &= x + r_i\cos(\theta + \alpha_i) \\
y_i &= y + r_i\sin(\theta + \alpha_i) \\
x_j &= x + r_j\cos(\theta + \alpha_j) \\
y_j &= y + r_j\sin(\theta + \alpha_j)
\end{aligned}
\qquad
\begin{aligned}
d_x &= x_e - x_s \\
d_y &= y_e - y_s \\
h &= \sqrt{d_x^2 + d_y^2} \\
a &= (-d_y)/h \\
b &= d_x/h \\
c &= -ax_s - by_s
\end{aligned}
\qquad
\begin{aligned}
l &= b(x_j - x_i) - a(y_j - y_i) \\
d_i &= ax_i + by_i + c \\
d_j &= ax_j + by_j + c \\
f &= \int (ax + by + c)^2 dl \\
f &= |l| \cdot \left(d_i^2 + d_i(d_j - d_i) + (d_j - d_i)^2/3 \right)
\end{aligned}
$$

There are three degrees of freedom (x, y, θ),and the gradient therefore has three components. The gradient calculation equations appear in Equation 2.

(EQ 2)

$$k_{4_i} = br_i \cos(\theta + \alpha_i) - ar_i \sin(\theta + \alpha_i)$$

$$\frac{\partial f}{\partial x} = |l| a (d_i + d_j)$$

$$\frac{\partial f}{\partial y} = |l| b (d_i + d_j)$$

$$\frac{\partial f}{\partial \theta} = |l| \left(2k_4 d_i + d_i l + k_4 (d_j - d_i) + \frac{2 (d_j - d_i) l}{3} \right) - (\mathrm{sgn}(l)\ (d_j - d_i)) f$$

The initial λ value for the line search is shown in Equation 4, where F is the total error and $d_{corresp}$ is the correspondence distance threshold. This initial λ limits the search such that the maximum change in x or y is $d_{corresp}$. The resolution of the search is typically submillimeter.

(EQ 3)

$$\lambda = d_{corresp} / MAX\left(\left|\frac{\partial F}{\partial x}\right|, \left|\frac{\partial F}{\partial y}\right|, \left|\frac{\partial F}{\partial \theta}\right| \right)$$

The minimization loop appears below:

1. Compute correspondence
2. Compute x and y components of gradient
3. Perform line search to refine x and y
4. Compute theta gradient
5. Perform line search to refine theta
6. Update pose estimate
7. Check termination conditions

It is necessary to recompute the correspondence on each iteration because updating the pose estimate changes the correspondence. On the first iteration the $d_{corresp}$ is set fairly high (typically 3 m), according to the maximum expected error in the initial pose estimate. On the second and later iterations, this threshold is decreased (typically by a factor of 0.9) to the final error expected based mostly on the expected noise level in the sensor data (typically 0.1 m).

It became clear that computing the linear degrees of freedom separately from the angular degree of freedom was necessary. When all three were computed simultaneously, convergence was very slow. The nature of the error is that it is very sensitive to changes in theta and much less so to changes in x and y, so there was oscillation around the correct orientation and very slow convergence to the proper x and y.

There are four termination conditions for the pose estimation loop:

- the gradient is small

- the computed λ is small
- the change in pose is small
- iteration limit is reached

Unfortunately, there has to an iteration limit because the error is not guaranteed to monotonically decrease throughout the minimization process. The reason for the nonmonotonicity is that the correspondence is recomputed at each iteration.

The resulting accuracy in the sensor pose estimate depends upon the accuracy of the sensor data, the quality of the map, the correctness of the correspondence determination, and the validity of the correspondence error metric.

A pose estimation example appears in Figure 19 on page 41.

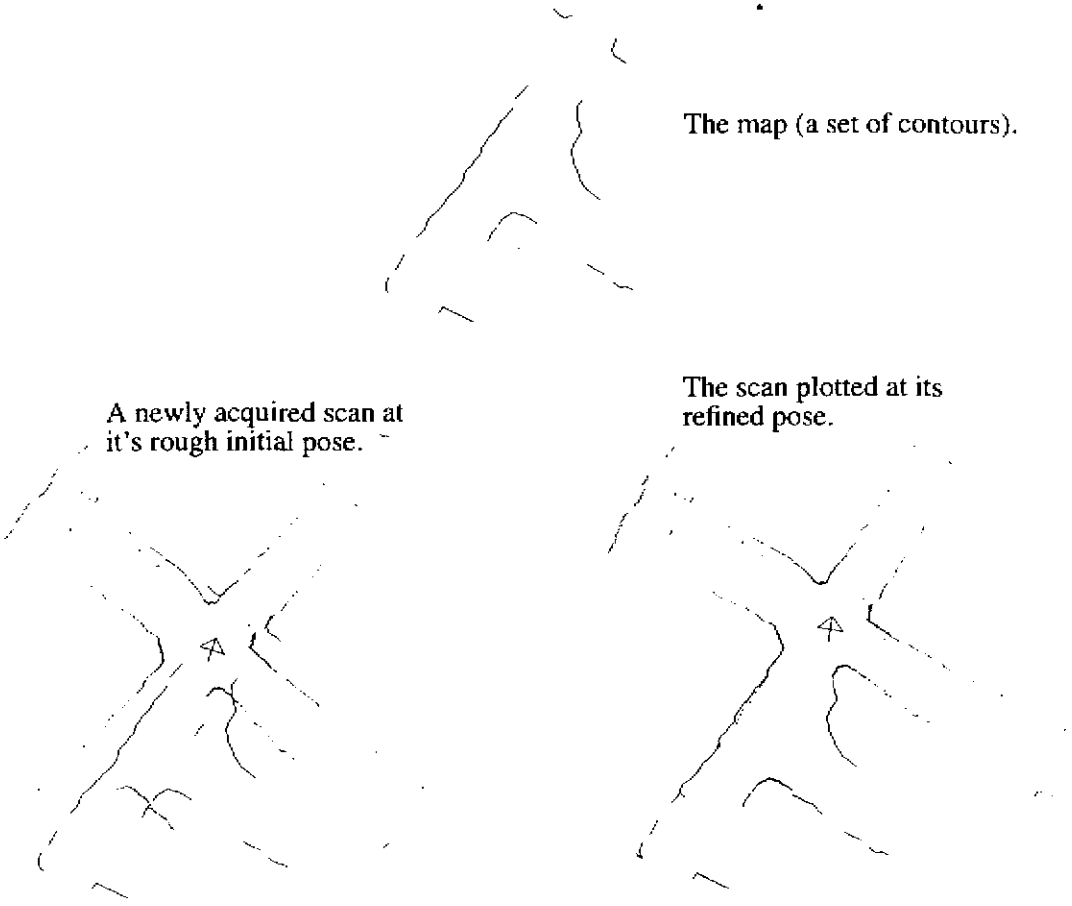


FIGURE 19. An example of pose estimation by matching a scan to a set of contours.

4.3 Map Expansion

After the pose of a new scan has been computed, the next step in the map-building cycle is to expand the map by creating new contour segments where the new scan does not overlap an existing contour. For now, we assume the world is static, so after computing the correspondence between a scan and the map, any scan points that do not correspond to the map are the result of viewing previously unscanned areas. One of three operations is performed to incorporate this new scan data into the map: create a new separate contour, extend an existing contour, or connect two existing contours with some new contour segments.

4.3.1 Creating a Contour

When no points of a partial scan correspond to any contour in the map, a new contour is created. We use a polyline contour representation with "small" nominal segment length. The length of the segment depends upon the quality of the range data, the desired level of detail, and the required speed. The first point of the contour is the first point of the partial scan, and the last point of the contour is the last point of the partial scan. The intermediate vertices of the contour also coincide with partial scan points, and they are determined by a minimum contour segment length requirement. After determining each contour vertex, the partial scan points are checked sequentially until a point is encountered that is at least the minimum segment length distant from the previous contour point. That point becomes the next contour point. This process is repeated until the last point of the partial scan is encountered. This is a very rough fit of the contour to the scan data, and it is greatly improved the first time the contour is refined.

4.3.2 Extending a Contour

When some point(s) of a partial scan corresponds the last segment of a contour, but all remaining points in the partial scan do not correspond to anything, the contour must be extended. New contour segments are created starting where the last correspondence occurred and ending at the last point of the partial scan. Extending the beginning of a contour happens analogously, adding new contour segments onto the beginning of the contour until the first point in the partial scan.

4.3.3 Connecting Contours

When there is at least one correspondence between a partial scan and the last segment of a contour, then some unmatched partial scan points, then correspondence to the first segment of a contour, the two contours should be connected. The partial scan overlaps the end of one contour and the beginning of another, which indicates that they are actually the same contour in the scene. The end of the first contour is extended using the above contour creation algorithm and connected to the beginning of the second contour. Note that this might connect the end of a contour to the beginning of the same contour, forming a closed (polygonal) contour.

4.4 Contour Fitting

After contours are created or modified in the Map Expansion step of the map building cycle, or if contours are overlapped by new scan data, they must be refined to better fit the scan data. An error minimization is performed using line search where the contour vertices are the variables in the minimization equations. As with pose estimation, there are two main components to the error minimization: computing correspondence and line search.

4.4.1 Determining Correspondence Between a Contour and Scans

To determine the correspondence between a contour and a set of scans, the Correspondence Engine (described in Section 4.2.1 on page 35) is run on the contour and every existing partial scan.

4.4.2 Minimizing Correspondence Error to Improve Contour Fit

To improve the fit of a contour to scan points, we use the line search method described in Section 4.2.2 on page 37 to minimize the correspondence error. The error computation is also described in Section 4.2.2 on page 37. There are 2N degrees of freedom (the coordinates of the N contour vertices), and the gradient therefore has 2N components. The equations for the gradient appear in Equation 4.

(EQ 4)

$$\frac{\partial f}{\partial x_s} = \left(\frac{1}{h}\right)\left(|l|\left(d_i\left(2\left(y_e - y_i\right) + \left(y_i - y_j\right)\right) + \left(d_j - d_i\right)\left(\left(y_e - y_i\right) + \frac{2\left(y_i - y_j\right)}{3}\right) + 2bf\right) + \operatorname{sgn}\left(l\right)\left(x_i - x_j + bl\right)\right)$$

$$\frac{\partial f}{\partial y_s} = \left(\frac{1}{h}\right)\left(|l|\left(d_i\left(2\left(x_i - x_e\right) + \left(x_2 - x_i\right)\right) + \left(d_j - d_i\right)\left(\left(x_i - x_e\right) + \frac{2\left(x_2 - x_i\right)}{3}\right) - 2af\right) + \operatorname{sgn}\left(l\right)\left(y_i - y_j - al\right)\right)$$

$$\frac{\partial f}{\partial x_e} = \left(\frac{1}{h}\right)\left(|l|\left(d_i\left(2\left(y_i - y_s\right) + \left(y_j - y_i\right)\right) + \left(d_j - d_i\right)\left(\left(y_i - y_s\right) + \frac{2\left(y_j - y_i\right)}{3}\right) - 2bf\right) + \operatorname{sgn}\left(l\right)\left(x_j - x_i - bl\right)\right)$$

$$\frac{\partial f}{\partial y_e} = \left(\frac{1}{h}\right)\left(|l|\left(d_i\left(2\left(x_s - x_i\right) + \left(x_i - x_j\right)\right) + \left(d_j - d_i\right)\left(\left(x_s - x_i\right) + \frac{2\left(x_i - x_j\right)}{3}\right) + 2af\right) + \operatorname{sgn}\left(l\right)\left(y_j - y_i + al\right)\right)$$

The initial λ value for the line search is found by the formula:

(EQ 5)

$$\lambda = d_{corresp}/MAXGRADIENT$$

The minimization loop appears below:

1. Remove degenerate contour segments
2. Compute correspondence

3. Compute gradient
4. Perform line search
5. Update contour vertices

The first step, removal of degenerate contour segments is necessary because in the course of fitting contours to scan data, some segments become very small and can cause the gradient computation to blow up. Additionally, if a contour segment is smaller than the noise level in the range scans, it allows the sensor noise to be propagated into the contour. In the current implementation, no new contour segments are added during the fitting process, although this is desirable in some cases and might be added in the future.

It is necessary to recompute the correspondence on each iteration because updating the contour vertices generally changes some correspondences, meaning some scan points correspond to different contour segments or to nothing at all.

There are five termination conditions:

- the number of contour segments is fewer than 2
- the maximum gradient is small
- the initial λ is very small
- the maximum change in any degree of freedom between iterations falls below a threshold
- iteration limit is reached

An example of contour refinement appears in Figure 20 on page 45.

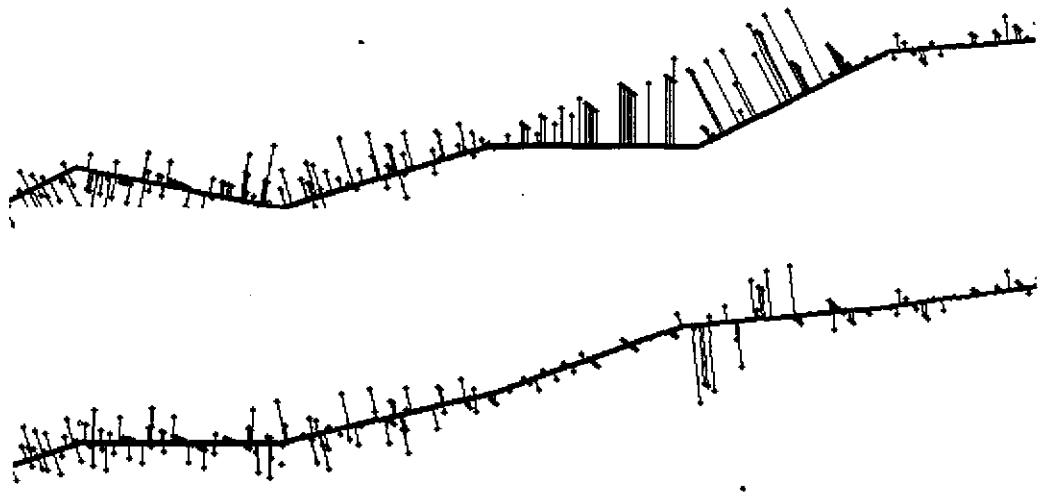


FIGURE 20. A portion of a contour overlaid on its corresponding scan points before and after contour refinement.The thin line segments are an indication of the correspondence between scan point and line.

4.5 Sensor Placement

The last step in the map-building cycle is moving the sensor to a new vantage point. Automatic sensor placement was not addressed in this thesis. Some researchers have looked into this, Sedas [Sedas93] and others. For the work presented in this thesis, sensor placement was done manually, choosing what seemed to be the obvious locations for scanning. In the mine, this amounted to taking a scan at each "intersection" and usually halfway between intersections. The density of scans required and the exact layout depends upon the geometry of the particular environment being mapped. One simple algorithm for automatic sensor placement is to record a scan at a chosen spatial interval.

4.6 Map Refinement

A contour map is constructed by iterating the map-building cycle presented at the beginning of this chapter, inserting one scan at a time, and extending the map. However, it is inevitable that some error will exist in the map since only relative pose estimation is performed. Without external reference, the best course of action is to reduce the speed that the error accumulates. Loops in the layout of the environment can be used to reduce the total error. Our technique is to iterate between re-refining each of the sensor poses, and then re-fitting all of the contours. Contours and scans are very slowly "pulled" toward each other as the total error in the map is gradually reduced. This can be an off-line process, so execution time is not of great importance.

An example appears in Figure 21 on page 46 and Figure 22 on page 47. A map formed from 9 scans is shown. The most apparent error is indicated with an arrow. At that location, a concrete block wall was scanned from both sides. Two contours were fit to the wall, which is correct -- one for each side of the wall. The error is that, in the figure, the two sides of the wall overlap each other, which is physically impossible. Looking at Figure 22 on page 47, after map refinement, the two sides of the wall are in the correct position in relation to each other. The refinement ran overnight on a Sparc 20. A graph of the improvement in the 9 poses appears in Figure 23 on page 48.

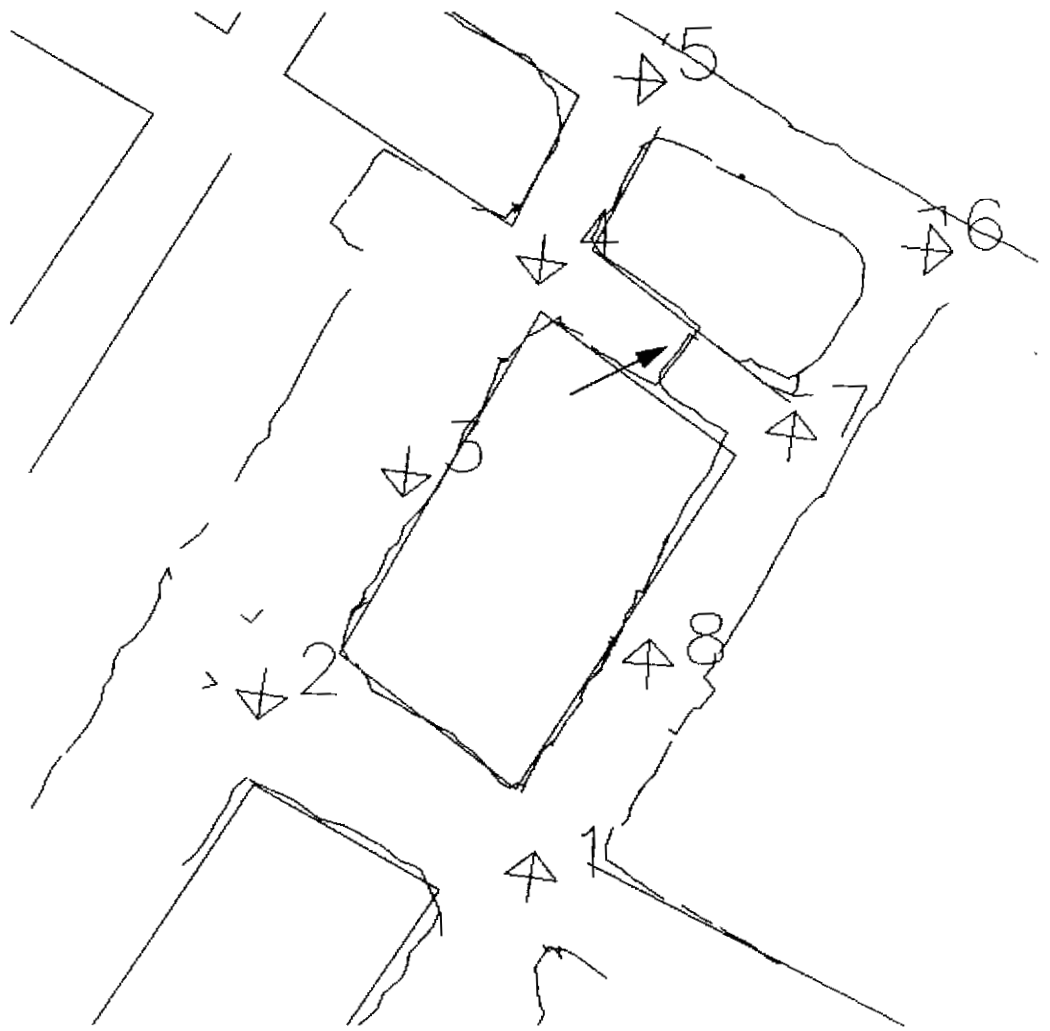


FIGURE 21. Before map refinement.

