Real-Time Vision-Based Robot Localization

Greg Hager

Research Report YALEU/DCS/RR-868
September 1991

Real-Time Vision-Based Robot Localization

Sami Atiya
Fakultät für Informatik der Universität Karlsruhe (TH)
Institut für Algorithmen und Kognitive Systeme
and
Fraunhofer-Institut für Informations- und Datenverarbeitung
Fraunhoferstr.1
D-7500 Karlsruhe 1

Greg Hager
Department of Computer Science
Yale University
P.O. Box 2158 Yale Station
New Haven, CT 06520-2158

Abstract

This article describes an algorithm for determining robot location from visual landmarks. This algorithm determines both the correspondence between observed landmarks (in this case vertical edges in the environment) and a stored map, and computes the location of the robot using those correspondences. The primary advantages of this algorithm are its use of a single geometric tolerance to describe observation error, its ability to recognize ambiguous sets of correspondences, its ability to compute bounds on the error in localization, and fast execution. The current version of the algorithm has been implemented and tested on a mobile robot system. In several hundred trials the algorithm has never failed, and computes location accurate to within a centimeter in less than half a second.

1 Introduction

The determination of observer location from sensory data is a central problem in many robotics applications. This problem is particularly difficult for systems that must estimate their location in complex environments using imprecise, unreliable, and incomplete sensor data. In many cases, it is also important to compute an accurate indication of localization *error*. For example, mating, docking, navigating in a confined space or going through a small opening require precise relative location information. If an estimate of robot location is too far from the correct value, planning errors or costly collisions may result. Conversely, the accuracy of localization may depend on the velocity of the observer and therefore affect the time needed or the path used to travel between points in the environment.

A solution to the localization problem involves determining *what* is being observed and *where* it is observed from. Although localization systems employ a wide variety of sensors and sensor processing algorithms, nearly all can be broken down into the following basic components: a means for detecting features in sensor data which may correspond to structures in the environment (hereafter referred as *landmarks*) used for localization, a method for matching observed features with a map of known landmarks, and a method for computing location and localization error from the matches. Detection and matching, often the most computationally expensive and error prone components of the system, can be simplified by using active emitters or artificial patterns [18]. However, in many cases it is cumbersome and expensive to lay out a sufficient number of well-calibrated artificial targets. Instead, most mobile systems employ common features computed from visual, acoustic, or laser-based depth information. Consequently, one of the major difficulties in mobile robot localization is to quickly and reliably establish the correspondence between observed features and known landmarks.

There are three basic approaches to solving the matching problem reported in the literature. First, if the location of the observing system and the identity of the observed features is *a priori* unknown, then the matching problem becomes one of geometric constraint satisfaction. Sugihara [37] has studied the problem of determining robot position from features detected in a single camera image, and Krotkov [22] extends this work with an analysis of the effects of observation uncertainty. The crux of both methods is an efficient algorithm for exploiting geometric invariants to match observed features to known landmarks.

A second approach assumes prior knowledge of location and a model of robot motion can be used to *predict* the appearance and/or location of landmarks. Many authors, including [2; 9; 24; 35] describe systems based on this approach. Briefly, a running estimate of location computed by an extended Kalman filter [12] is used to predict the appearance of points, lines or surfaces (the landmarks) in the environment. These landmarks are projected into the image domain and surrounded by "validation regions" derived from the error covariance terms of the Kalman filter. An observed feature falling into a validation region is matched to the landmark used to generate the region. Data on the matched pairs is fed back into an extended Kalman filter to update the robot location. Methods for handling ambiguous and missing matches are also described.

Finally, a third approach assumes that the features themselves can be *tracked* over time in the image domain. In this case, features and landmarks need only be matched once at the very beginning and are available for computing location (or motion or scene structure depending on

the problem formulation) as long as the features stay in view [10; 20; 39].

Despite the quantity of work described above, there are still several problems that remain to be solved. The success of methods relying on using prior location estimates or tracking to compute location ultimately rely on an initial match of observed features to known landmarks. Later matching depends on maintaining good localization information and accurate error estimates. Thus, the problem of matching using no prior location information is crucial for initialization, for independently verifying the consistency of matches over time, and for reinitializing the system if localization information degrades or fails altogether.

None of the matching methods using no prior information reported in the literature provide a completely satisfactory treatment of observation error. The work reported by Sugihara [37] assumes that observations are error-free. Krotkov [21] suggests means for making the algorithm less sensitive to observation error, but fails to provide a complete analysis or proof of correctness. Neither author adequately addresses the issues involved with errors in the locations of the landmarks in the map. They also do not analyze the effects of observing *false positives*—detecting features that do not correspond to a known landmark, although this is very common in mobile robot applications.

Location estimates are usually computed using optimization techniques, *e.g.* least squares, or using statistical considerations, *e.g.* the extended Kalman filter. In both cases, a quantitative estimate of localization error relies on making distributional assumptions about sensing errors. Furthermore, the extended Kalman filter employs a linearization. If the location estimate is guaranteed to remain close to the correct value, the linearization is usually well-behaved. However, in circumstances where this is not true the extended Kalman filter may exhibit poor convergence and unreliable error estimates [17]. Consequently, reliable operation must be verified through simulation or extensive testing, and filter parameters can require time-consuming empirical tuning.

This article presents an alternative formulation of the localization problem in which the error in sensor observations is represented by a tolerance. Both the matching problem, with or without prior location information, and the location estimation problem are formulated using this error model. This setting of the problem does not require strong distributional assumptions about sensing errors or a particular structural form (e.g. linear) for the observing system. Bounded errors lead to a natural representation of location as the complete set of locations consistent with observed data. This set provides a quantitative description of localization error.

This problem formulation leads to a set-based algorithm for solving the matching problem and computing the location of a mobile robot in typical indoor environments. The primary features of this algorithm are:

- It runs very quickly—the current version requires less than 0.5 seconds on a Sun-4/280[1] to identify up to 5 landmarks and compute robot location and bounds on localization error.
- It treats errors using tolerances, thereby avoiding the difficult issues surrounding the distributional modeling required to employ statistical techniques.

[1]Sun is a trademark of Sun Inc.

- It computes both robot location *and* a geometric, worse-case accuracy that is derived from a *single* empirically determined observation tolerance.
- It determines when matching ambiguities arise that make it impossible to solve the problem uniquely.

These algorithms have been integrated into a working, vision-based, mobile robot navigation system that has been tested in many hundred simulated and real trials.

The remainder of this article is organized as follows. The next section introduces the basic concepts of set-based estimation methods and reviews previous work in this area. In Section 3 the localization problem is formulated using set-based concepts. Section 4 describes a solution to the stated problem and presents an analysis of its complexity as well as its limitations. Section 5 describes a mobile robot system and presents the results of numerous simulations and experimental trials. Section 6 describes several extensions to the basic algorithm that permit the use of odometric information, tolerate uncertainty in landmark location, and increase the speed of matching using heuristics. The final section discusses these results and describes a set of problems to be addressed in future research.

Notation and Terminology: The remainder of this article uses the following notational conventions. The closed interval from a to b, $a \leq b$ is written $[a, b]$. If a and b are vectors in $\Re^n$, $a_i \leq b_i,\ i = 1 \ldots n$ then $[a, b] = [a_1, b_1] \times \ldots \times [a_n, b_n]$ is a closed "box" in $\Re^n$. Point-valued and interval-valued variables are distinguished by writing the latter in bold-face type. For example, if $x \in \Re^n$, it is possible to write $x \in \boldsymbol{x} = [\underline{x}, \overline{x}]$, indicating that a real vector x falls within some real interval vector $\boldsymbol{x}$ with lower vector $\underline{x}$ and upper vector $\overline{x}$. A point value, x, occurring in expressions with interval values should be thought of as the degenerate interval $\boldsymbol{x} = [x, x]$. Two special interval operators are the width function $w(\boldsymbol{x}) = \overline{x} - \underline{x}$; and the center function $c(\boldsymbol{x}) = (\overline{x} + \underline{x})/2$. Finally, the evaluation of a function $h : \Re^n \rightarrow \Re^m$, on interval arguments is defined as

$$h(\boldsymbol{x}) = \{y \mid y = h(x), x \in \boldsymbol{x}\}.$$

A methodology for developing explicit expressions for functions evaluated on interval arguments may be found in [1; 28].

2 The Manipulation of Set-Based Uncertainties

Set-based approaches to estimation assume that all information about an unknown quantity is represented by a set of values consistent with observation and other internal system constraints. Though not as actively pursued as its statistical counterpart, set-based methods have been continually developed and applied over the past three decades. The originating works in set-based estimation discuss the problems of state estimation and prediction of linear systems [6; 34; 38]. More recent work extending the set-based paradigm to a wider variety of estimation problems includes [4; 27; 33] and our own work [14; 15; 16].

A central concept of set-based theories is the combination of information via intersection. With reference to Figure 1, if both set A and set B are known to contain the true value of some

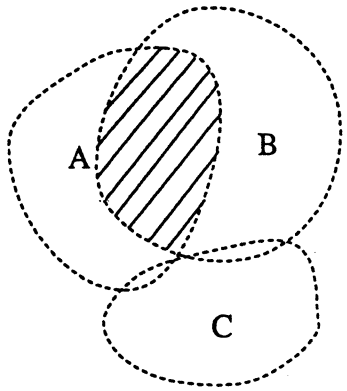

Figure 1: Combining information via intersection.

parameter, then $A \cap B$ must also contain that value. As more and more independent constraints are added to the problem (for example, by continued observation), it is reasonable to expect the intersection of the corresponding sets to grow smaller and smaller, providing a tighter and tighter bound on the unknown quantity. Returning to the diagram, if C is also supposed to contain the correct parameters, then, as it is disjoint from $A \cap B$, at least one of A, B, or C must be in error. This indicates some structural assumption used to construct A, B, or C has been violated. Uncertainty sets can be transformed by projecting every point in the uncertainty set. For example, suppose it is known that $d = f(a) + b$. Then the correct transformation of sets A and B is $D = \{d \mid d = f(a) + b, \quad a \in A, b \in B\}$.

