Map-making and Localization for Mobile Robots using Shape Metrics

Russell G. Brown[*], L. Paul Chew[†] and Bruce R. Donald[*]

Computer Science Department

Cornell University

Ithaca, New York

Abstract

We consider the problem of enabling an autonomous mobile robot to get from one specified point in its environment to another. The solution to this problem requires the robot to build the representations of its environment to enable it to determine its current location, its destination, and how it can get from one to the other. Many pitfalls have been found trying to achieve the goal of a robustly navigating robot. Among these are: difficulty in getting good sensor information about the robot's world and difficulty dealing with changing environments. Given these two problems, a third question emerges which is, how does the robot deal with uncertainty both in its environment and in its own position within that environment.

In this paper, we report on algorithms and sensor designs intended to improve the performance of a mobile robot in navigational tasks. These include (i) a simple laser rangefinder designed to give accurate point-and-shoot range readings, (ii) an algorithm enabling the robot to identify which areas of its environment have remained static over time and those which have been particularly time-varying, and (iii) an algorithm for localization of a mobile robot using a previously constructed model of the world and current sensor readings. These ideas allow a robot to plan and perform experiments to obtain needed information about its environment. We are implementing these ideas on mobile robotic platforms at the Cornell Robotics and Vision Laboratory. All of these ideas are at least partially implemented, and preliminary experimental results are included to demonstrate proof of concept for these algorithms and sensors.

[*]This paper describes research done in the Robotics and Vision Laboratory at Cornell University. Support for our robotics research is provided in part by the National Science Foundation under grants No. IRI-8802390, IRI-9000532 and by a Presidential Young Investigator award, and in part by the Air Force Office of Sponsored Research, the Mathematical Sciences Institute, Intel Corporation, and AT&T Bell laboratories.

[†]This author's work was supported by the Advanced Research Projects Agency of the Department of Defense under ONR Contract N00014-88-K-0591, and by ONR Contract N00014-89-J-1946, NSF Contract IRI-9006137, and AFOSR Contract AFOSR-91-0328.

1 Introduction

We would like to build a robot which can, for a reasonable class of environments, fulfill tasks of the form "go to x and get y," or perhaps "take z to x." We would like to be able to specify x, y, and z in the same form to our robot that we would to another human being. We realize that we will not soon be able to specify such x or y as "Debbie's office," or "the Intel package," but we still wish to enable the robot to build a representation of its environment which we can annotate with information such as "location x is Debbie's office," or "the Intel package is an object meeting description y, and have the robot be able to go to x and locate y. Much work has been done on building and using representations of the environment (particularly representations built by the robot without human intervention) for navigation, and some progress has been made. A lot of work remains to be done.

A number of difficulties have prevented the completely successful implementation of a navigating robot. Most of these problems stem from sensing and control uncertainty in any navigational system, and from the difficulty of maintaining an accurate representation of a robot's environment as it changes over time. The following sections briefly discuss some previous efforts made to overcome these difficulties, and present our approaches to solving some of these problems. Section 2 describes the experimental apparatus which we are using to explore these issues. Section 3 describes some map-making tactics for dealing dynamic environments. Section 4 describes algorithms for reducing a robot's uncertainty of position. These results are computational geometry algorithms providing metrics for shape comparison. These algorithms were originally designed for use in model-based recognition systems for computer vision applications. We will show that, by applying these algorithms to rasterized maps, in place of greyscale images, several of these algorithms have significant potential to improve the performance of mobile robots performing navigation-oriented tasks.

1.1 Key Ideas

Here are some ideas that our paper foregrounds. We discuss how mobile robotic systems may benefit from tailoring sensors to meet the spec of map-making

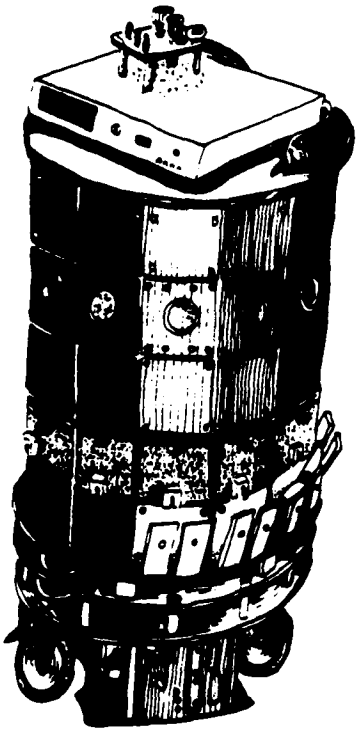

Figure 1: Tommy, the mobile robot

and localization algorithms. This exploits a kind of "impedance matching" between the uncertainties inherent in the sensors and those tolerated by the algorithms. In particular, we consider time-varying, uncertain environments of which the robot has no prior model. Second, we attempt to place landmark recognition on a firm algorithmic footing. Third, we attempt to integrate algorithms for map-making, localization, and landmark recognition with path planning and navigation algorithms. We do this within the common framework of rasterized computational geometry. Fourth, the techniques above permit us to investigate a robot system which can perform experiments to gain information about the environment, as driven by the demands of the task. Finally, we attempt both predictive and post-hoc analyses of the performance and competence of our system. The predictive analyses devolve to exploiting of the mathematical properties of the underlying algorithms (particularly shape metrics) and their computational complexity. The post-hoc analyses result from experimental studies performed on our implemented mobile robot system.