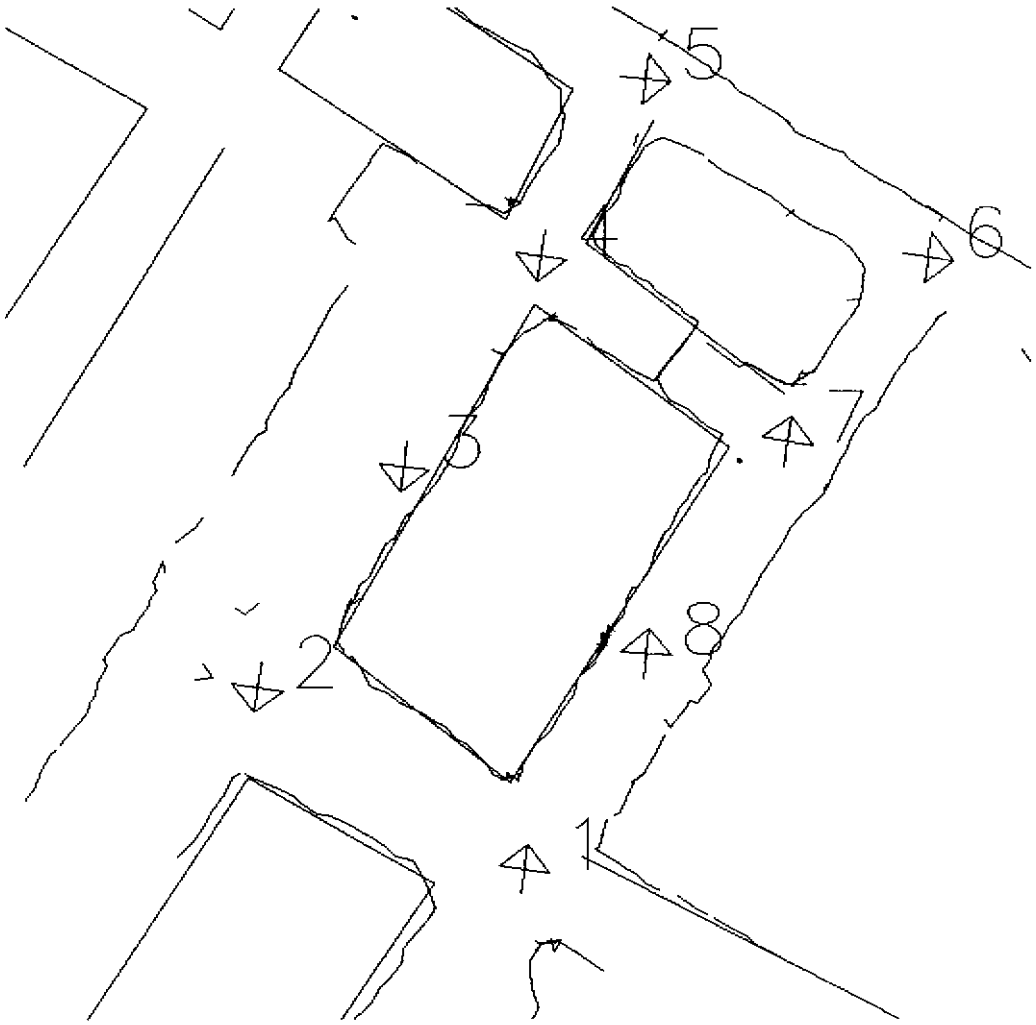


FIGURE 22. After map refinement.

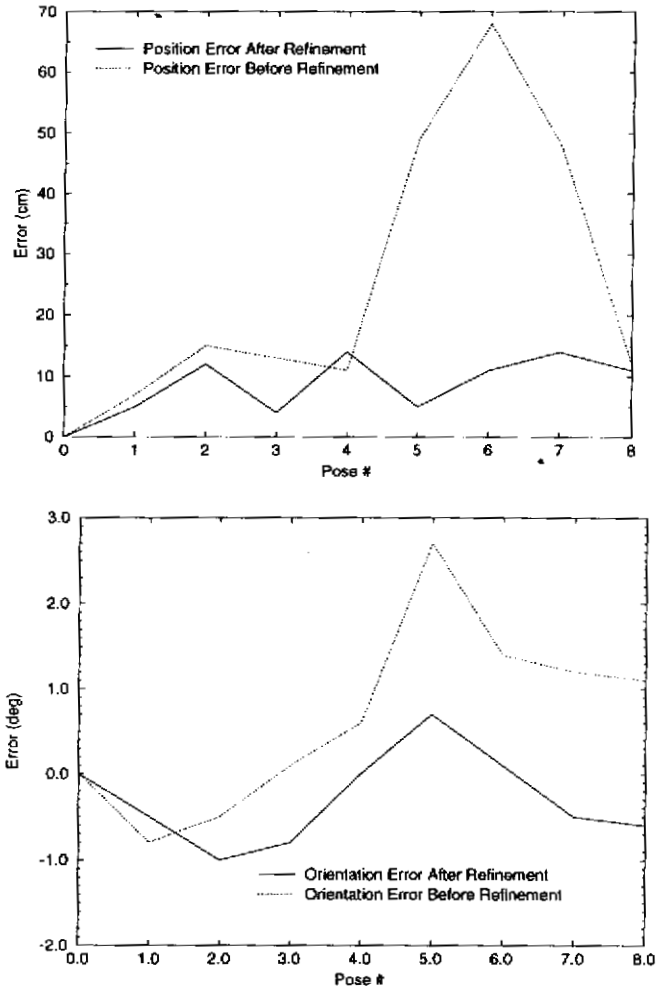


FIGURE 23. Improvement in sensor poses as a result of map refinement.

4.7 Dynamic Environments

Up to this point, it's been assuming that the environment is static. What happens when this assumption is lifted? No longer can we scan an area once and never need to look again. Every time we pass a given area, we must look and make sure nothing has changed, or modify our map if something has changed. There are several philosophies in dealing with a "dynamic" environment. Of course all environments are dynamic at some level. For our purposes, a dynamic environment is one that exhibits changes over time that are detectable by our sensor. Philosophies - One could assume that only new objects are appearing, so only add to and extend the map. (This is necessary anyway because you have to build the map one scan at a time.) One could assume that objects only disappear. In general, both are occurring. One philosophy is to say that if the information

(scan or map segments) is older than n, it is too out of date to be any good. Throw it away. This is fine if your purpose is navigation, and you only care about the immediate surroundings (stuff that was scanned recently).

So then, what are the ways to deal with this dynamic environment? Moravec's grid maps handle dynamic environments gracefully -- it makes no difference if something appears or disappears. There are no conditions or special cases detected. The probability for the effected cells is simply updated. The downside is that it takes a while for this information to sink in. The picture is fuzzy for a while. One new scan cannot wipe out several old ones. That is good if the new scan is wrong for whatever reason, but it's bad if the new scan is correct.

Map expansion was included in the previous chapter because it's necessary in a map building system, even in completely static environments. Scans of previously unscanned areas require addition of new map segments. However, there are several more cases that crop up when the environment is no longer static. In a static environment, all that is necessary to expand the map is adding completely new contours, extending existing contours, and possibly connecting the end of a contour to the beginning of a contour. Map expansion in a dynamic environment requires all of the same capabilities as expanding a static environment plus the ability to splice into existing contours. The nuts and bolts of doing the splicing operation are straightforward. Detecting where the splicing needs to be done is the tricky part.

As a simple example, consider the situation when a previously mapped wall (say 10 meters long) has a vending machine placed against it in the middle of the wall. Assume the vending machine is slightly trapezoidal so it's possible for three faces of it to be seen from one vantage point. The next time that wall is scanned, there are a bunch of scan points that don't correspond to any part of the map. (These are the hits on the vending machine.) What do you do to the contours to reflect this change in the environment? Probably everyone would agree that the three sides of the vending machine that do not face the wall should have new contour segments fit to them, but what to do about the fourth side of the vending machine, the side that faces the wall? It's very unlikely that it will ever be scanned. Should the wall contour segments that are now occluded be deleted? Should the vending machine contour segments be connected to the wall contour segments? These are somewhat philosophical questions, so here's my philosophy. The occluded wall contour segments should be left alone and the vending machine segments should be connected to the wall contour segments. My reasoning is as follows. There is no direct evidence that the section of the wall disappeared -- it is simply occluded. If the vending machine is moved, that part of the wall is probably still there. The second half of the question (should the vending machine be spliced into the wall) is a little tougher to answer. If the wall had not been scanned before the appearance of the vending machine, the vending machine would simply be part of the wall. (Range sensors can't really tell the difference between vending machine and wall.) For this reason, I splice the vending machine into the wall to form one contour. This simple example actually illustrates all of the issues, and other examples are only subparts of this example.

Disappearing objects can be handled at two levels: the contour level and the scan point level. In other words, given a new scan, new contour segments could be fit to all of the new scan and then compared against existing contour segments to determine where discrepancies exist. The other possibility is to compare new scan points against existing contour segments. The first alternative is attractive because there is less data involved. However the interpretation becomes more difficult and prone to errors. The brute force method seems to be the better alternative, comparing new scan points to existing contour segments. However, another philosophical point surfaces - what to assume regarding the laser beam divergence. Should each scan point be modelled as a line with gaps between neighboring scan points? Or should two neighboring scan points form a triangle of coverage because of the laser beam divergence? I chose the latter because there is a significant beam divergence and we have a pretty good idea of what it is. Thus handling disappearing objects amounts to a single unit operation of intersecting a triangle with a line segment.

This remainder of this chapter details the cases that can arise and presents a few examples to illustrate these points.

4.7.1 Appearance

The approach in contour editing is to logically reason about the causes of conflicts in the map after a new scan is placed and correspondence is computed. These conflicts occur in various flavors and combinations. To detect these flavors, it is necessary to step through each new partial scan sequentially, point by point, from start point to end point, looking at correspondences (or lack of correspondence) and logically deduce what happened in the scene. At the transitions, the deduction takes place and the required surgery is performed.

When looking at the sequence of correspondence and loss of correspondence, there are eight cases that arise. These eight are actually built up from two basic occurrences: intersection with an interior segment, intersection with a start/end contour segment. Intersection, in this context, is when a scan point corresponds to a contour segment.

There are a few operations that might be required, depending on the condition that is encountered. They are all simple operations: cut, detach, extend, splice, join, create.

- Cut - divide a contour segment into two (not necessarily in half), creating two new endpoints where there was none before
- Detach - disconnect two adjacent contour segments such that the shared endpoint is duplicated
- Extend - create new contour segments starting from an existing contour point
- Splice - cut a segment and connect another segment to one of the newly created endpoints
- Join - create new contour segments starting from an existing contour point, either to start or end of partial scan

1. condition: beginning of partial scan, transition from no correspondence to correspondence to first segment of a contour

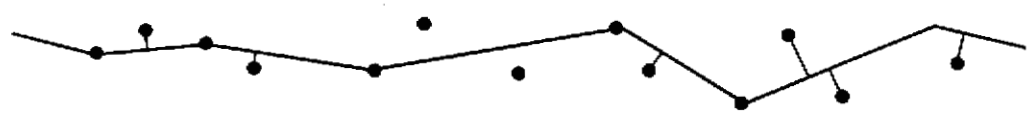
action: extend beginning of contour

2. beginning of partial scan, transition from no correspondence to correspondence to non-first segment of a contour

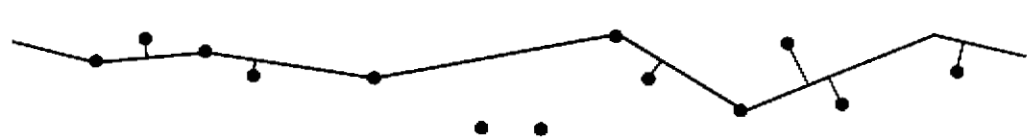
action: join contour

3. partial scan corresponds to end segment of one contour and start segment of another

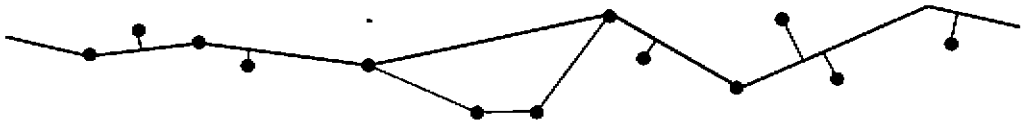
action: connect contours



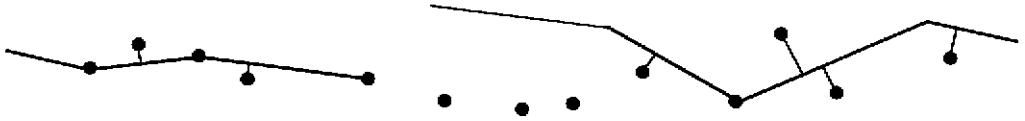
4. partial scan corresponds to two interior segments of a contour(s)



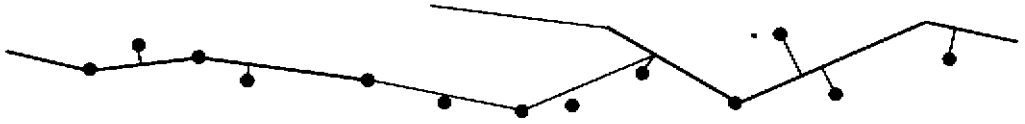
action: splice into both contour segments and create new segment(s)



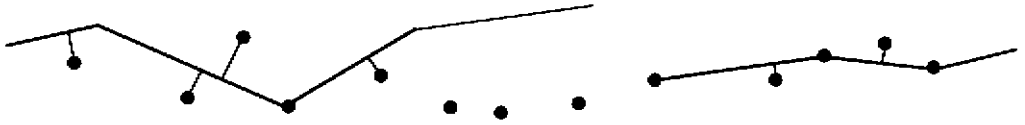
5. partial scan corresponds to end of one contour and interior segment of another



action: extend end of first contour and splice into segment

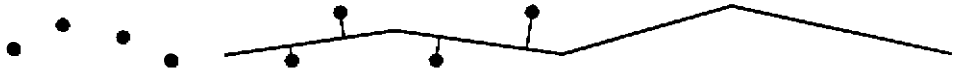


6. partial scan corresponds to beginning of one contour and interior segment of another



action: extend beginning of contour and splice into segment

7. partial scan corresponds to end segment of a contour and then nothing else



action: extend end of contour

8. partial scan corresponds to an interior segment of a contour and then nothing else

action: splice into segment and then extend end

4.7.2 Disappearance

When part of the scene disappears after it has been mapped, it will need to be scanned again and the map will need to be modified accordingly. Some researchers have used a fadeaway technique to handle this problem. This may be appropriate for highly dynamic environments, but it is a very pessimistic approach for a mostly static environment. It might also be appropriate for systems that rely on dead reckoning, and therefore the position error builds very quickly. I do not remove any data from the map due to age alone, only after conflicting scans are incorporated into the map. I assume that the laser beam divergence provides full coverage between adjacent scan points, and that the pose estimate for the scan is pretty good. So, if the triangle formed by a pair of adjacent scan points (and the pose of the scanner) overlaps a contour segment, the intersection of triangle and segment is removed from the map. More specifically, if the following conditions are true, the intersection is removed:

1. the triangle overlaps the segment such that the segment intersects the two sides of the triangle that share the scanner pose
2. the distance from scan point to segment is over a threshold (does not "correspond")
3. the view angle from the scanner pose to the segment is not too glancing (don't want to wipe out whole segments that the scanner is "looking along")

The approach is brute force and conservative.

4.8 Summary

This chapter described a contour-based mapping system for a static 2D environment and then extended the theory to handle a dynamic environment. The map is constructed incrementally by inserting range scans using a series of steps. These steps are referred to as the map building cycle. They are: scan acquisition, pose estimation, map expansion, contour refinement, and sensor placement. The first scan is fixed and contours are fit to it. The next scan is matched to the existing contours to determine its pose, and then the map is expanded using the new information provided by the new scan. In areas where the new scan overlaps existing contours, those contours are refined to take advantage of the new information. In areas where the scan does not overlap existing contours,

new contours are created. The sensor is then moved and the cycle repeats for the third and subsequent scans. Figure 24 on page 54 illustrates the first two cycles of a map-building run.

