Mobile Robot Localization using Landmarks

(Extended Abstract)

Margrit Betke*
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

Leonid Gurvits†
Siemens Corporate Research
755 College Road East
Princeton, NJ 08540

Abstract

We describe an efficient algorithm for localizing a mobile robot in an environment with landmarks. We assume that the robot has a camera and maybe other sensors that enable it to both identify landmarks and measure the angles subtended by these landmarks. We show how to estimate the robot's position using a new technique that involves a complex number representation of the landmarks. Our algorithm runs in time *linear* in the number of landmarks. We present results of our simulations and propose how to use our method for robot navigation.

1 Introduction

Consider an autonomous agent, which could be a mobile robot or a human traveler, who uses a map to navigate through an environment that contains landmarks. The landmarks are marked on the agent's map. The autonomous agent also has a tool which can measure angles; this tool might be a compass. The agent may use the following algorithm to identify its location in the environment:

1. Identify surrounding landmarks in the environment.
2. Find the corresponding landmarks on the map.
3. Measure the angles subtended at your position by the landmarks.
4. Compute your position efficiently.

If the first three steps can be executed without errors three landmarks are sufficient to compute the position of the agent, unless the agent's position and these three landmarks either form a circle, or lie on one line. Then the localization problem does not have a unique solution.

However, in real life an agent makes two kinds of mistakes: (1) some angles are measured with small errors and (2) some landmarks are misidentified. Another source of mistakes could be errors in the map itself. Suppose that the majority of mistakes are of the first type. In this situation we address the following problems:

- Estimating the position of the agent efficiently.
- Finding misidentified landmarks and large angle measurement errors.

In this paper we present an algorithm that estimates the agent's position in $O(n)$ operations where n is the number of landmarks on the 2-D map. Our simulations show that large errors due to misidentified landmarks and erroneous angle measurements can be found, discarded and the algorithm can be repeated without them with improved results.

Our work is motivated by a mobile robot called Ratbot that was built at the Learning Systems De-

*Part of this research was done while the author was visiting Siemens Corporate Research. The author is supported by NSF grant ASC-9217041. Author's net address: margrit@lcs.mit.edu

†Author's net address: gurvits@scr.siemens.com

partment at Siemens Corporate Research. Ratbot is used as a test bed for various machine learning approaches to robot navigation. Ratbot navigates through the corridors of the building at Siemens Corporate Research. It is equipped with a camera that points upwards onto a reflective ball which acts like a mirror of the surroundings. The straight lines of objects like pictures, doors, and walls look like arcs in the images taken by Ratbot's camera. Only a circular, one-dimensional strip of the brightness of the image is analyzed. Landmarks like doors, pictures, and fire extinguishers appear as dark bands on a strip. The strips provide information on the angles subtended at the robot's position by two landmarks, but not on the distance of the landmarks to the camera (i.e., depth information).

To find the corresponding features of an image that is taken during navigation and an image that is taken in advance and stored in a database, only the one-dimensional strips of the two images need to be compared. The *correspondence problem* is solved by an algorithm due to Hancock and Judd [5]. Once corresponding landmarks are identified, a description of the matching landmarks within the global map of the robot's environment is given. The map of the environment is a two-dimensional description of the office building through which the robot navigates. We call the coordinate system that describes the global map of the environment an *external* coordinate system. We distinguish it from the *camera* coordinate system whose origin is at the mobile robot's location in the environment. The position of the robot with respect to the external coordinate system is unknown and needs to be determined. This problem is similar to exterior orientation problems in machine vision or hand-eye calibration in robotics [6].

The problem of relating angle information provided by landmarks to the position of a navigating robot has been addressed previously [13, 9, 10]. Many authors have studied landmark navigation and position estimation [1, 2, 7, 8, 12, 14]; Mataric [11] gives a survey of related work.

Our position estimation algorithm has been used by Greiner and Isukapalli [3] to determine which of the landmarks that are visible to Ratbot are useful for its position estimate. However, our algorithm is not restricted to robots that have the same geometry as Ratbot's camera system. Our techniques are general and can be used for other navigation or exterior orientation problems.

