Mobile Robot Localization using Landmarks

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

Leonid Gurvits†
NEC Research Institute
4 Independence Way
Princeton, NJ 08540

April 27, 1995

Abstract

We describe an efficient method for localizing a mobile robot in an environment with landmarks. We assume that the robot can identify these landmarks and measure their bearings relative to each other. Given such noisy input, the algorithm estimates the robot's position and orientation with respect to the map of the environment. The algorithm makes efficient use of our representation of the landmarks by complex numbers. The algorithm runs in time *linear* in the number of landmarks. We present results of simulations and propose how to use our method for robot navigation.

Keywords: Robotics, mobile robot localization, landmark navigation, map algorithms, triangulation.

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

†Author's net address: `gurvits@research.nj.nec.com`

1 Introduction

We describe an efficient algorithm for localizing a mobile robot in an environment with landmarks. The robot has sensors that both identify landmarks and measure their bearings relative to each other. Such sensor information is generally uncertain and contains noise. Given the positions of possibly misidentified landmarks on a 2-D map of the environment and noisy measurements of their bearings relative to each other, the algorithm estimates the robot's position with respect to the map of the environment. The algorithm makes efficient use of the geometry of the problem; specifically, the representation of the landmarks by complex numbers. The algorithm runs in time linear in the number of landmarks. Results of simulations are presented that explore the strength of the algorithm.

Why is mobile robot localization important? A robot cannot accurately execute its commands. As a mobile robot moves through its environment, its actual position and orientation always differs from the position and orientation that it is commanded to hold. Wheel slippage is a major source of error. The errors accumulate and the location uncertainty increases over time. Dead-reckoning is not sufficient to locate the robot. Therefore, sensory feedback is needed to locate the robot in its environment.

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. 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 bearings of the landmarks relative to each other.
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 and orientation of the agent efficiently.
- Finding misidentified landmarks and large angle measurement errors.

This chapter presents an algorithm that estimates the agent's position in $O(n)$ operations where n is the number of landmarks on the 2-D map. 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 Department 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 (see Figure 1). Straight lines of objects like pictures, doors, and walls look like arcs in the images taken by Ratbot's camera. The camera setup is due to Judd [18]. 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 bearing of one landmark relative to another landmark, 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

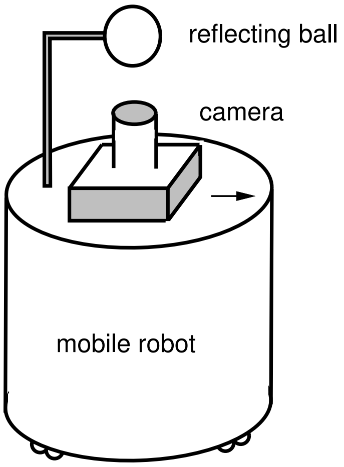

Figure 1: Illustration of a robot with a camera set-up that provides "fish-eyed" circular images of the surroundings.

and an image that is taken in advance and stored in a database, only the strips of the two images need to be compared. Therefore, the correspondence problem, i.e., the problem of identifying features in two images that are the projections of the same object in the environment, is much easier for pictures taken by Ratbot. Only one-dimensional strips of pixel values of two pictures need to be compared. Hancock and Judd provide an algorithm that finds matching features in two image strips [16].

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. The position of the robot with respect to this map is unknown and needs to be determined.

Our position estimation algorithm has been used by Greiner and Isukapalli [14] 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 camera setups. Our algorithm could be used not only for a robot moving in an office building, but also for a robot moving in an assembly plant, or for an autonomous vehicle in a city.

The paper is organized as follows. Section 2 describes work related to our problem. Section 3 gives a formal description of our problem. Section 4 discusses how the robot's position can be estimated using a triangulation approach. Section 5 describes our position estimation algorithm which uses a different approach that represents landmarks with complex numbers. Section 6 proposes how to incorporate our algorithm in an on-line navigation algorithm that estimates the robot's position while the robot is moving.

2 Related work

In recent years several authors have used triangulation approaches for mobile robot localization. Triangulation is a well-known technique for finding a position by means of bearings from fixed points in the environment.

Sugihara [30] addresses a version of the localization problem where the points in the environment look identical (to the contrary, we use landmarks that are distinguishable). The robot measures the direction of rays each of which emanates from the robots position and "pierces" through an observed point. Sugihara gives an $O(n^3 \lg n)$ algorithm for finding the robot's position and orientation such that each ray pierces at least one of the n points in the environment. Avis and Imai [2] and Krotkov [21] extend Sugihara's result to the case that angle measurements include errors.

Sutherland and Thompson [31] address the localization problem using *distinguishable* landmarks in the environment (as we do). They show that for a given error in angle measurement, the size of the localization error varies depending on the configuration of landmarks.

There is a wide variety of approaches to handle position uncertainties associated with a navigating robot. Some approaches differ from our approach in the way the robot's environment is represented, e.g. Brooks [3], Mataric [26], Elfes [12], and Levitt et al. [25, 24, 22].

Many authors use a statistical approach for robot navigation, for example, Chatila and Laumond [5], Smith and Cheeseman [28], Kosaka and Kak [20], Crowley [10], Leonard and Durrant-Whyte [23], Watanabe and Yuta [32], Burlina, DeMenthon and Davis [4] and Matthies and Shafer [27].

Approaches to handle uncertainties in a robot's position can also be distinguished by the kind of data available to the robot. Our method uses visual input. Other sensors that have been used for position estimation are sonar sensors [11], odometry [6], GPS [8], and ultrasonic beacons [19].

We address mobile robot localization as a problem of relating robot-centered information with global map information. Relating a camera-centered coordinate system to an external coordinate system is called *exterior orientation* in the field of photogrammetry (see for example [17]). Photogrammetry deals with the problem of how objects in the environment are related to their images in camera systems with different orientations and positions. For robot manipulators, the problem of relating a camera-centered coordinate system to a world-centered coordinate system is called the *hand-eye transformation.* The camera information (eye) is used to guide a robot arm or a mobile robot (hand) [9].

3 The position estimation problem

In this section we define the problem of estimating a robot's position and orientation in its environment given a global map of the environment and bearings of landmarks measured relative to each other at the robot's position.

We call the coordinate system of the map of the environment the *external* coordinate system. It is spanned by axes $x^{(e)}$ and $y^{(e)}$. We distinguish it from the *robot-centered* coordinate system spanned by axes $x^{(r)}$ and $y^{(r)}$. A landmark z can be described in both coordinate systems. Vector $z^{(e)}$ describes a landmark in the external coordinate system (see Figure 2, left). Vector $z^{(r)}$ describes a landmark in the robot-centered coordinate system (see Figure 2, right).

The robot's *position* is described by vector $p = (p_x, p_y)$ in the external coordinate system. Vector p links the origins of both coordinate systems. The robot is oriented in direction of axis $x^{(r)}$. The robot's *orientation* is described by angle θ between the external coordinate axis $x^{(e)}$ and the robot-centered coordinate axis $x^{(r)}$. Figure 3 illustrates how an external and a robot-centered coordinate system are linked given position vector p and orientation angle θ.