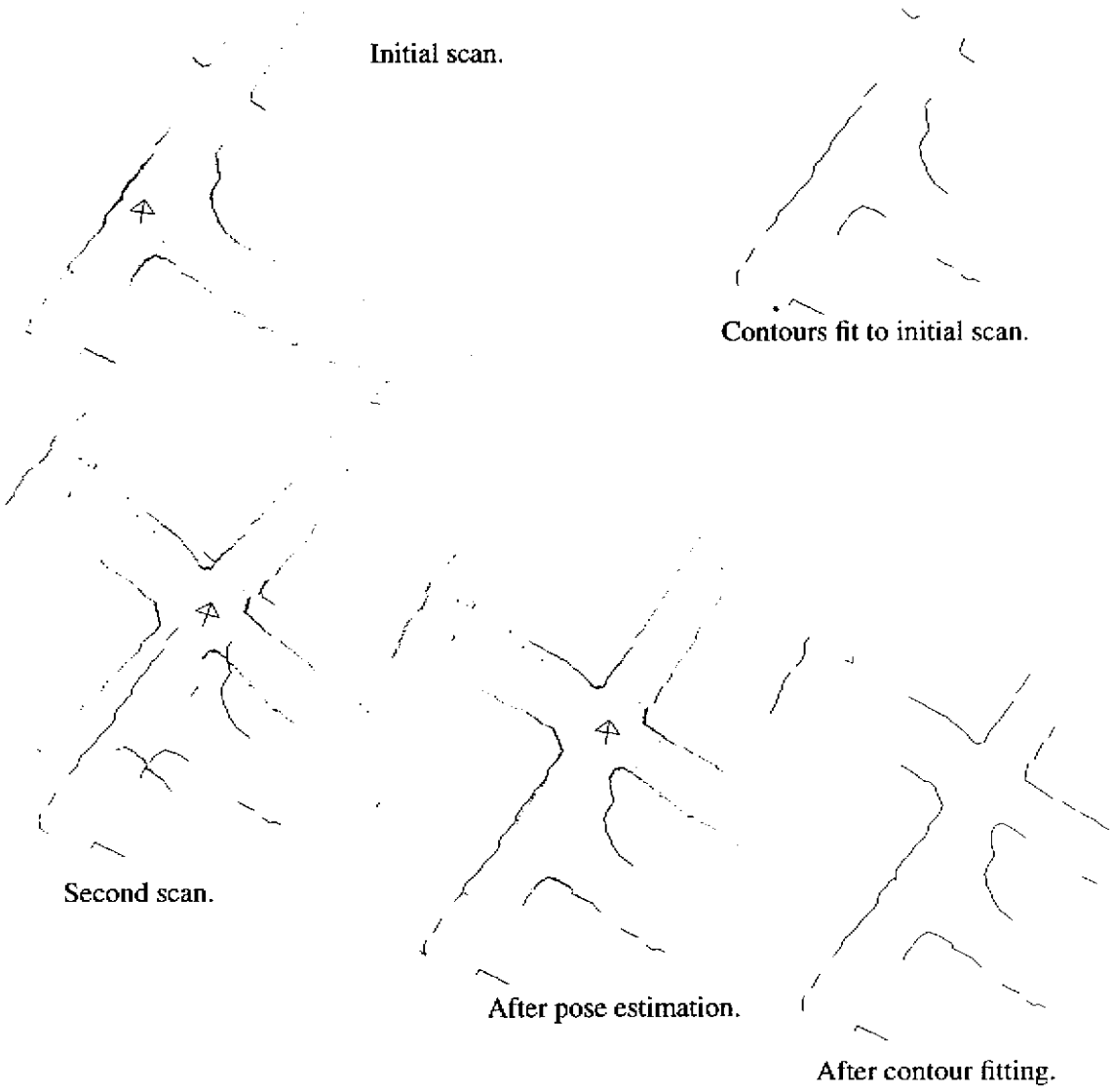


FIGURE 24. The first two cycles of a contour map building run.

5 Experimental Results

To test the contour-based mapping system in an expansive unstructured area, a room-and-pillar coal mine was chosen. This type of mine is modelled as 2D by mine engineers, because it follows coal "seams" that are fairly level. Workable seam heights are typically 3 - 8 feet and don't vary by much throughout the mine. The room-and-pillar technique of mining removes coal, leaving a matrix of untouched coal pillars behind which support the load from the overburden, the earth above the mine. There is a network of tunnels ("rooms") between the pillars left behind where the coal was removed. The exact dimensions of the rooms and pillars are different depending on several factors including the amount of overburden. To give an example, a current production-oriented mine we visited used 40' x 40' square pillars and 20' room widths. A small section of their mine map from their survey office appears in Figure 25 on page 55.

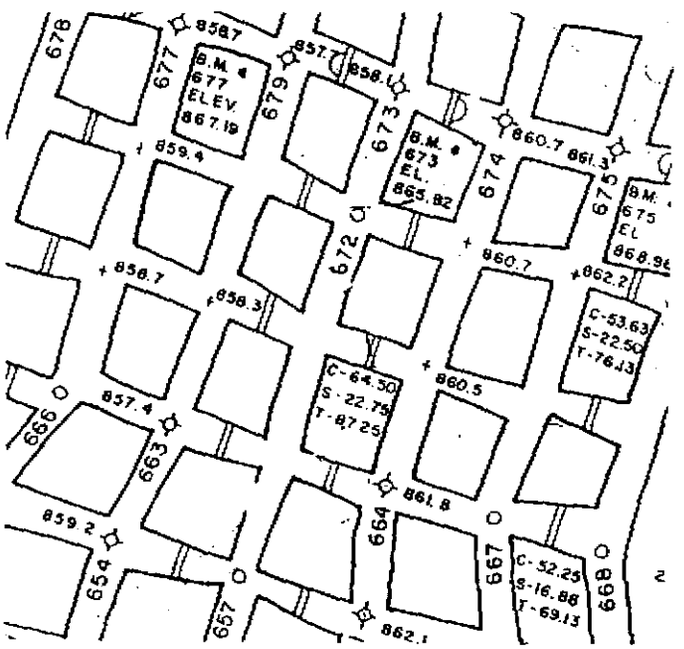


FIGURE 25. Mine survey map of a high-production room-and-pillar coal mine.

The quadrilaterals in Figure 25 on page 55 are pillars of coal, roughly 40'x40'. The rooms between the pillars are roughly 20' wide. The narrow double lines connecting the pillars represent concrete block walls that were built to control ventilation. Some of the surveyed locations are marked with circles and numbered. These are the numbers in the 600's. The other numbers, the ones in the 800's designate the elevation of the mine in feet. This is only a small part of a "production mine" that covers many square kilometers. It was mined with a coal excavating machine called a "continuous miner" that removes the coal with a vertically sweeping cutter. It is this vertical shearing motion and the fact that the coal seam is nearly level that allows the mine to be modelled with a 2D map.

Preliminary to getting a large data set of a mine, two pretests were performed. First, a test was performed to verify that our laser range sensor would perform properly in the mine environment. Then the work on the iconic pose estimator and eventually the contour-based system took place. The second pretest, used throughout development of the mapping code, was to test the system on synthetically generated scans. Finally, a large data set from an underground mine was collected for the ultimate test of the mapping system.

5.1 Real Scans

To verify that the Cyclone laser range scanner (see Appendix B for description) would function properly in a mine, we tested it in the mine pictured in Figure 26 on page 56. This was more or less a binary test. Would the reflectance of the coal mine walls provide enough of a return signal for the sensor to get an accurate range reading? Figure 26 on page 56 shows three overlapping 1000-point scans. A graphical user interface allowed a person to visually determine the relative poses of the three scans by manipulating poses with a mouse.

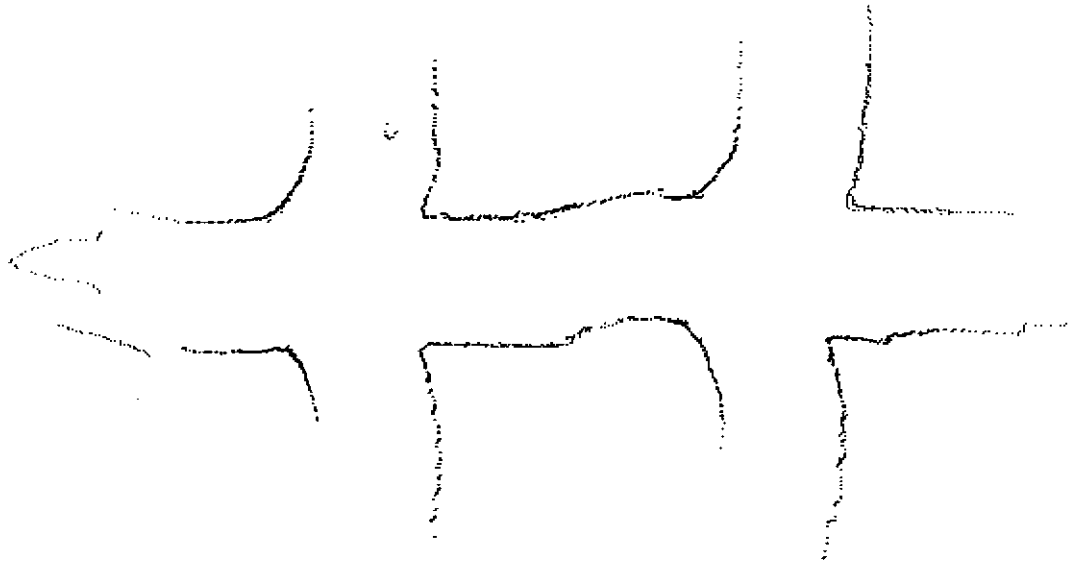


FIGURE 26. Three 1000-point Cyclone range scans from the Marrowbone mine.

5.2 Simulated Mapping of a Mine

Synthetic range scans facilitated testing of the mapping system. Systematic and random error in the sensor could be controlled (and eliminated) to allow testing the mapping system first on perfect range data. After the system worked on perfect data, random and/or systematic error was added to ensure that the mapping system could handle realistic amounts of error. Synthetic scans were generated through software by "scanning" a 2D world model composed of line segments. Ground truth (the sensor pose) was given, and therefore not a source of error. And because the scans were

generated from a given line segment model, the ultimate goal of the mapping system was also known. If all worked as it should, the resulting map should agree with the given 2D model that the scans were generated from. The two most illustrative tests used a large synthetic room-and-pillar mine similar to the real mine in Figure 25 on page 55.

5.2.1 Generating Synthetic Data

Before synthetic scans could be generated, a synthetic world had to be constructed. A simple fractal technique was used to add a random component to a matrix of squares representing a room-and-pillar coal mine. The approximate dimensions from the production mine in Figure 25 on page 55 were used: pillars 12m on a side and 6m wide rooms. Each line segment was split with random error added to the midpoint until segment size fell below a threshold, about 2m, resulting in the synthetic world pictured in Figure 27 on page 57.

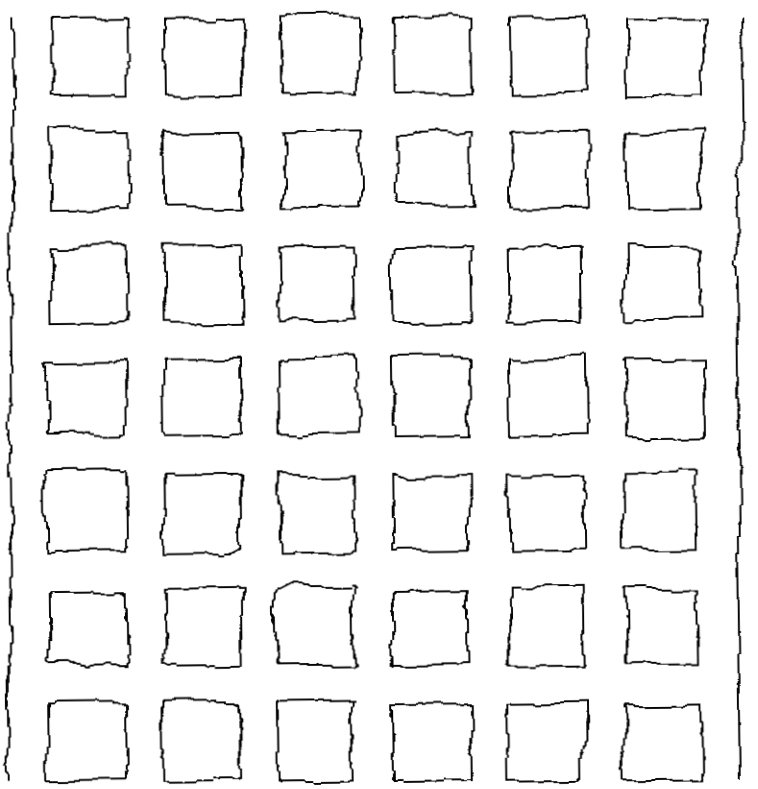


FIGURE 27. Synthetically generated line model of a room-and-pillar type of mine.

Synthetic scans were generated by intersecting rays form a given sensor pose with every line segment in the world model, and taking the distance to the nearest intersection. Random noise was added to the range. The following parameters controlled synthetic scan generation:

- number of scan points

- maximum range
- minimum range
- start angle
- angular resolution
- range resolution
- standard deviation of range

Initial pose estimates were created by adding random noise to the known poses from which scans were generated. The set of 42 scans plotted at the initial poses appears in Figure 28 on page 58.

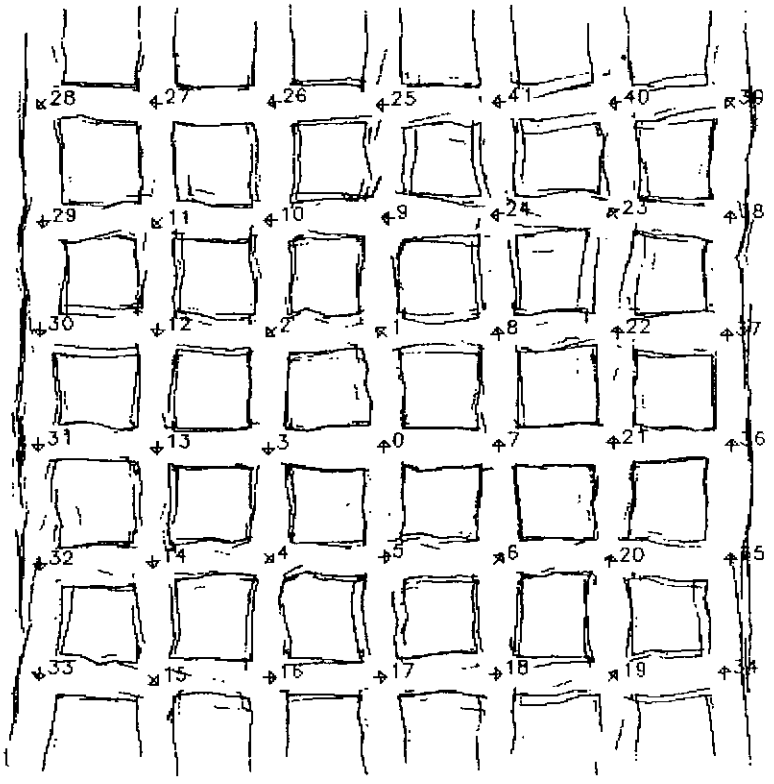


FIGURE 28. A set of 42 synthetically generated 2048 point range scans *plotted at initial pose estimates. The initial poses were generated by adding* random noise to the actual poses.

Figure 29 on page 59 shows the same set of scans plotted at the pose estimates resulting from running the contour-based mapper. The maximum pose estimate error is 4cm an 0.05°. Figure 30 on page 60 shows the resulting map. It consists of 109 contours with a nominal segment length of 0.5m.

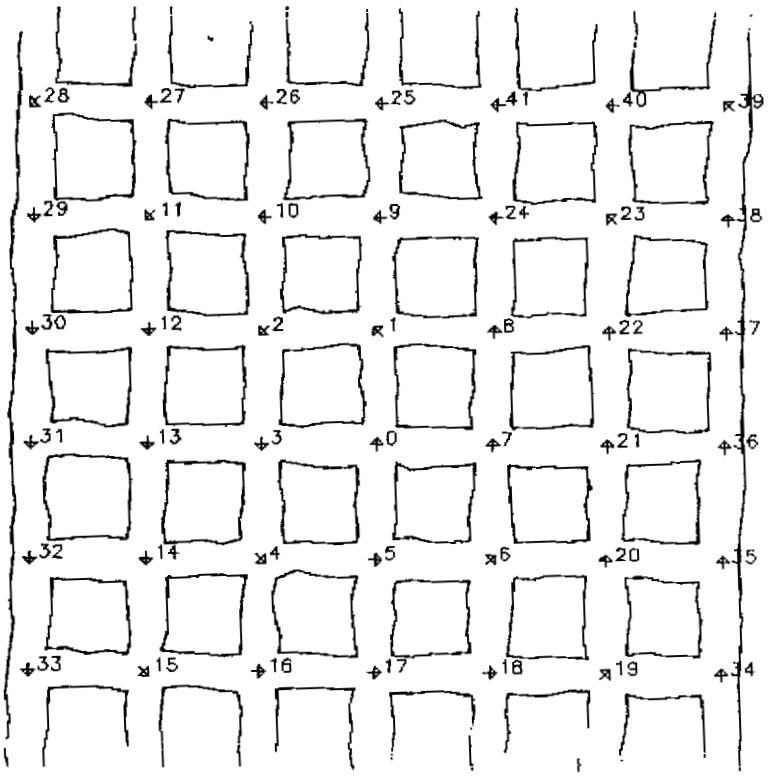


FIGURE 29. The set of 42 synthetically-generated scans plotted at their refined poses resulting from the contour-based mapping system.