Section 2 gives a formal description of our problem. Section 3 describes our position estimation algorithm which uses a new linear approach that represents landmarks with complex numbers. Our algorithm runs in time linear in the number of landmarks. Section 4 proposes how to incorporate our algorithm in an on-line navigation algorithm that estimates the robot's position while the robot is moving.

2 The robot position problem

In this section we define the problem of estimating a robot's position in its environment given a global map of the environment and the angles subtended by landmarks that it measures at its position.

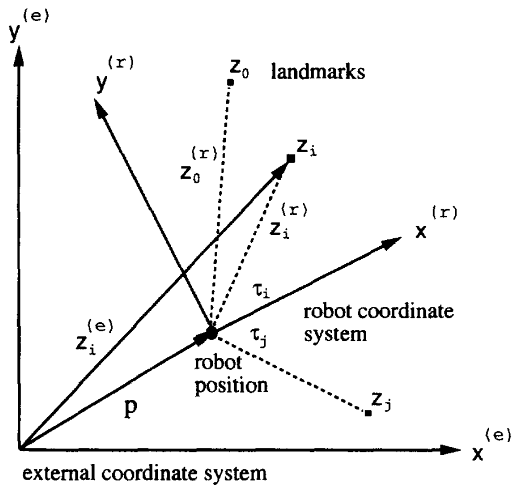

Figure 1: Environment with external coordinate system $(x^{(e)}, y^{(e)})$ and robot coordinate system $(x^{(r)}, y^{(r)})$.

The position of a landmark $z^{(e)}$ is given by two coordinates $(x^{(e)}, y^{(e)})$. The superscript e denotes that landmark $z^{(e)}$ is given in the external coordinate system. A landmark which is described by a vector in the robot-centered camera coordinate system is denoted by $z^{(r)}$. We illustrate the environment with its coordinate systems in Figure 1.

The positions $z_0^{(e)}, \dots z_n^{(e)}$ of the landmarks in the external coordinate system are given on the map. The angles subtended by the landmarks from the robot's position are measured by the robot's sensors. The angle subtended at the robot's position by the perpendicular from the x-axis of the robot-centered coordinate system to landmark $z_i^{(r)}$ is denoted by τ_i. Note that the measurements are noisy in practice.

Now we can formulate our problem: Given the external positions $z_0^{(e)}, \dots, z_n^{(e)}$ of n landmarks and corresponding (noisy) angle measurements $\tau_0, \dots \tau_n$, estimate the position p of the robot in the environment.

3 Linear position estimation

In order to solve the robot position problem we can apply the law of cosine to all possible sets of two landmarks and get a system of nonlinear equations. This system is overdetermined and can be solved using a least squares approach. Standard algorithms that solve large numbers of nonlinear equations take too long for real-time robot navigation. Therefore, in this section we introduce a new, *linear* approach to the problem. We describe an efficient algorithm that runs in time linear in the number of landmarks. Instead of applying the law of cosine, we use a complex number representation of the landmarks and get a set of linear equations. To solve this particular system of m equations, we do not need an $O(m^3)$ matrix inversion algorithm. Instead we introduce an $O(m)$ algorithm which takes advantage of the special structure of the matrix and the right side of the equation.

The $(x^{(r)}, y^{(r)})$ coordinates of a landmark $z^{(r)}$ can be written as a complex number $z^{(r)} = x^{(r)} + jy^{(r)} = l\, e^{j\tau}$ in the robot-centered coordinate system. Then the ith landmark is

$$z_i^{(r)} = l_i\, e^{j\tau_i}.$$

The robot measures the angles subtended by a reference landmark $z_0^{(r)}$ and the other landmarks $z_1^{(r)}, \dots, z_n^{(r)}$ at its position. (Landmark $z_0^{(r)}$ may be chosen to be the most reliable landmark.) Dividing the complex number $z_i^{(r)}$ by $z_0^{(r)}$ for $i = 1, \dots, n$ gives us a set of equations that includes the measured angles φ_i subtended by landmark z_i and landmark z_0:

$$\frac{z_i^{(r)}}{z_0^{(r)}} = \frac{l_i}{l_0}\, e^{j(\tau_i - \tau_0)} = \frac{l_i}{l_0}\, e^{j\varphi_i} \qquad (1)$$