Three set representations discussed in the literature are: convex hulls described by support functions, polygonal regions, and elliptical regions. Support functions [38] have the advantage of mathematical generality and elegance, but do not generally lead to computationally tractable algorithms. Polygonal representations take advantage of the fact that the linear transformation of a convex polytope, and the intersection of two convex polytopes again results in a convex polytope [27]. However, the number of vertices describing the intersection of two polytopes is often larger than the number of vertices of either of the operands. Hence, repeated intersection can lead to an excessive computational burden. Elliptical regions are also closed under linear transformations. Moreover, the bounding ellipse for the intersection of two ellipses both be calculated algebraically in fixed time. Consequently, a theory of elliptical sets applied to linear measurement and dynamic systems can be thought of as a set-based analog to the classical Kalman filter [34].

Both elliptical and polyhedral representations share many of the same advantages and disadvantages of the Kalman filter. Updated estimates and their errors can be computed and manipulated easily and efficiently as long as all transformations involved are affine. However, if any of the transformations involved are nonlinear, then the computed bounding set is no longer guaranteed to contain the true solution. Consequently, as with the Kalman filter, applications involving nonlinear transformation of estimates may require extensive empirical tuning and verification to ensure consistent, correct performance of these set-based methods.

The problems discussed in this paper lead naturally to polygonal uncertainty sets. However, these sets are approximated by rectangles (in two dimensions) or boxes (in three dimensions)

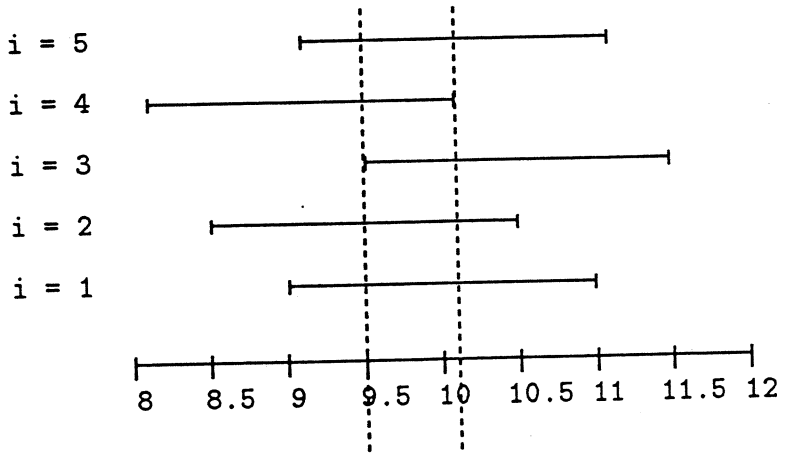

Figure 2: The evolution of set intersections over time.

when information is combined. This interval-based representation requires a fixed number of minimum and maximum operations to compute the intersection, and general nonlinear transformations of uncertainty sets can be computed using interval arithmetic [28]. One of the results of this article is to show that, by careful consideration of all of the problem constraints, it is possible to compute effective bounding intervals for robot location in spite of the nonlinearities involved in the problem.

2.1 The Convergence of Set-Based Methods

In principle, set-based methods make no assumptions about the statistical nature of errors. However, in practice, most observation errors *are* time-varying to some degree. In such cases, repeated intersection over time leads to increasingly tighter bounds on the unknown quantity. For example, by straightforward manipulation the system $z_i = x + v_i$, $v_i \in [-1, 1]$, yields the constraint $x \in [z_i - 1, z_i + 1]$ for each i. Given the sequence of observations $10, 9.5, 10.5, 9.1, 10.1$, the final value *must* fall in the interval

$$[\max(9, 8.5, 9.5, 8.1, 9.1), \min(11, 10.5, 11.5, 10.1, 11.1)] = [9.5, 10.1].$$

Figure 2 illustrates the sequence of intervals, and their intersection. Assuming v varies within its uncertainty interval, the intersection will never be empty and will become continually smaller. More generally, the following properties hold:

Let $X_1, X_2, \ldots X_n$ be a sequence of independent random variables such that

- $X_i \in [\theta - a_i, \theta + b_i]$, and
- for any $\delta > 0$, there is an $\epsilon > 0$ such that $P(X_i < \theta - a_i + \delta) \geq \epsilon$ and $P(X_i > \theta + b_i - \delta) \geq \epsilon$ (the distribution places mass around the endpoints of the interval).

Let $S_n = \bigcap_{i=1}^{n} [X_i - b_i, X_i + a_i]$. Then

1. For all $n > 0$, $\theta \in S_n$.
2. For any $c > 0$, $\lim P(w(S_n) > c) \rightarrow 0$ as $n \rightarrow \infty$. (the sequence of sets converges in probability [7]).

Proof

Since $X_i \in [\theta - a_i, \theta + b_i]$, by algebraic manipulation $X_i - b_i \leq \theta \leq X_i + a_i$. Let $l = \max(X_1 - b_1, X_2 - b_2, \ldots, X_n - b_n)$ and $u = \min(X_1 + a_1, X_2 + a_2, \ldots, X_n + a_n)]$. By definition, it follows that $S_n = [l, u]$, $l \leq \theta \leq u$, and hence $\theta \in S_n$.

Next, note that $w(S_n) = u - l$. Consequently,

$$P(w(S_n) > c) = \prod_{i=1}^{n} P(\theta - a_i + c \leq X_i \leq \theta + b_i - c) \leq \prod_{i=1}^{n} (1 - 2\epsilon) = (1 - 2\epsilon)^n$$

for some $0 < \epsilon \leq 0.5$. Hence $0 \leq 1 - 2\epsilon < 1$, and therefore $\lim(1 - 2\epsilon)^n \rightarrow 0$ as $n \rightarrow \infty$.

From the final expression, it is also clear that the *rate* of convergence towards small intervals depends on the likelihood of the observed quantities taking values near the endpoints of their associated tolerance interval. Convergence is slowest for random variables with a large concentration of probability mass near θ, and swift for those that often take values near the endpoints of their set of support.

An immediate consequence of the above is the following:

Let $X_1, X_2, \ldots X_n$ be a sequence of random variables as above and let f be a continuous function defined on $\bigcup_{i=1}^{n} [\theta - a_i - b_i, \theta + a_i + b_i]$. Then

1. $f(\theta) \in S_n^f = \bigcap_{i=1}^{n} f([X_i - b_i, X_i + a_i])$.
2. For any $c > 0$, $\lim P(w(S_n^f) > c) \rightarrow 0$ as $n \rightarrow \infty$.
3. $\theta \in \{a \mid f(a) \in S_n^f\}$.

In other words, convergence is preserved under continuous transformations. Furthermore, if f^{-1} is also a continuous function then the uncertainty interval on θ is a closed interval.

The following facts make a connection to optimal estimation techniques:

Let θ be a uniformly distributed random variable on the interval $[a, b]$, and let $X_1, X_2, \ldots, X_n$ be a sequence of independent random variables each having a conditional distribution $f(X_i \mid \theta)$ that is uniform on the interval $[\theta - a_i, \theta + b_i]$, $i = 1, 2, \ldots, n$. Define $l = \max(a, X_1 - b_1, X_2 - b_2, \ldots, X_n - b_n)$, and $u = \min(b, X_1 + a_1, X_2 + a_2, \ldots, X_n + a_n)$. Then

1. $f(\theta \mid X_1, \ldots, X_n)$ is also uniform on the interval $[l, u]$.
2. The minimum mean square estimator of θ is the conditional mean

$$E[\theta \mid X_1, X_2, \ldots, X_n] = (l + u)/2.$$

In short, combining information via intersection corresponds to computing Bayes' Theorem for *uniform* prior and conditional (sampling) distributions. Choosing the center of the interval remaining after combining information via intersection corresponds to the minimum mean square estimate under the same assumptions.

Though all of the preceding analysis has been done for scalar quantities, the results extend to vectors of random variables with only minor modifications. Hence, in many ways set-based methods provide a simple, viable form of robust estimation. They can be used to estimate unknown quantities to a high degree of accuracy using very weak statistical assumptions and no distributional information about the behavior of observation errors. Moreover, many applications are naturally described in terms of uncertainty sets. For example, a robot moving to a prearranged point for loading requires that the set of possible robot locations falls within the tolerances on the loading apparatus. If this requirement is fulfilled, then no docking error is possible, otherwise additional sensing must be done to reduce the localization error of the robot.

3 Localization Problem Formulation

Following Krotkov [22] and Sugihara [37], let $p_1, p_2, \ldots p_n$ denote the positions of fixed landmarks or beacons expressed in a fixed two-dimensional world coordinate system W. Let $\Gamma = (x, y, \theta)$ parameterize the location of a local robot coordinate system R with respect to W. It is assumed that multiple images of landmarks are available, and the image-image correspondence can be solved. Let $o = (o^l, o^r)$ denote the horizontal position of a vertical edge in two camera images which are labeled "left" and "right", and let $o_1, o_2, \ldots, o_m$ denote a series of these observations. The imaging of points by the camera in a coordinate system C is described using a linear spatial transformation ${}^{C}T_{W} = {}^{C}T_{R}{}^{R}T_{W}$ followed by a nonlinear imaging transformation $I(\cdot)$ which computes both the left and right image locations. Unless otherwise noted, all positional quantities (including robot location) will be expressed in millimeters and all angles will be expressed in degrees.

A *correspondence* between an observation o_i and a world point p_j is a tuple $\lambda = \langle o_i, p_j \rangle$, and a *labeling* Λ is a set of such correspondences. A labeling is *consistent* if each observation is in correspondence with no more than one feature in the world coordinate frame.

In the ideal case where the camera model, the world map, and sensor observation of landmarks are all error free, the localization problem can be precisely stated as:

> **Problem 0:** Given n points $p_1, p_2, \ldots, p_n$ in a world coordinate system and m observations $o_1, o_2, \ldots, o_m$ taken at two camera positions with known relative relationship, determine if there is a unique, consistent labeling Λ and fixed pose Γ such that $o_i = I({}^{C}T_{W}(\Gamma)p_j)$ for all $\langle o_i, p_j \rangle \in \Lambda$.