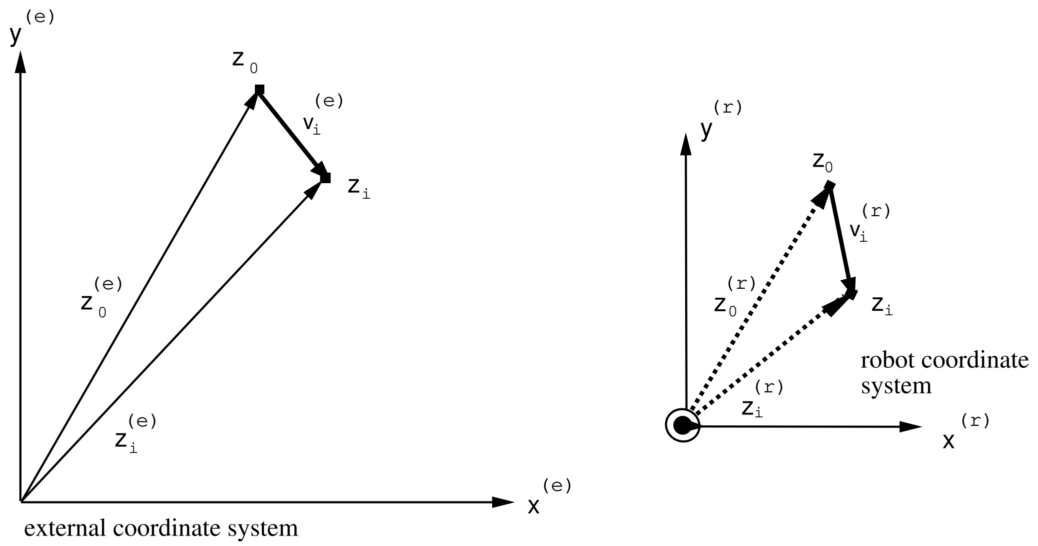

Figure 2: On the left: Environment with external coordinate system $(x^{(e)}, y^{(e)})$. Two landmarks, z_0 and z_i, are shown. On the right: Environment with robot coordinate system $(x^{(r)}, y^{(r)})$. The robot's position is illustrated by a black circled disk. The robot is orientated in the direction of the coordinate axis $x^{(r)}$.

Having a map of the robot's environment means that the vectors $z_0^{(e)}, \ldots z_n^{(e)}$ in the external coordinate system are given. The robot measures the bearing of each landmark relative to the direction in which the robot is oriented: Angle τ_i is the angle subtended by landmark vector $z_i^{(r)}$ and axis $x^{(r)}$ for $i = 0, \ldots, n$:

$$\tau_i = \angle(z_i^{(r)}, x^{(r)}).$$

We call the angle obtained from measuring the bearing of one landmark relative to another landmark the "visual angle" between the two landmarks [31].

The robot does not have a way to measure depth, i.e., it does not know its distance $|z_i^{(r)}|$ to the i-th landmark z_i for all i.

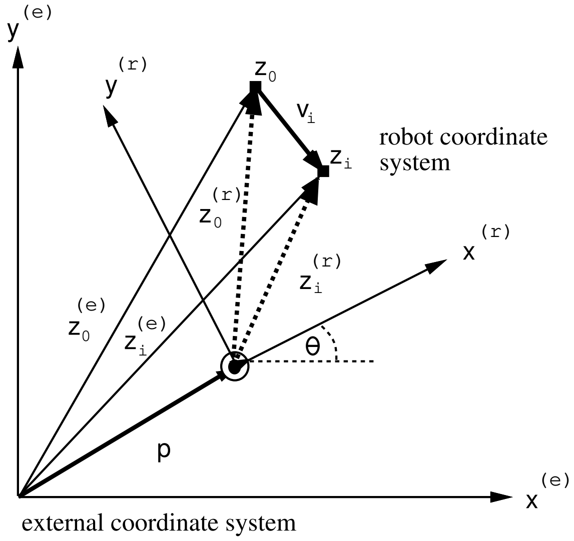

Figure 3: Environment with external and robot coordinate system: Both vector $z_i^{(e)}$ in the external coordinate system and vector $z_i^{(r)}$ in the robot coordinate system point to landmark z_i. The robot's location is defined by position vector p and orientation angle θ.

Note that if the orientation angle θ is zero then we can express the vector between two landmarks z_0 and z_i as vector $v_i^{(r)} = z_0^{(r)} - z_i^{(r)} = v_i^{(e)} = z_0^{(e)} - z_i^{(e)}$ (see Figure 3).

Now we can formulate our problem: Given the external positions $z_0^{(e)}, \ldots, z_n^{(e)}$ of $n+1$ landmarks and corresponding angle measurements $\tau_0, \ldots \tau_n$, estimate the position p and the orientation θ of the robot in the environment.

4 A triangulation approach

This section shows how to apply well-known triangulation techniques to the robot localization problem. First we consider the localization problem for perfect data, then for noisy data.

4.1 Triangulation with perfect data

If the visual angle φ between two landmarks z_0 and z_1 measured at an unknown position p and the distance between the landmarks is known, then position p lies on an arc of a circle spanned by landmarks z_0 and z_1 [13]. (See Figure 4 (a).)

A third landmark z_2 is needed to determine position p uniquely: it is the intersection of the circle through landmarks z_0 and z_1 and the circle through landmarks z_1 and z_2 (see Figure 4 (b)). Note that p cannot be determined uniquely if landmarks z_0, z_1 and z_2 lie on a circle.

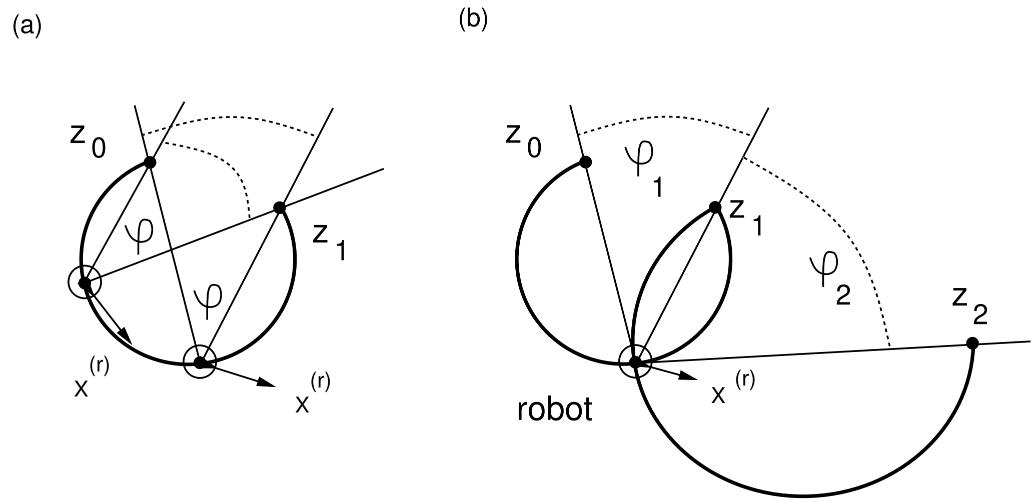

Figure 4: Triangulation with perfect data: (a) Scenario with two landmarks: The robot's position is somewhere on the arc shown. The robot is oriented in the direction of $x^{(r)}$. (b) Scenario with three landmarks: The robot's position is at the point of intersection of the two arcs shown. The robot's orientation can be uniquely determined from $x^{(r)}$.

One approach to compute position p is to describe the two circles analytically and determine their intersection. The law of cosine can be used to compute the radius of each circle given the visual angles and distances between the landmarks. The coordinates of the center of the circles can then be computed from the coordinates of landmarks z_0, z_1 and z_2. Once vector p is known, the orientation of the robot, i.e., the direction of $x^{(r)}$ is known.

Another way of triangulating position p is to determine the distance between the robot and the three landmarks. We follow this approach in the next section.

4.2 Triangulation with noisy data

The triangulation method described above does not compute the exact position if the input data is noisy. Using several landmarks yields a more robust position estimate. Given n landmarks, there are $\binom{n}{3}$ combinations of three landmarks that can be used to compute $\binom{n}{3}$ position estimates by the triangulation method described above. One way of combining these estimates to obtain a final position estimate is to compute their average.

In the following we outline a different triangulation method that uses n landmarks and is based on estimating the distance between the robot and the landmarks. (Note that this is not the method that we propose).

Applying the law of cosine the relationship between two landmarks z_i and z_j can be written as

$$|v_{ij}|^2 = |z_i^{(r)}|^2 + |z_j^{(r)}|^2 - 2|z_i^{(r)}||z_j^{(r)}| \cos\varphi_{ij}$$

where $|v_{ij}|$ is the (known) distance between z_i and z_j, φ_{ij} is the visual angle between z_i and z_j at the robot's position, and $|z_i^{(r)}|$ and $|z_j^{(r)}|$ are the unknown distances to the landmarks (see Figure 5). Angle φ_{ij} is computed from measured angles τ_i and τ_j: $\varphi_{ij} = \tau_i - \tau_j$.