The difference between two vectors in the robot-centered coordinate system is the same as the difference between the corresponding vectors in the external coordinate system. Thus, the vectors between the landmarks in the robot-centered coordinate system are known. The vector v_i between landmark z_0 and z_i can be written as

$$v_i = z_i^{(e)} - z_0^{(e)} = z_i^{(r)} - z_0^{(r)}.$$

Given that $z_i^{(r)} = z_0^{(r)} + v_i$, equation (1) becomes

$$\frac{z_0^{(r)} + v_i}{z_0^{(r)}} = \frac{l_i}{l_0}\, e^{j\varphi_i} \quad \textit{for} \ i = 1, \dots, n$$

After some algebra we obtain a set of equations whose only unknowns are $z_0^{(r)}$ and v_i for $i = 1, \dots, n$:

$$\frac{1}{z_0^{(r)}} = \frac{l_i}{l_0} \frac{1}{v_i}\, e^{j\varphi_i} - \frac{1}{v_i} \quad \textit{for} \ i = 1, \dots, n \qquad (2)$$

To remove the dependence on $z_0^{(r)}$, we substitute the left-hand side of Equation (2) with the expression on the right-hand side for a different index k:

$$\frac{l_k}{l_0} \frac{1}{v_k} e^{j\varphi_k} - \frac{1}{v_k} = \frac{l_i}{l_0} \frac{1}{v_i} e^{j\varphi_i} - \frac{1}{v_i} \qquad (3)$$

for $i, k = 1, \dots, n$, and $k \neq i$. The set of equations (3) can be transformed into a matrix equation $A\, r = c$ where r, c, and A can be defined as follows: Vector $r = (l_1/l_0, \dots, l_n/l_0)$ is the vector of the unknown ratios of the length of vectors $z_0^{(r)}$ and $z_1^{(r)}, \dots, z_n^{(r)}$. Vector c is a $n(n-1)$ dimensional vector of differences of complex numbers $c_i = 1/v_i$ such that

$$c = \begin{pmatrix} c_1 - c_2 \\ c_1 - c_3 \\ \vdots \\ c_1 - c_n \\ c_2 - c_1 \\ c_2 - c_3 \\ \vdots \\ c_2 - c_n \\ \vdots \\ c_n - c_1 \\ c_n - c_2 \\ c_n - c_3 \\ \vdots \\ c_n - c_{n-1} \end{pmatrix}$$

and matrix A is a $n(n-1) \times n$ matrix consisting of complex numbers $b_i = c_i\, e^{j\varphi_i}$:

$$A = \begin{pmatrix} b_1 & -b_2 & 0 & \dots & 0 & 0 \\ b_1 & 0 & -b_3 & \dots & 0 & 0 \\ \vdots & & & & & \\ b_1 & 0 & 0 & \dots & 0 & -b_n \\ -b_1 & b_2 & 0 & \dots & 0 & 0 \\ 0 & b_2 & -b_3 & \dots & 0 & 0 \\ \vdots & & & & & \\ 0 & b_2 & 0 & \dots & 0 & -b_n \\ \vdots & & & & & \\ -b_1 & 0 & 0 & \dots & 0 & b_n \\ 0 & -b_2 & 0 & \dots & 0 & b_n \\ 0 & 0 & -b_3 & \dots & 0 & b_n \\ \vdots & & & & & \\ 0 & 0 & 0 & \dots & -b_{n-1} & b_n \end{pmatrix}$$