2 The Robot

Our lab has four Cornell Mobile robots (figure 1), built on Real World Interfaces B12 Mobile Robot Bases. We have equipped these bases with RWI modular enclosures, enabling us to put computational resources and sensory devices onto the mobots in a modular fashion. The robot on which we performed most of the experiments written about here is equipped with a ring of RWI-built sonar transducers, an ring of infrared proximity sensors, a set of bump sensors, and a laser rangefinder. The laser rangefinder is the most important sensory device for this paper, providing accurate point-and-shoot distance measurements. It is worth spending a couple of paragraphs discussing the design of this sensor, and showing a map which our robot has built using data obtained from the sensor. The robot is run by a Cornell Generic Controller, based on the Intel 80C196KB microcontroller, which is used to control the sensors and actuators on the robot, and a Gespak 68K single board which is running a version of Scheme modified to run on the robot. The Scheme environment allows the mobile robot to be programmed in a high level language, and with an easy to use execution and debugging environment. This is accomplished by tying the Scheme read-eval-print loop running on the Gespak to a Sparcstation running Sun Common LISP. The symbol table and other sorts of information which are needed for a symbolic debugger are kept in the Sparcstation, allowing the Scheme to run efficiently on the smaller processor in the robot, while supplying the user with a nice environment. Scheme programs running in the robot are self-contained, and we can run the robot independently by unplugging the communications tether from the read-eval-print loop. When unplugged, the robot can execute Scheme programs; when plugged in to the Sparcstation, it can be debugged easily—the environment supports downloading of byte-compiled code to the robot, as well as the defining of new functions within the read-eval-print loop running on the robot. For a more detailed survey of this environment, see [RD92].

The laser rangefinder on the robot consists of three laser diodes, a video camera, and a simplified "partial" frame grabber designed in our lab. The diodes are configured to project three parallel beams. The video camera and the frame grabber are configured to take a picture of the projection of the diodes into the environment. Once the picture is grabbed, it is simple to find the bright spots in the image which image the intersection of the laser beams with the environment (we put a red filter on the camera to reduce the brightness of ambient light and use lowpass filtering to emphasize the spots we are looking for). A small amount of simple trigonometry computes a pair of range values which are very accurate over the range of 300 to 3000 mm. The accuracy is typically better than 10mm in cases where the returned value is approximately correct; in the vast majority of cases, the reading returned by the ranger is either correct to within 10mm, or takes on an impossible value (less than 200mm or greater than 5-10 meters). In addition, since the range information is contained primarily in the locations of the left and right diode spots, the ranger is capable of observing fairly sharp discontinuities in the world—the spot being measured is small enough that boundaries are typically not blurred. We designed the ranger to provide a pair of measurements with a center reference point (the middle beam) so that we can also compute a local surface normal. The partial frame grabber digitizes only a portion of the video picture. This enables the grabber to utilize a fairly modest hardware design. The grabber consists of an Intel 80C196KB microcontroller, 8K of EPROM, 56K of static RAM divided between the digitizer and the processor, and some support hardware. It allows the acquisition of anywhere from a single scan line up to 1/4 of a full frame. Image acquisition can be done at full frame rate if processing