We can apply the law of cosine to all possible sets of two landmarks and get a system of equations which is overdetermined and can be solved using a least squares approach. Once the lengths $|z_i^{(r)}|, \ldots, |z_n^{(r)}|$ are found, another overdetermined set of equations needs to be solved to find the position $p = (p_x, p_y)$ of the robot: $|z_i^{(r)}|^2 = (x_i - p_x)^2 + (y_i - p_y)^2$ for $i = 0, \ldots, n$ where x_i and y_i are the coordinates of $z_i^{(r)}$. Once vector p is estimated, we can get an estimate for the orientation angle θ by computing vector $z_i^{(r)} = z_i^{(e)} - p$ for some i and then $\theta = \angle(z_i^{(r)}, x^{(e)}) - \tau_i$.

In the following we illustrate the method with three landmarks z_0, z_i and z_j (see also Figure 5). The law of cosine gives us

$$|v_{0i}|^2 \quad = \quad |z_0^{(r)}|^2 + |z_i^{(r)}|^2 - 2|z_0^{(r)}||z_i^{(r)}| \cos\varphi_{i0}$$

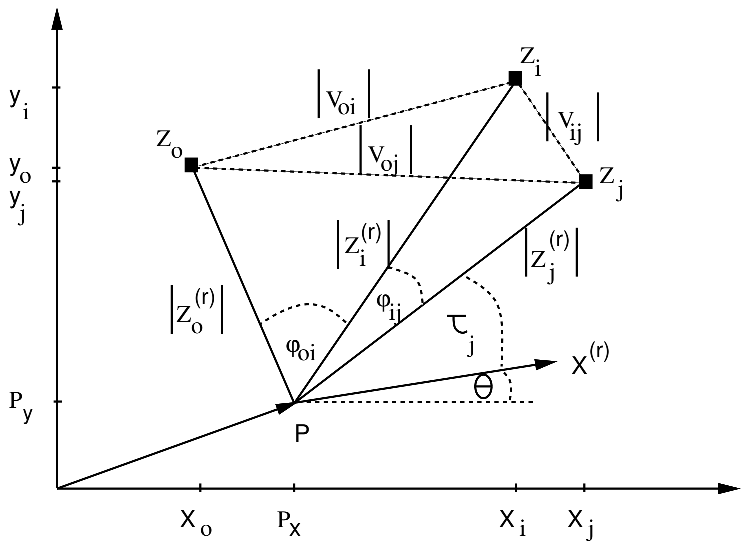

Figure 5: Position estimation using the law of cosine.

$$
\begin{aligned}
|v_{0j}|^2 &= |z_0^{(r)}|^2 + |z_j^{(r)}|^2 - 2|z_0^{(r)}||z_j^{(r)}| \cos\varphi_{j0} \\
|v_{ij}|^2 &= |z_i^{(r)}|^2 + |z_j^{(r)}|^2 - 2|z_i^{(r)}||z_j^{(r)}| \cos\varphi_{ij}
\end{aligned}
$$

First we could solve this system of nonlinear equations using a least squares method. (A review of the least squares method for solving linear equations is given in Section 5.1.) Then we could express the now known lengths $|z_0^{(r)}|, |z_i^{(r)}|$ and $|z_j^{(r)}|$ by the (unknown) coordinates of the position $p = (p_x, p_y)$ of the robot and the coordinates (x, y) of the landmarks:

$$
\begin{aligned}
|z_0^{(r)}|^2 &= (x_0 - p_x)^2 + (y_0 - p_y)^2 \\
|z_i^{(r)}|^2 &= (x_i - p_x)^2 + (y_i - p_y)^2 \\
|z_j^{(r)}|^2 &= (x_j - p_x)^2 + (y_j - p_y)^2
\end{aligned}
$$

Subtracting the second and third equation from the first we get two equations linear in the unknowns p_x and p_y.

$$
\begin{aligned}
|z_0^{(r)}|^2 - |z_i^{(r)}|^2 &= x_0^2 - x_i^2 + 2p_x(x_i - x_0) + y_0^2 - y_i^2 + 2p_y(y_i - y_0) \\
|z_0^{(r)}|^2 - |z_j^{(r)}|^2 &= x_0^2 - x_j^2 + 2p_x(x_j - x_0) + y_0^2 - y_j^2 + 2p_y(y_j - y_0)
\end{aligned}
$$

Finally, we can solve for p_x and p_y and obtain an estimate for the position $p = (p_x, p_y)$.

Triangulation with noisy data as described above is based on solving nonlinear equations with complicated closed-form solutions. However, standard algorithms that provide least-squares solutions for large numbers of nonlinear equations take too long for real-time robot navigation. In the following section, we introduce a new *linear-time* approach to localizing a mobile robot which is based on a least-squares solution of a linear set of equations.

5 Linear position estimation

This section describes an efficient localization algorithm that runs in time linear in the number of landmarks. Instead of using the triangulation method described above, we use a method based on the complex number representation of the landmarks. This representation is the key idea to get a set of linear equations whose solution is a set of position estimates. (Triangulation methods may also provide a set of position estimates but as a solution to nonlinear equations). We describe this set of m linear equations as a vector equation. To solve this particular vector equation, 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 vector equation. In the following, we describe how to set up and solve this vector equation.

A landmark z can be written as a complex number in the robot-centered coordinate system. For each landmark $z_i^{(r)}$ we have an expression

$$z_i^{(r)} = l_i \, e^{\,j\tau_i} \qquad\qquad \text{for } \; i = 1, \ldots, n$$

where length l_i is the unknown distance of the robot to landmark $z_i^{(r)}$, angle τ_i is the measured angle between $z_i^{(r)}$ and the coordinate axis $x^{(r)}$, and $j = \sqrt{-1}$.

We pick a reference landmark $z_0^{(r)}$ and compute the angles $\varphi_i = \tau_i - \tau_0$ for $i = 1, \ldots, n$. (Landmark $z_0^{(r)}$ may be chosen to be the most reliable landmark if such reliability information is available.) Dividing the complex number representation of landmarks $z_1^{(r)}, \ldots, z_n^{(r)}$ by $z_0^{(r)}$ for $i = 1, \ldots, n$ yields a set of equations that includes the visual angles φ_i between landmark z_i and landmark z_0:

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

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

$$\frac{z_0^{(r)} + v_i^{(r)}}{z_0^{(r)}} = \frac{l_i}{l_0}\, e^{\,j\varphi_i} \qquad \text{for } i = 1, \ldots, n.$$

After some algebra we obtain a set of equations whose unknowns are vectors $z_0^{(r)}$ and $v_i^{(r)}$ and ratios l_i/l_0, for $i = 1, \ldots, n$:

$$\frac{1}{z_0^{(r)}} = \frac{l_i}{l_0}\frac{1}{v_i^{(r)}}\, e^{\,j\varphi_i} - \frac{1}{v_i^{(r)}} \qquad \text{for } i = 1, \ldots, n. \tag{2}$$

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:

$$\frac{l_k}{l_0}\frac{1}{v_k^{(r)}} e^{j\varphi_k} - \frac{1}{v_k^{(r)}} = \frac{l_i}{l_0}\frac{1}{v_i^{(r)}} e^{j\varphi_i} - \frac{1}{v_i^{(r)}} \qquad \text{for } i, k = 1, \ldots, n, \quad \text{and } k \neq i. \tag{3}$$