In practice, errors in edge localization and landmark position must be accommodated and observations that have no corresponding landmark in the map must be culled. To accommodate the former, an observation *tolerance*, $\epsilon \in \Re^2$, and landmark position tolerances $\delta_i \in \Re^2$, $i = 1, 2, \ldots, n$ are introduced. The interpretation of ϵ is a maximal error in sensor measurement, and likewise δ_i is a maximal error in the absolute position of landmark i. Define $\boldsymbol{p}_i = [p_i - \delta_i, p_i + \delta_i]$, and $\boldsymbol{o}_i = [o_i - \epsilon, o_i + \epsilon]$.

In the nonideal case considered here, the localization problem is divided into two separate subproblems:

> **Problem 1:** Given n point location intervals $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n$ in a world coordinate system and m observation intervals $\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_m$ taken at two camera positions with known relative relationship, determine if there is a unique, consistent labeling Λ so that for *some* fixed location Γ,
>
> $$I({}^{C}T_{W}(\Gamma)\boldsymbol{p}_j) \cap \boldsymbol{o}_i \neq \emptyset$$
>
> for all $\langle o_i, p_j \rangle \in \Lambda$.

> **Problem 2:** Given a labeling Λ as described above, determine the complete set of robot positions, $\mathcal{P}$, consistent with Λ—that is, compute the set
>
> $$\mathcal{P} = \{\Gamma \mid I({}^{C}T_{W}(\Gamma)\boldsymbol{p}_j) \cap \boldsymbol{o}_i \neq \emptyset \text{ for all } \langle o_i, p_j \rangle \in \Lambda\}.$$

The first problem is a qualitative determination of *what* is being viewed and is hereafter referred to as the *labeling* problem. The second problem is the quantitative determination of *where* the viewer is located, and is hereafter referred to as the *location estimation* problem.

4 A Set-Based Solution

The approach used to solve the labeling problem is to transform both observed data and stored map points into a representation that is invariant to translation and rotation, and thereby permits direct comparison of observed and stored landmarks. Once the correct point correspondences are known, the location estimation problem can be solved directly. The original motivation for this approach came from [32] where the labeling of star fields was done from sighting data, but similar ideas appear in many other sources [8; 13; 19; 25].

In overview, note that any three non-colinear points in the plane determine a triangle with three angles α, β, γ and three sides L, R, B of length l, r, b, respectively (see Figure 3). These six values are translation and rotation invariant, and therefore independent of the coordinate frame in which the points are expressed. Hence, these quantities can be directly compared to determine if two triplets of points lie in the same geometric configuration relative to one another. Furthermore, by computing the range of lengths and angles resulting from observation error tolerances this comparison can be made tolerant to observation errors.

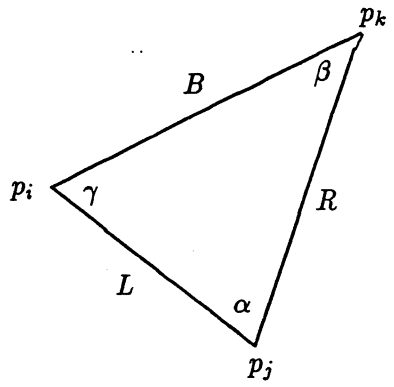

Figure 3: Triangle labeling conventions.

Thus, an algorithm for determining the solution to Problem 1 is to store a list of angles and distances between the points in a map of the environment. At runtime the same quantities are computed for every combination of three stereo data pairs and the results, now encoded as *intervals* on angles and distances, are compared with the network of known landmarks. From this comparison, it is possible to determine all consistent pairings of point triples, henceforth referred to as *matches*. The matches determine a consistent labeling that is then used to compute a set of locations known to contain the robot.

The matching process will now be described in complete detail. This solution assumes that landmark positions (though not observations) are error-free ($\delta_i = 0$). The modifications required to handle landmark position errors are presented in Section 6.

4.1 Computing Correspondences

Imaging Model and Calibration Camera imaging is modeled in four stages following [23]:

1. A geometric transformation $e = {}^{c}T_{W}p$ depending on six parameters describing the spatial transform ${}^{c}T_{W}$,
2. the perspective transformation of the point $e = (x_c, y_c, z_c)$ into undistorted image coordinates
$$\begin{bmatrix} u_u \\ v_u \end{bmatrix} = \frac{f}{z_c} \begin{bmatrix} x_c - b \\ y_c \end{bmatrix} \tag{1}$$
depending on the focal length f and a baseline parameter b,
3. A mapping of (u_u, v_u) into the radially distorted point (u_d, v_d) depending on the distortion coefficient κ:
$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = \frac{2}{1 + \sqrt{1 - 4\kappa R^2}} \begin{bmatrix} u_u \\ v_u \end{bmatrix}; \quad R^2 = u_u^2 + v_u^2, \tag{2}$$

4. A conversion from the image coordinates (u_d, v_d) to pixel coordinates (u, v) depending on image center (c_u, c_v) and scale factors (s_u, s_v):

$$u = u_d/s_u + c_u; \qquad v = v_d/s_v + c_v. \tag{3}$$

The transform ${}^{C}T_{R}$ describes the camera at the origin. Step 2 states that the camera translation is parallel to the imaging plane and directed along the x axis, so motions only require adjustment by a known offset b. Significant departures from parallel may require incorporating rotations into this expression. The focal length of the camera is taken to be that given by the manufacturer and the remaining 11 calibration parameters are determined by observing a known constellation of points and performing nonlinear least squares regression.

Stereo-Based Position Determination Imaging geometry is reduced to 2 dimensions by assuming that vertical edges are imaged at a fixed height $y_c = 0$ corresponding to the scan line $v = c_v$.

Let $o^r = (u, v)^r$ be the image of a vertical edge at the origin position of the camera and $o^l = (u, v)^l$ be the image of the same vertical edge at a distance b from the origin. Expression (3) can be inverted to get distorted image coordinates (u_d, v_d). Then, the distortion-corrected image coordinates $(u_u, v_u)^r$ and $(u_u, u_u)^l$ are computed by inverting (2) giving

$$\begin{bmatrix} u_u \\ v_u \end{bmatrix} = \frac{1}{(1 + \kappa R_d^2)} \begin{bmatrix} u_d \\ v_d \end{bmatrix}; \quad R_d^2 = u_d^2 + v_d^2. \tag{4}$$

Using (1) the (planar) location $e = (z_c, x_c)$ of an observed point in the camera coordinate system can be computed as:

$$z_c = \frac{bf}{u_u^r - u_u^l}, \quad x_c = \frac{z_c u_u^r}{f}. \tag{5}$$

Under the assumption that image distortion is locally constant, perturbing u_d^r and u_d^l in the above expression yields a diamond-like area defined by the four points computed from all combinations of $u_d^r \pm \epsilon$ and $u_d^l \pm \epsilon$ [26; 36]. The region enclosed by this polytope, henceforth referred to as a *stereo region*, contains all possible locations for the observed point up to sensing error.

Transformation To Triangles For any three given points, p_i, p_j, and p_k, define $L = p_i - p_j$, $R = p_k - p_j$, and $B = p_k - p_i$. The six vector of lengths and angles describing this triangle are given by:

$$S_{i,j,k} = \begin{bmatrix} l \\ r \\ b \\ \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} \|L\| \\ \|R\| \\ \|B\| \\ \cos^{-1}\left[\frac{R \cdot L}{\|R\|\|L\|}\right] \\ \cos^{-1}\left[\frac{B \cdot R}{\|B\|\|R\|}\right] \\ \cos^{-1}\left[\frac{L \cdot B}{\|L\|\|B\|}\right] \end{bmatrix}.$$

In the ideal case, this is a redundant description as a triangle is determined by any combination of three values including at least one length. Due to observation errors, there is a family of triangles consistent with any given triplet of stereo regions. This family is represented by determining intervals on angle and length as follows.

Given two regions a and b, the maximum possible distance between points in a and points in b occurs on the vertices defining the regions. Hence, determining this value is straightforward and requires $4 * 4$ operations. The minimum possible distance also occurs at vertices *except* when a perpendicular to a segment of one region can be made to pass through a vertex of the other region. To solve for minimum distance let b_k be a vertex of region b and a_i and a_j, $j \neq i$ be two adjacent vertices of region a. The minimum distance between the point b_k and the line $a_i + \lambda(a_j - a_i)$ is given by:

$$\lambda = \frac{(b_k - a_i) \cdot (a_j - a_i)}{\|a_j - a_i\|^2}.$$

If λ falls between 0 and 1, the minimum distance corresponding to that value of λ is used, otherwise the minimum distance is given by $\|b_k - a_i\|$ if $\lambda < 0$ and $\|b_k - a_j\|$ otherwise. The above expression is computed for all combinations of vertices taken from a and segments taken b and vice versa ($4 * 4 * 2$ operations), and the minimum of those values is the minimum distance sought.

Given three stereo regions a, b and c, the minimum angle between $a-b$ and $c-b$ occurs at the extreme points of all three regions,[2] while the maximum angle sometimes occurs at the extreme points and sometimes occurs by choosing two vertices of a and c and a point along a segment forming the region b. Let a_i and c_j be vertices of regions a and c respectively, and b_k and b_l be two adjacent vertices of region b. Define $s_1 = a_i - (b_k + \lambda(b_l - b_k))$ and $s_2 = c_j - (b_k + \lambda(b_l - b_k))$. Then, the latter maximization problem is

$$\max_{\lambda} \cos^{-1}\left[\frac{s_1 \cdot s_2}{\|s_1\|\|s_2\|}\right] \equiv \min_{\lambda} \frac{s_1 \cdot s_2}{\|s_1\|\|s_2\|},$$

the above equivalence holding for interior angles.

Appendix A derives the closed-form solution. Because of the complexity of this solution a faster approximation is used to compute the maximal angle. For regions a, b, and c with the angle situated at b, the angle between the vertices of a and c for each endpoint and *midpoint* of the segments comprising b is computed ($4*4*8$ evaluations). The maximum and minimum of these values comprise the upper and lower bounds on angle, respectively.

By carrying out these computations for stereo regions a, b, and c, a closed interval $\mathbf{S}_{a,b,c}$ consisting of six components can be computed. Three points p_i, p_j, and p_k are consistent with regions a, b and c if $S_{i,j,k} \in \mathbf{S}_{a,b,c}$. (Recall boldface type is used to distinguish between point values and interval quantities as in the above expression.)