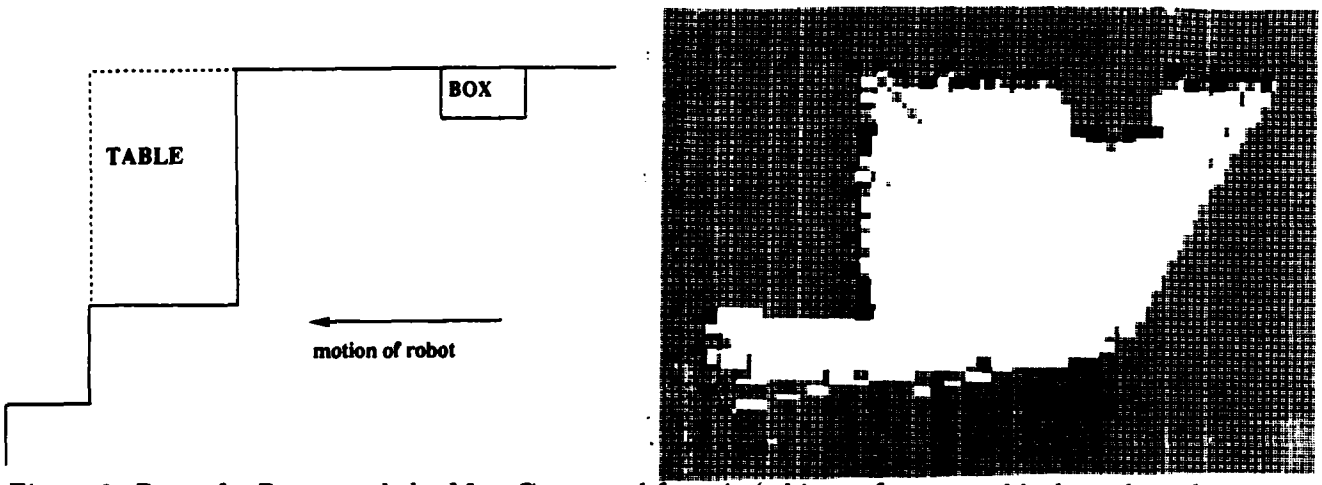

Figure 2: Part of a Room and the Map Generated from it (white = freespace, black = obstacle, grey = unsure/unknown)

requirements are small; we can acquire an image and perform ranging computations at a rate of 15 rangings per second. The rangefinder (the diodes, the camera, and the frame grabber) is mounted on a pan/tilt head designed and fabricated in our lab. This is a two stage platform which enables the rangefinder to be servoed in a circle (parallel to the ground) at speeds of up to at least 1Hz, and tilted to point in any direction within plus and minus 30 degrees of horizontal. This is an important part of the system, as it allows us to point the ranger in any direction we want without having to turn the robot, and allows us either to automatically scan the world around the robot, or to direct the ranger's attention to a particular point in the world.

Figure 2a shows a sketch of part of the lab. It shows a couple of walls, a cardboard box, and a table turned on its side. This is a slightly engineered environment, in the sense that the upturned tables are easier to see than they would be when upright. This is due entirely to the height of the table being greater than that of the robot and to the difficulty in noticing small items like table legs in a small number of ranging probes (it's too easy to look straight "through" the leg). Figure 2b shows a piece of map obtained by scanning the rangefinder around a circle several times while moving the robot in a straight line. The map was generated using a simplified form of the statistical occupancy grid representation used by ([ME85, Elf87]), described in more detail in section 3.1. The robot updated the map with each range reading by lowering the probability of occupancy of any cell lying along the interior of the range line, and raising the probability of occupancy of the cell which contains the terminus. The update rule used was the following: If a cell registers as empty during a particular update, its probability of occupancy is multiplied by a value α less than one; if it registers as occupied, its probability of emptiness (one minus the probability of occupancy) is multiplied by α. We have used a value for α of 1/3 with good results. The map is a three valued representation obtained by thresholding the occupancy grid at two different values (in this case, 0.25 and 0.55), in which black represents object boundaries, white represents free space (with relatively high confidence), and grey represents places of unknown occupancy (either from contradictory readings or from not having looked—grey includes places the ranger has never seen because of occlusion). The rangefinder did a reasonable job of mapping the part of the environment it was able to examine. Comparison of the map to the physical environment shows that the parts of the map which appear flat are, in fact, placed with good accuracy. Whether the recognition of these clusters of dark pixels as line segments can be successfully automated has yet to be tested in this particular application, but the specific map shown proved amenable in experiment to edge detection and segmentation, using algorithms developed for computer vision applications.

3 Map-Making Algorithms

A *map* is a data structure representating part of the world. In the context of robotics, a map is a representation which can be used to accomplish tasks. Many researchers have developed algorithms for building representations of the environment from data returned by range sensors, primarily using sonar transducers. This section will begin by reviewing a few important results in map-making. From there, we'll go on to show how these results can be used to launch more sophisticated systems, by using an improved range sensor and by creating more sophisticated algorithms.

3.1 Previous Work in Map-Making

Maps for robotic navigation come in several flavors: *grid-based* and *feature-based* are the two largest divisions in the classification of maps, with feature-based maps divided into those with and those without a "strong" sense of geometry. *Strong geometry* in this context means that features are annotated with spatial coordinates, as opposed to simply topological connectivity information. Grid-based maps and feature-based maps equipped with geometry are typically built

where the robot can use some form of odometry as at least a part of its navigational program. Feature-based maps without strong geometry are generally used when the navigation program is designed to go from recognizable place to recognizable place without using coordinates or odometry, either of places in the world, or of its current position. Conceptually, grid-based maps resemble large checkerboards, while feature-based maps are planar graphs with nodes representing locations and edges representing actions to move between them.