The only unknowns in Equation (3) are vectors $v_i^{(r)}$ and ratios l_i/l_0. Since the angles $\varphi_1, \ldots, \varphi_n$ in Equation (3) are independent of the orientation of the robot-centered coordinate system, we can rewrite Equation (3) for a robot-centered coordinate system with a different orientation:

$$r_k \frac{1}{\hat{v}_k^{(r)}} e^{j\varphi_k} - \frac{1}{\hat{v}_k^{(r)}} = r_i \frac{1}{\hat{v}_i^{(r)}} e^{j\varphi_i} - \frac{1}{\hat{v}_i^{(r)}} \qquad \text{for } i, k = 1, \ldots, n, \quad \text{and } k \neq i \tag{4}$$

where landmark z_i is described by vector $\hat{z}_i^{(r)}$, $r_i = |\hat{z}_i^{(r)}|/|\hat{z}_0^{(r)}|$ and $\hat{v}_i^{(r)} = \hat{z}_i^{(r)} - \hat{z}_0^{(r)}$. We choose the robot-centered coordinate system that is oriented the same way as the external coordinate system. Thus, its axes $\hat{x}^{(r)}$ and $\hat{y}^{(r)}$ are parallel to axes $x^{(e)}$ and $y^{(e)}$, respectively (see Figure 6). We pick this particular coordinate system, because we can then write

$$\hat{v}_i^{(r)} = v_i^{(e)} = z_i^{(e)} - z_0^{(e)}$$

for $i = 1, \ldots, n$ which yields equations

$$r_k \frac{1}{v_k^{(e)}} e^{j\varphi_k} - \frac{1}{v_k^{(e)}} = r_i \frac{1}{v_i^{(e)}} e^{j\varphi_i} - \frac{1}{v_i^{(e)}} \qquad \text{for } i, k = 1, \ldots, n, \quad \text{and } k \neq i \tag{5}$$

whose only unknowns are ratios $r_1, \ldots, r_n$.

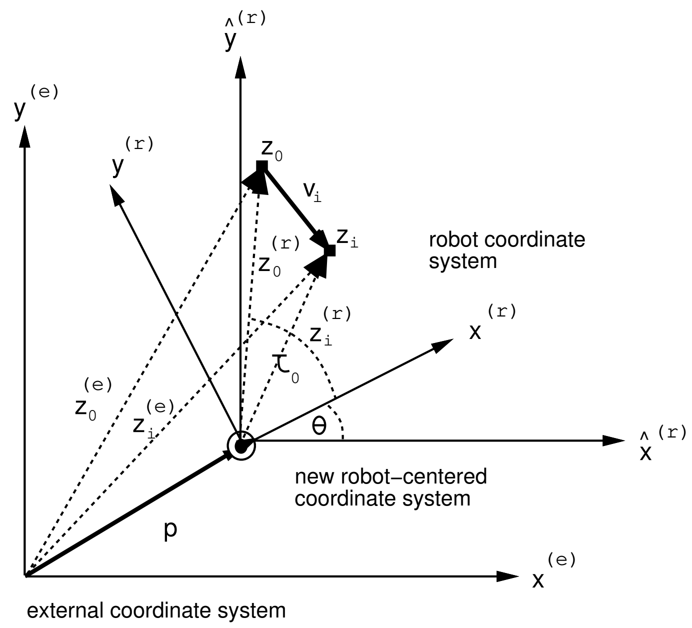

Figure 6: External coordinate system spanned by $x^{(e)}$ and $y^{(e)}$, robot-centered coordinate system spanned by $x^{(r)}$ and $y^{(r)}$ and robot-centered coordinate system spanned by $\hat{x}^{(r)}$ and $\hat{y}^{(r)}$. The robot is facing in direction of axis $x^{(r)}$.

Note that using the robot-centered coordinate system that is spanned by axes $\hat{x}^{(r)}$ and $\hat{y}^{(r)}$ does **not** imply that the robot knows its orientation in the environment a priori. It does not know its orientation angle $\theta = \angle(x^{(r)}, x^{(e)}) = \angle(x^{(r)}, \hat{x}^{(r)})$. Instead, once equations (5) are solved we can compute vector $\hat{z}_0^{(r)}$ which can then be used to compute the robot's orientation

$$\theta = \angle(\hat{z}_0^{(r)}, \hat{x}^{(r)}) - \tau_0 \tag{6}$$

as illustrated in Figure 6.

The set of equations (5) can be transformed into a matrix equation $A\,r = c$ where r, c, and A can be defined as follows: Vector $r = (r_1, \ldots, r_n)$ is the vector of the unknown ratios of the length of vectors $\hat{z}_1^{(r)}, \ldots, \hat{z}_n^{(r)}$. Vector c is a $n(n-1)$ dimensional vector of differences of complex numbers $c_i = (c_{xi}, c_{yi}) = 1/\hat{v}_i^{(r)}$ and matrix A is a $n(n-1) \times n$ matrix consisting of complex numbers $b_i = (b_{xi}, b_{yi}) = c_i\, e^{\,j\varphi_i}$:

$$A = \begin{pmatrix} b_1 & -b_2 & 0 & 0 & \ldots & 0 & 0 \\ b_1 & 0 & -b_3 & 0 & \ldots & 0 & 0 \\ b_1 & 0 & 0 & -b_4 & \ldots & 0 & 0 \\ \vdots & & & & & & \\ b_1 & 0 & 0 & 0 & \ldots & 0 & -b_n \\ -b_1 & b_2 & 0 & 0 & \ldots & 0 & 0 \\ 0 & b_2 & -b_3 & 0 & \ldots & 0 & 0 \\ 0 & b_2 & 0 & -b_4 & \ldots & 0 & 0 \\ \vdots & & & & & & \\ 0 & b_2 & 0 & 0 & \ldots & 0 & -b_n \\ \vdots & & & & & & \\ -b_1 & 0 & 0 & 0 & \ldots & 0 & b_n \\ 0 & -b_2 & 0 & 0 & \ldots & 0 & b_n \\ 0 & 0 & -b_3 & 0 & \ldots & 0 & b_n \\ \vdots & & & & & & \\ 0 & 0 & 0 & 0 & \ldots & -b_{n-1} & b_n \end{pmatrix} \quad \text{and} \quad c = \begin{pmatrix} c_1 - c_2 \\ c_1 - c_3 \\ c_1 - c_4 \\ \vdots \\ c_1 - c_n \\ c_2 - c_1 \\ c_2 - c_3 \\ c_2 - c_4 \\ \vdots \\ c_2 - c_n \\ \vdots \\ c_n - c_1 \\ c_n - c_2 \\ c_n - c_3 \\ \vdots \\ c_n - c_{n-1} \end{pmatrix}$$

A solution of equation $A\,r = c$ yields a vector r whose components r_i can be used to solve equation

$$\frac{1}{\hat{z}_0^{(r)}} = r_i \frac{1}{v_i^{(e)}}\, e^{\,j\varphi_i} - \frac{1}{v_i^{(e)}} \tag{7}$$

for some i to obtain vector $\hat{z}_0^{(r)}$. (This equation corresponds to Equation (2) and describes the vectors with respect to the robot-centered coordinate system spanned by axes $\hat{x}^{(r)}$ and $\hat{y}^{(r)}$.) Once $\hat{z}_0^{(r)}$ is known, the robot position p is

$$p = z_0^{(e)} - \hat{z}_0^{(r)}.$$

Note that the system of linear equations $A\,r = c$ is overdetermined. We have n unknowns $r_1, \ldots, r_n$ and $n(n-1)$ equations. These equations may be inconsistent, since we have measurement errors. Therefore, we use a “least squares method” to find a solution for which the average error in the $n(n-1)$ equations is minimized.

5.1 Least squares errors and the pseudo inverse

The "least squares method" is a standard method for solving overdetermined systems of equations [29]. In the following we rederive this method for our particular overdetermined system of equations.

To solve the matrix equation $A\,r = c$ we choose ratio vector r so as to minimize the "average" error in the $n(n-1)$ equations. 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 solve the minimization problem

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

The necessary condition of the squared error E^2 to be minimal is that its derivative with respect to vector r must be zero. The derivative is

$$\begin{aligned}
\frac{d}{dr}E^2 &= \frac{d}{dr}||A\,r - c||^2 \\
&= \frac{d}{dr}(A\,r - c)^T(A\,r - c) \\
&= \frac{d}{dr}(r^T A^T A r - r^T A^T c - c^T A r + c^T c) \\
&= A^T A r + (A^T A)^T r - 2A^T c \\
&= 2\,A^T A r - 2A^T c
\end{aligned}$$

where A^T is the conjugate transpose of A. Setting the derivative to zero gives us

$$\begin{aligned}
0 &= 2A^T A r - 2A^T c \\
A^T A r &= A^T c \\
r &= (A^T A)^{-1} A^T c,
\end{aligned}$$

provided matrix $A^T A$ is nonsingular. In Section 5.3 we give the necessary and sufficient conditions for $A^T A$ being nonsingular and argue that they will almost always be fulfilled in practice.