Note that the system of linear equations $A\,r = c$ is overdetermined. We have n unknowns $r_1, \dots, r_n$ and $n(n-1)$ equations. These equations may be inconsistent, since we have measurement errors. Therefore, we need to find a solution for which the "average error" in the equations is minimized. Such a solution is obtained by the "least squares method." We define the average error by the sum of squares $E^2 = ||A\,r - c||^2$. If there is an exact solution to $A\,r = c$, the error is $E = 0$. In practice it is unlikely that there is no error, therefore, we find the ratio vector r for which E^2 is minimized. The solution to

$$\min\{r : E^2\} = \min\{r : ||A\,r - c||^2\}$$

is r in terms of the *pseudo inverse* $A^+ = (A^TA)^{-1}A^T$ of matrix A

$$r = (A^TA)^{-1}A^Tc$$

providing matrix A^TA is nonsingular. In Section 3.2 we give the necessary and sufficient conditions for A^TA being nonsingular and argue that they will almost always be fulfilled in practice. In the following section we describe an algorithm called POSITION ESTIMATOR which efficiently calculates vector $r = A^+c = (A^TA)^{-1}A^Tc$ such that the squared error E^2 in the equations is minimized.

3.1 The position estimation algorithm

In this section we describe procedure POSITION ESTIMATOR that estimates the position of the robot. Procedure POSITION ESTIMATOR takes as an input the positions of the landmarks $z_0^{(e)}, \dots, z_n^{(e)}$ (as given in the external coordinate system) and the measured angles $\varphi_1, \dots \varphi_n$. After the initialization of vectors v, c, b and $\sum_{i=1}^n c_i$ in lines 1–5, the algorithm first calculates vector $s = \frac{1}{2}A^Tc$ in lines 6 and 7, and then determines $r = 2(A^TA)^{-1}s$ by calling procedure RATIOS-CALCULATOR in line 8. It next calculates n position estimates $p_1, \dots, p_n$ given by vector r (lines 9-11) and uses them to estimate the position of the robot p in line 12.

```
POSITION-ESTIMATOR($z_0^{(e)}, \dots, z_n^{(e)}, \varphi_1, \dots \varphi_n$)
 1  for $i = 1$ to $n$                    (initialization)
 2      $v_i = z_0^{(e)} - z_i^{(e)}$
 3      $c_i = 1/v_i$
 4      sum-of-$c_i$ = sum-of-$c_i + c_i$
 5      $b_i = c_i e^{j\varphi_i}$
 6  for $i = 1$ to $n$                    (calculates $\frac{1}{2}A^Tc$)
 7      $s_i = nb_i^Tc_i - b_i^T$ sum-of-$c_i$
 8  RATIOS-CALCULATOR($b_1, \dots, b_n, s_1, \dots, s_n$)
                                          (returns $r$)
 9  for $i = 1$ to $n$
10      $z_{0,i}^{(r)} = 1/(r_ib_i - c_i)$
11      $p_i = z_0^{(e)} - z_{0,i}^{(r)}$
12  $p = \frac{1}{n}\sum_{i=1}^n p_i$                (robot position $p$)
```

Note that in the following detailed description of the algorithm we use two kinds of multiplications of complex numbers $a = (Re(a), Im(a))$ and $b = (Re(b), Im(b))$: the standard complex multiplication ab and the dot product $a.b = a^Tb = (Re(a)Re(b) + Im(a)Im(b))$. For each component of $s = \frac{1}{2}A^Tc$ we have

$$s_i = (n-1)b_i^Tc_i - \sum_{j,j\neq i} b_i^Tc_j$$

$$s_i = nb_i^T c_i - b_i^T \sum_{j=1}^{n} c_j \qquad (4)$$

In line 8 procedure RATIOS-CALCULATOR returns the vector $r = (A^TA)^{-1}A^Tc$. In line 10 of procedure POSITION ESTIMATOR a set of n solutions $z_{0,1}^{(r)} \dots, z_{0,n}^{(r)}$ for the position of landmark z_0 in the robot-centered coordinate system is calculated using equation (2):

$$z_{0,i}^{(r)} = \frac{1}{r_i b_i - c_i} = \frac{1}{\frac{l_i}{l_0}\frac{1}{v_i}e^{j\varphi_i} - \frac{1}{v_i}} \quad \textit{for } i = 1, \dots, n$$