One important method is the statistical occupancy grid method used by Moravec and Elfes [Elf87, ME85]. This method was developed to make maps from data returned by sonar transducers, using statistical methods to produce a map which could be expected to be far more accurate than any individual sonar reading used to generate it. The method is essentially the following: initially, every cell in the grid is occupied with probability one half. After a sonar ping (or other range reading), the cell which is in the direction pinged at the distance value returned by the sensor has its probability of occupancy raised using a bayesian model. All cells in the direction pinged closer to the robot than that distance value have their occupancy probability lowered using the same model. This is based on the notion that if one gets a range value at (say) ten feet, then there *probably* is something there, and there *probably* is not anything in that direction closer than ten feet. Since we don't trust our range sensors unconditionally, we use the statistical occupancy method rather than just a binary map because it greatly simplifies the problems of integrating newly measured data into the representation built from previously acquired data. This is particularly useful for integrating potentially conflicting range data taken from different positions and angles.

Leonard, Durrant-Whyte, and Cox recently presented a feature-based map-making process which uses geometry [LDWC90]. Their technique is to locate landmarks, which are locations in the environment which appear the same when ranged from different directions. These landmarks are typically things like planes and corners, and they identify them using model-based Kalman filtering. Using this method, it is possible to build maps which consist of sets of landmarks, along with their coordinates, and from what angles they are visible.

Mataric's master's thesis describes a feature-based map-making process which uses only limited geometry[Mat90]. The technique she uses is to find landmarks, which are similar in many ways to "natural" beacons, but to maintain the map at a topological level only, building a symbolic map which represents landmarks as nodes in a graph, with edges connecting pairs of landmarks between which it is easy for the robot to move. Landmarks are chosen by hand so that they are (a) easily distinguishable from one another when sensed from nearby and (b) close enough together that purely local operations suffice to get a robot from a given landmark to an adjacent landmark in the connectivity graph.

There is a lot of previous work not covered here. We chose to describe these particular works because they are representative of the trends and themes existing throughout work on this topic.

3.2 Map-Making Improvements

The previous section outlined a number of techniques for creating maps which a robot can use to represent its environment and to navigate within it. This section concentrates on an enhancement of the map-making process which allows the robot access to information about the time-varying behavior of various parts of its environment. In other words, the robot gathers information about what parts of its environment are likely to change (for example, doors and chairs, as opposed to walls). The enhancement is presented as a modification of the statistical occupancy grid algorithm, but it should be stressed that the idea of annotating map features with variability values can be applied to many different world representations. The main purposes for selecting the statistical grid are the ease of implementation, and also the ease of explanation that a simple representation such as this one has to offer. Another major advantage of a grid-based representation is that it admits easy use of rasterized algorithms. A *rasterized algorithm* employs the following technique: given a geometric description (for example, lists of edges or vertices), convert that description to a grid by digitizing the primitives (in essence, "plotting" geometric primitives in the description on an initially blank bitmap). Given an algorithm that runs on geometric descriptions, convert (by hand) the algorithm to perform on a discretized grid. This procedure is analogous to discrete approximation of differential equations describing a physical system, enabling finite grid methods to be used to simulate the system. We are finding that algorithms operating on discretized arrays are useful, in that they are often easier to implement than the original combinatorial algorithms, and they often run faster. [LRDG90] exemplifies how rasterization can be applied to robotics algorithms.

We propose adding the following to the grid maintenance algorithm: Keep track, for each cell in the grid, not only of probability of occupancy, but of the derivative, with respect to time, of that probability. To be more precise, let ρ be the statistical occupancy grid representing a given environment. $\rho(x, y, t)$ is the probability of occupancy of cell (x, y) at time t. We want to keep track of the time derivative of this function, $\dot{\rho} = \frac{\partial \rho}{\partial t}$. $\rho(x, y, t)$ is the rate at which cell (x, y) is changing at time t. Now, define Γ to be the time integral of the absolute value of $\dot{\rho}$:

$$\Gamma(x, y, t) = \int_{t_0}^{t} |\dot{\rho}(x, y, \tau)| d\tau \qquad (1)$$

$\Gamma(x, y, t)$ is a measure of the total amount of change in cell (x, y) since some specified time t_0. In practice, we approximate (1) using finite differences. If we were to compare Γ for cells in a grid to the "real world" areas they represent, we would expect to find the following: Things like cabinets and walls which seldom move would have very small values of Γ; the same should be true for open spaces which are seldom if ever occupied. On the other hand, if we found a group of cells which had large values for Γ, we'd likely find that they were things like doors, chairs, or very slow graduate students, to name a few. This is particularly true if we use a high-accuracy sensor, such as a