Matrix $A^+ = (A^T A)^{-1} A^T$ is often called the *pseudo inverse* of matrix A (see for example [1, 29]). In the following section we describe an algorithm called Position estimator which efficiently calculates vector $r = A^+ c = (A^T A)^{-1} A^T c$ such that the average error in the equations is minimized.

5.2 The position estimation algorithm

In this section we describe procedure Position estimator that estimates the position and the orientation of the robot. In the 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

$$ab = (Re(a)Re(b) - Im(a)Im(b),\ Re(a)Im(b) + Re(b)Im(a))$$

and the inner product

$$\bar{a}^T b = Re(a)Re(b) + Im(a)Im(b)$$

where $\bar{a} = (Re(a), -Im(a))$ is the conjugate of a. For notational convenience we write the inner product of complex numbers a and b as $a^T b$.

Procedure Position estimator takes as an input the positions of the landmarks $z_0^{(e)}, \ldots, z_n^{(e)}$ (as given in the external coordinate system) and the angles $\varphi_1, \ldots \varphi_n$.

After the initialization of vectors v, c, b and $\sum_{i=1}^n c_i$ in lines 2–6, Position estimator first computes vector $s = \frac{1}{2} A^T c$ in lines 7 and 8 as follows: For each component of vector $s = \frac{1}{2} A^T c$ the procedure computes

$$\begin{aligned} s_i &= (n-1) b_i^T c_i - \sum_{j, j \neq i} b_i^T c_j \\ &= n b_i^T c_i - \sum_{j=1}^{n} b_i^T c_j \\ &= n b_i^T c_i - b_i^T \sum_{j=1}^{n} c_j \end{aligned}$$

Position-estimator($z_0^{(e)}, \ldots, z_n^{(e)}, \varphi_1, \ldots \varphi_n$)

1. $\textit{sum-of-}c_i = 0$
2. **for** $i = 1$ **to** n (initialization)
3. $\quad v_i = z_0^{(e)} - z_i^{(e)}$
4. $\quad c_i = 1/v_i$
5. $\quad \textit{sum-of-}c_i = \textit{sum-of-}c_i + c_i$ (calculates $\sum_{i=1}^{n} c_i$)
6. $\quad b_i = c_i e^{j\varphi_i}$
7. **for** $i = 1$ **to** n (calculates $\frac{1}{2}A^T c$)
8. $\quad s_i = n b_i^T c_i - b_i^T \ \textit{sum-of-}c_i$
9. Ratios-calculator($b_1, \ldots, b_n, s_1, \ldots, s_n$) (returns r)
10. **for** $i = 1$ **to** n
11. $\quad \hat{z}_{0,i}^{(r)} = 1/(r_i b_i - c_i)$
12. $\quad p_i = z_0^{(e)} - \hat{z}_{0,i}^{(r)}$
13. $p = \frac{1}{n}\sum_{i=1}^{n} p_i$ (robot position p)
14. $\theta = \angle(\hat{z}_0^{(r)}, \hat{x}^{(r)}) - \tau_0$

$$= \quad n(b_{xi}c_{xi} + b_{xi}c_{xi}) - (b_{xi}\sum_{j=1}^{n} c_{xj} + b_{yi}\sum_{j=1}^{n} c_{yj}). \tag{8}$$

Procedure Position estimator then determines $r = 2(A^T A)^{-1}s$ by calling procedure Ratios-calculator in line 9. It next calculates n position estimates $p_1, \ldots, p_n$ (lines 10-12).

In line 11 of procedure Position estimator, a set of n solutions $\hat{z}_{0,1}^{(r)} \ldots, \hat{z}_{0,n}^{(r)}$ for the position of landmark $\hat{z}_0^{(r)}$ in the robot-centered coordinate system is calculated using equation (7):

$$\hat{z}_{0,i}^{(r)} = \frac{1}{r_i b_i - c_i} \quad \text{for } i = 1, \ldots, n.$$

In line 13 of procedure Position estimator, a set of estimates $p_1, \ldots, p_n$ 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 $\hat{z}_{0,i}^{(r)}$ 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 $p_1, \ldots, p_n$ as an average to obtain an estimate of the position p of the robot (line 13 of procedure POSITION ESTIMATOR)[1]. In line 14 the orientation angle is computed using Equation (6).

Procedure RATIOS-CALCULATOR takes as an input the vectors b and s and returns the vector $r = 2(A^TA)^{-1}s$. First RATIOS-CALCULATOR calculates vectors $b_x = (b_{x1}, \ldots, b_{xn})$ and $b_y = (b_{y1}, \ldots, b_{yn})$ where b_{xi} is the real coordinate of b_i and b_{yi} is the imaginary coordinate of b_i. Then it exploits the special form of matrix A^TA:

$$A^TA = 2\begin{pmatrix} (n{-}1)\, b_1^Tb_1 & -b_1^Tb_2 & -b_1^Tb_3 & \ldots & -b_1^Tb_n \\ -b_2^Tb_1 & (n{-}1)\, b_2^Tb_2 & -b_2^Tb_3 & \ldots & -b_2^Tb_n \\ \vdots & & & & \\ -b_n^Tb_1 & -b_n^Tb_2 & -b_n^Tb_3 & \ldots & (n{-}1)\, b_n^Tb_n \end{pmatrix}$$

Instead of inverting the matrix A^TA directly, we write matrix A^TA as a sum of a diagonal matrix and two matrices that are outer products (or Grammian matrices, see Albert [1]). In this special form, the matrix A^TA can be inverted efficiently:

$$A^TA = 2\,(nD - b_xb_x^T - b_yb_y^T)$$

where D is a diagonal matrix whose ith entry is $b_i^Tb_i = b_{xi}^2 + b_{yi}^2$, and the outer product $b_xb_x^T$ is defined as follows

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

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 $nD - b_xb_x^T - b_yb_y^T$ directly) using the following well-known lemma [1]:

[1]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 [15]. They may not be the same for the x- and y-coordinates of p_i.

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

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

Proof: The proof follows directly by verifying equation

$$I = (K - hh^T)(K^{-1} + \frac{K^{-1}hh^T K^{-1}}{1-(K^{-1}h)^T h})$$

where I is the identity matrix. □

We apply the formula given in Lemma 1 twice to invert matrix $A^T A$ (We show in Section 5.3 that the requirements of Lemma 1, i.e., matrix K is nonsingular and $(K^{-1}h)^T h \neq 1$, are indeed met.)

Since $r = (A^T A)^{-1} A^T c = ((nD - b_x b_x^T) - b_y b_y^T)^{-1} s$, we can apply Lemma 1 to $K = (nD - b_x b_x^T)$ and $h = b_y$ and get

$$\begin{aligned} r &= ((nD - b_x b_x^T) - b_y b_y^T)^{-1} s \\ &= K^{-1}s + \frac{K^{-1} b_y b_y^T K^{-1}}{1-(K^{-1}b_y)^T b_y} s \\ &= K^{-1}s + \frac{(K^{-1}b_y)((K^{-1}b_y)^T s)}{1-(K^{-1}b_y)^T b_y} \end{aligned} \tag{9}$$

Applying Lemma 1 again, the first term $K^{-1}s$ can be expressed as

$$K^{-1}s = (nD - b_x b_x^T)^{-1} s = (nD)^{-1}s + \frac{(nD)^{-1} b_x ((nD)^{-1} b_x)^T s}{1-((nD)^{-1}b_x)^T b_x}, \tag{10}$$

Vector $K^{-1}b_y = (nD - b_x b_x^T)^{-1} b_y$ in Equation (9) is calculated by applying the formula given in Lemma 1 which yields

$$K^{-1}b_y = (nD)^{-1}b_y + \frac{(nD)^{-1} b_x ((nD)^{-1} b_x)^T b_y}{1-((nD)^{-1}b_x)^T b_x}. \tag{11}$$

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 $1/(nb_i^T b_i) = 1/(n(b_{xi}^2 + b_{yi}^2))$.