In line 11 of procedure POSITION ESTIMATOR a set of estimates $p_1, \dots, p_n$ for the robot position is calculated using the solutions for the position of landmark z_0. If there is no noise in the measurements, the vectors $z_{0,i}^{(r)}$ are the same for all indices i. In practice there will be noise, so we take the centroid

$$p = \frac{1}{n}\sum_{i=1}^{n} p_i$$

of the position estimates $p_1, \dots, p_n$ as an average to obtain an estimate of the position p of the robot (line 12 of procedure POSITION ESTIMATOR). Note that taking the centroid may not be the best way to average the n estimates. We are currently working on determining the best averaging coefficients. They may not be the same for the x and y coordinates of p_i [4].

Procedure RATIOS-CALCULATOR takes as an input the vectors b and s and returns the vector $r = 2(A^TA)^{-1}s$. It exploits the special form of matrix $A^TA =$

$$2\begin{pmatrix} (n-1)\,b_1^Tb_1 & -b_1^Tb_2 & \dots & -b_1^Tb_n \\ -b_2^Tb_1 & (n-1)\,b_2^Tb_2 & \dots & -b_2^Tb_n \\ & \vdots & & \\ -b_n^Tb_1 & -b_n^Tb_2 & \dots & (n-1)\,b_n^Tb_n \end{pmatrix}$$

which can be written as: $A^TA = 2\,(nD - b_xb_x^T - b_yb_y^T)$ where b_x is the vector of the real coordinates and b_y is the vector of the imaginary coordinates of $(b_1, \dots, b_n)$, D is a diagonal matrix whose ith entry is $b_i^Tb_i = b_{xi}^2 + b_{yi}^2$, and $b_xb_x^T$ is defined as follows

$$b_xb_x^T = \begin{pmatrix} b_{x1}^2 & b_{x1}b_{x2} & \dots & b_{x1}b_{xn} \\ b_{x2}b_{x1} & b_{x2}^2 & \dots & b_{x2}b_{xn} \\ & \vdots & & \\ b_{xn}b_{x1} & b_{xn}b_{x2} & \dots & b_{xn}^2 \end{pmatrix}.$$

Matrix $b_yb_y^T$ can be written analogously. Then $r = (A^TA)^{-1}A^Tc = (A^TA)^{-1}\,2\,s = 1/2\,(nD - b_xb_x^T - b_yb_y^T)^{-1}\,2\,s = (nD - b_xb_x^T - b_yb_y^T)^{-1}s$. In this form r can be calculated (without inverting matrix $(nD - b_xb_x^T - b_yb_y^T)$ directly) using the following well-known lemma:

Lemma 1 *If matrix K is nonsingular and $(K^{-1}h)^Th \neq 1$, then*

$$(K - hh^T)^{-1} = K^{-1} + \frac{K^{-1}hh^TK^{-1}}{1 - (K^{-1}h)^Th} \qquad \square$$

We apply the formula given in Lemma 1 twice to invert matrix A^TA (We argue in Section 3.2 that the conditions of Lemma 1 are fulfilled.)

Since $r = 2(A^TA)^{-1}s = ((nD - b_xb_x^T) - b_yb_y^T)^{-1}s$, we can apply Lemma 1 to $K = (nD - b_xb_x^T)$ and $h = y$ and get

$$\begin{aligned} r &= ((nD - b_xb_x^T) - b_yb_y^T)^{-1}s \\ &= K^{-1}s + \frac{K^{-1}b_yb_y^TK^{-1}}{1 - (K^{-1}b_y)^Tb_y}s \\ &= K^{-1}s + \frac{(K^{-1}b_y)((K^{-1}b_y)^Ts)}{1 - (K^{-1}b_y)^Tb_y} \qquad (5) \end{aligned}$$