Searching For Matches Each interval of lengths and angles computed from observed data will be consistent with some collection of triangles computed from map points. The crucial point

[2]The only exception is when a single line passes through all three regions, a case that is easy to check for.

of designing a good algorithm is to make the search for these matches as fast as possible.

The approach taken here is to construct six lists of ordered pairs from six vectors describing triangles of landmarks in the map. The first element of an ordered pair is an angle or length and the second member is a pointer to the six vector that the angle or length came from. The lists are sorted on the value component. The sorted list can be searched quickly to isolate a range of values contained within an angle or length interval, and the pointers are used to mark the associated six vectors. Only vectors that are marked six times are consistent with an interval six vector of angles and lengths.

In detail, an algorithm for finding matches works as follows:

Initialization:

- For each unique grouping[3] of map points, compute (the point) $S_{i,j,k}$ and add it to a list M. Call the final length of this list q.
- Let $M_{j,i}$ denote the ith element of the jth vector in M. For each element $M_{j,i}$, $j = 1, \ldots, q$ and i, $i = 1 \ldots 6$, add the pair $\langle M_{j,i}, j \rangle$ to a list L^i.
- Sort the elements of each L^i on the first (value) component yielding six lists of pairs of indices and values sorted on value.

Runtime: For each triplet of observed stereo pairs o_a, o_b, o_c,

- Compute (the interval) $\mathrm{S}_{a,b,c} = [l, u]$ encoding the permissible range of the six triangle parameters for the given tolerance ϵ.
- For each coordinate $i = 1, \ldots, 6$
  - Find the first index r such that the value field of L^i_{r-1} is smaller than l_i.
  - Find the first index s such that the value field of L^i_{s+1} is larger than u_i.
  - For each $\langle v_j, k_j \rangle = L^i_j$, $r \leq j \leq s$, mark M_{k_j} as found.
- For each $S_{i,j,k} \in M$ that has been marked six times, add $(\langle o_a, p_i \rangle, \langle o_b, p_j \rangle, \langle o_c, p_k \rangle)$ to Λ.

To illustrate, suppose that the map consists of four landmarks. These landmarks are represented by a list, M, consisting of four triangles:

	j \ i	1	2	3	4	5	6	# found
$S_{1,2,3}$	1	55	60	65	100	107	95	1
$S_{1,2,4}$	2	40	60	80	100	153	134	1
$S_{1,3,4}$	3	60	70	50	95	84	134	2
$S_{2,3,4}$	4	30	90	60	107	84	153	3

[3]Triangles that are merely a permutation of map points indices p_i, p_j, p_k corresponding to a relabeling of the triangle axes are redundant.

A triple of observed points characterized by an interval vector with $\alpha = [l_1, u_1] = [25, 46]$, $\beta = [l_2, u_2] = [65, 91]$ and $\gamma = [l_3, u_3] = [50, 65]$ leads to lists L^1, L^2 and L^3 with values of r and s as noted:

	α	index
r	30	4
s	40	2
	55	1
	60	3

	β	index
	60	1
	60	2
r	70	3
s	90	4

	γ	index
r	50	3
	60	4
s	65	1
	80	2

The lists for the remaining 3 components are similar and are not shown.

Marking the referenced entries between r and s from each table leads to the counts noted in the final column of the list M. To this point only $S_{2,3,4}$ can be consistent with the observed data as it is marked three times. The process continues for the remaining three components, and only those members of M marked 6 times are chosen.

4.2 Choosing A Labeling

The set of matched pairs computed by the previously described algorithm can be sorted into c maximal consistent categories $\Lambda_1, \Lambda_2 \ldots \Lambda_c \subseteq \Lambda$. A category Λ_i is called a labeling *of size* n if Λ_i places n observations in correspondence with n landmarks. Ideally, $c = 1$ and there is no labeling ambiguity. In practice, ambiguities do arise. For example, the following is a consistent partitioning of matches:

Λ_1	Λ_2	Λ_3
$o_1, o_2, o_3 \Longleftrightarrow p_1, p_2, p_3$	$o_1, o_2, o_3 \Longleftrightarrow p_1, p_2, p_4$	$o_1, o_2, o_3 \Longleftrightarrow p_1, p_2, p_5$
$o_1, o_2, o_4 \Longleftrightarrow p_1, p_2, p_4$		$o_1, o_2, o_4 \Longleftrightarrow p_1, p_2, p_4$
$o_1, o_3, o_4 \Longleftrightarrow p_1, p_3, p_4$		
$o_2, o_3, o_4 \Longleftrightarrow p_2, p_3, p_4$		

Because of correspondence conflicts, the matches in Λ_2 and in Λ_3 are not consistent with each other or with those in Λ_1. Λ_2 is of size 3 while both Λ_1 and Λ_3 are of size 4. Only Λ_1 contains the correct number of matches for a labeling of size 4 (4 matching triangles). Does this mean Λ_1 is the correct match? The answer depends on a number of factors.

There are two types of matching failure to consider: *true* failure—returning an incorrect labeling, and *ambiguity* failure—returning no response because there is no unique labeling. The latter is not a catastrophic occurrence—in fact it is common and unavoidable. The former is cause for serious concern.

True failure *cannot* happen under the following assumptions:

1. When an observed landmark can be matched to a map landmark the algorithm does so (the algorithm is correct and complete).

2. A labeling is accepted only when there is no other consistent labeling.

3. Any image of the world contains *at least* three landmarks.

4. The sensor always detects landmarks when they are present.

To verify this claim, suppose that at least one non-landmark point is observed, and that an error results. Then there are n observed points that result in a unique labeling. At least 3 of these points must be correctly matched. Hence, any point that is incorrectly matched must lie within a stereo diamond of a true landmark. But, since all true landmarks are observed, these landmarks must also be present. But these landmarks combined with the correctly matched points form a competing consistent labeling. This is a contradiction, so the assumption must have been false, and no labeling errors are possible.

In practice, assumptions 3 and 4 are nearly impossible to guarantee, and so a true failure can occur. In addition, if there are consistent labelings, but one possibility establishes more correspondences, then that labeling has a high likelihood of being correct. Hence, it is not unreasonable to violate assumption 2. By defining the likelihood of the sensor detecting false positives (landmarks that are not there) and false negatives (missing landmarks that are there) it is possible to determine the probability of failure for a given map, and a given robot location. Unfortunately, this probability can only be computed by simulation or by Monte-Carlo methods, and may vary widely for different maps, and different robot positions. However, several general rules can be stated:

1. The larger the number of correspondences in a labeling, the higher the reliability of that labeling: larger labelings place more constraints on the individual correspondences, and thereby reduce the likelihood that some spurious observations were able to satisfy all constraints.

2. The fewer observations matched by competing consistent labelings, the higher the match reliability: for example, suppose labeling a establishes 5 correspondences, and a competing labeling, b, establishes 3. Then a must contain at least two false positives to be in error. If b matched 4 observations, then a need contain only one false positive to be incorrect.

3. The more landmarks in the map, the lower the labeling reliability: more landmarks in the map lead to more possibilities for a spurious measurement to be consistent with a landmark.

4. The larger the uncertainty intervals on angles and lengths, the lower the labeling reliability: larger uncertainty intervals provide less stringent matching conditions, and raise the likelihood of a chance match between a false positive and a landmark.

Hence, an algorithm designer can determine the reliability of labelings by choosing two parameters: the minimum size of an acceptable labeling and the minimal size difference between the two largest labelings. For example, the vision sensing used in the robot system described in Section 5 almost never misses a detectable landmark though false positives occur quite often. Requiring labelings of at least four correspondences, and accepting any labeling that is larger

than all competing labelings (a size difference of only 1) has proved sufficient for very reliable operation.

It is possible that the stored map itself is ambiguous—that is, it contains constellations of landmarks that are indistinguishable (or nearly so) from one another. Such redundancies lead to ambiguous labelings and slightly increase the probability of true failure. The obvious solution is to inspect the list of triangle representations generated from the map, and to enforce a minimal level of separation among the entries.

In summary, by careful choice of landmarks, good engineering of the detection algorithm, and simulation analysis of the labeling algorithm, it is possible to reduce the probability of true failure to very low levels. Further means of ensuring correct labeling by using temporal continuity are presented in Section 6.

4.3 Determining Robot Position

Given a labeling, a point estimate of location can be computed using least squares. The solution can be derived in closed form, can be computed quickly, and usually delivers good estimates. However, least squares does not give a concrete error determination relative to the original observation error tolerance. This section discusses computing bounds on localization error by approximating the solution set $\mathcal{P}$ (as defined in Problem 2 of Section 2) by an enclosing interval on Γ.

Given two observed points e_i and e_j corresponding to p_i and p_j in the world coordinate system, it is possible to compute robot position by: determining the rotation that makes segments between p_i and p_j and e_i and e_j parallel, and then determining the translation that causes the endpoints of the rotated segments to overlap. By examining the extreme values of the above quantities on the stereo regions, it is possible to calculate the extreme values of computed position.

The maximal and minimal angles consistent with stereo regions a and b occur when a segment of length $d = \|p_i - p_j\|$ is placed such that one endpoint falls on a vertex a_i of region a and the other falls on a segment $(b_k - b_j)$ of region b or the dual case. This intersection is determined by solving the equation $\|b_j + \lambda(b_k - b_j) - a_i\| = d$ for λ and keeping those λ such that $\lambda \in [0, 1]$. Defining $r = p_j - p_i$, $t = b_j - a_i$, and $s = b_k - b_j$, for all consistent solutions, the angle of the observed line, the line in the world system, and their difference are computed as

$$\theta_o = \operatorname{atan}\left(\frac{t_y + \lambda s_y}{t_x + \lambda s_x}\right), \quad \theta_d = \operatorname{atan}\left(\frac{r_y}{r_x}\right), \quad \theta = \theta_o - \theta_d. \tag{6}$$

For each pair of matched stereo regions the algorithm computes up to 32 $(2 * 4 * 4)$ values of θ, and then forms the minimal interval $\boldsymbol{\theta}$ containing all 32 values. The intersection of the computed intervals over all matched pairs yields $\boldsymbol{\theta}^*$.