We can now obtain the full expression for $r = (A^T A)^{-1} A^T c = (nD - b_x b_x^T - b_y b_y^T)^{-1} s$

by substituting $K^{-1}s$, $K^{-1}b_y$, and $(nD)^{-1}$ into Equation (9).

If the i-th landmark is very close to the robot's position, the least squares method described above may yield a negative r_i. In this case ratio r_i should be positive, but almost zero. Thus, we assign it to be zero in procedure RATIOS-CALCULATOR.

```
RATIOS-CALCULATOR(b_1, ..., b_n, s_1, ..., s_n)
      compute diagonal matrix (nD)^{-1}
      compute vector K^{-1}s using Equation (10)
      compute vector K^{-1}b_y using Equation (11)
      compute vector r = (r_i, ..., r_n) using Equation (9)
      for i = 1 to n
          if r_i < 0 then r_i <- 0
```

5.3 Correctness and analysis of POSITION ESTIMATOR

In order to establish the correctness of procedure POSITION ESTIMATOR we first need to prove the following lemmas. We use the notation $D = diag(d_i)$ to denote a diagonal matrix D whose ith diagonal element is d_i. Given some matrix M we write $M > 0$ if M is positive definite and $M \geq 0$ if M is positive semidefinite.

Lemma 2 *Let matrix $D = diag(b_{xi}^2 + b_{yi}^2)$, $b_x^T = (b_{x1}, \ldots, b_{xn}$, and $b_y^T = (b_{y1}, \ldots, b_{yn}$. Matrix $A^TA = 2(nD - b_x b_x^T - b_y b_y^T)$ is nonsingular if and only if the vectors b_x and b_y are linearly independent.*

Proof: Since $A^TA \geq 0$, we know that A^TA is singular if and only if there exists a vector z, $z \neq 0$ such that $z^TA^TAz = 0$ or equivalently

$$
\begin{aligned}
z^T(nD - b_x b_x^T - b_y b_y^T)z &= 0 \\
z^T nDz &= z^T b_x b_x^T z + z^T b_y b_y^T z \\
nz^T diag(b_{xi}^2 + b_{yi}^2)z &= (z^T b_x)(b_x^T z) + (z^T b_y)(b_y^T z) \\
n\sum_{i=1}^{n}(b_{xi}^2 + b_{yi}^2)z_i^2 &= (\sum_{i=1}^{n} b_{xi}z_i)^2 + (\sum_{i=1}^{n} b_{yi}z_i)^2.
\end{aligned}
$$

Note that for any numbers $a_1, \ldots, a_n$ we know that $(\sum_{i=1}^n a_i)^2 \leq n \sum_{i=1}^n a_i^2$ and the equality holds if and only if $a_i = a_j$ for all indices i and j. Therefore, $A^T A$ is singular if and only if

$$b_{xi} z_i = b_{xj} z_j \quad \text{and} \quad b_{yi} z_i = b_{yj} z_j$$

for all indices i and j. This means that $b_x = \alpha z$ and $b_y = \beta z$ for some constants α and β. Thus we get that $b_x = (\alpha/\beta)\, b_y$ which means that b_x and b_y are linearly dependent. $\square$

The following lemma gives the geometrical interpretation of Lemma 2.

Lemma 3 *Matrix $A^T A = 2(nD - b_x b_x^T - b_y b_y^T)$ is singular if and only if the landmarks and the robot's position are arranged in a perfect circle or on a line.*

Proof: According to Lemma 2 we have to find a geometric criterion for vectors b_x and b_y to be linearly dependent. The rank of the $n \times 2$ matrix $b_x b_y$ whose two columns consist of vectors b_x and b_y is one if vectors b_x and b_y are collinear. Note that

$$rank(b_x b_y) = rank \begin{pmatrix} b_x^T \\ b_y^T \end{pmatrix}$$

where $\binom{b_x^T}{b_y^T}$ is the matrix whose rows are vectors b_x^T and b_y^T. By definition (see Section 5.2) we have

$$\begin{pmatrix} b_x^T \\ b_y^T \end{pmatrix} = \begin{pmatrix} b_{x1} \ldots b_{xn} \\ b_{y1} \ldots b_{yn} \end{pmatrix} = (b_1 \ldots b_n)$$

since $b_k = (b_{xk}, b_{yk})^T$, for $1 \leq k \leq n$. Therefore, it is sufficient for the proof to find the geometrical interpretation of the collinearity of the 2-D vectors b_k, for $1 \leq k \leq n$. That is, we need a geometrical interpretation for the existence of a complex number η such that $b_k = \alpha_k \eta$, where α_k is a real number.

We also have

$$b_k = c_k e^{j\varphi_k} = \frac{e^{j\varphi_k}}{v_k} = \frac{e^{j\varphi_k}}{z_k - z_0}$$

where vector v_k connects landmark z_0 with the landmarks z_k and φ_k is the visual angle between z_0 and z_k from the robot's position for $1 \leq k \leq n$ as illustrated in

Figure 7. (Note that $b_k = (b_{xk}, b_{yk})^T$ is *not* a vector in the robot-centered coordinate system.)

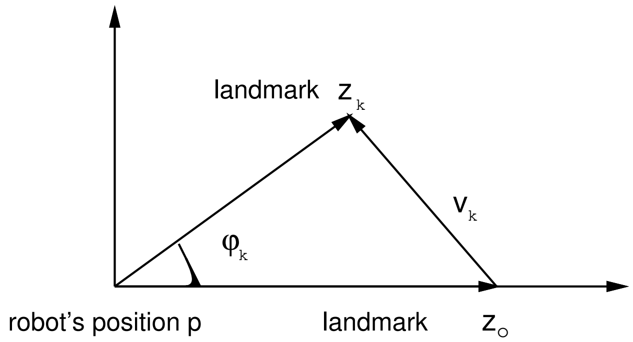

Figure 7: Robot coordinate system defined as a z-plane in the proof of Lemma 3.

Without loss of generality, we use a robot-centered coordinate system in which the x-axis is aligned with the vector to landmark z_0 (see Figure 7). The landmarks in this coordinate system are defined as $z_k = |z_k|e^{j\varphi_k}$ for $0 < k \leq n$. Thus, b_k can also be expressed as

$$b_k = \frac{z_k}{z_k - z_0} \; \frac{1}{|z_k|} = \frac{z_k}{z_k - z_0} k$$

where k is defined to be $1/|z_k|$. Then vectors b_k, $1 \leq k \leq b_n$, are linearly independent if for all indices k

$$b_k = \frac{z_k}{z_k - z_0} k = \alpha_k \eta$$

for some fixed complex number η and real α_k. We can also write this as

$$\frac{z_k}{z_k - z_0} = \beta_k \eta$$

for $\beta_k = \alpha_k / k$ which is equivalent to $\beta_k \eta z_0 = \beta_k \eta z_k - z_k$ and

$$\frac{\beta_k \eta z_0}{\beta_k \eta - 1} = z_k$$

for $\beta_k \eta - 1 \neq 0$. We can view this as a mapping f of points $w = \beta_k \eta$ in the complex w-plane into the complex z-plane where each landmark z_k is a point in the z-plane:

$$f(w) = \frac{z_0 w}{w - 1} = z.$$

This mapping is called a Möbius transformation (see for example [7]). The Möbius transformation maps the line $w = \beta\eta$ in the w-plane into a circle in the z-plane. Since $f(0) = 0$, the circle passes through the origin of the z-plane. Thus, landmarks $z_1, \ldots, z_n$ and the robot at $(0,0)$ lie on a circle if the vectors b_k, $1 \leq k \leq b_n$, are linearly independent. Landmark z_0 lies on the same circle since

$$\lim_{\beta \to \infty} \frac{\beta\eta z_0}{\beta\eta - 1} = \lim_{\beta \to \infty} \frac{\eta z_0}{\eta - 1/\beta} = z_0.$$

Note that we assumed $\beta_k\eta - 1 \neq 0$. If $\beta_k\eta = 1$, then we know that $b_k = \frac{e^{j\varphi_k}}{z_k - z_0} = k\beta_k\eta = k$ which means that $\varphi_k = 0$. In this case it follows that all landmarks $z_0, \ldots, z_n$ lie on the real axis of the z-plane. Thus, the landmarks and the robot lie on one line.