Applying Lemma 1 again, the first term $K^{-1}s = (nD - b_xb_x^T)^{-1}s$ can be expressed as

$$K^{-1}s = (nD)^{-1}s + \frac{(nD)^{-1}b_x((nD)^{-1}b_x)^Ts}{1 - ((nD)^{-1}b_x)^Tb_x} \qquad (6)$$

Vector $K^{-1}b_y = (nD - b_xb_x^T)^{-1}y$ in Equation (5) is calculated by applying the formula given in Lemma 1 which yields

$$K^{-1}b_y = (nD)^{-1}b_y + \frac{(nD)^{-1}b_x((nD)^{-1}b_x)^Tb_y}{1 - ((nD)^{-1}b_x)^Tb_x}. \qquad (7)$$

The inverse of a diagonal matrix is a matrix whose entries on the diagonal are inverted. Thus, the ith diagonal entry of $(nD)^{-1}$ is $1/(nb_i^Tb_i)$. We can now obtain the full expression for $r = (nD - b_xb_x^T - b_yb_y^T)^{-1}s$ by substituting $K^{-1}s$, $K^{-1}b_y$, and $(nD)^{-1}$ into Equation (5).

```
RATIOS-CALCULATOR($b_1, \ldots, b_n, s_1, \ldots, s_n$)
1   for $i = 1$ to $n$
2       $b_{xi}$ = real part of $b_i$
3       $b_{yi}$ = imaginary part of $b_i$
4       $diag_i = 1/(n(b_{xi}^2 + b_{yi}^2))$        $((nD)^{-1})$
    calculate vector $K^{-1}s$:
5       for $i = 1$ to $n$
6           $diagb_{xi} = diag_i * x_i$        $((nD)^{-1}b_x)$
7           $diags_i = diag_i * s_i$        $((nD)^{-1}s)$
8       calculate dot products $diagb_x.b_x$
            and $diagb_x.s$
9       calculate vector $(diagb_x.s)\,diagb_x$
10      use results of lines 8 and 9 to calculate
            $K^{-1}s$ with formula (6)
    calculate vector $K^{-1}b_y$:
11      for $i = 1$ to $n$
12          $diagb_{yi} = diag_i * b_{yi}$        $((nD)^{-1}b_y)$
13      calculate vector $(diagb_x.b_y)\,diagb_x$
14      use vectors $diagy$ and $(diagb_x.b_y)\,diagb_x$
            to calculate $K^{-1}b_y$ with formula (7)
    calculate vector $r$:
15      calculate vector $(K^{-1}b_y)((K^{-1}b_y).s)$ and
            scalar $(K^{-1}b_y).b_y$
16      use results of lines 10, and 15 to
            calculate $r$ with formula (5)
17  return $r = (r_1, \ldots, r_n)$
```

3.2 Correctness and analysis of the position estimation algorithm

In order to establish the correctness of procedure POSITION ESTIMATOR we can first prove that matrix $A^TA = 2(nD - b_xb_x^T - b_yb_y^T)$ is singular iff the landmarks and the robot's position are arranged in a perfect circle or on a line. This singularity condition is easily tested through some preprocessing that makes sure that the landmarks are not all arranged in a circle or on a line when procedure POSITION ESTIMATOR is called. In practice it is highly unlikely that the landmarks all lie on a perfect circle or line including the robot's position.

We can also show that if the landmarks are correctly identified and there is no noise in the angle measurements procedure POSITION ESTIMATOR returns the actual position of the robot.

Theorem 1 *Given a set of noisy measurements of landmarks which are not all arranged in a circle or line algorithm* POSITION ESTIMATOR *determines the position of the robot such that the squared error $||Ar - c||^2$ is minimized.* □

Theorem 2 *The algorithm* POSITION ESTIMATOR *runs in time linear in the number of landmarks.* □

3.3 Quality of POSITION ESTIMATOR

Consider the position estimates p_i that are calculated in line 11 of POSITION ESTIMATOR. The centroid p equals each estimate p_i only in a situation in which there is no noise in the measurements. Errors in the angle measurements result in errors in the position estimates p_i. Averaging these estimates p_i properly will give us an unbiased estimate p as long as the linearization is valid. We are currently working on the problem of finding optimal averaging coefficients [4]. In the implementation of our algorithm we use $1/n$ for the coefficients. We argue that the centroid obtained by uniform averaging produces an error that is small according to our simulation results. Our simulations also show that it is possible to find outlier estimates p_i which can be removed to improve the estimate for the centroid p.