Given a stereo region a_i, it is straightforward to compute a bounding interval with components $\boldsymbol{s}_i$ and $\boldsymbol{t}_i$ for that region. Then by using interval arithmetic [1; 28], the intervals on robot

translation can be computed as:

$$\begin{aligned} x_i &= p_{i,x} + \sin(\theta^*)t_i - \cos(\theta^*)s_i, \\ y_i &= p_{i,y} - \sin(\theta^*)s_i - \cos(\theta^*)t_i \end{aligned} \tag{7}$$

where $p_i = (p_{i,x}, p_{i,y})$ is the match for region a_i. Again, the intersection of these sets computed for all regions yields x^* and y^*.

Based on the discussion in Section 2, if the observation errors are stochastic, then combination of multiple samples of the same scene may further reduce the size of these intervals. In the absence of approximation errors, and assuming independence of observations, the size of these intervals will tend toward zero in the limit. This point is further discussed in Section 5.

4.4 Analysis of Solution Complexity

As before, let n be the number of map points, and m be the number of observed points. The offline portion of the algorithm consumes $O(n^3 \log(n))$ time to compute and sort all triangle parameters. However, this is only done once and stored as a compiled table of size $O(n^3)$ that is read in at runtime.

At runtime, the search for the lower and upper point of an interval takes $O(\log(n))$. The worst case of the marking phase would be if *all* map triangles were consistent with an observed triangle, yielding $O(n^3)$ marking operations. Thus, the worst case complexity is $O(m^3(n^3 + \log(n)))$. The algorithm used to partition c matched triangles is $O(c^2)$. In the worst case, there are $O(m^3 n^3)$ matches, so this could take up to $O(m^6 n^6)$ comparisons. The determination of robot location requires, in the worst case, $O(m^2)$ computations to determine θ^* and $O(m)$ computations to determine position.

In order to place these results in perspective, Sugihara [37] reports a labeling complexity of $O(n^3)$ for the *ideal* case (no observation error) using a fixed number (4) points observed from a fixed location with a camera (only monocular information is available). There is no final partitioning step in the case of ambiguity. Under the same conditions, the runtime complexity of the above algorithm is $O(\log(n))$—the time needed to search the lists. The additional algorithm complexity can be attributed to the effects of observation uncertainty and labeling ambiguity.

In practice, the worst case limits are never even approached. By keeping information about the minimal and maximal marked values, marking and scanning can be done very efficiently and are never carried out over the entire array. Furthermore, these operations are very cheap. Partitioning the matches need only be done if the set of possible matches is small enough to generate a unique solution. For example, if 5 observed points generate 200 matches, then elementary combinatorics shows that no unique labeling could be found. No partitioning needs to be done.

An implementation of this algorithm on a Sun IV yields the following timing figures when processing five observed landmarks with 40 stored landmarks:

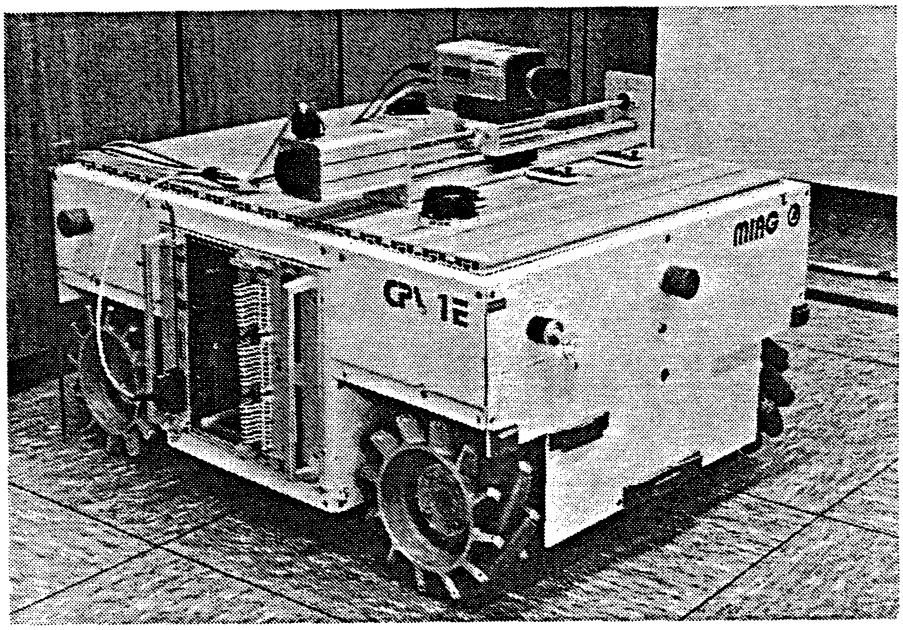

Figure 4: The University of Karlsruhe robot KAMRO with the slider stereo apparatus.

Stereo Solution	< 0.02 sec
Interval Calculation	0.20 sec
Labeling Solution	0.20 sec
Position Determination	0.07 sec
Total	0.49 sec

For 25 stored landmarks, the labeling timing drops to under 0.066 seconds.

Algorithm optimization would probably decrease these values by a factor of two. Furthermore, there are many opportunities for parallelism. The table lookups can be carried out using parallel hardware, and the marking handled by independently addressable bits. It is reasonable to assume that hardware could also be designed to partition matches. Other means of decreasing execution time are discussed in Section 6.2.

5 Experimental Results

The correspondence and localization algorithms have been implemented and tested on a mobile robot system under development at the University of Karlsruhe in coorporation with the Fraunhofer-Institut für Informations- und Datenverarbeitung in Karlsruhe, Germany [31].

The vision system consists of a CCD-camera (resolution 780 by 580 pixels with an 8mm objective) mounted on a controllable slider (positioning precision to 0.02mm). The camera is connected to the VISTA real-time image processing system [30]. This system is in turn connected to an ethernet, and sends information about images (the positions of vertical stripes) to a Sun

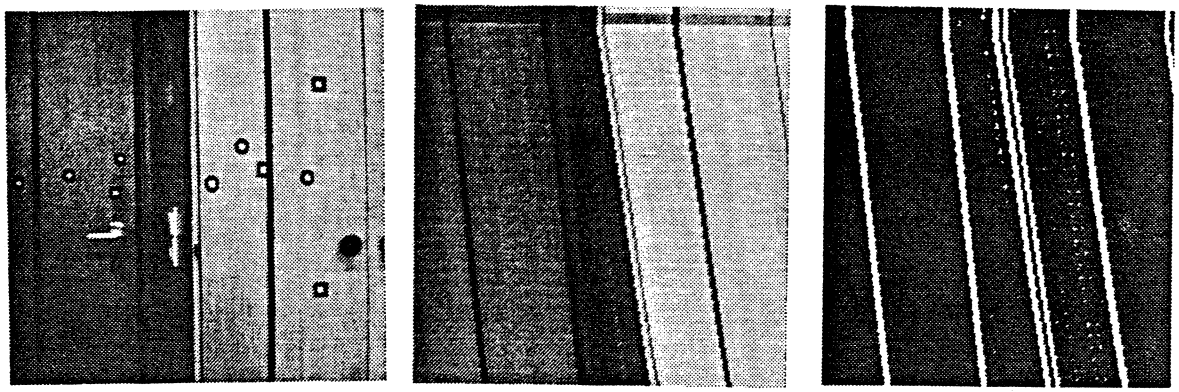

Figure 5: "Band image" image processing steps. Left, the actual scene; middle, the "band image"; and right, the processed image.

4/280 computer. The slider system is mounted on the Karlsruhe mobile robot (KAMRO) [31] pictured in Figure 4.

The camera was calibrated using the procedure described in Section 4. In justification of the second order image distortion model, the distortion coefficient for this lens was calculated to range between -0.094 and -0.1112. Without this coefficient, the error in stereo calculation is equivalent to an observation error of several pixels. If this error is neglected the distortion of edges near the edge of the field of view of the camera results in the rejection of matches that are correct. Accounting for this error by using a larger tolerance would cause the number of false matches to grow quite large, leading to ambiguous matches in many cases.

The notion of image continuity [3; 29] is used to follow the path of a vertical stripe from left to right as the slider moves a pre-set distance. Using the VISTA system, small image slices are taken with the slider in motion. The resulting "band image" is processed with a low-pass (smoothing) and high-pass (differentiation) operator followed by a non-maximal suppression. A line is fit to contiguous patches of pixels, and the coordinates of intersection with the upper and lower edges of the image are used in the stereo calculation. Figure 5 shows a "typical" scene, the resulting "band image," and the filtered and thresholded image.

The error in observation was determined by taking several time series of data and looking at the data spread. Under the assumption that the maximal observation errors are symmetrically distributed about the "true" value, the required tolerance can be read directly from the histogram. Figure 6 shows two representative frequency histograms of x (vertical) position in the image. In all cases, the error was up to and including one half pixel. The effects of the slider maximal error of 0.02mm considered to act on a point observed at a distance of one meter lead to a 0.02 pixel error in image coordinates. Finally, due to the approximation used to compute the maximal angle, and the possibility of other small model errors, the observation error tolerance was inflated slightly to 0.55 pixels.

5.1 Simulation Tests

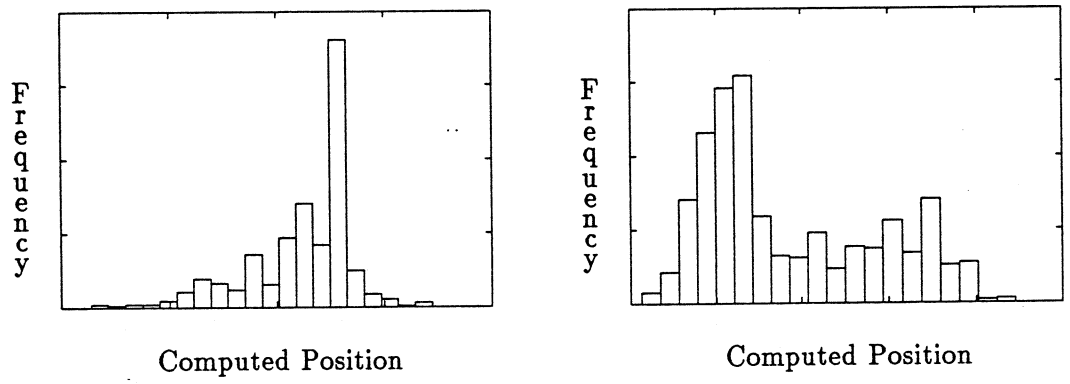

Figure 6: Two data histograms showing the frequency distribution of the horizontal position of two vertical stripes in an image.