laser ranger, where we expect few spurious changes in occupancy probabilities; it is less true if we use sensors which are noisy or which have large sensing areas, such as sonars. That is, Γ measures Δ (signal), equal to Δ (world + noise). In order for Γ to be a useful value, Δ (noise) must be small. It may be possible to recognize doors, not by their shape but by the fact that they are thin rectangular areas which have high Γ values, from being closed part of the time and open part of the time. It's certainly the case that, even if the robot cannot segment out and recognize things like doors, one can configure the planner used by the robot (for example, our system incorporates a path planner, based on [LRDG90]) to treat spaces which, at a current time t_1 have low values for $\rho(x, y, t_1)$ but high values for $\Gamma(x, y, t_1)$ as dangerous. Unless the map's information on such areas is very recent or the robot can directly sense that the space is, in fact, open, the planner should avoid them. This information is very useful to a navigational planner; when the robot using that planner is called on to navigate from one part of the world to another, the planner can plan its navigation strategy based on the current values of Γ: If, in the map which represents the current value of ρ, there is an open path on which all cells have low Γ values, then the navigator can choose to follow that path, with reasonable confidence that it will be able to achieve its objective (this can be implemented easily by thresholding on the value of Γ). If, on the other hand, the most attractive path to the goal passes through cells with high Γ values, the planner will need to engage in active sensing by planning an experiment to go and find out whether those cells are, in fact, currently open. It could find this out by going where it can see those cells and using its ranger to determine if it can see through those cells. If the cells are currently open, the robot can proceed; otherwise it must recover by planning another path to its goal. Thus, the robot can base its actions on its need for information. If the shortcut is open, the robot can pass on through, but it must be prepared to plan for going the long way around, if necessary.

The preceeding paragraphs presented some of ideas used to build internal representations of the world. In addition, we presented some advanced ideas for making better internal representations, representations which are more suitable for a robot operating over an extended period of time in a dynamic environment—we provide the robot not only with information about the state of parts of the environment the last time it saw them, but with some idea of how meaningful older pieces of information about the environment are. This allows the robot to plan experiments to gather information it needs from its environment. It should be emphasized that, given appropriate sensor technology, for example the laser ranger described earlier, the ideas presented here are easy to implement, due to their simplicity. So, the extent to which they are useful depends a lot on the accuracy of the sensors being used, and in the time-varying case, on whether the robot senses and computes quickly enough to be able to sense changes in the environment—if the world changes too quickly, then the robot will have difficulty keeping up. Our robot's behavior in experiments run with a preliminary implementation of these ideas has been promising; further testing is required and is ongoing; further results will be forthcoming.

4 Mobile Robot Localization

Localization is the process of determining the robot's location within the environment. More precisely, it is a procedure which takes as input a map, an estimate of the robot's current pose, and a set of sensor readings. Any of these may be incomplete, and all are toleranced by error bounds. The output is a new estimate of the robot's current pose. By *pose*, we mean either the position and orientation of the robot in the world or, "equivalently" (in a sense we make precise below), the translation and rotation necessary to make a set of sensor readings (or a partial map, built from those readings) best match an *a priori*, global map. In this section, we will explore some work that has been done on localization and our chief result for this paper, which is the application and modification of some computational geometry algorithms, developed primarily for model-based recognition in computer vision, to the task of localizing a point robot in a planar environment.

4.1 Why Localization?

In the domain of robotic manipulator arms, position sensing is a sensing primitive—it is typically possible to get a position reading which is within some (small) error bound of the actual position of the manipulator. This is because, with a manipulator arm, there is typically a stationary base to which the robot is rigidly attached; determining position becomes a matter of reading joint angles and performing kinematic calculations. With mobile robots, these assumptions to not hold, and odometry from wheel shaft encoders is fraught with error. One early method devised for tracking the position of a moving object is *dead reckoning*. Dead reckoning is, simply put, starting from a known position and computing current position by integrating velocity over time. The idea is, if you move distance r in direction θ, you can add the vector $\langle r, \theta \rangle$ to your original position to obtain your new position. This is fine, in principle, but it works extremely poorly in practice. The reason is control and sensing uncertainty. For example, suppose a robot has perfect orientation sensing and can sense distance moved to within one centimeter per motion. Then, after 100 motions, it only knows its position to within one meter. Even these assumptions (perfect orientation, 1cm accuracy in translation) are actually optimistic, particularly if the robot is allowed to translate and rotate simultaneously. *Used as the only sensor*, dead reckoning is practically useless.

With this in mind, researchers have developed a number of techniques for reducing the robot's positional uncertainty. One of the easiest ways to reduce uncertainty is to have the robot build a graph (whose nodes represent individual locations in its environment) chosen so that movement between states can be represented with deterministic transitions. Since the nodes of these graphs are essentially the recognizable states of the robot's environment, the robot is perfectly localized with respect to its map as long as it knows with certainty that it is in a particular single node of its map. If dead reckoning is good enough to get it from one state to another reliably, then using

dead reckoning to move between states introduces no extra uncertainty. This is one of the underlying concepts behind, for instance, the navigation techniques used in [Mat90].