The properties of a Möbius transformation also hold for inverse transforms. Therefore, given landmarks arranged on a circle or on a line, it follows that vectors b_k, $1 \leq k \leq b_n$, are linearly independent and matrix $A^T A$ is singular. $\square$

The 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.

Lemma 4 *If landmarks are correctly identified and there is no noise in the measurements of the visual angles between the landmarks, then procedure* Position Estimator *returns the actual position of the robot.*

Proof: Let $\hat{z}_i^{(r)} = \hat{l}_i e^{j\hat{\tau}}$ be the vector to the i-th landmark in the robot coordinate system that is oriented as the external coordinate system. Note that the actual position of the robot is at $p_{actual} = z_0^{(e)} - \hat{z}_0^{(r)}$. The algorithm calculates $p = z_0^{(e)} - \frac{1}{n}\sum_{i=1}^{n} \hat{z}_{0,i}^{(r)}$ which is equivalent to

$$\begin{aligned} p &= z_0^{(e)} - \frac{1}{n}\sum_{i=1}^{n} \hat{z}_{0,i}^{(r)} \\ &= z_0^{(e)} - \frac{1}{n}\sum_{i=1}^{n} \frac{1}{r_i b_i - c_i} \end{aligned}$$

$$
\begin{aligned}
&= z_0^{(e)} - \frac{1}{n}\sum_{i=1}^{n} \frac{1}{\frac{\hat{l}_i}{\hat{l}_0}\frac{1}{\hat{v}_i}e^{j\varphi_i} - \frac{1}{\hat{v}_i}} \\
&= z_0^{(e)} - \frac{1}{n}\sum_{i=1}^{n} \frac{\hat{l}_0 e^{j\hat{\tau}_0}(\hat{z}_i^{(r)} - \hat{z}_0^{(r)})}{\hat{l}_i e^{j\hat{\tau}_i} - \hat{l}_0 e^{j\hat{\tau}_0}} \\
&= z_0^{(e)} - \frac{1}{n}\sum_{i=1}^{n} \frac{\hat{z}_0^{(r)}(\hat{z}_i^{(r)} - \hat{z}_0^{(r)})}{\hat{z}_i^{(r)} - \hat{z}_0^{(r)}} \\
&= z_0^{(e)} - \frac{1}{n} n \hat{z}_0^{(r)} \\
&= z_0^{(e)} - \hat{z}_0^{(r)}.
\end{aligned}
$$

Thus, the algorithm calculates $p = p_{actual}$. □

Lemma 2 and Lemma 3 establish the correctness of our algorithm:

Theorem 1 *Given a set of noisy measurements of landmarks which are not all arranged in a circle or line algorithm* POSITION ESTIMATOR *is an algorithm that determines the position of the robot such that the square of errors in the equations $Ar = c$ is minimized.* □

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

Proof: Due to the special form of vector $s = \frac{1}{2}A^T c$ it can be calculated in time linear in n by first computing $\sum_{j=1}^{n} c_{jx}$ and $\sum_{j=1}^{n} c_{jx}$ and then computing each entry s_i of vector s following Equation (8).

Algorithm POSITION ESTIMATOR calls procedure RATIOS-CALCULATOR which calculates vector r by substituting matrix $(nD)^{-1}$ and vectors $K^{-1}s$ and $K^{-1}b_y$ into Equation (9). Matrix $(nD)^{-1}$ can be computed in $O(n)$ time if we represent this diagonal matrix by a vector of its diagonal entries. The ith diagonal entry of $(nD)^{-1}$ is $1/(n(b_{xi}^2 + b_{yi}^2))$. Using this vector representation means that multiplying matrix $(nD)^{-1}$ with vectors s and b_x can also be done in linear time. Dot products $((nD)^{-1}b_x)^T s$ and $((nD)^{-1}b_x)^T b_x$ take $O(n)$ time to compute. Given vectors $(nD)^{-1}s$

and $(nD)^{-1}b_x$ and scalars $((nD)^{-1}b_x)^T s$ and $((nD)^{-1}b_x)^T b_x$, the computation of Equation (10) can be concluded in $O(1)$ time.

With a similar argument we can show that Equations (11) and (9) can be computed in linear time. □

5.4 Quality of Position Estimator

This section reports results of simulation experiments using procedure Position Estimate. In the experiments, the actual robot position $p_{actual} = (p_{x,actual}, p_{y,actual})$ is compared to the position $p = (p_x, p_y)$ that is computed by procedure Position Estimate. The actual robot orientation θ_{actual} is compared to the orientation θ that is also computed by procedure Position Estimate.

In the experiments, the errors $|p_{x,actual} - p_x|$ and $|p_{y,actual} - p_y|$ in the x- and y-coordinates of the position are computed. The length of the error vector $\|p_{actual} - p\|$ and the orientation error $|\theta_{actual} - \theta|$ are also computed.

Without loss of generality, the actual robot position p_{actual} is assumed to be at the origin of the map, i.e., $p_{actual} = (0, 0)$ and the robot is assumed to be oriented in the direction of the x-axis, i.e., $\theta_{actual} = 0°$.

First we randomly place eleven landmarks in a $10m \times 10m$ area using a uniform distribution where the robot is at a corner of this area. Note that this scenario uses only a quarter of the robot's surrounding area. It assumes that the robot cannot get omnidirectional input as Ratbot, but instead has a conventional camera setup.

In an attempt to simulate worst-case measurement errors, we first add a constant fraction $\Delta\varphi$ of a degree to *each* angle. Figure 8 shows results of experiments in which the added noise $\Delta\varphi$ ranges from 0° to 5°. We averaged our results over 100,000 scenarios with 11 landmarks, distributed uniformly in the 10 by 10 meter area. The length of the error vector $p_{actual} - p$ is 57 cm for 1° noise, 1.1 m for 2° noise, and 2.57 m for 5° for noise in *every* angle. The error in the position angle θ is 2° for 1° angle noise, 4.2° for 2° angle noise, and 9.8° for 5° angle noise.