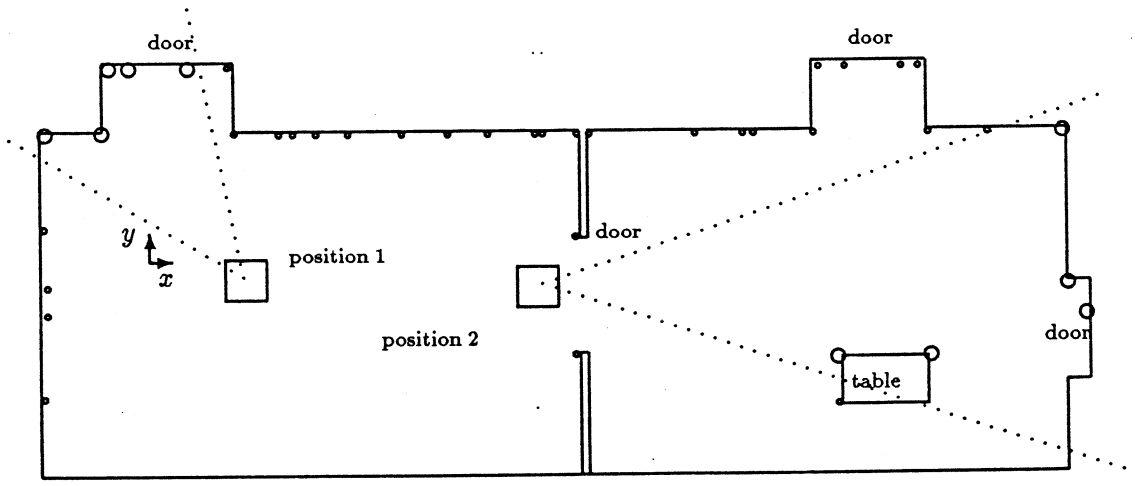

Figure 7: A map of the navigation testing area. Each room is approximately five meters by five meters in size. The small circles indicate landmarks used in simulation experiments.

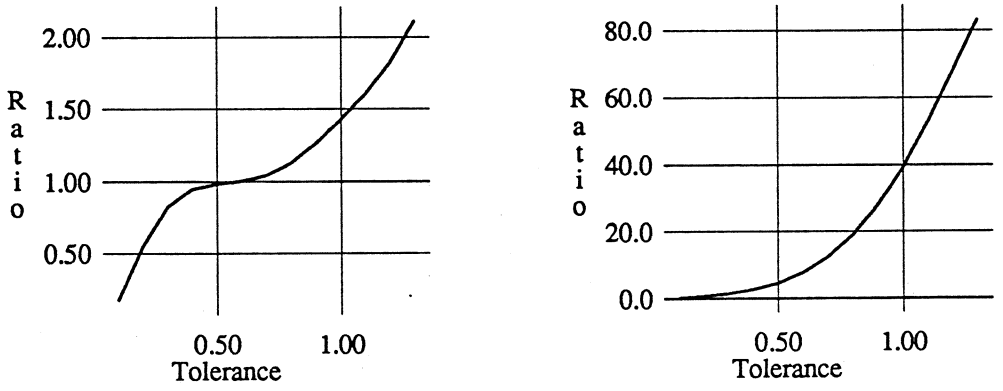

Figure 8: A graph of the ratio of computed matches to expected matches as a function of observation error in two simulations.

Several simulations have been performed to examine the robustness of the labeling algorithm to errors in choosing the observation tolerance parameter ϵ. In general, the performance of the algorithm can vary greatly depending on the geometry of the observed landmarks. The following two cases show the "typical" behavior of the algorithm, and its performance in adverse circumstances.

Simulation 1: The robot position was chosen randomly from the interval $x = 560 \pm 100$, $y = 20 \pm 100$, $\theta = -26 \pm 2$ (position 1 of Figure 7). At each position, the observation of the landmarks (drawn slightly larger) in the viewing cone of the robot from this position was simulated with uniformly varying error in the range of ± 0.55 pixels. In both cases, the ratio of the actual number of matches to the ideal number of matches was computed. In Figure 8, the left graph shows the mean of this ratio as a function of the tolerance ϵ used in the algorithm. In this case, the number of matches was relatively insensitive to the choice of error tolerance. Choosing the tolerance in a range of 0.15 pixels (30%) about the correct value leads to a match ratio within 5% of optimal (1.0). Moreover, the labeling was correct in all cases up to $\epsilon = 1.3$, more than 200% of the correct value.

Simulation 2: The robot position was chosen randomly from the interval $x = 2500 \pm 100$, $y = -200 \pm 100$, $\theta = 90 \pm 2$ (position 2 of Figure 7). At each position, the observation of the landmarks (drawn slightly larger) in the viewing cone of the robot from this position was simulated with uniformly varying error in the range of ± 0.55 pixels. Figure 8 shows the mean of the ratio between the number of computed matches and the ideal number of matches as a function of the observation tolerance ϵ. In this case, because of the large distance from the observed landmarks and structural ambiguities in the stored map even the nearly correct value of $\epsilon = 0.5$ leads to 4.7 times more matches than the ideal case. However, even with this explosive growth the correct labeling was found for all values of ϵ up to 0.8. By slightly modifying the stored map (removing some ambiguities) the correct labeling was found up to $\epsilon = 1.1$.

These tests are by no means conclusive, but they suggest that the exact choice of ϵ is not

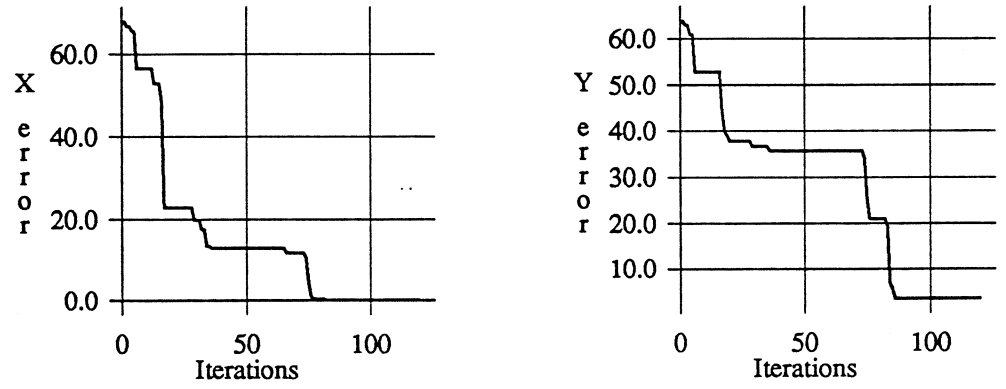

Figure 9: The actual time evolution of the maximal error in robot localization: x component (left) and y component (right).

crucial to good algorithm performance, provided a sufficient number of landmarks (in this case, five) are observed. Choosing the smallest value known to be correct will improve algorithm performance in marginal cases.

5.2 Experiments With Real Data

A typical image taken in the test room may contain from zero to approximately seven observed vertical edges. On the average, the images contain five vertical edges, zero or one of which is a false detection. If the number of observed edges is less than three, then the algorithm cannot be run (a minimum of three points is required to describe a triangle). Unless the observed landmarks are very close (within about a meter), a minimum of four observed stripes is required to provide some labeling redundancy. If the robot does not see a scene with four stripes, there is an error-handling procedure that rotates the robot a small amount and takes another image. This process continues until a satisfactory labeling is found.

Experiment 1: As discussed in Section 2, if the errors in observation across time are stochastic and independent then continued observation of the same scene will drive the maximal error in positioning (the size of the tolerance intervals on the position parameters) toward zero.

To test this idea, the robot was programmed to continually observe, compute location sets, and to combine the estimates over time. Figure 9 shows the rate of convergence of the intervals toward a single point. Figure 10 shows the *expected* rate of convergence over time when observing the same scene with observation error distributed uniformly in the range $[-0.5, 0.5]$.

On this trial the initial localization accuracy is consistent with simulation, though the convergence is much faster than expected. Based on the results in Section 2, even a uniform error distribution appears to be an optimistic assumption for this data. This experiment was carried out to 1000 observations, however the accuracy was not reduced after the 86th observation. The final accuracy of the solution was 0.5 millimeters in x position, 3.3 millimeters in y position,

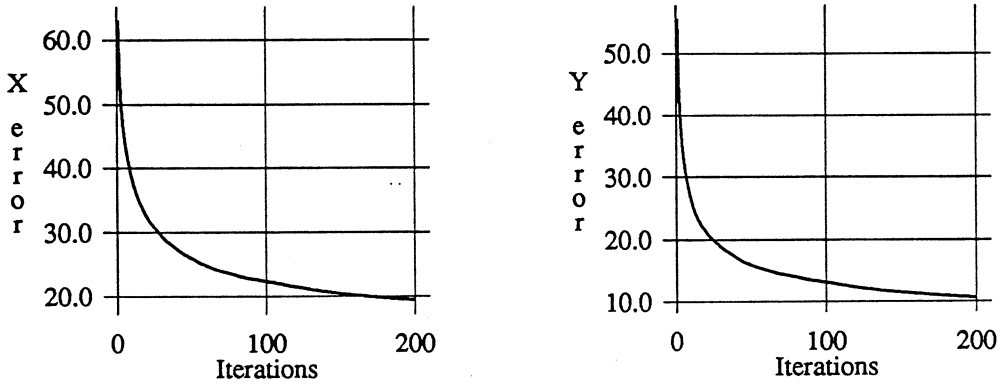

Figure 10: The simulated time evolution of the maximal error in robot localization: x component (left) and y component (right).

and 0.07 degrees of orientation. Hand measurements verified that the error in location estimation was less than one centimeter—approximately the expected level of accuracy relative to the possible systematic errors in calibration. Recomputation of the camera calibration reduced this error to a few millimeters.

Experiment 2: The complete system has been tested by point to point navigation in the two rooms shown in Figure 7. In particular, to move from room to room includes navigating the door opening which required a positioning precision of 3 cm. This particular path has been tested well over 100 times without failure suggesting that the error in positioning is less than 0.5 cm.