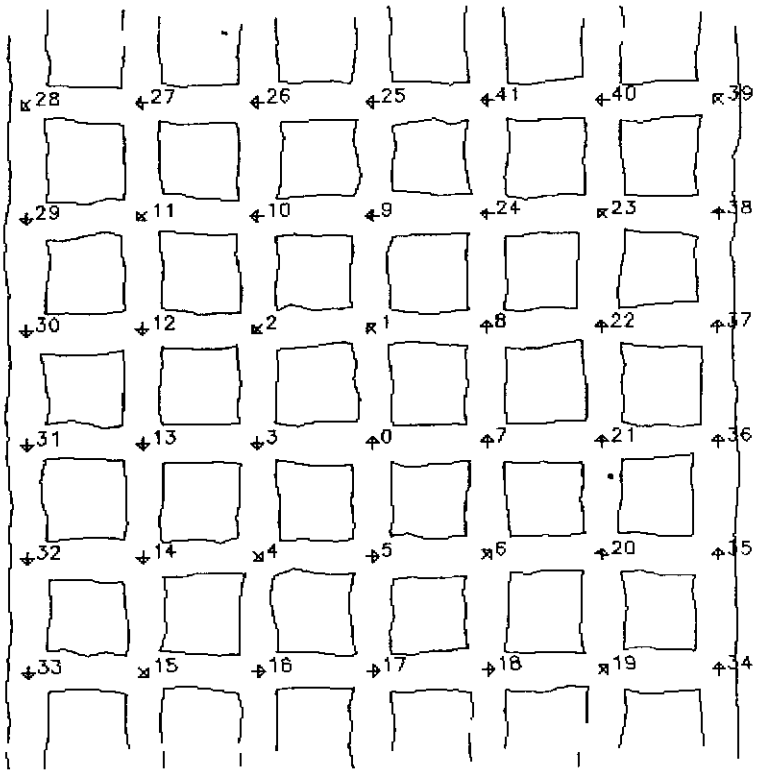


FIGURE 30. The resulting contour map. It is virtually indistinguishable from the ground truth map in Figure 27 on page 57

5.3 Mapping a Real Mine

Figure 31 on page 61 shows part of another room-and-pillar coal mine map, from a research mine in which we were able to get a dataset. It is an old mine which now serves as a research mine for the U.S. Bureau of Mines in Pittsburgh. It was mined over at least a 90 year period with a small amount of mining still occurring for research purposes. The map in Figure 31 on page 61 was drawn from approximately 200 hand-surveyed points. It is not complete. (Some of the pillars aren't fully represented.) The layout is not nearly as regular as the modern production mine from Figure 25 on page 55. There are many more "crib block" supports to augment the roof support than in the production mine, and there are even some 2' square timber pillars by design. The room widths are generally smaller, often only about 10' to 12' wide.

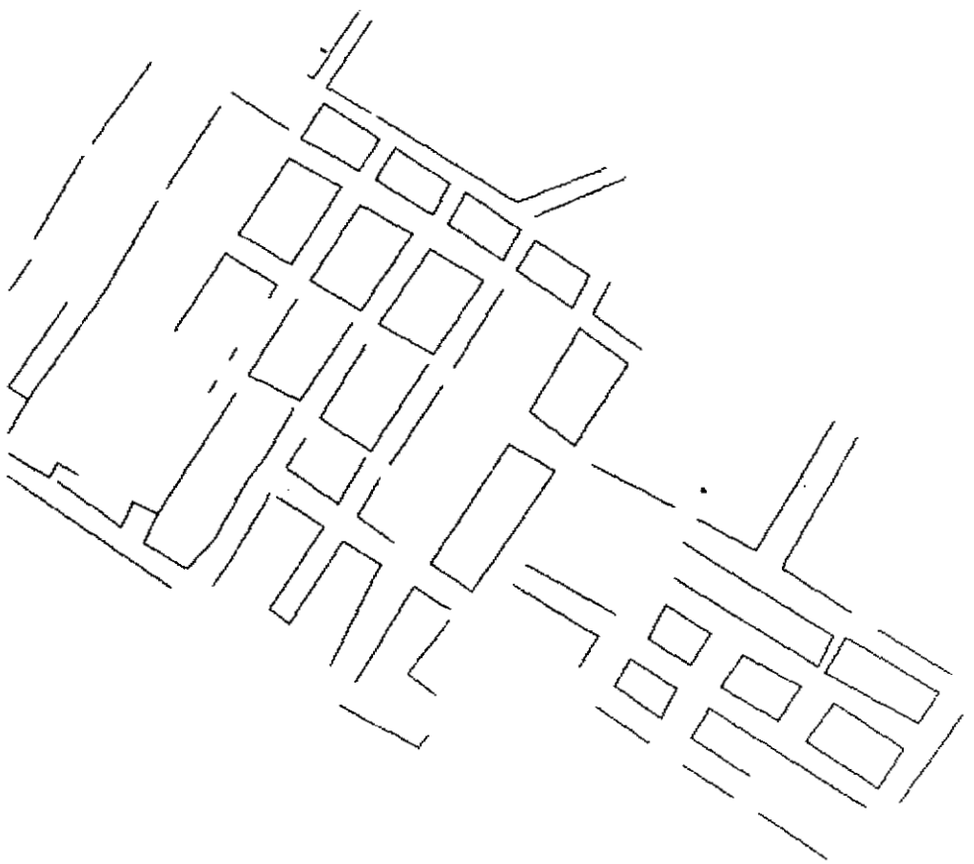


FIGURE 31. Hand surveyed mine map from a research (low-production) mine.

This 100x200 meter area of the mine was scanned at 102 locations. The location and orientation of the sensor was surveyed using a total station at each location for evaluation purposes. A data set consisting of laser range scans, inertial navigation system packets, and total station survey data was collected. The total station survey data served as ground truth for the position and orientation of the range scanner and the inertial navigation system. The ultimate comparison is between the automatically generated map from laser range data and the hand-measured map.

5.3.1 Sensor Platform

The laser range scanner and the inertial navigation system were mounted to a 4' x 8' sheet of 3/4" plywood. The INS was bolted rigidly. The laser was mounted in a 2-axis gimballed mount. Three reflective targets were attached to the scanner, and three to the platform to allow measuring the 3D positions (in the mine coordinate system) of the scanner and the INS (platform) using the total station. The laser scanner was levelled manually using a SmartLevel (resolution of 0.1°).

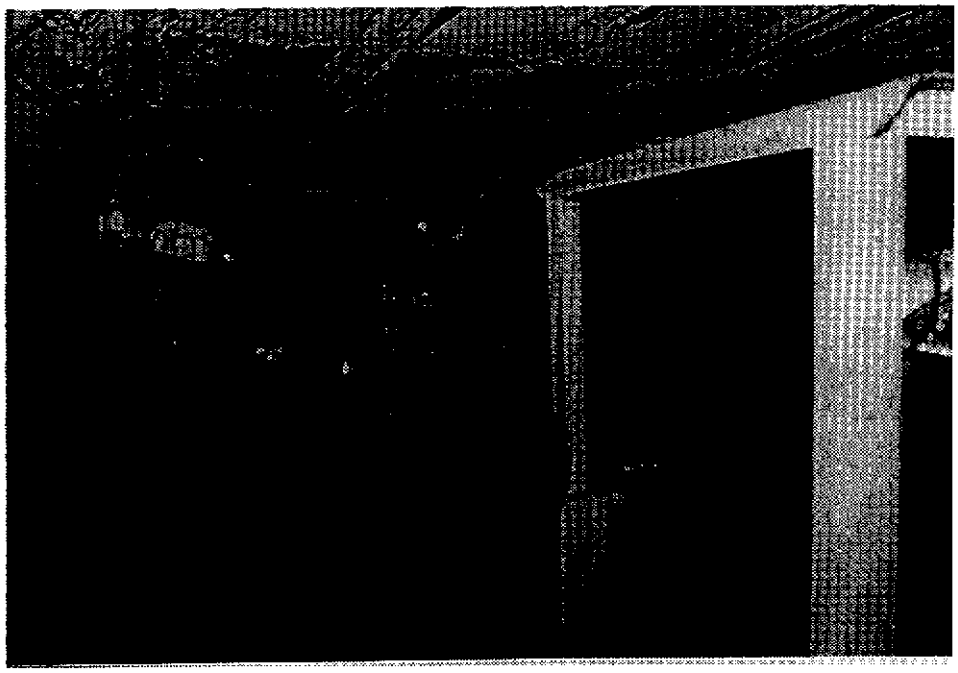


FIGURE 32. Photo of the sensor platform.

Scans were acquired with the sensor platform stationary and surveyed. A total of 102 scan locations spaced approximately 10 m apart. For part of the area, the sensor platform was carried on a flatcar on rails. For the areas that were not accessible by rail, it was carried by a Cushman, which also allowed more control over the placement of the sensor. The height of the sensor varied with the elevation of the bottom of the mine, by as much as 2 feet between overlapping scans. The levelling of the scanner probably had more of an effect on the overlap. It was levelled to $\pm 0.1°$.

At each scan location, the following data was collected:

- thirty 2048-point single line range scans
- one 2048x200 range image
- INS northing, easting, elevation, azimuth, roll, pitch
- roll and pitch of sensor platform measured with SmartLevel
- roll and pitch of Typhoon according to internal clinometers
- 7 reflectors' distance, vertical angle, horizontal angle measured with total station (3 on Typhoon, 3 on sensor platform, and 1 on INS)

An example 2048-point scan appears in Figure 33 on page 63.

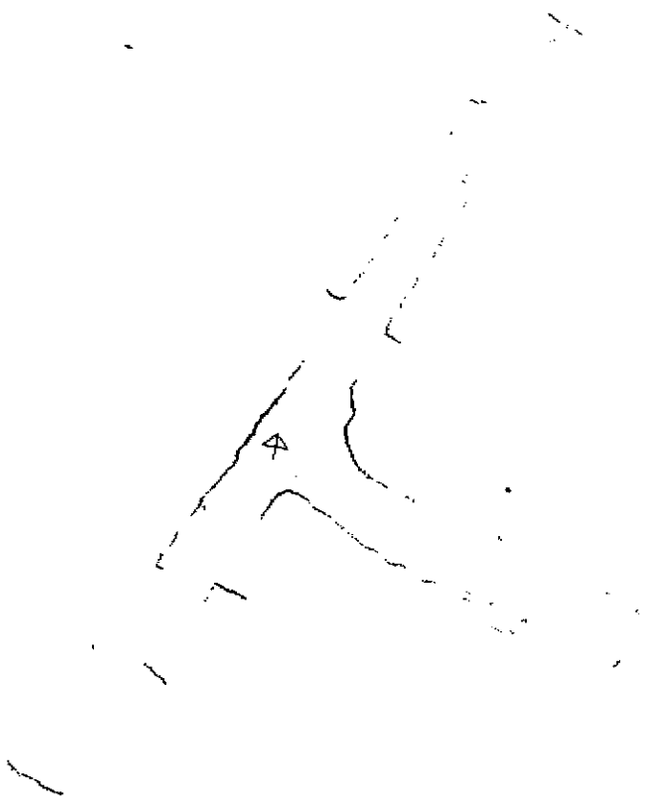


FIGURE 33. An example 2048 point range scan.

Taking 30 single line scans allows for using some statistical methods (such as averaging or taking the minimum) to generate an improved single scan. The 2048x200 range image data will not be used for this thesis. The INS position was initialized each morning to the location reported by the total station.

Using the laser scans, and a rough estimate of scanner position from either corrupted INS or corrupted total station, a 2D map can be generated. In an actual mapping system, the scans could be more closely spaced and keep track of its own position without the need for INS or total station. Alternatively, an operator could estimate the displacement between scans by stepping it off or by indicating the approximate location on a graphical display.

An example of mapping an expansive area of an underground coal mine is presented in Figure 34 on page 64 through Figure 39 on page 68. Figure 34 on page 64 shows a set of 33 2048-point range scans plotted at their surveyed poses and overlaid on the surveyed mine map for comparison. The pose of the robot is indicated by the arrows. The crosshairs within the arrowhead mark the position of the sensor, and the direction of the arrow indicates the orientation of the sensor. To give a sense of scale, the arrowhead is 2m wide. Figure 35 on page 65 shows the same set of scans, but plotted at erroneous poses. These poses were generated by adding random error to the sur-

veyed poses with $\sigma_x = 0.33$, $\sigma_y = 0.33$, and $\sigma_\theta = 3.33$. So the maximum error in x and y is 1m and the maximum error in orientation is 10 degrees. These numbers were chosen based on the fact that the corridor width is roughly 4m, so it should be possible to place the sensor closer to the center than the walls without measuring, and he should be able to pace off 10 meters with 1 meter accuracy. Likewise, a person should be able to place the sensor to within 10 degrees, using a compass if necessary. If the sensor is deployed on a robot, dead reckoning should suffice, given that the spacing between poses is roughly 10m.

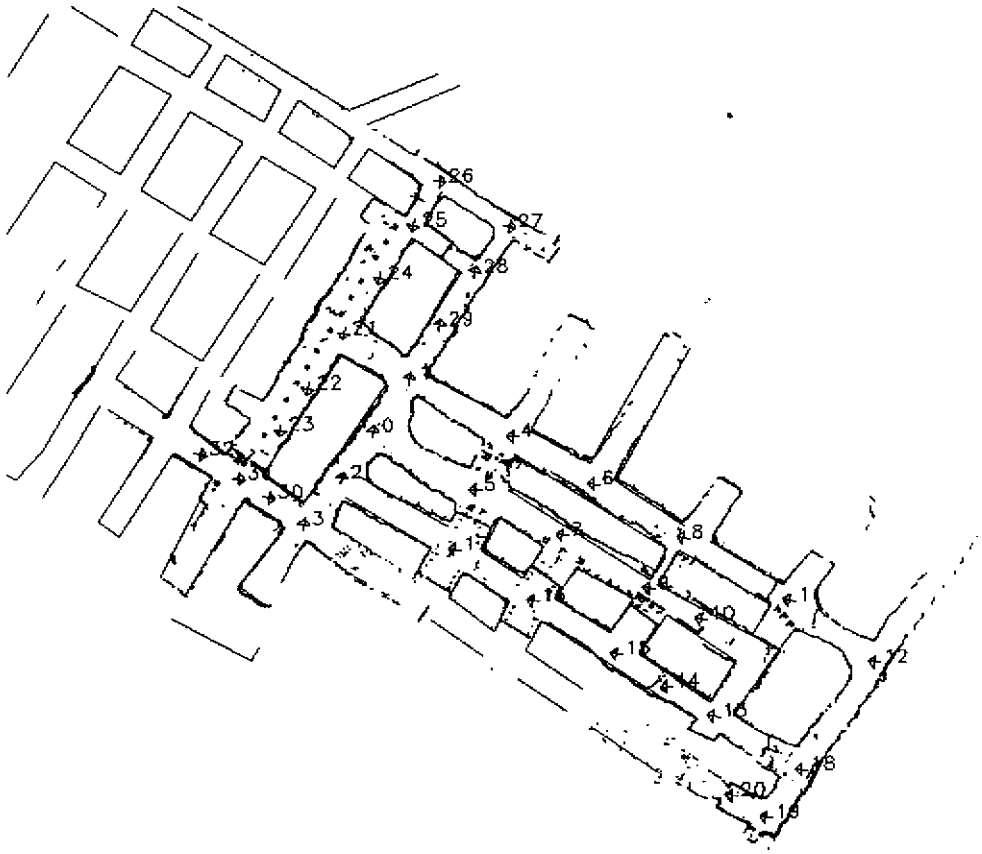


FIGURE 34. Set of 2048-point range scans from the Bureau of Mine Safety Research Mine plotted at their surveyed poses.

Figure 36 on page 66 shows the improved sensor poses after running the contour mapper. The contours that resulted are shown in Figure 38 on page 67 overlaid on the presurveyed map. The generated contour map is shown alone in Figure 39 on page 68. The presurveyed map was only used for display; the contour mapper software did not use it for any other purpose.

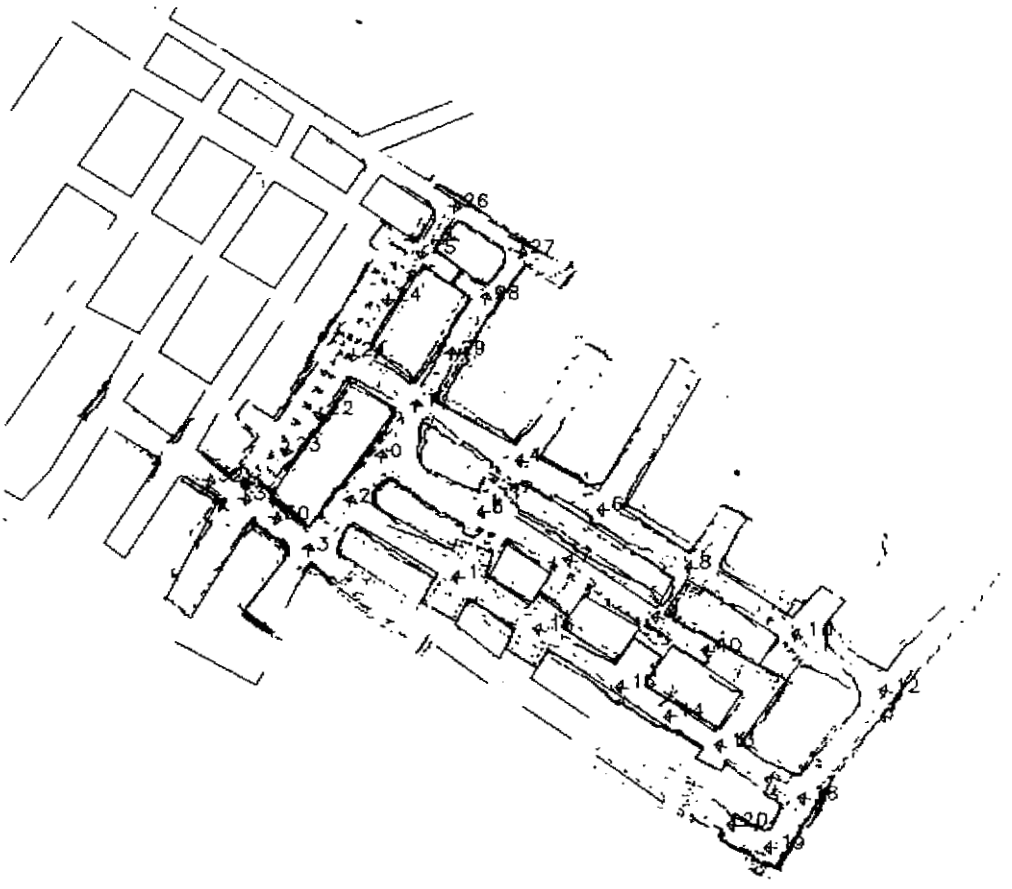
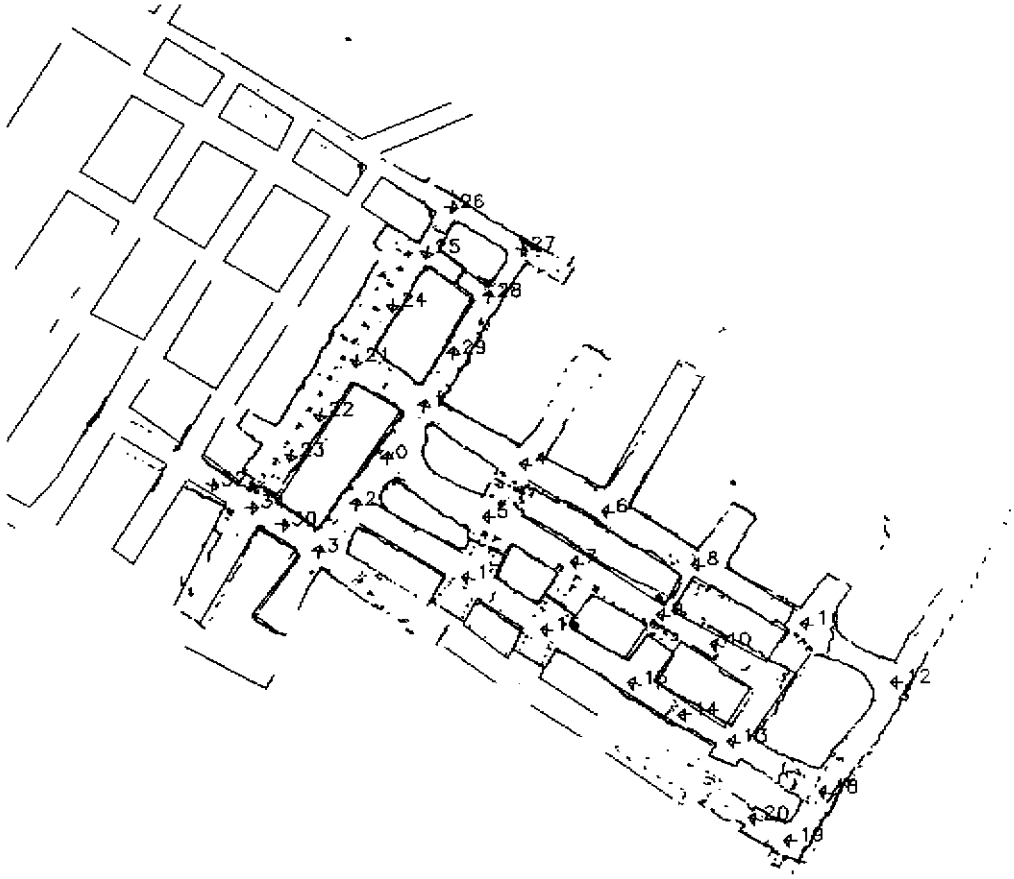