In our experiments we calculated the length of the error vector in a scenario in which the robot is at the origin of our map and the landmarks are randomly distributed in a 10 meters by 10 meters area. The robot is at the corner of this area. The distance of the landmarks to the robot is at most 14.1 m and the angles are at most 45°. Note that this scenario is quite restricted, it uses only a quarter of the robot's surrounding area. We expect better results for scenarios in which the robot can take advantage of information provided by all of its surrounding.

Figure 2 shows the results of experiments in which we added an amount of noise to the angles which was a certain fraction of the true angle. The length $||p_{actual} - p||^2$ of the error vector between p_{actual} and the computed position p is $9.3cm$ for 1% angle noise, $37cm$ for 5% angle noise, and $1m$ for 10% angle noise in *every* landmark.

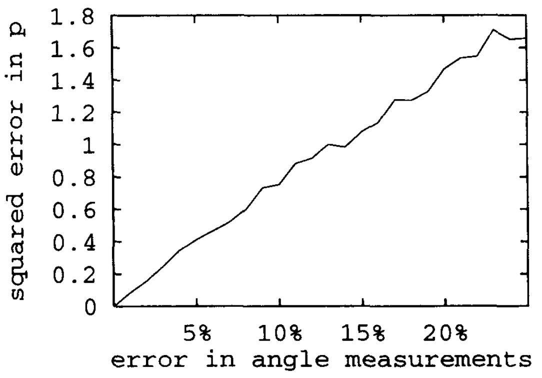

Figure 2: Simulation results using algorithm POSITION ESTIMATOR on an input of noisy angle measurements. The squared error in the position estimate p (in meters) is shown as a function of measurement errors (in percent of actual angle).

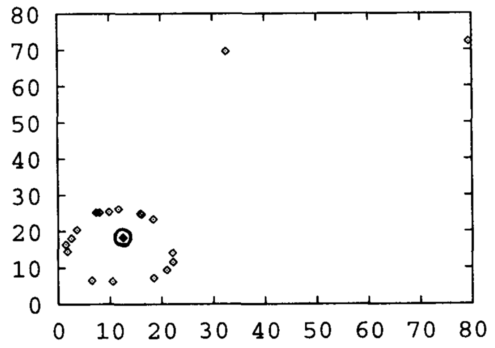

Figure 3: Two outliers landmarks result in bad position estimates at $(79\,cm, 72\,cm)$ and $(32\,cm, 69\,cm)$. The centroid calculated without the outliers is illustrated by a filled diamond. The position estimate after POSITION ESTIMATOR is called again after outlier landmarks are removed is at $(6.5\,cm, 6.5\,cm)$ (shown circled).

There may be some *outliers* that are not as close to the actual position. However, note that there is a one-to-one correspondence between each landmark z_i and estimate p_i. Each outlier p_i corresponds to either a misidentified landmark z_i or a noisy angle measurement for landmark z_i.

In the simulation shown in Figure 3 we used our algorithm to calculate position estimates $p_1, \ldots, p_{19}$ given 20 landmarks $z_0, \ldots, z_{19}$. The errors introduced can either result from a noisy measurement of an angle or from misidentified landmarks. We assumed that the angle measurement error is 10% of the actual angle for landmarks z_5 and z_{18} and 1% for the other landmarks. Our experiments show that landmarks z_5 and z_{18} produce position estimates p_5 and p_{18} that are far from the actual position of the robot compared to the other estimates. So instead of taking the centroid of all position estimates $p_1, \ldots, p_{19}$, we can obtain better results by modifying the algorithm such that the outliers p_5 and p_{18} are discarded and the centroid of the remaining points is calculated. Instead, we call algorithm POSITION ESTIMATOR on an input of 18 landmarks (without z_5 and z_{18}). Then the centroid is at $(6.5\,cm, 6.5\,cm)$. This is a considerable improvement over the original centroid which was at $(17\,cm, 24\,cm)$.