6 Extensions

This section describes three extensions to the basic localization algorithm. First, the matching algorithm is extended to allow for uncertain absolute landmark locations. Second, a linear time algorithm for recognizing matches is described. Finally, a method for integrating information across time is presented

6.1 Algorithm Modifications for Landmark Uncertainty

The algorithm in Section 4.1 represents landmark positions as single points. In fact the location of landmarks will often be slightly in error. Hence, the robustness of the labeling and localization algorithms is increased by treating landmarks as intervals. In addition, the use of interval-based landmarks provides the basis for map learning. Newly observed landmarks can be added to the map as large intervals, and these intervals refined as the landmarks are continually reobserved.

The major difficulty in extending the labeling algorithm to intervals on landmark positions

is that triples of three landmarks must be represented by intervals on angles and lengths. The labeling algorithm must determine if the interval of values for a map triangle intersects the interval of values for an observed triangle. However, intervals cannot be totally ordered, and hence the search and marking phase of the labeling algorithm does not apply. It is possible to construct a search and marking algorithm for interval landmarks that has exactly the same computational complexity as the algorithm for point-based landmarks, but the space complexity of that algorithm is $O(n^6)$. Thus, it is practical only for very small maps.

An $O(n^3)$ space complexity solution to the labeling problem that allows uncertainty on landmark position proceeds as follows:

- For each of the three angles and lengths, make *two* lists of value/pointer pairs sorted by lower and upper value, respectively.
- Given the ranges of angles and length for an observed triangle pair, search for the first value in each list of lower values that is less than or equal to the *upper value* of the corresponding observed triangle value. Mark this and all smaller entries.

  Likewise, find the first value in the list of upper values that is no smaller than the *lower value* of the corresponding observed triangle value. Mark this and all larger entries.
- After this has been done for all six triangle parameters, take only those values that have been marked twelve times.

It is easy to show that this algorithm is correct: it finds the exact set of landmark triples that match an observed triple of points. Its primary drawback is the excessive number of marking operations.

The number of marking operations can be reduced by noting that the error tolerance on map points should be small in order for the labeling algorithm to work efficiently. Consequently, the range of angles and lengths computed from triples of map points will often be quite small. Suppose that the maximum size of any interval on $\alpha = [\alpha_l, \alpha_u]$ is w. Then any interval that intersects α must have a lower value in the range $[\alpha_l - w, \alpha_u]$, and likewise any interval that intersects α must have an upper value in the range $[\alpha_l, \alpha_u + w]$. Using this information, the final labeling algorithm can be described as follows:

Initialization:

- For each unique grouping of map points, compute (the interval) $S_{i,j,k}$ (using the methods described in Section 4.1) and add it to a list M. Call the final length of this list q.
- For each element $M_j = [l, u]$, $j = 1, \ldots, q$, for each i, $i = 1 \ldots 6$, add the pair $\langle l_i, j \rangle$ to a list L^i and the pair $\langle u_i, j \rangle$ to a list U^i
- Sort the elements of each L^i and U^i, $i = 1, \ldots, 6$ on the first (value) component yielding twelve lists of pairs of indices and values sorted on value.
- Determine the maximal width of each of the six components of elements in M. Call this vector W.

Runtime: For each triplet of observed stereo pairs o_a, o_b, o_c,

- Compute (the interval) $\mathbf{S}_{a,b,c} = [l, u]$ encoding the permissible range of the six triangle parameters for the given tolerance ϵ.
- For each coordinate $i = 1, \ldots, 6$
  - Find the first index r_u such that the value field of $L^i_{r_u}$ is no larger than u_i.
  - Find the first index r_l such that the value field of $L^i_{r_l-1}$ is smaller than $l_i - W_i$.
  - For each $\langle v_j, k_j \rangle = L^i_j$, $r_l \leq j \leq r_u$, mark M_{k_j} as found.
  - Find the first index s_l such that the value field of $U^i_{s_l}$ is no smaller than l_i.
  - Find the first index s_u such that the value field of $U^i_{s_u+1}$ is larger than $u_i + W_i$.
  - For each $\langle v_j, k_j \rangle = U^i_j$, $s_l \leq j \leq s_u$, mark M_{k_j} as found.
- For each $S_{i,j,k} \in M$ that has been marked twelve times, add $(\langle o_a, p_i \rangle, \langle o_b, p_j \rangle, \langle o_c, p_k \rangle)$ to Λ.

Matches are disambiguated as before. Likewise, robot position is computed as described in Section 4.3 except that (6) and (7) must be modified to account for landmark position uncertainty. This is easily accomplished by using interval arithmetic [28].

This algorithm has been tested on simulated and real data with landmark uncertainty of 5 mm. There was no noticeable degradation of the labeling algorithm performance, and the increase in the size of the robot location interval was minimal. However, the speed of the algorithm decreased due to a larger number of triangles found to be consistent with observed data.

6.2 A Faster Partitioning Algorithm

As noted before, the major bottleneck in labeling, particularly when the landmarks are represented as intervals, is partitioning the matches. Instead of partitioning the matches, it is possible to construct a *match matrix* with m rows and n columns, where the entry in row i and column j indicates a match between observed feature i and map landmark j. For the example of Section 4.2, the match matrix is:

$$\begin{bmatrix} 6 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 3 & 1 & 1 \\ 0 & 0 & 0 & 4 & 0 \end{bmatrix}$$

For m observations to be labeled, the following conditions must hold:

1. Each observation must be matched in $(m-1)(m-2)/2$ triangles. In other words, exactly one value in each row must attain or exceed this threshold.

2. The above condition must hold for m rows.

3. The maxima must occur in m different columns.

If these conditions are fulfilled, then the matches for the rows with a member that passed the threshold value are chosen. Otherwise, the number of assumed correct landmarks is reduced by 1, and the process repeated. This continues until some row contains more than one value exceeding the threshold, indicating an ambiguity that cannot be resolved using this algorithm.

This algorithm is an approximation: it can miss unique labelings and can also commit labeling errors, though both cases, in particular the latter, are highly unlikely. For example, the following example shows a labeling that would be missed:

Λ_1	Λ_2	Λ_3	Λ_4
$o_1, o_2, o_3 \Longleftrightarrow p_1, p_2, p_3$	$o_1, o_2, o_3 \Longleftrightarrow p_2, p_1, p_4$	$o_1, o_2, o_4 \Longleftrightarrow p_2, p_3, p_4$	$o_1, o_3, o_4 \Longleftrightarrow p_2, p_3, p_5$
$o_1, o_2, o_4 \Longleftrightarrow p_1, p_2, p_4$			
$o_1, o_3, o_4 \Longleftrightarrow p_1, p_3, p_4$			
$o_2, o_3, o_4 \Longleftrightarrow p_2, p_3, p_4$			

The match matrix is:

$$\begin{bmatrix} 3 & 3 & 0 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 \\ 0 & 0 & 4 & 1 & 0 \\ 0 & 0 & 0 & 4 & 1 \end{bmatrix}$$

In this case, the algorithm will indicate that the correct correspondence for observation 1 is ambiguous, even though this is not strictly the case. In general, many similar constellations of landmarks in the map lead to this type of error.

The following six matches lead to a labeling error:

$$o_1, o_2, o_3 \Longleftrightarrow p_1, p_2, p_3 \quad o_1, o_2, o_3 \Longleftrightarrow p_1, p_2, p_4 \quad o_1, o_2, o_3 \Longleftrightarrow p_1, p_2, p_5$$
$$o_1, o_3, o_4 \Longleftrightarrow p_6, p_4, p_5 \quad o_1, o_3, o_4 \Longleftrightarrow p_7, p_4, p_5 \quad o_1, o_3, o_4 \Longleftrightarrow p_8, p_1, p_5$$

The match matrix is:

$$\begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \end{bmatrix}$$

The correspondences $o_1 \Longleftrightarrow p_1$, $o_2 \Longleftrightarrow p_2$, $o_3 \Longleftrightarrow p_4$, $o_4 \Longleftrightarrow p_5$, would be chosen. In general, for such an error to happen one triple of observed points must be matched in several

ways, all the time keeping one or two elements of the match fixed. In order for this to happen, the map must contain several nearly ambiguous pairs, or the uncertainty on the observations must be quite high. Conversely, the uncertainty cannot be so large that an ambiguity arises. The conjunction of these circumstances is quite small. This case has never arisen in several tests of the algorithm.

For c matches, the time required to choose a labeling is $O(c)$ (the list of matches must be processed once to develop the matrix). With this algorithm, the time needed to compute a labeling solution for the problem given in Section 4.4 drops from 0.2 sec to 0.1 sec. The time improvements are even more significant for larger problems with 6 to 8 observed landmarks.

6.3 Using Information About Robot Motion

Location estimation often does not have to be solved from first principles. If system motions can be modelled, prior location information can be projected using that model and combined with new location information. One of the principle advantages of so-called "recursive" (*e.g.* the Kalman filter) schemes is the use of a system dynamics model to project and combine past information with new information.

A similar scheme is possible using interval-based uncertainties. To illustrate, assume the only uncertainty in motion is in translation, and that it is proportional to distance travelled. If s is a commanded speed, k is an error ratio, t is a period of time, and $\mathcal{P} = (x, y, \theta)$, then a model for the projection of location estimates is

$$\begin{bmatrix} x^-_{k+t} \\ y^-_{k+t} \end{bmatrix} = \begin{bmatrix} x^+_k \\ y^+_k \end{bmatrix} + st(1 + [-k, k]) \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}$$

Since $\mathcal{P}$ is a vector of closed intervals, the above expression can be computed using interval arithmetic.

If $\hat{x}$ and $\hat{y}$ are estimated locations at time $k + t$, then the consistent locations must be

$$x^+_{k+t} = \hat{x} \cap x^-_{k+t}, \quad y^+_{k+t} = \hat{y} \cap y^-_{k+t} \tag{8}$$

Figure 11 depicts the results of combining motion and observation over time. The unshaded squares represent projected locations, the shaded squares represent estimated position, and the darkly outlined squares represent the jointly consistent regions computed by (8).