FIGURE 35. Set of 2048-point range scans from the Bureau of Mine Safety Research Mine plotted at their initial pose estimates.



Research Mine plotted at their refined pose estimates.

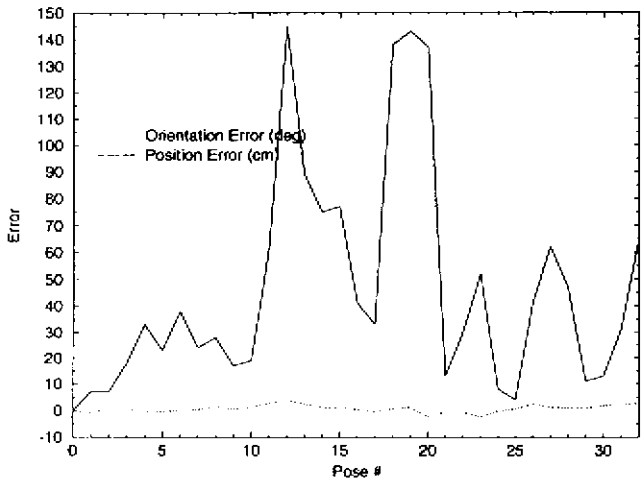


FIGURE 37. A graph of the pose error corresponding to Figure 36 on page 66. The orientation error peaks at 3.9°, although it is usually less than 2°.

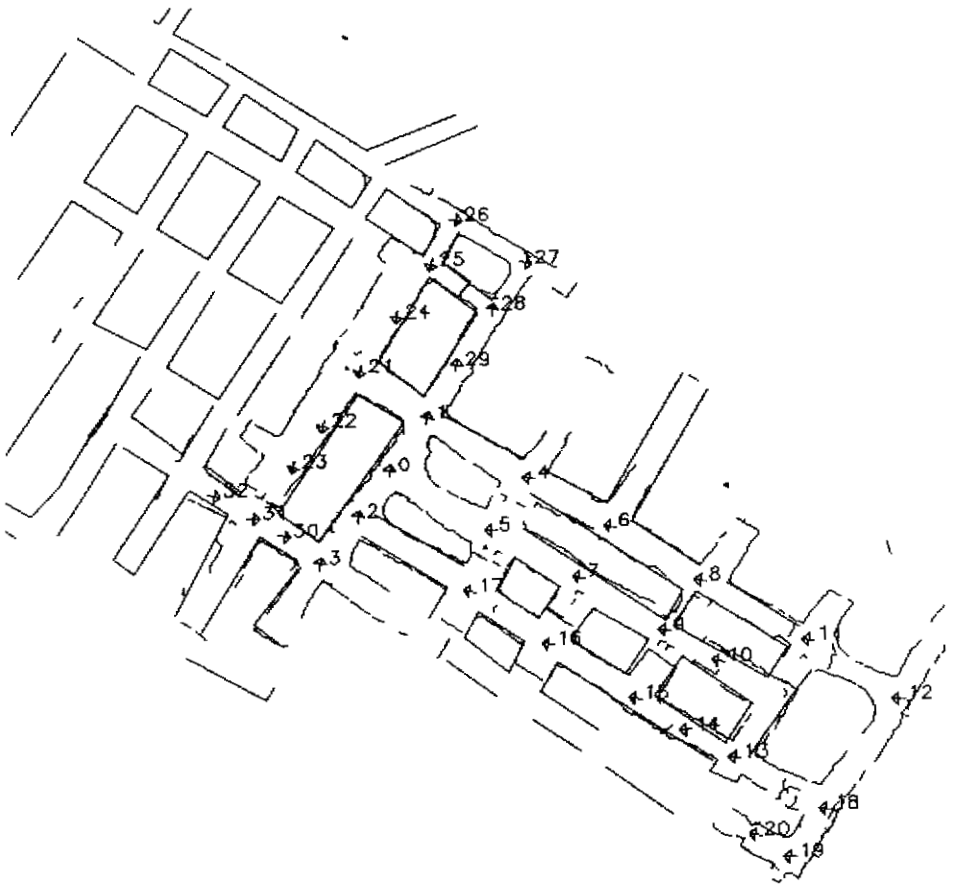


FIGURE 38. Contours overlaid on ground truth.

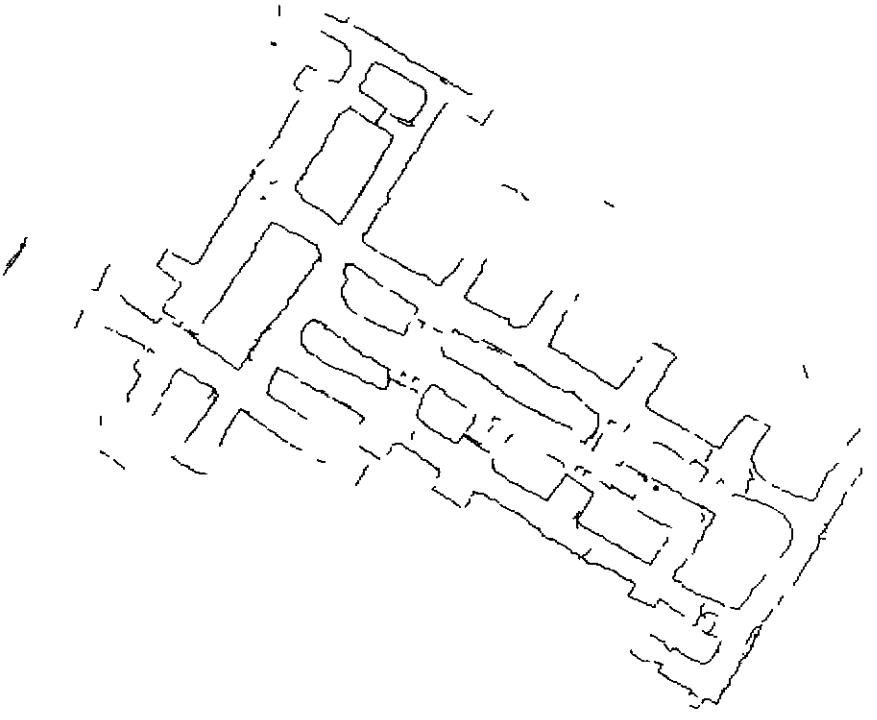


FIGURE 39. The generated map by itself. It consists of 359 contours. Over 60,000 range measurements were processed to map this area of approximately 5,000 m^2.

5.4 Discussion

The fundamental assumption of this thesis is that the area to be mapped can be accurately modelled as 2D. That the walls are vertical and the floor is flat. The research mine violated this assumption significantly. The floor was not flat, causing a height difference of as much as 2 feet between overlapping scans. This would not be a problem if the walls were perfectly vertical and the scanner was perfectly level. The walls were deteriorated due to age. Significant portions had crumbled away. In addition, part if not all of this mine was mined by undercutting and blasting as opposed to the shearing action of a continuous miner. The vertical shearing action of a continuous miner leaves nearly vertical walls behind. The scanner was levelled as well as we could level it, but even a tenth of a degree can cause a problem. The entries were narrow, which reduced the useful range of the scanner. In hindsight, I would have spaced the scans more closely to allow even more overlap of neighboring scans.

6 An Application for Robotic Mapping

Automated mine equipment could enhance productivity, access unworkable mineral seams, and reduce human exposure to the inhospitable environment of dust, noise, gas, water, moving equipment and roof-fall. Room-and-pillar mining is accomplished by repetition of a well-defined cycle of cutting coal, removing coal, and supporting the roof. Each of these operations is performed using different machines, thus allowing incremental automation of the entire mining process. This chapter describes a system that automates the first step in the cycle: cutting of coal with a continuous miner.

Automation of the continuous miner's task requires the ability to maneuver the miner in highly constrained corridor-like environments, and to accurately position it for cutting coal. Our system (MINENAV) consists of a modified continuous miner, a scanning laser range sensor, a SPARCstation, and software for autonomous control. Data from the laser range scanner is used to model the environment and to compute the position of the miner. The various cutting and loading appendages of the miner are all controllable from the SPARCstation.

The system has been tested on a mobile robot in a mock coal mine and on an actual continuous miner in a real coal mine. This system is the first instance of an intelligent robotic system for cutting coal. This chapter first describes the system and each of its components, and then presents the results of system tests.

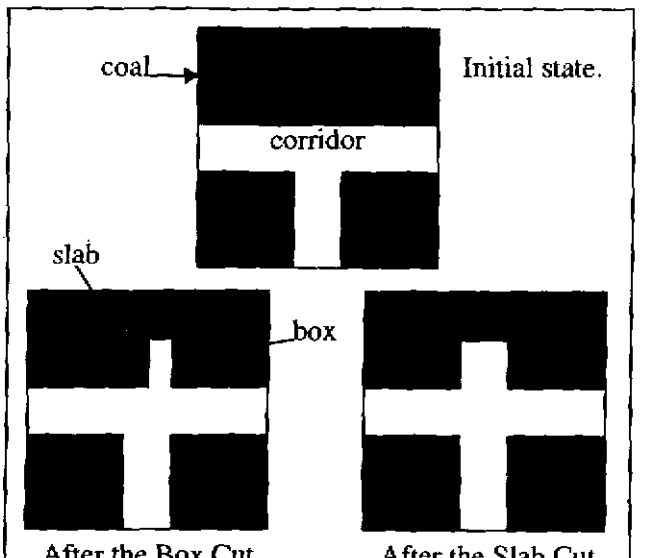


FIGURE 40. Extending a corridor.

6.1 System Overview

The general concept of the system is simple: use a laser rangefinder to guide a continuous miner to navigate and to cut coal. In room-and-pillar mining, a corridor is extended by executing two paral-

lel cuts: the box cut and the slab cut. See Figure 40 on page 69. The miner must precisely maneuver into position before the box cut, then maneuver to reposition for the slab cut. After the slab cut, the miner must back out to make way for the roof bolter. To accomplish this advancing of the section, each maneuver is planned in advance, and the perception and control modules guide the vehicle along the preplanned trajectories to get in position and cut coal.

The system configuration is shown in Figure 41 on page 71. The ovals represent software modules while the rectangles represent hardware devices. The software modules can be classified into three broad categories: planning, perception, and control. Most of the software modules are integrated through a central Task Control Architecture (TCA) [Fedor91] module, which provides communication and coordination services between modules.

The Task Planner computes the sequence of maneuvers and cuts that are necessary to accomplish a goal such as advancing the section. The motion trajectories for each maneuver are planned by the Path Planner.

The resulting task plan is executed by the control modules with assistance from perception. The vehicle control modules include the Cutter module and the Path Tracker module. The Cutter module controls the various cutting and loading appendages of the miner while cutting coal. The Path Tracker controls the locomotion of the vehicle to maneuver it between cuts. The perception modules process data from the range sensor to compute the position and heading of the vehicle as it moves. Two modules, the Feature-Based Pose Estimator (FPE)and the Iconic Pose Estimator (IPE) are employed depending on the navigation requirements. Each of these software modules and hardware devices is described in the following section.

6.2 Planning

There are two levels of planning in the coal mining software: task planning and path planning. The Task Planner generates a sequence of subtasks consisting of miner maneuvers in open space and cuts into the coal face. The motion trajectories for maneuvers are then planned by the Path Planner and queued for execution.

6.2.1 Task Planning

Based on the coarse geometry of the problem, the Task Planner determines the sequence of subtasks required to advance the coal face. Two types of subtasks are employed: maneuvering and cutting. Maneuvering involves locomotion of the miner in the corridor to position for coal cutting, while cutting involves locomotion of the miner and activation of its appendages to physically engage the coal face. Each subtask is parameterized by geometric constraints on the motion of the miner, goal points to permit the fusing of adjacent subtasks in the sequence, and ordering con-

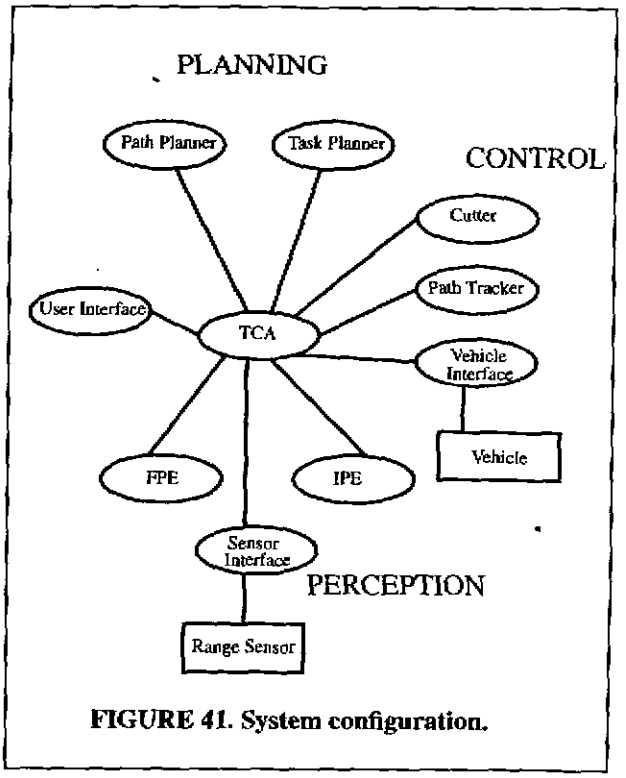


FIGURE 41. System configuration.

straints on the appendage operations. The Task Planner utilizes task tree facilities of the Task Control Architecture to assemble the subtask sequence.

6.2.2 Path Planning

The Path Planner module uses a line segment map of the environment to plan a safe sequence of poses for the miner to travel between an initial pose and a goal pose. A geometric model of the miner (a bounding polygon) is tested for intersections with the map during a search process to find a safe path. The search space is the set of all poses or *configurations*. The search space is three-dimensional, with each configuration in the space consisting of the miner's position and heading. A gradient descent scheme is used to find trajectories efficiently for simple cases. An A* approach [Nilsson71] is employed for more difficult cases. For storage efficiency, the search space is stored as an octree. Since the complete task is composed of several subtasks queued by TCA, planning of subtask n+1 can occur concurrently with execution of subtask n, so the effective planning time is usually zero. The result of the path planner is a piecewise linear path that connects the initial point to the goal point. This path has two important characteristics: it is both *safe* and *kinematically feasible*. It is safe in that if the path is tracked by the miner properly, it will not collide with any obstacles that are in the map. The path is kinematically feasible in that the miner is physically capable of the motions required to properly track the path.

6.3 Perception

In order to properly track paths and cut coal, the perception subsystem must be able to measure the position and heading (pose) of the continuous miner. There are two interchangeable perception subsystems: map-based pose estimation and triangulation-based pose estimation. The triangulation-based system is much more limited in capability: it has a 35 foot maximum range of operation, and requires line of sight between miner-mounted targets and external lasers. However, the sensor that is currently used for map-based perception is not rugged enough to survive on a continuous miner, and is not explosion-proof, so until one is built, we use triangulation in our underground tests.

6.3.1 Map-Based Pose Estimation

Map-based perception matches range data to an a priori map of the miner's environment to compute the pose of the miner. Both the feature-based and iconic map-based pose estimation systems use data from a radial scanning laser range sensor. A typical range scan from an underground coal mine appears in Figure 42 on page 72. It consists of 1000 points in a 360° horizontal planar field of view. The cross indicates the sensor position. Both feature-based and iconic pose estimation are described in earlier chapters of this thesis.