Work has also been done toward developing localization techniques and algorithms for geometric representations. Having developed the statistical occupancy grid representation, Moravec and Elfes [ME85] went on to develop an algorithm for matching two maps. This algorithm returned the pose which would best match one map to the other. This allowed the robot to compare its current map, constructed from current sensor data, to an a priori, earlier map, and determine what positional and orientational error might have been introduced during motion about the environment. This early localization algorithm was unsatisfactory for a variety of reasons. One disadvantage was that their algorithm compared two *complete* maps, rather than comparing a purely local map, reconstructed from local sensory data, to a previously constructed global map. Another disadvantage was that the measure of similarity in [ME85] lacked a mathematical property that we desire: We want the similarity values obtained by comparing a pair of maps at adjacent (or nearby) poses to be roughly the same. We call this property *locality*, and define it more formally in section 4.2.

A later algorithm, from the computational geometry community, provides a theoretical approach to robot localization. Guibas, et al, in [GMR91], present the localization problem as a matching problem between an environment polygon and a "star-shaped" visibility polygon representing the output of a swept rangefinder. This algorithm provides a list of poses; each is a point within the environment polygon from which the view is consistent with the star-shaped polygon. The two main *lacunae* of this result are (i) there is no attempt to deal with uncertainty, and (ii) this result assumes a static environment—the algorithm is optimized with preprocessing to allow quick processing of queries. Note that these problems are not insurmountable, since the algorithm may well be adapted to compensate for these problems, but they remain to be extended.

4.2 Shape Metrics for Localization

Suppose we wish to determine what change in the robot's pose will produce the best correspondence between its current local map (based on local sensing) and its global map. In order to give meaning to the notion of "best correspondence", we will need a *measure of similarity* which gives a numerical value to the quality of a given correspondence, so that we will be able to say that one match is better than another. More precisely, let $\alpha \in \Re^2 \times S^1$. Let A and B be geometric maps. Let $A(\alpha)$ denote A transformed by α, i.e., the action of α on all points in A. Our computational problem is: Find the α that minimizes the quantity $M(A(\alpha), B)$. We would like the measure we use to satisfy metric properties. In general, a metric, M, is a measure which satisfies the following criteria: (i) for all A and B, $M(A,B) \geq 0$. (ii) $M(A,B) = 0 \iff A = B$. (iii) (the *triangle inequality*) for all A, B, and C, $M(A,C) \leq M(A,B) + M(B,C)$. To explain this concept in words, we would like the measure to always be positive or zero, with zero occurring only when the measure is applied to two identical things. In addition, if A is near B and C is near B, then A and C are not too far apart. In our case, A, B, and C would be maps. Having a measure of similarity which is a metric provides several advantages which other measures might not. One is that because the metrics we use have a sound mathematical basis, we can be rigorous in analyzing the results of our experiments. The metric we use here was easily formulated as a simple modification to a previously published shape-matching metric, the hausdorff distance, designed for use with model-based recognition. This metric is described in [HK90]; our reasons for desiring a measure satisfying metric properties are similar to those developed more explicitly in [HK90, ACH+91].

Another property which we would like our measure of similarity to provide is some sense of *locality* which allows us to treat our measure as being continuous in all of its parameters (two translational, one rotational). More precisely, we want the measure of similarity of two poses which are spatially very close to be nearly equal. We also would like a fixed value indicating a perfect match. A metric provides such a value, namely zero. The reason why locality and continuity are important is that they transform localization into a minimization problem. The difference in the measure between two nearby locations provides a discrete gradient. By following the gradient, we can find the locally best correspondence between the partial map (built from local sensory data) and the global map. Gradient following is not part of the algorithm we describe in this section; it is, however, part of at least one modification to this algorithm which might be used in practice to enhance performance. The algorithm that we outline next is similar to [GMR91] in the sense that the result is a set of points where the robot might be; after explaining the basic algorithm, we will go on to show several ways in which this algorithm can be converted to a metric-oriented algorithm (one which utilizes metrics to determine the "best" pose, and which returns the best pose, together with a value representing how well the local map and the global map compare at that pose).

Let us represent a range reading as a vector. In order for a given position within a geometric map to be consistent with a particular range reading, two things must be true. The head of the range vector must not intersect any obstacles. This comes from the interpretatation of a range reading which says, "a reading of 57cm means that there is nothing *within* 57cm of the rangefinder in the direction which it currently points, but there is something *at* 57cm." Now: in order to find obstacles given a particular position of the robot, place the tail end of the vector where the robot is, and look for the head of the vector. In order to find possible positions of the robot given the positions of the obstacles, reverse the process: any place the tail of the vector can be positioned such that the head of the vector lies on an obstacle but no point of the body of the vector lies on an obstacle, represents a place where the robot can be and get that range reading. Part of this process is the process of inverting the vector and computing the Minkowski sum of the vector with the obstacles as described in [LP83]. Consider the vector to be a white line segment with a black point at one end. Anywhere the vector can be placed so that white matches white and black matches black is a possible location of the robot. Figure 3 shows several diagrams