This temporal constraint also provides a means for verifying labelings. If multiple consistent labelings are computed, but only one is consistent with the projection of the previous location, then that labeling has a high likelihood of being correct. Likewise, if no labeling is consistent with the previously projected location, then a labeling error is a possible cause.

7 Discussion and Future Work

The set-based matching and localization algorithms described in this article have several advantages in problem areas where a bounded error model is appropriate:

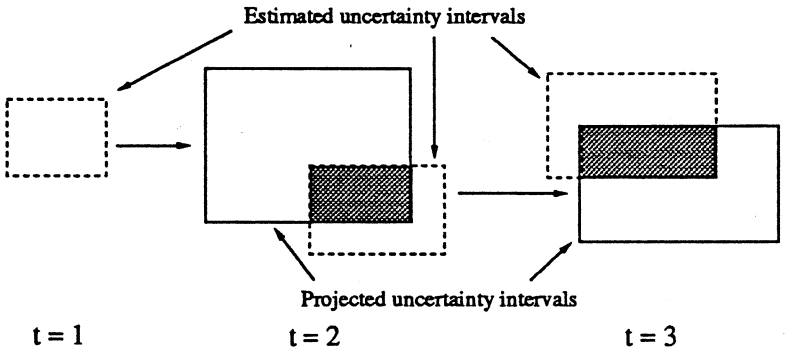

Figure 11: Updating over time. The shaded area denotes the intersection of regions consistent with past information (solid boxes) and current estimated location (dashed boxes).

1. The model parameters are purely geometric and easily recovered by examining series of data.
2. The algorithms are simple and fast.
3. The methodology can be proved convergent under extremely weak statistical assumptions.
4. The algorithms have a very low probability of failure, and deliver quantitative, provably conservative bounds on localization error.

The localization algorithms are also easily parallelized as an initial implementation on a Transputer^{TM} network has shown. With specialized hardware, it may be possible to perform localization at video rates.

The methods are comparable with the widely-published Kalman filter-based methods in terms of simplicity and execution time, though they rely on a different set of assumptions. In particular, the correspondence algorithm requires no prior location information or prior statistical assumptions. In comparison, the solutions reported in [2; 11; 10] and other related Kalman filter-based work require a good initial solution and good statistical models to "bootstrap" the data association algorithms used.

Likewise, the computation of location deals with errors explicitly and quantitatively. Again, since set-based estimation does not rely on strong statistical assumptions, the algorithms are more robust than estimation methods that do rely on distributional assumptions. In general, we believe the computation of solution *sets* is an important approach to robotics problems. That is, rather than computing a single point, or a single point with some type of (often heuristic) figure of merit, it is better to compute the complete set of possible solutions based on observation uncertainty intervals. Many types of robotic tasks that depend on the configuration space of the robot can be easily formulated and solved in terms of sets of locations.

All of the analysis in this article rests on the idea that correspondence is solved from first principles at every time step. The next phase of this work includes mounting a second camera on the robot and using feature tracking to support continuous stereo. The methods presented

above will be used to locate features and solve the static localization problem. Thereafter, landmarks will be tracked and a running location estimate will be computed. Both Kalman filter and set-based methods will be tested for accuracy, suitability, and efficiency.

Furthermore, some aspects of "active" vision will be investigated. For example, if the correspondence cannot be solved or localization accuracy becomes unacceptable, the system can enlarge the stereo baseline. This will improve accuracy, although at the cost of having fewer landmarks common to both images. In addition, a wider baseline may complicate feature tracking. Slowing the robot motion increases the effective sampling rate and thereby increases the accuracy of localization without complicating correspondence or tracking. However, the task is performed more slowly. Hence, the goal is to optimize parameters such as baseline or speed with respect to task requirements and algorithm performance. The use of decision-theoretic methods [5] for formalizing these notions will be investigated.

Acknowledgements The navigation system for the mobile robot of the University of Karlsruhe is funded by the Deutsche Forschungsgemeinschaft as part of a cooperative research project on artificial intelligence (Sonderforschungsbereich Künstliche Intelligenz). The second author would also like to acknowledge the contribution of the Fulbright Commission which supported him as a visiting scientist for one year at the above-mentioned institutions. The following agencies also contributed to the support of the second author: DARPA Grant N0014-88-K-0630 (administered by ONR), AFOSR Grants 88-0244, AFOSR 88-0296; Army/DAAL 03-89-C-0031PRI; NSF Grants CISE/CDA 88-22719, IRI 89-06770; and Du Pont Corporation.

We wish to thank H.-H Nagel for his valuable advice and encouragement and for careful reading of earlier drafts of this paper. We also wish to thank Jerome Kodjabachian and Max Mintz for their comments on earlier drafts of this paper, and Jerome Kodjabachian and H.-H. Nagel for independent derivations of the closed-form solution to the angle maximization problem. We are also grateful to Professors U. Rembold and R. Dillmann for their supportive cooperation.

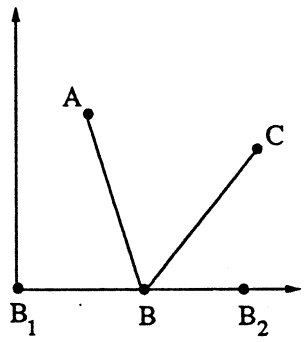

Figure 12: Labeling conventions for determining maximum angle.

A A Closed Form Solution for Maximal Angle

Suppose A, B_1, B_2, and C are arranged as in Figure 12. The problem is to find the point B along the line B_1B_2 where the expression

$$\frac{(A-B)\cdot(C-B)}{\|A-B\|\|C-B\|} \tag{9}$$

attains its minimum value.

In order to simplify the problem, all points will be translated by $-B_1$ and rotated so that the line B_1B_2 is aligned with the x axis. Under this transformation, the above expression expands to:

$$\frac{(A_x-x)(C_x-x)+A_yC_y}{\sqrt{((A_x-x)^2+A_y^2)((C_x-x)^2+C_y^2)}}.$$

The point B becomes $(x,0)$ and the problem is to find the values of x that minimizes the above expression.

Assuming neither A or C lies on the x axis, differentiating the above expression, equating it to zero, and rearranging the terms yields the polynomial:

$$\begin{aligned} P &= (A_x+C_x-2x)((A_x-x)^2+A_y^2)((C_x-x)^2+C_y^2) \\ &\quad -((A_x-x)(C_x-x)+A_yC_y)[(A_x-x)((C_x-x)^2+C_y^2)+(C_x-x)((A_x-x)^2+A_y^2)] \\ &= [(A_x-x)C_y-(C_x-x)A_y]\left[((A_x-x)^2+A_y^2)C_y-((C_x-x)^2+C_y^2)A_y\right] \\ &= P_1P_2. \end{aligned}$$

Thus, the polynomial P is 0 only when P_1 or P_2 is zero. Recall both C_y and A_y are positive quantities. Under these conditions, the solution leading to $P_1 = 0$ is the case when A, C and B are colinear. The corresponding angle is either 0 or 180 degrees.

P_2 is a quadratic in x of the form $ax^2 + bx + c$ where

$$\begin{aligned} a &= (C_y - A_y), \\ b &= 2(A_y C_x - C_y A_x), \\ c &= C_y(A_x^2 + A_y^2) - A_y(C_x^2 + C_y^2). \end{aligned}$$

Solving this quadratic and simplifying leads to the final solution

$$x = \frac{(C_y A_x - A_y C_x) \pm A_y C_y((A_x - C_x)^2 + (A_y - C_y)^2)}{C_y - A_y}.$$

If one of these values lies between B_1 and B_2, then substituting that value into (9) and computing the inverse cosine yields the maximum angle. Otherwise the maximum angle results when $B = B_1$ or $B = B_2$ in (9).

References

[1] G. Alefeld and J. Herzberger. *Introduction to Interval Computations.* Academic Press, New York, N.Y., 1983.

[2] N. Ayache and O. D. Faugeras. Building, registrating, and fusing noisy visual maps. *The International Journal of Robotics Research*, 7(6):45–65, 1988.

[3] H. Baker and R. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. In *Proceedings of the 1988 DARPA Image Understanding Workshop*, pages 1022–1030. Morgan Kaufmann, Los Altos, CA, 1988.

[4] B. R. Barmish and J. Sankaran. The propagation of parametric uncertainty via polytopes. *IEEE Journal on Automatic Control*, AC-24(2):346–349, April 1979.

[5] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis.* Springer-Verlag, New York, N.Y., 2nd edition, 1985.

[6] D. P. Bertsekas and I. B. Rhodes. Recursive state estimation for a set-membership description of uncertainty. *IEEE Journal on Automatic Control*, AC-16(2):117–128, April 1971.

[7] P. Bickel and K. Doksum. *Mathematical Statistics.* Holden-Day, Oakland, CA, 1977.

[8] T. M. Breuel. Model based recognition using pruned correspondence search. In *Proceedings of the 1991 Conference on Computer Vision and Pattern Recognition*, pages 257–262. IEEE Computer Society Press, Washington, D.C., 1991.

[9] R. Chatila and P. Moutarlier. Stochastic multisensor data fusion for mobile robot location and environment modelling. In *Proceedings of the 5th International Symposium of Robotics Research*, pages 207–216. MIT Press, Cambridge, MA, 1989.

[10] J. L. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pages 674–680. IEEE Press, Washington, D.C., 1989.

[11] H. Durrant-Whyte. *Integration, Coordination, and Control of Multi-Sensor Systems.* Kluwer, Boston, MA, 1988.

[12] A. Gelb, editor. *Applied Optimal Estimation.* MIT Press, Cambridge, MA, 1974.

[13] W. E. Grimson. Sensing strategies for disambiguating among multiple objects in known poses. *IEEE Journal on Robotics and Automation*, RA-2(4):196–213, 1986.

[14] G. D. Hager. Set-based estimation: Towards task-directed sensing. In *Proceedings of Melecon '91*, pages 1205–1209. IEEE Computer Society Press, Washington, D.C., 1991.

[15] G. D. Hager. Deciding not to decide using resource-bounded sensing. In *Sensor Fusion III: 3-D Perception and Recognition*, volume 1383, pages 379–390. SPIE, Bellingham, WA, 1990.

[16] G. D. Hager. *Task-Directed Sensor Fusion and Planning.* Kluwer, Boston, MA, 1990.