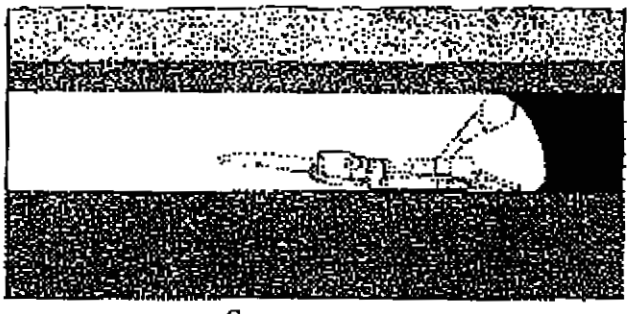
FIGURE 42. An example range scan from a mine.

6.3.2 Triangulation-Based Pose Estimation

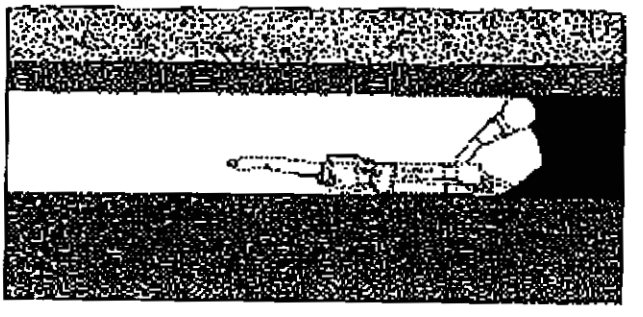
The CMU range sensor, the Cyclone[Singh91], does not meet the requirements for operation on a continuous miner (ruggedness, explosion proof, etc.), so our underground tests have used an alternate pose estimation system. Lasernet is a commercially available system which consists of a scanning laser and a reflective target. Using two externally mounted lasers, and two miner-mounted targets, the pose of the miner can be determined accurately to a maximum range of about 35 feet. The Cyclone is vehicle-mounted, and therefore does not have a limited range of operation. For details on the triangulation-based system, see Anderson [Anderson91].

6.4 Control

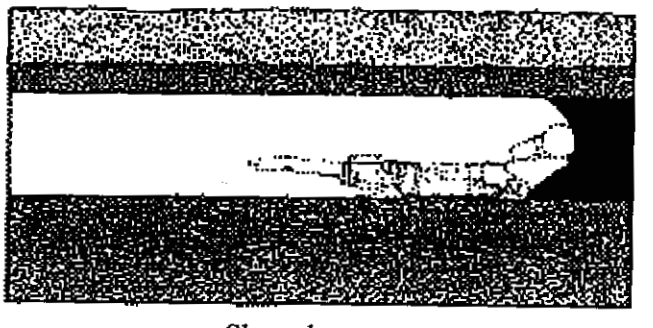
The continuous miner must be controlled to execute two types of operations: coal cutting and maneuvering between cuts.The miner is a tracked (skid-steer) vehicle that has a large rotating cut-



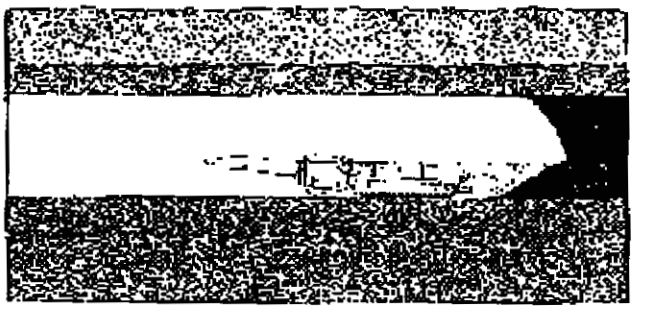
Square up.



Sump in.



Shear down.



Remove cusp.

FIGURE 43. The sump-shear cycle.

ter head on the front and a conveyor from the cutterhead to the rear of the machine.

6.4.1 Cutting

The coal cutting cycle consists of four steps: squaring up to the coal face, sumping in, shearing down, and removing the cusp. (See Figure 43 on page 74.) This cycle is repeated until the desired depth of penetration into the wall of coal is achieved. Squaring up refers to aligning the miner with the desired heading of the cut. For the initial sump, this is accomplished by driving the miner into the wall, letting the tracks continue to drive after impact, and stopping when the heading has reached the desired value.

Sumping in means turning on the rotating cutterhead and driving into the wall of coal. Shearing down involves stopping the tracks and allowing gravity to pull the rotating cutterhead down through the coal until it reaches the floor. Due to the cylindrical shape of the cutterhead, there is a "cusp" of coal left on the floor. Cusp removal is the process of driving the machine backwards with the cutterhead spinning to shave off the cusp.

The cutting aspect of the control is straightforward: sequence the appendages in the proper order, and repeat the cycle until the perception system (the pose estimator) confirms that the desired depth of cut has been accomplished.

6.4.2 Maneuvering

Control of the continuous miner during maneuvering is the responsibility of the Path Tracker. The module is an implementation of the "pure pursuit" algorithm [Wallace85]. At each time interval, the current position and heading of the vehicle are sensed, and a trajectory that will intersect the desired path at a desired lookahead distance is computed. The correction trajectory usually consists of an arc followed by a straight line, but if the vehicle is far off the path, a turn-in-place might be required. Since the miner is currently capable of only one turning radius other than zero, some corrections would be impossible without turning in place. The cruising speed of the miner is about three inches per second, and the minimum time interval between corrections for stable control is a few seconds.

6.5 Results

The system was tested in three environments: an indoor mine mock-up using the Locomotion Emulator (LE) mobile robot, a pseudo-coal corridor using a Joy CM16 continuous miner, and a real underground coal mine using a Joy CM14 continuous miner.

For the first test setup (shown in Figure 44 on page 76) a wooden frame with the dimensions of a continuous miner was attached to the LE, and the LE controller emulated the skid-steer continuous miner. The Cyclone range sensor was mounted on the LE along with a SPARCstation. The system

successfully maneuvered through a wooden mine corridor mock-up, and simulated the coal cutting cycle.

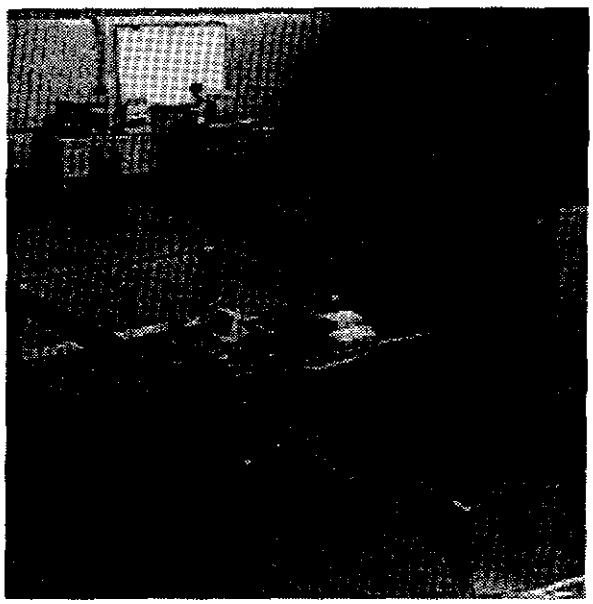


FIGURE 44. LE test setup.

The second test setup (shown in Figure 45 on page 76) involved a real continuous miner (Joy CM16), and a more realistic mine mock-up. The spinning cutterhead sumping into the pseudo-coal is shown. The Cyclone sensor could not be used since it is not rugged enough to be mounted on a continuous miner. Instead, the Lasernet system provided the needed position and heading feed-back. The mine mock-up was more realistic in that it was a poured block of imitation coal that

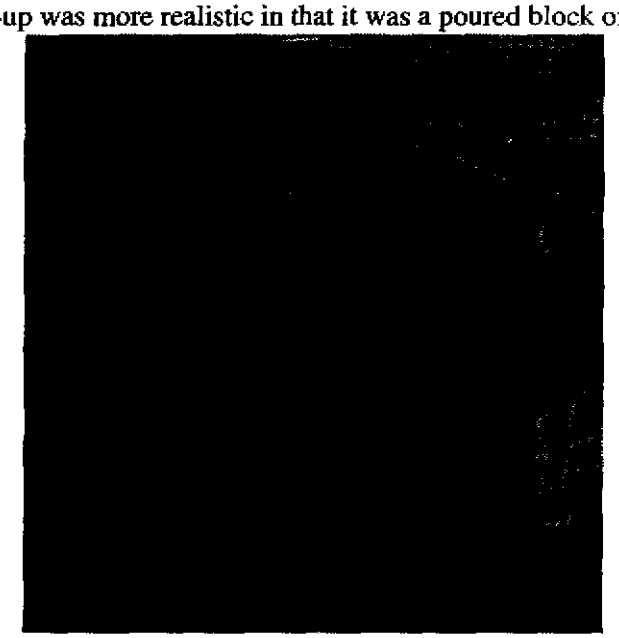


FIGURE 45. CM16 test setup.

allowed us to test the cutting software. This environment was still ideal in that the floor was flat concrete instead of bumpy rock.

The third and ultimate test setup was in a real underground coal mine using a Joy CM14 continuous miner and the Lasernet sensor. (See Figure 46 on page 77.) The SPARCstation was off-board the miner and connected through a single serial line. The system successfully extended the mine corridor several feet by executing a series of box and slab cuts.

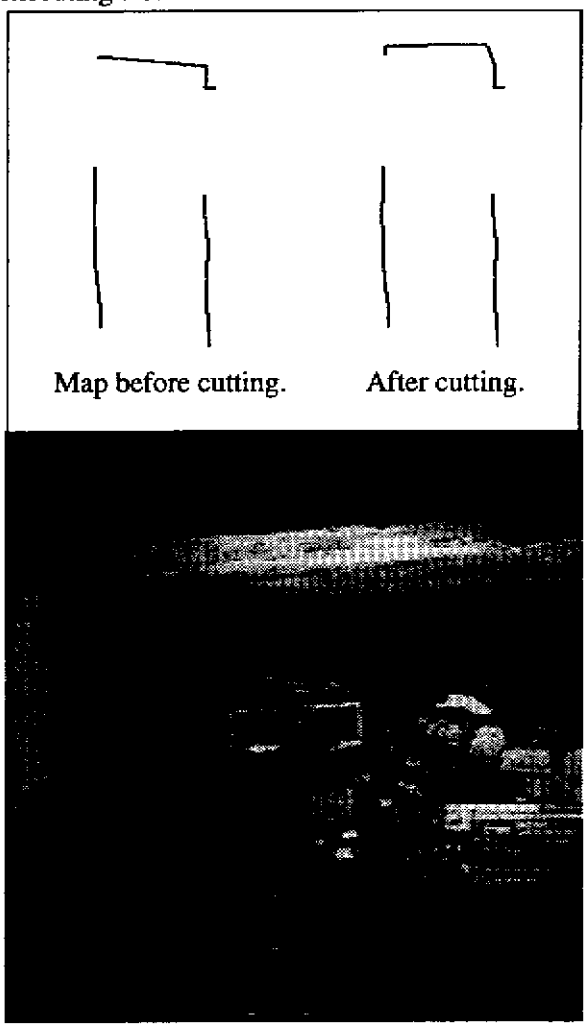


FIGURE 46. CM14 engaging real coal.

6.6 Summary

A robotic system for automating a continuous miner was presented. A laser range sensor was used to guide the vehicle while maneuvering in tight corridors and positioning for coal cutting. The system succeeded in initial tests in the lab and in the mine. In contrast to previous script-based systems, this work is the first instance of a robotic system with potential for fully automating a continuous miner.

Given a mine-hardened laser range sensor that could survive aboard a continuous miner, the iconic and contour-based pose estimation systems could be used to navigate mine equipment and to build mine maps autonomously.

7 Conclusion

7.1 Summary

This dissertation demonstrates the feasibility of using a long maximum range laser range scanner for mapping expansive unstructured environments. A progression of techniques was presented starting with feature-based pose estimation, then iconic pose estimation, and finally contour maps. Results were presented for each of the methods indicating advantages and disadvantages for each of the methods. Finally some ideas for how the ideas presented in this dissertation could be integrated into an application (autonomous underground mining) were presented.

The feature-based pose estimation system matched a range scan to a pre-surveyed line segment map to determine the pose of the robot in the map's coordinate frame. Corner and line segment features from the scan were matched to corner and line segment features from the map.

Several iconic pose estimation systems were presented. All of them shared a common core: matching every point of a scan to a map. The map can be a presurveyed line segment map (the same as the feature-based system), it can be automatically generated from an initial scan, or it can be the unprocessed initial scan. In the second and third case, many scans can be matched to each other simultaneously. However, the time required for an iteration grows unacceptably quickly as the number of scans involved increases, making this impractical for expansive environments. The real shortcoming for the iconic system is that it doesn't result in a compact and concise map representation. A set of scans can serve as a crude map, but a better representation is desirable.

The contour mapping system builds on the capabilities of the feature-based and iconic pose estimators, resulting in a system capable of assembling an unlimited number of scans of an expansive unstructured dynamic environment into a detailed, complete, compact, and concise representation.

Finally, underground room-and-pillar mining was described and the applicability of encircling laser range sensing was discussed. The pose estimation and mapping algorithms developed in this thesis are relevant to the needs of underground mining.

7.2 Contributions

This work has extended the state of the art in 2D mapping for mobile robots working in unstructured environments and has demonstrated potential for automating mining. Specifically, the contour-based mapper is:

- the first system to map an expansive unstructured environment without external information
- the first such system to be shown valid for use in a coal mine
- first such system to increase accuracy by taking advantage of heavy overlap, including cycles
- first system to be able to revise existing maps to account for dynamic change

7.2.1 Mapping of an Expansive Unstructured Area

This thesis demonstrated 2D mapping of an expansive unstructured area. Specifically, a large portion of an underground room-and-pillar coal was mapped automatically. We are not aware of another automated mapping system that can build a model of similar quality of an area so large and unstructured. Most previous results have demonstrated mapping of relatively simple indoor environments.

7.2.2 Mining

An encircling laser range sensor entered an underground mine for the first time. The potential for this type of sensor to assist mining operations was demonstrated, as a mine mapper and as a pose estimator for mobile equipment in an unstructured environment.

7.2.3 Map Refinement

Two methods of whole map refinement were demonstrated. One of the incarnations of the iconic pose estimation system allowed refinement of multiple poses simultaneously. Every point of every scan was used to reduce the total error in the map by matching all the scans to each other. Contour maps can be refined if the scans used in making them are stored. The procedure is simply to iterate between refining all of the contours in the map and then refining each of the scan pose estimates individually.

7.2.4 Handling of Dynamic Environments

The flexibility of the contour representation, the pinpoint accuracy of the laser range sensor, and the accuracy of pose estimation allowed strong assumptions to be made regarding conflicts between successive scans of the same area.

7.3 Improvements and Future Work

There are many areas of the existing system that have room for improvement.

7.3.1 Extending to 3D

Extending the mapping system to build a 3D model from laser range data would provide a much more correct and complete picture of the environment. The sensing technology exists, and it's becoming more affordable every day. The results presented in this thesis used the single line scans that were collected in a coal mine. That data set also included a set of 360° x 40°, 2048 pixel x 200 pixel range images. An example is shown in Figure 47 on page 80. Imagine the 3D model that could be constructed from 102 of these images.

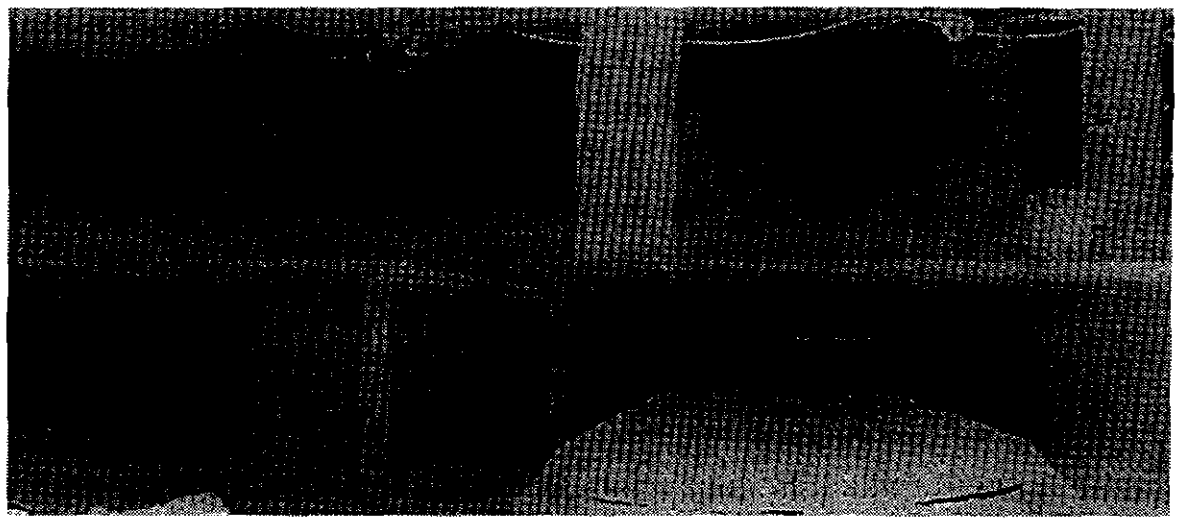


FIGURE 47. A Typhoon range image of a mine. The top half of the image shows the front 180° x 40°; the bottom half shows the rear 180° x 40°.

7.3.2 Approximations in Error and Gradient Computations

For the contour mapping system, the error and gradient calculations are integrals that are calculated piecewise. In the current implementation, the calculation is simplified by ignoring the error contributed around the vertices of contours. The effect of this approximation is assumed to be insignificant, but has not been verified.

7.3.3 Front End on Pose Estimation

When mapping, accuracy is more important than execution time, so whenever there was a tradeoff, speed was traded for accuracy. The pose estimation process is slow because the correspondence is recalculated on every iteration and the convergence thresholds are small. Line search is slower than the Gauss-Newton approach used in the iconic estimator. One idea for speeding up the contour-based pose estimator without sacrificing efficiency is to use the iconic estimator to bring the pose error down from its initial level to a much smaller level, and then to use the contour-based pose estimator to do the fine adjustment.