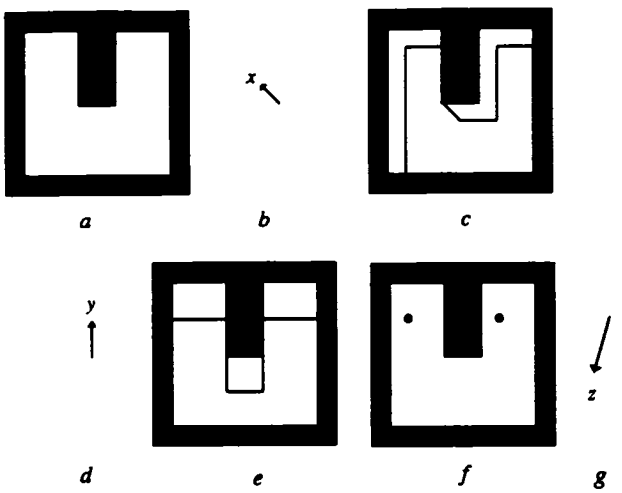

Figure 3: The Localization Algorithm at Work

which will be useful to understanding this algorithm. 3*a* shows an example of a geometric map. 3*b* shows a vector x which might represent a range reading. 3*c* shows the map with lines added which show all points where the robot could be and get the range reading shown in 3*b*. The new lines show the places where the vector x covers white space in the map, except for at its tip. 3*d* shows another possible range vector, while 3*e* shows the same information as 3*c*, but for the range vector y. 3*f* shows the original map with a set of singular points added. These points are the places within the map which are consistent with both the range reading shown in 3*b* and that shown in 3*d*. In this instance, two readings are sufficient to determine the robot's position to within one of two points; a third reading in direction z (fig. 3*g*) would enable the robot to locate itself completely unambiguously.

This describes the zero-uncertainty version of the algorithm; the whole algorithm becomes "take m readings, $r_1, \ldots, r_m$; for each reading r_i, determine its *feasible pose set*: the set of points P_i corresponding to the consistent poses in the map for that reading; intersect those sets of points: $P = \bigcap_{1 \le i \le m} P_i$. Any point in the intersection is a place where the robot could be, given that map and those readings." The algorithm can be described more precisely, in terms of the computational geometry tools and techniques used; however, this description is beyond the scope of this paper. We are currently working on a more formal treatment based on the configuration space algorithms of [LP83] and an output-sensitive plane sweep algorithm([CE92], modified per [Don89]). We deal with translational uncertainty by one of two mechanisms: the *consistent reading* method is to say, "if P is empty (there are no points in the intersection of the results of all range readings), what is the set of all points which are contained in a maximal number of P_i?" In other words, if there are 24 range readings taken (say one every 15 degrees, just for example), and no point in the map is in all 24 feasible pose sets, then look for the set of points contained in any 23 of the feasible poses sets, or any 22, and so on (this can be done without increased complexity, as explained in the following paragraph). This introduces a quality measure to the result: a point which is contained in all 24 point sets is a better match than one which is only contained in 23, or 22. This tactic enables the algorithm to be somewhat robust in the presence bad range readings. It also can handle the case where a movable obstacle in the robot's vicinity is sensed by one or more range readings, rather than a stationary obstacle behind the movable object which might be reflected in the robot's map. The *varying-epsilon* method for adding uncertainty to the algorithm is to use epsilon balls. Before, we described the vector as a white segment with a black point at one end. If we make the end a black ball of some radius, instead of a single point, and require that at least one point of that black ball lie on an obstacle, then we can treat the range reading as being, say, "57cm plus or minus 2cm". Once we do this, the P_i become sets of regions, rather than sets of points (since each point in a zero-uncertainty P_i expands to a region with the same size as the error ball); the intersection of two P_i is now typically a set of regions, rather than a set of points. We can now also measure the quality of a match by saying, "how large do we need to make epsilon in order to find a point which is in all of the feasible pose sets?" We can also combine approaches, by asking questions like, "How large do we need to make epsilon so that we can find a point which is in 90% of the feasible pose sets?"

In the translation only case, requiring a match on all range readings (disallowing the consistent reading method described above), a complexity of $O(m^3n^3)$ can be easily obtained, where m is the number of range readings taken and n is the complexity of the map. Using a plane sweep algorithm [PS85], this can be gotten down at least as far as $O(m^2n^2 \log(mn))$ [AST92]. For performance reasons, we prefer to use rasterized algorithms when we implement algorithms involving geometry on the robot. The rasterized version of this can be implemented in time $O(mn^2e^2l)$, where the rasterized environment is n by n, m readings are taken, we use an error ball of radius e, and the longest reading has length l. Adding the ability to consider nonperfect matches (points where fewer than m range readings match) does not increase the complexity of the algorithm: we have to spend $O(mn^2)$ time computing the intersection P of all of the P_i. The P_i are n by n bitmaps, with 1s in the i^{th} bitmap corresponding to points in P_i. If, instead of performing intersection by ANDing together these bitmaps, we add them together pixelwise, we need only look for locations in the resulting array which have the maximal summed value.