4 Robot navigation

The key advantages of our POSITION ESTIMATOR over the standard nonlinear approach described in the beginning of Section 3 is that (1) we *linearized* the problem of determining the robot position in a noisy environment, (2) the algorithm runs in time *linear* in the number of landmarks, (3) the algorithm provides a position estimate which is very close to the actual robot position, and (4) large errors in some measurements (i.e., outliers) can be found.

Our algorithm does not use information about the motion of the mobile robot: the history of position estimates, the commands that make the robot move, and the uncertainties in these commands. We are currently working on two methods to incorporate our POSITION ESTIMATOR into a navigation algorithm that keeps track of this information. One approach is *Kalman filtering*, the other is a *regularization method*. We are also working on the following questions:

Is it possible to obtain a similar algorithm for 3-dimensional representations of landmarks? For the 3-D case we propose to use quaternions (see for example [6]) instead of complex numbers.

Given a probability distribution on the measurement errors, what are the probabilistic bounds for the quality of our solution?

Acknowledgments

We would like to thank Russell Greiner, Tom Hancock, Stephen Judd, Nick Makris, Glenn Meredith, and Ron Rivest for helpful discussions.

References

[1] B. Crespi, C. Furlanello, and L. Stringa. Memory-based navigation. In *Proc. 13th Interl. J. Conf. on Artificial Intelligence*, pages 1654–1658, 1993.

[2] J. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proc. IEEE Interl. Conf. on Robotics and Automation*, pages 674–680, 1989.

[3] R. Greiner and R. Isukapalli. Learning useful landmarks. Technical report, Siemens Corporate Research, Princeton, 1993.

[4] Leonid Gurvits and Margrit Betke. Robot navigation using landmarks. In *preparation*, 1994.

[5] R. T. Hancock and J. S. Judd. Hallway navigation using simple visual correspondence algorithms. Technical Report SCR-94-479, Siemens Corporate Research, Princeton, 1993.

[6] B. K. P. Horn. *Robot Vision*. MIT Electrical Engineering and Computer Science Series. The MIT Press/McGraw-Hill, 1986.

[7] A. Kosaka and A. Kak. Fast vision-guided mobile robot navigation using model-based reasoning and rediction of uncertainties. *CVGIP: Image Understanding*, 56(3):271–329, 1992.

[8] E. Krotov. Mobile robot localization using a single image. In *Proc. IEEE Interl. Conf. on Robotics and Automation*, pages 978–983, 1989.

[9] B.J. Kuipers and T. S. Levitt. Navigation and mapping in large-scale space. *AI Magazine*, 9(2):25–43, 1988.

[10] T.S. Levitt, D. T. Lawton, D. M. Chelberg, K. V. Koitzsch, and J. W. Dye. Qualitative navigation II. In *Proc. DARPA Image Understanding Workshop*, pages 319–326, 1988.

[11] M. Mataric. A distributed model for mobil robot environment-learning and navigation. Technical Report AI-TR-1228, MIT, 1990.

[12] G. P. Roston and E. P. Krotkov. Dead reckoning navigation for walking robots. In *Proc. IEEE/RSJ Interl. Conf. on Intelligent Robots and Systems*, pages 607–612, 1992.

[13] K. T. Sutherland and W. B. Thompson. Inexact navigation. In *Proc. IEEE Interl. Conf. on Robotics and Automation*, pages 1–7, 1993.

[14] Y. Watanabe and S. Yuta. Position estimation of mobile robots with internal and external sensors using uncertainty evolution technique. In *Proc. IEEE Interl. Conf. on Robotics and Automation*, pages 2011–2016, 1990.