7.3.4 Better Iteration Cutoffs

The iteration cutoffs are very fine for both the contour-based pose estimation and the contour-fitting minimization loops. Hopefully there's a good way to cut off the iteration without sacrificing accuracy. I haven't found it yet.

7.3.5 Scan Preprocessing

The density of range data is highly variable, depending on the geometry of the scene. Close to the scanner, the density is higher than it needs to be. For the contour mapper, efficiency could be gained by preprocessing the scans to reduce the density. This would reduce the computation time greatly for all subsequent correspondence, error, and gradient computations. Little accuracy would be lost by cutting the density to a reasonable level.

Another preprocessing technique that could be exploited if the robot is stationary during scanning is to take several scans and use some statistical means to generate an improved composite scan. Some quick checks have indicated that using the minimum of a set of readings provides a more precise measurement.

7.3.6 Better Map Expansion

Because of noise in the scan data and several interrelated thresholds, there are some defects in the contours that are generated. Some of this is best fixed by removing redundant contours. Some of it can only by fixed by a better implementation.

7.3.7 The Need for Speed

Because the priority of the mapping system is accuracy, speed has been sacrificed for higher accuracy wherever the two came in conflict. I suspect that for purposes of navigation, a version of the iconic estimator that uses a relatively small number of points could provide a nice trade-off between speed and accuracy.

References

[Anderson91] D. Anderson, "Laser-Based Guidance for Underground Mobile Mining Equipment", *First International Design for Extreme Environments Assembly,* November 1991.

[Cox91] I. Cox, "Blanche - An Experiment in Guidance and Navigation for an Autonomous Mobile Robot", *IEEE Transactions on Robotics and Automation,* vol 7, no 2, 1991.

[Crowley85] J. Crowley, "Navigation for an Intelligent Mobile Robot", *IEEE Journal of Robotics and Automation,* vol RA-1, no 1, March 1985.

[Crowley89] Crowley, J., "World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging", *IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, May 1989.

[Dahlquist74] Dahlquist et al, Numerical Methods, Prentice-Hall, Inc., 1974.

[Drumheller87] M. Drumheller, "Mobile Robot Localization Using Sonar", *PAMI-9,* 1987.

[Elfes87] Elfes, A., 1987, "Sonar-Based Real-World Mapping and Navigation", *IEEE Transactions on Robotics and Automation,* Vol. 3, No. 3., June.

[Fedor91] C. Fedor and R. Simmons, "Task Control Architecture User Manual", *The Robotics Institute, Carnegie Mellon University,* July 1991.

[Fitzpatrick89] K. Fitzpatrick and J. Ladd, "Locomotion Emulator: A Testbed for Navigation Research", in proceedings, *1989 World Conference on Robotics Research: The Next Five Years and Beyond*, May 1989.

[Gonzalez92] J. Gonzalez et al, "An Iconic Position Estimator for a 2D Laser Rangefinder", in proceedings, *IEEE International Conference on Robotics and Automation,* Nice, France, May 1992.

[Gonzalez93] Gonzalez, J., "Estimacion de la Posicion y Construccion de Mapas para un Robot Movil equipado con un Escaner Laser Radial", Ph.D. thesis, University of Malaga, Spain, 1993.

[Kelly95] A. Kelly, "An Intelligent Predicitive Control Approach to the High Speed Cross Country Autonomous Navigation Problem", Ph.D. dissertation, The Robotics Institute, Carnegie Mellon University, CMU-RI-TR-95-33.

[Krotkov89] E. Krotkov, "Mobile Robot Localization Using a Single Image", in proceedings, *IEEE Conference on Robotics and Automation*, 1989.

[Kuc87] R. Kuc and M. Siegel, "Physically Based Simulation Model for Acoustic Sensor Robot Navigation", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol PAMI-9, no. 6, Nov 1987.

[Leonard91] Leonard, J. J. and Durrant-Whyte, H., "Simultaneous Map Building and Localization for an Autonomous Mobile Robot", *IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS '91)*, Osaka, Japan, Nov. 1991.

[Leonard92] Leonard, J. J., and Durrant-Whyte, H., Directed Sonar Sensing for Mobile Robot Navigation, Kluwer Academic Publishers, 1992.

[Moravec85] H. Moravec and A. Elfes, "High Resolution Maps from Wide-Angle Sonar", in proceedings, *IEEE Conference on Robotics and Automation,* March 1985.

[Nilsson71] N. Nilsson, "Problem-Solving Methods in Artificial Intelligence", McGraw-Hill, New York, 1971.

[Sears82] F. Sears et al, University Physics, *Addison-Wesley, 1982.*

[Sedas93] S. Sedas, "Algorithms for Automatic Sensor Placement to Acquire Complete and Accurate Information", *Ph.D. dissertation, The Robotics Institute, Carnegie Mellon University, 1993.*

[Shaffer92b]G. Shaffer et al, "Comparison of Two Range-Based Pose Estimators for a Mobile Robot", *SPIE Mobile Robots VII,* Boston, 1992.

[Singh91] S. Singh and J. West, "Cyclone: A Laser Scanner for Mobile Robot Navigation", *Technical Report, The Robotics Institute, Carnegie Mellon University, CMU-RI-TR-91-18,* August 1991.

[Wallace85] R. Wallace et al, "First Results in Robot Road-Following", *IJCAI-85,* August 1985.

[Wallace88] A. Wallace, "A Comparison of Approaches to High-Level Image Interpretation", Pattern Recognition, 21(3), 1988.

Appendix A Gauss-Newton Error Minimization

The following symbols are used in this appendix.

- r = array of range measurements from an arbitrary scan
- α = array of angle measurements corresponding to r
- (x, y, θ) = pose of scan composed of r and α
- r_m= array of range measurements for scan #m
- α_m = array of angle measurements corresponding to r_m
- i = index into r_m and α_m or r and α
- (x_m, y_m, θ_m) = pose of scan #m
- r_n= array of range measurements for scan #n
- α_n = array of angle measurements corresponding to r_n
- j = index into r_n and α_n
- (x_n, y_n, θ_n) = pose of scan #n
- l = length of a map line segment
- (x_s, y_s) = start point of a map line segment
- (x_e, y_e) = end point of a map line segment

A.1 Feature Matching

For this section only, the following symbols will be used in addition to those already defined:

- (x_p, y_p) = predicted point feature in robot frame
- (r_p, θ_p) = predicted line feature radius from origin and orientation in robot frame
- (x_r, y_r) = robot (sensed) point feature
- (r_r, q_r) = robot (sensed) line feature radius from origin and orientation

To carry out the minimization, we transform the predicted features from the world frame into the robot frame. There are generally fewer predicted features than sensed features, so we chose to transform the smaller set and work in the robot frame. The transformation formulas appear in Equation 6 with the untransformed predicted feature parameters primed.

(EQ 6)

$$
\begin{aligned}
x_p &= (x_p' - x)\cos\theta + (y_p' - y)\sin\theta \\
y_p &= -(x_p' - x)\sin\theta + (y_p' - y)\cos\theta \\
r_p &= r_p' - x\cos\theta - y\sin\theta \\
\theta_p &= \theta_p' - \theta
\end{aligned}
$$

A.1.1 Computing F

The vector of error components, F, is composed of each of the individual correspondence errors. There are two cases: point and line. For points, the correspondence error is separated into x and y components. For line correspondences, the error is represented as difference in orientation and difference in perpendicular distance from origin. See Equation 7.

(EQ 7)

$$f_x = x_p - x_r$$
$$f_y = y_p - y_r$$
$$f_r = r_p - r_r$$
$$f_\theta = \theta_p - \theta_r$$

A.1.2 Computing J

There are twelve derivative which make up the Jacobean, three for each of the types of error. The formulas are listed in Equation 8.

(EQ 8)

$$\frac{\partial f_x}{\partial x} = -\cos\theta \qquad \frac{\partial f_r}{\partial x} = -\cos\theta$$
$$\frac{\partial f_x}{\partial y} = -\sin\theta \qquad \frac{\partial f_r}{\partial y} = -\sin\theta$$
$$\frac{\partial f_x}{\partial \theta} = -(x_p' - x)\sin\theta + (y_p' - y)\cos\theta \qquad \frac{\partial f_r}{\partial \theta} = 0$$
$$\frac{\partial f_y}{\partial x} = \sin\theta \qquad \frac{\partial f_\theta}{\partial x} = 0$$
$$\frac{\partial f_y}{\partial y} = -\cos\theta \qquad \frac{\partial f_\theta}{\partial y} = 0$$
$$\frac{\partial f_y}{\partial \theta} = -(x_p' - x)\cos\theta - (y_p' - y)\sin\theta \qquad \frac{\partial f_\theta}{\partial \theta} = -1$$

A.1.3 Computing D

There are two elements in the F vector for each correspondence. There are six elements in the J matrix for each correspondence. See Equation 9.

(EQ 9)

$$F = \begin{bmatrix} f_{1a} \\ f_{1b} \\ f_{2a} \\ f_{2b} \\ f_{3a} \\ f_{3b} \\ \cdots \\ f_{ma} \\ f_{mb} \end{bmatrix} \qquad J = \begin{bmatrix} \frac{\partial f_{1a}}{\partial x} & \frac{\partial f_{1a}}{\partial y} & \frac{\partial f_{1a}}{\partial \theta} \\ \frac{\partial f_{1b}}{\partial x} & \frac{\partial f_{1b}}{\partial y} & \frac{\partial f_{1b}}{\partial \theta} \\ \frac{\partial f_{2a}}{\partial x} & \frac{\partial f_{2a}}{\partial y} & \frac{\partial f_{2a}}{\partial \theta} \\ \frac{\partial f_{2b}}{\partial x} & \frac{\partial f_{2b}}{\partial y} & \frac{\partial f_{2b}}{\partial \theta} \\ \frac{\partial f_{3a}}{\partial x} & \frac{\partial f_{3a}}{\partial y} & \frac{\partial f_{3a}}{\partial \theta} \\ \frac{\partial f_{3b}}{\partial x} & \frac{\partial f_{3b}}{\partial y} & \frac{\partial f_{3b}}{\partial \theta} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_{ma}}{\partial x} & \frac{\partial f_{ma}}{\partial y} & \frac{\partial f_{ma}}{\partial \theta} \\ \frac{\partial f_{mb}}{\partial x} & \frac{\partial f_{mb}}{\partial y} & \frac{\partial f_{mb}}{\partial \theta} \end{bmatrix} \qquad D = \begin{bmatrix} \Delta_x \\ \Delta_y \\ \Delta_\theta \end{bmatrix}$$

$$\varphi = F^T F$$

$$F + JD = 0$$

$$D = \left(J^T J\right)^{-1} J^T F \equiv J^{\dagger} F$$

A.2 Iconic Matching

The distance between a scan point and its corresponding map line segment is defined as the minimum of:

- distance between scan point and map line segment end point
- distance between scan point and map line segment start point
- perpendicular distance between scan point and line which contains map line segment (provided that the scan point lies between the endpoints of the segment)

Let the unknown sensor pose be (x, y, θ) the scan (in the sensor reference frame) be a vector of range values, r, and a vector of azimuth angles, α, and consider a representative map line segment with start point (x_s, y_s) and end point (x_e, y_e). The three different distances (d_s, d_e, and $d_\perp$) are defined in Equations 10 through 12.

(EQ 10)

$$d_s = \sqrt{\left(x + r_i \cos\left(\theta + \alpha_i\right) - x_s\right)^2 + \left(y + r_i \sin\left(\theta + \alpha_i\right) - y_s\right)^2}$$

(EQ 11)

$$d_e = \sqrt{(x + r_i\cos(\theta + \alpha_i) - x_e)^2 + (y + r_i\sin(\theta + \alpha_i) - y_e)^2}$$

(EQ 12)

$$l = \sqrt{(x_e - x_s)^2 + (y_e - y_s)^2}$$

$$a = \frac{-(y_e - y_s)}{l}$$

$$b = \frac{(x_e - x_s)}{l}$$

$$d_\perp = a(x + r_i\cos(\theta + \alpha_i) - x_s) + b(y + r_i\sin(\theta + \alpha_i) - y_s)$$

Matching a set of scan points to a set of line segments is an overdetermined nonlinear optimization problem. The unknowns are the deltas in the three degrees of freedom of the sensor pose (Δ_x, Δ_y, Δ_θ) and the quantity to be minimized is the sum of the squared correspondence errors and is denoted by φ. The Gauss-Newton method [Dahlquist74] is summarized below in Equation 17.

The number of point-to-segment correspondences is *m*. *F* is an *m x 1* column matrix of correspondence errors (d_s, d_e, or $d_\perp$), *J* is an *m x 3* Jacobean matrix of *F* with respect to the sensor pose (x, y, θ), and *D* is a *3 x 1* column matrix of deltas for each sensor pose degree of freedom. By subtracting the delta from the sensor pose at each iteration, the sum of the correspondence error is reduced and the sensor pose estimate is improved.

A.2.1 Computing F

Computing the *F* matrix is straightforward once the correspondence between overlapping parts of scans has been computed as described above. For each of the *m* correspondences, there is an element in the *F* matrix which is the distance between point and line segment. The formula for an error component, *f*, due to an endpoint correspondence appears in Equation 13. The formula for the error due to a point to segment correspondence appears in Equation 14.

(EQ 13)

$$f = d_s = \sqrt{(x + r_i\cos(\theta + \alpha_i) - x_s)^2 + (y + r_i\sin(\theta + \alpha_i) - y_s)^2}$$

$$f = d_e = \sqrt{(x + r_i\cos(\theta + \alpha_i) - x_e)^2 + (y + r_i\sin(\theta + \alpha_i) - y_e)^2}$$

(EQ 14)

$$l = \sqrt{(x_e - x_s)^2 + (y_e - y_s)^2}$$

$$a = \frac{-(y_e - y_s)}{l}$$

$$b = \frac{(x_e - x_s)}{l}$$

$$f = d_\perp = a\left(\left(x + r_i\cos\left(\theta + \alpha_i\right)\right) - x_s\right) + b\left(\left(y + r_i\sin\left(\theta + \alpha_i\right)\right) - y_s\right)$$

A.2.2 Computing J

Computing the J matrix is straightforward; there are two cases.

- The correspondence is between a scan point and a map line segment endpoint, in which case the correspondence error is either d_s or d_e.
- The correspondence is between a scan point and an interior point of a map line segment, in which case the correspondence error is $d_\perp$.

The Jacobeans for each case appear in Equations 15 and 16.

(EQ 15)

$$\frac{\partial f}{\partial x} = \frac{x + r_i\cos\left(\theta + \alpha_i\right) - x_s}{f}$$

$$\frac{\partial f}{\partial y} = \frac{y + r_i\sin\left(\theta + \alpha_i\right) - y_s}{f}$$

$$\frac{\partial f}{\partial \theta} = \frac{\left(r_i\cos\left(\theta + \alpha_i\right)\right)\left(y - y_s\right) - \left(r_i\sin\left(\theta + \alpha_i\right)\right)\left(x - x_s\right)}{f}$$

(EQ 16)

$$\frac{\partial f}{\partial x} = a$$

$$\frac{\partial f}{\partial y} = b$$

$$\frac{\partial f}{\partial \theta} = -a\left(r_i\sin\left(\theta + \alpha_i\right)\right) + b\left(r_i\cos\left(\theta + \alpha_i\right)\right)$$

A.2.3 Computing D

Computing the D matrix is accomplished by iteratively applying the formula shown in Equation 17. On each iteration, the sensor pose is adjusted by subtracting the three components of D (Δ_x, Δ_y, Δ_θ) from the sensor pose.

(EQ 17)

$$F = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \cdots \\ f_m \end{bmatrix} \qquad J = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial \theta} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial \theta} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial \theta} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_m}{\partial x} & \frac{\partial f_m}{\partial y} & \frac{\partial f_m}{\partial \theta} \end{bmatrix} \qquad D = \begin{bmatrix} \Delta_x \\ \Delta_y \\ \Delta_\theta \end{bmatrix}$$

$$\varphi = F^T F$$

$$F + JD = 0$$

$$D = \left(J^T J\right)^{-1} J^T F \equiv J^{\dagger} F$$

One can iterate computation of F, J, and D until a desired level of convergence is achieved (the deltas are sufficiently small). The resulting accuracy in the sensor pose estimate will depend upon the accuracy of the sensor data, the correctness of the correspondence determination, and the validity of the correspondence error metric.

A.3 Multiple Iconic Matching

A.3.1 Computing F

The correspondence is between a point of an adjustable scan (x_a, y_a) and a segment formed by connecting two adjacent points (x_s, y_s) and (x_e, y_e) in another scan. The error is the minimum of the distance from (x_a, y_a) to (x_s, y_s) or (x_e, y_e), or if (x_a, y_a) lies between (x_s, y_s) and (x_e, y_e), the perpendicular distance from (x_a, y_a) to the line containing (x_s, y_s) and (x_e, y_e). The three possibilities are summarized in Equation 18.

(EQ 18)

$$l = \sqrt{(x_e - x_s)^2 + (y_e - y_s)^2}$$

$$a = \frac{-(y_e - y_s)}{l}$$

$$b = \frac{(x_e - x_s)}{l}$$

$$f_s = \sqrt{(x_a - x_s)^2 + (y_a - y_s)^2}$$

$$f_e = \sqrt{(x_a - x_e)^2 + (y_a - y_e)^2}$$

$$f_\perp = a(x_a - x_s) + b(y_a - y_s)$$

A.3.2 Computing J

Computing the J matrix is straightforward; there are four cases. The correspondence is between a point of an adjustable scan (referred to as scan #m) and:

1. a point from an adjustable scan #n
2. a segment from an adjustable scan #n
3. a point from a fixed scan #n
4. a segment from a fixed scan #n

The Jacobeans for cases 1 through 4 cases appear in Equations 19 through 23.