In practice, we could make the following improvements: given a bounded error estimate of the robot's current pose, consider only a local window of the map, thereby reducing the n^2 part of the complexity to a constant. Or, given a large number of readings, take a random sampling of some constant fraction of the readings; etc. These, however, are for the future, and are only mentioned here. A portion of this work which is not dealt with here is the problem of rotational uncertainty. Fortunately, RWI bases have good rotational odometry(i. e., orientation sensing), at least in our experience. [CHKK92] provides a description of work involving Hausdorff distance match-

ing and the rotational case; this work includes a description of a combinatorial algorithm for localization with rotation and translation, obtaining a time bound of $O(m^3n^3 \log(mn))$.

5 Conclusions

In this paper, we considered mobile robot navigation. We described the experimental platform on which we implemented our algorithms, including a laser range finder designed and built in our lab, and presented evidence to support the suitability of the ranger for a mobile robot performing navigation and navigation-related tasks. We also described several techniques for using range sensors to perform two tasks in robot navigation, namely map-making and localization, where we define localization as determination of the robot's pose within its environment.

We discussed a method for building a geometric map of the robot's environment. Next we explored some ways that the robot can maintain its map over time, even in dynamic environments. Finally, we discussed an algorithm, based in computational geometry and originally designed for use in model-based recognition, which can be used to compare a robot's current range data to its model of the world so as to get improved estimates of its current position within that world.

We intend to improve our implementations to make them more versatile in their treatment of the available range data. Over the longer term, it is our hope to improve the robustness of our robot, its sensors, and its algorithms to the point where the robot can perform navigate successfully in office environments which are at most minimally engineered.

6 Acknowledgements

We would like to thank Dan Huttenlocher and Jim Jennings for their assistance with the development of the ideas in this paper. Many of the key ideas presented here arose in discussions with them. They have been very generous with their time and ideas. We would also like to thank Craig Becker, Mark Battisti, and Kevin Newman for their valuable assistance in the design, construction, and maintenance phases of the development of our experimental apparatus, as well as Jonathon Rees for developing the Mobot Scheme system which played a big role in moving our lab from the building stage to the experimentation stage.

Thanks to Loretta Pompilio for drawing the illustration in figure 1.

References

[ACH+91] E. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell. An efficiently computable metric for comparing polygon shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**(3):209–216, 1991.

[AST92] P. K. Agarwal, M. Sharir, and S. Toledo. Applications of parametric searching in geometric optimization. In *Proceedings Third ACM-SIAM Symposium on Discrete Algorithms*, pages 72–82, 1992.

[CE92] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *JACM*, 39:1–54, 1992.

[CHKK92] L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. M Kleinberg. On robot localization, approximate placement, and the minimum hausdorff distance. Technical report, Cornell University Department of Computer Science, 1992.

[Don89] B. Donald. *Error Detection and Recovery in Robotics*, volume 336 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, 1989.

[Dru87] Drumheller. Mobile robot localization using sonar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-9**(2), 1987.

[Elf87] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, **RA-3**(3), 1987.

[GMR91] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem in two dimensions. In *Symposium on Discrete Algorithms*, pages 259–268, 1991.

[HK90] D. P. Huttenlocher and K. Kedem. Efficiently computing the hausdorff distance for point sets under translation. In *Proceedings of the Sixth ACM Symposium on Computational Geometry*, pages 340–349, 1990.

[LDWC90] J. Leonard, H. Durrant-Whyte, and I. J. Cox. Dynamic map building for an autonomous mobile robot. In *Proceedings of the 1990 IEEE/RSJ International Conference on Intelligent Robot Systems*, 1990.

[LP83] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C–32(2):108–120, 1983. Also MIT A.I. Memo 605, December 1982.

[LRDG90] J. Lengyel, M. Reichert, B. Donald, and D. Greenberg. Real-time robot motion planning using rasterizing computer graphics hardware. Department of Computer Science Technical Report TR 90-1122, Cornell University Department of Computer Science, May 1990.

[Mat90] M. J Mataric. A model for distributed mobile robot environment learning and navigation. Master's thesis, Massachusetts Institute of Technology, 1990.

[ME85] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 116–121, 1985.

[PS85] F. Preparata and M. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[RD92] J. A. Rees and B. R. Donald. Program mobile robots in scheme. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, 1992.