(EQ 19)

$$f = \sqrt{\left(\left(x_m + r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right) - \left(x_n + r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)\right)^2 + \left(\left(y_m + r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right) - \left(y_n + r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)\right)^2}$$

$$\frac{\partial f}{\partial x_m} = \frac{\left(x_m + r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right) - \left(x_n + r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)}{f}$$

$$\frac{\partial f}{\partial y_m} = \frac{\left(y_m + r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right) - \left(y_n + r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)}{f}$$

$$\frac{\partial f}{\partial \theta_m} = \frac{\left(r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right)\left(y_m - \left(y_n + r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)\right) - \left(r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right)\left(x_m - \left(x_n + r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)\right)}{f}$$

$$\frac{\partial f}{\partial x_n} = \frac{\left(x_n + r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right) - \left(x_m + r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right)}{f}$$

$$\frac{\partial f}{\partial y_n} = \frac{\left(y_n + r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right) - \left(y_m + r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right)}{f}$$

$$\frac{\partial f}{\partial \theta_n} = \frac{\left(r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)\left(\left(x_m + r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right) - x_n\right) - \left(r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)\left(\left(y_m + r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right) - y_n\right)}{f}$$

(EQ 20)

$$l = \sqrt{\left(r_{n_{j+1}}\cos\left(\theta_n + \alpha_{n_{j+1}}\right) - r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)^2 + \left(r_{n_{j+1}}\sin\left(\theta_n + \alpha_{n_{j+1}}\right) - r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)^2}$$

$$a = \frac{-\left(r_{n_{j+1}}\sin\left(\theta_n + \alpha_{n_{j+1}}\right) - r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)}{l}$$

$$b = \frac{\left(r_{n_{j+1}}\cos\left(\theta_n + \alpha_{n_{j+1}}\right) - r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)}{l}$$

$$f = a\left(\left(x_m + r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right) - \left(x_n + r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)\right) + b\left(\left(y_m + r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right) - r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)$$

$$\frac{\partial f}{\partial x_m} = a$$

$$\frac{\partial f}{\partial y_m} = b$$

$$\frac{\partial f}{\partial \theta_m} = -a\left(r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right) + b\left(r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right)$$

$$\frac{\partial f}{\partial x_n} = -a$$

$$\frac{\partial f}{\partial y_n} = -b$$

$$\frac{\partial f}{\partial \theta_n} = a\left(\left(y_m + r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right) - y_n\right) - b\left(\left(x_m + r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right) - x_n\right)$$

(EQ 21)

$$f = \sqrt{\left(\left(x_m + r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right) - \left(x_n + r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)\right)^2 + \left(\left(y_m + r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right) - \left(y_n + r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)\right)^2}$$

$$\frac{\partial f}{\partial x_m} = \frac{\left(x_m + r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right) - \left(x_n + r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)}{f}$$

$$\frac{\partial f}{\partial y_m} = \frac{\left(y_m + r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right) - \left(y_n + r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)}{f}$$

$$\frac{\partial f}{\partial \theta_m} = \frac{\left(r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right)\left(y_m - \left(y_n + r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)\right) - \left(r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right)\left(x_m - \left(x_n + r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)\right)}{f}$$

$$\frac{\partial f}{\partial x_n} \equiv 0$$

$$\frac{\partial f}{\partial y_n} \equiv 0$$

$$\frac{\partial f}{\partial \theta_n} \equiv 0$$

(EQ 22)

$$l = \sqrt{\left(r_{n_{j+1}}\cos\left(\theta_n + \alpha_{n_{j+1}}\right) - r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)^2 + \left(r_{n_{j+1}}\sin\left(\theta_n + \alpha_{n_{j+1}}\right) - r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)^2}$$

$$a = \frac{-\left(r_{n_{j+1}}\sin\left(\theta_n + \alpha_{n_{j+1}}\right) - r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)}{l}$$

$$b = \frac{\left(r_{n_{j+1}}\cos\left(\theta_n + \alpha_{n_{j+1}}\right) - r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)}{l}$$

$$f = a\left(\left(x_m + r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right) - \left(x_n + r_{n_j}\cos\left(\theta_n + \alpha_{n_j}\right)\right)\right) + b\left(\left(y_m + r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right) - r_{n_j}\sin\left(\theta_n + \alpha_{n_j}\right)\right)$$

$$\frac{\partial f}{\partial x_m} = a$$

$$\frac{\partial f}{\partial y_m} = b$$

$$\frac{\partial f}{\partial \theta_m} = -a\left(r_{m_i}\sin\left(\theta_m + \alpha_{m_i}\right)\right) + b\left(r_{m_i}\cos\left(\theta_m + \alpha_{m_i}\right)\right)$$

$$\frac{\partial f}{\partial x_n} \equiv 0$$

$$\frac{\partial f}{\partial y_n} \equiv 0$$

$$\frac{\partial f}{\partial \theta_n} \equiv 0$$

A.3.3 *Computing D*

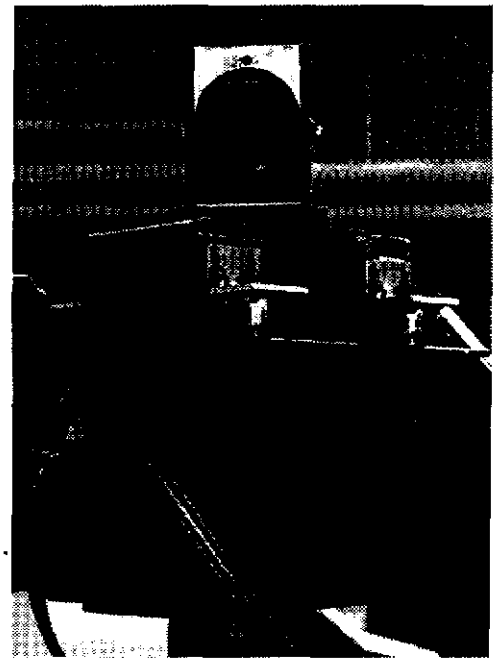
The method of solving for D is the same as in the single pose case in Equation 17. The matrices are just larger. See Equation 23. In our implementation, to reduce the storage requirements, we don't actually build the J matrix. Instead, we incrementally compute the J^TJ matrix. Inverting the J^TJ matrix takes a long time.

(EQ 23)

$$F = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \cdots \\ f_m \end{bmatrix} \qquad J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial y_2} & \frac{\partial f_1}{\partial \theta_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial y_3} & \frac{\partial f_1}{\partial \theta_3} & \cdots & \frac{\partial f_1}{\partial x_n} & \frac{\partial f_1}{\partial y_n} & \frac{\partial f_1}{\partial \theta_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial y_2} & \frac{\partial f_2}{\partial \theta_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial y_3} & \frac{\partial f_2}{\partial \theta_3} & \cdots & \frac{\partial f_2}{\partial x_n} & \frac{\partial f_2}{\partial y_n} & \frac{\partial f_2}{\partial \theta_n} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial y_1} & \frac{\partial f_3}{\partial \theta_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial y_2} & \frac{\partial f_3}{\partial \theta_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial y_3} & \frac{\partial f_3}{\partial \theta_3} & \cdots & \frac{\partial f_3}{\partial x_n} & \frac{\partial f_3}{\partial y_n} & \frac{\partial f_3}{\partial \theta_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial y_1} & \frac{\partial f_m}{\partial \theta_1} & \frac{\partial f_m}{\partial x_2} & \frac{\partial f_m}{\partial y_2} & \frac{\partial f_m}{\partial \theta_2} & \frac{\partial f_m}{\partial x_3} & \frac{\partial f_m}{\partial y_3} & \frac{\partial f_m}{\partial \theta_3} & \cdots & \frac{\partial f_m}{\partial x_n} & \frac{\partial f_m}{\partial y_n} & \frac{\partial f_m}{\partial \theta_n} \end{bmatrix} \qquad D = \begin{bmatrix} \Delta_{x_1} \\ \Delta_{y_1} \\ \Delta_{\theta_1} \\ \Delta_{x_2} \\ \Delta_{y_2} \\ \Delta_{\theta_2} \\ \Delta_{x_3} \\ \Delta_{y_3} \\ \Delta_{\theta_3} \\ \cdots \\ \Delta_{x_n} \\ \Delta_{y_n} \\ \Delta_{\theta_n} \end{bmatrix}$$

Appendix B Cyclone

The Cyclone laser range sensor was developed at the CMU Field Robotics Center [Singh91] to acquire fast, precise scans of range data over long distances (up to 50m). The sensor consists of a pulsed Gallium Arsenide infrared laser transmitter/receiver pair, aimed vertically upward. Like a lighthouse, a mirror in the tower part of the scanner rotates about the vertical axis and deflects the laser so that it shoots parallel to the ground, creating a 360° field of view. The tower rotation is belt-driven by a motor at speeds from 0.5Hz to 5Hz. The resolution of range measurements is 10 cm; the accuracy is ±20 cm. The angular resolution is selectable from one of 256, 512, 1024, 2048, 4096, or 8192.



Developer

- CMU Field Robotics Center
- 1987

Physical Specifications

- approximately 6"x16"x24"
- approximately 80 lbs

Range Sensor Specifications

- time of flight
- minimum range approx 4 m
- maximum range approx 100 m
- range resolution 10 cm
- range accuracy 1 std dev = approx 10 cm (depends on conditions)

Laser Specifications

- custom made by Riegl Inc.
- wavelength 905 nm
- output energy 600 nJ/pulse
- pulse width 30 to 40 ns
- maximum pulse rate 7.2 kHz
- beam divergence 4 mrad

Scanning Specifications

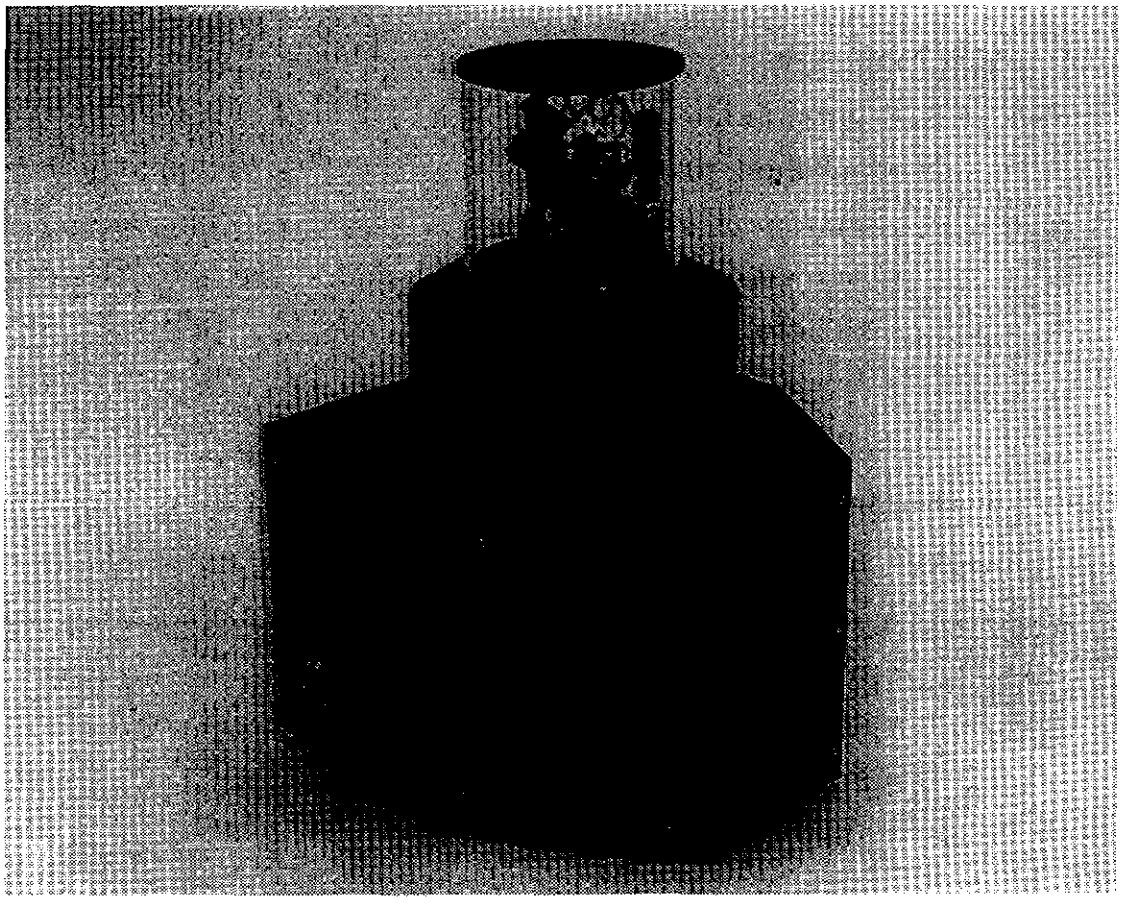
- rotation rate programmable up to 7 Hz
- vertical field of view n/a
- horizontal field of view 360 deg
- vertical angular resolution n/a
- horizontal angular resolution approx 0.044 deg
- resolution is selectable between 512 and 8192 samples/rev

Interface

- 16 bit parallel

Appendix C Typhoon

The Typhoon laser range sensor was developed by K2T Inc. to serve as a mapping, navigation, and/or obstacle detection sensor for underground coal mining. The sensor consists of a laser which points vertically upward at a rotating mirror. The mirror deflects the laser in a programmed scan pattern, capable of a maximum 40 degree vertical by 360 degree horizontal field of view, and a maximum resolution of 200 lines of 2048 points.



Manufacturer

- K2T Inc., Pittsburgh, PA
- 1994

Physical Specifications

- approximately 1 cubic foot
- spproximately 20 lbs

Range Sensor Specifications

- time of flight
- minimum range approx 1.5 m
- maximum range approx 45 m
- range resolution approx 4.4 cm
- range accuracy 1 std dev = approx 4.4 cm (depends on conditions)

Laser Specifications

- custom made by Schwartz Electro-Optic Inc.
- wavelength 905 nm
- output energy 50 nJ/pulse
- pulse width 7.5 ns
- maximum pulse rate 20 kHz
- beam divergence 1x3 mrad

Scanning Specifications

- rotation rate approx 10 Hz
- vertical field of view 40 deg
- horizontal field of view 360 deg
- vertical angular resolution 0.2 deg
- horizontal angular resolution approx 0.17 deg
- scan pattern and resolution are highly programmable

Interface

- high speed 16 bit parallel

Appendix D Locomotion Emulator

The Locomotion Emulator (LE) is a mobile robot that was developed at the CMU Field Robotics Center as a testbed for development of mobile robotic systems. The LE consists of a *locomotor*, a mechanism capable of completely general locomotion on a flat surface, and an *emulator*, a software environment that enables this mechanism to mimic the characteristics of different vehicles. The locomotor is a powerful all-wheel steer, all-wheel drive base with a rotating payload platform. The steering motions of the LE's three wheel modules are belt driven by one motor; its three drive motions are belt driven by another. Two shock-isolated enclosures, located between the wheel modules, house the onboard electronics and computing. Shore power is provided to the LE via a tether. The emulation software supports the four most common steering modules of wheeled robots -- unicycle, Ackerman, skid, and articulated. See Fitzpatrick [Fitzpatrick89]. An operator or computer host can issue commands specific to any of these configurations, and the LE generates appropriate motions of its steer, drive, and payload platform to replicate the motions of the target vehicle.

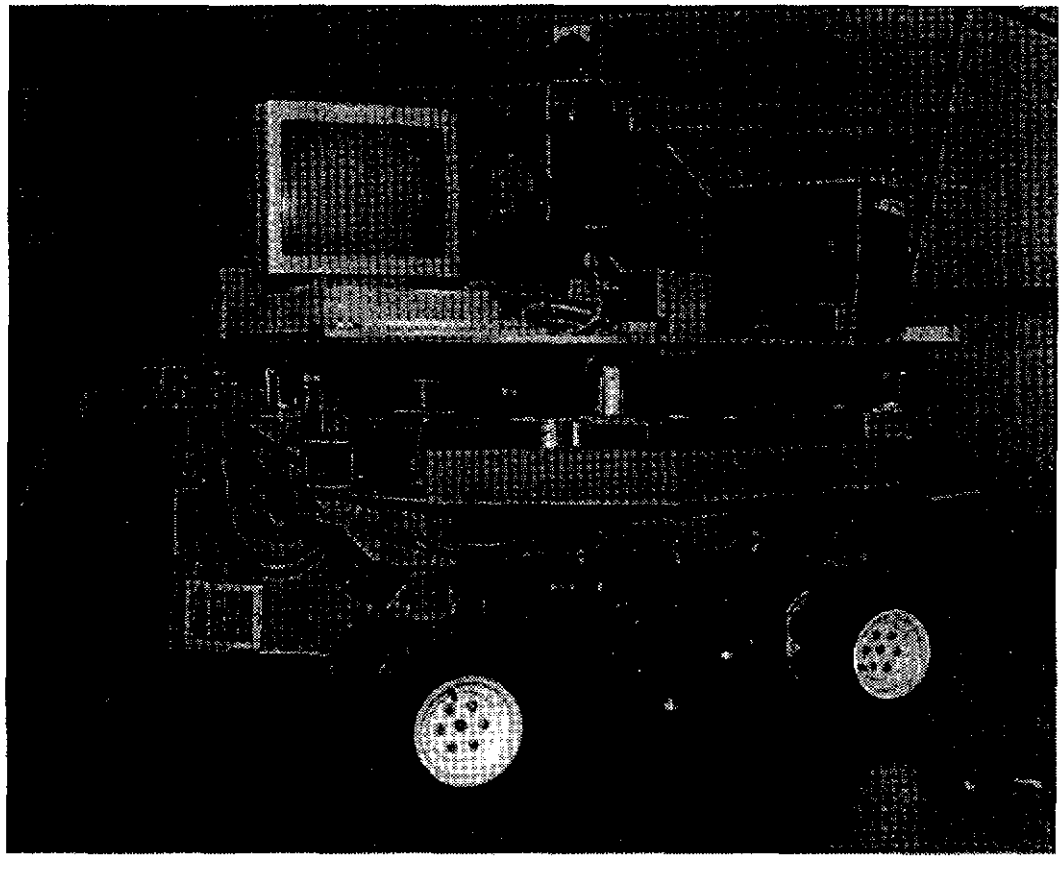


FIGURE 48. The Locomotion Emulator with Cyclone, a Sun 3/60 and a color monitor on top. Photo is from 1990.