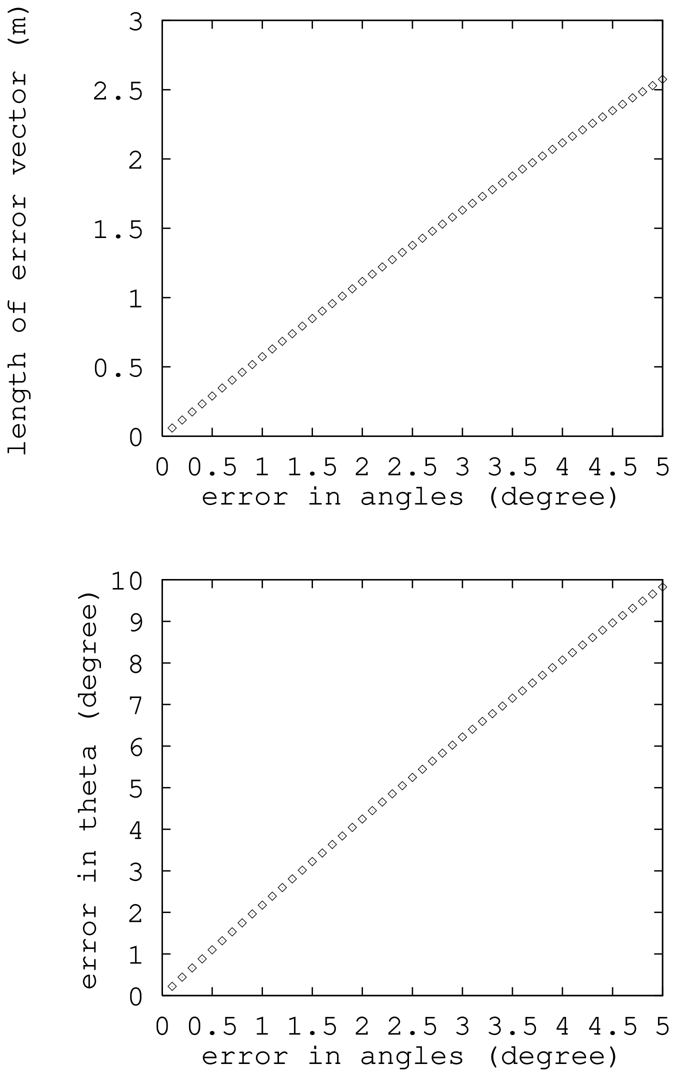

Figure 8: Simulation results using algorithm POSITION ESTIMATOR on an input of 11 landmarks and angle measurements with constant error. The error $\|p_{actual} - p\|$ in meters (top) and the error $|\theta_{actual} - \theta|$ in degrees (bottom) are shown as functions of $\Delta\varphi$ in degrees.

Next we show results of experiments in which the angle errors are not fixed, but uniformly distributed between $[-\Delta\varphi, \Delta\varphi]$ where $\Delta\varphi$ also ranges from $0°$ to $5°$. Figures 9 and 10 show the results of the experiments with 40,000 different environments that include 11 landmarks. Figure 9 shows the length of the error vector as a function of the visual angle error. The length of the error vector $p_{actual} - p$ is 19 cm for angle noise randomly chosen within $[-1°, 1°]$, 38 cm for angle noise randomly chosen within $[-2°, 2°]$ and 90 cm for angle noise randomly chosen within $[-5°, 5°]$. The error in position angle θ is on average zero.

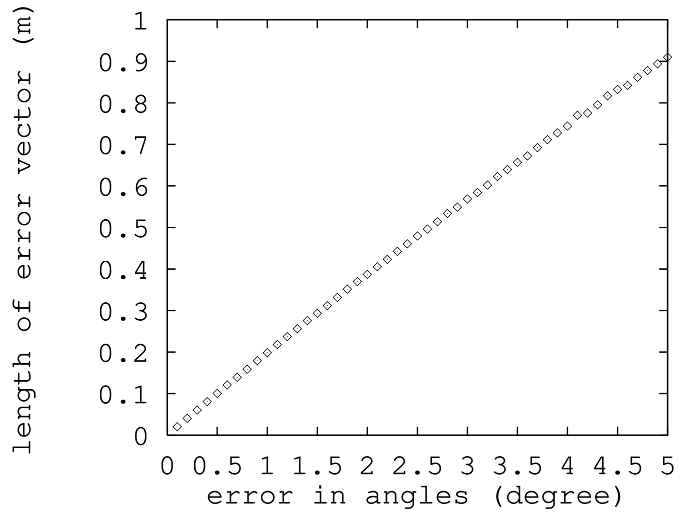

Figure 9: Simulation results using algorithm Position estimator on an input of 11 landmarks and angle measurements with error uniformly distributed over $[-\Delta\varphi, \Delta\varphi]$ where $\Delta\varphi$ ranges from 0 to 5 degrees. The error $\|p_{actual} - p\|$ in meters is shown as a function of $\Delta\varphi$ in degrees.

Figure 10 shows how the errors, $\|p_{actual} - p\|$, $|p_{x,actual} - p_x|$, $|p_{y,actual} - p_y|$, and $|\theta_{actual} - \theta|$ are distributed for $\Delta\varphi = 1°$. Figure 11 shows 200 positions of the robot in the two-dimensional environment computed by Position estimator with input error $\Delta\varphi = 1°$, $\Delta\varphi = 2°$, and $\Delta\varphi = 5°$. Note that the distributions for position errors $|p_{x,actual} - p_x|$ and $|p_{y,actual} - p_y|$ are almost normal.

In the next set of experiments, the robot is in the center of a $20\ m \times 20\ m$ area with eleven landmarks. We expect better results for this scenario in which the robot can take advantage of information provided by all of its surrounding. Indeed, our

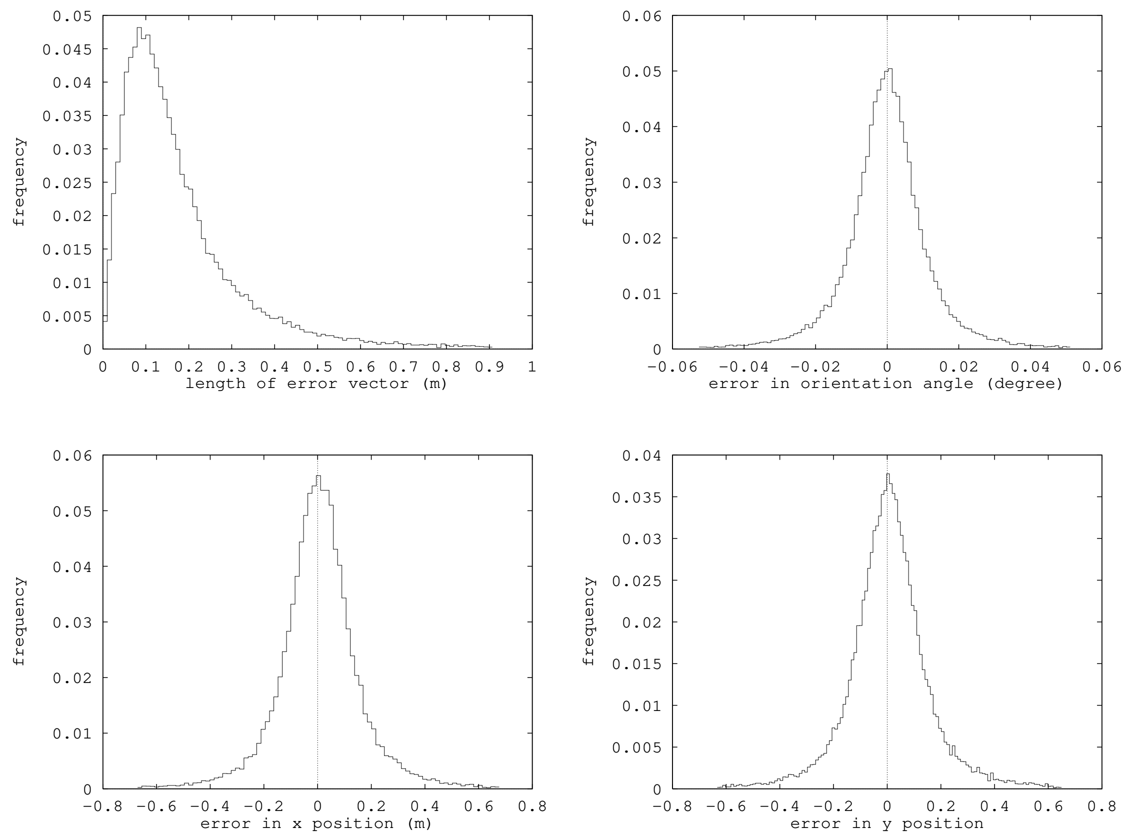

Figure 10: Distribution of errors for $\Delta\varphi = 1°$. On the top left, the distribution of the length of error vector $\|p_{actual} - p\|$ with mean $19.7\,cm$ and standard deviation $23.9\,cm$. On the top right, the distribution of the error in orientation $|\theta_{actual} - \theta|$ with almost zero mean and standard deviation $0.017°$. On the bottom left, the distribution of error in x position $|p_{x,actual} - p_x|$ with mean $0.93\,cm$ and standard deviation $0.22\,cm$. On the bottom right, the distribution of error in y position $|p_{y,actual} - p_y|$ with mean $10.6\,cm$ and standard deviation $21.4\,cm$.

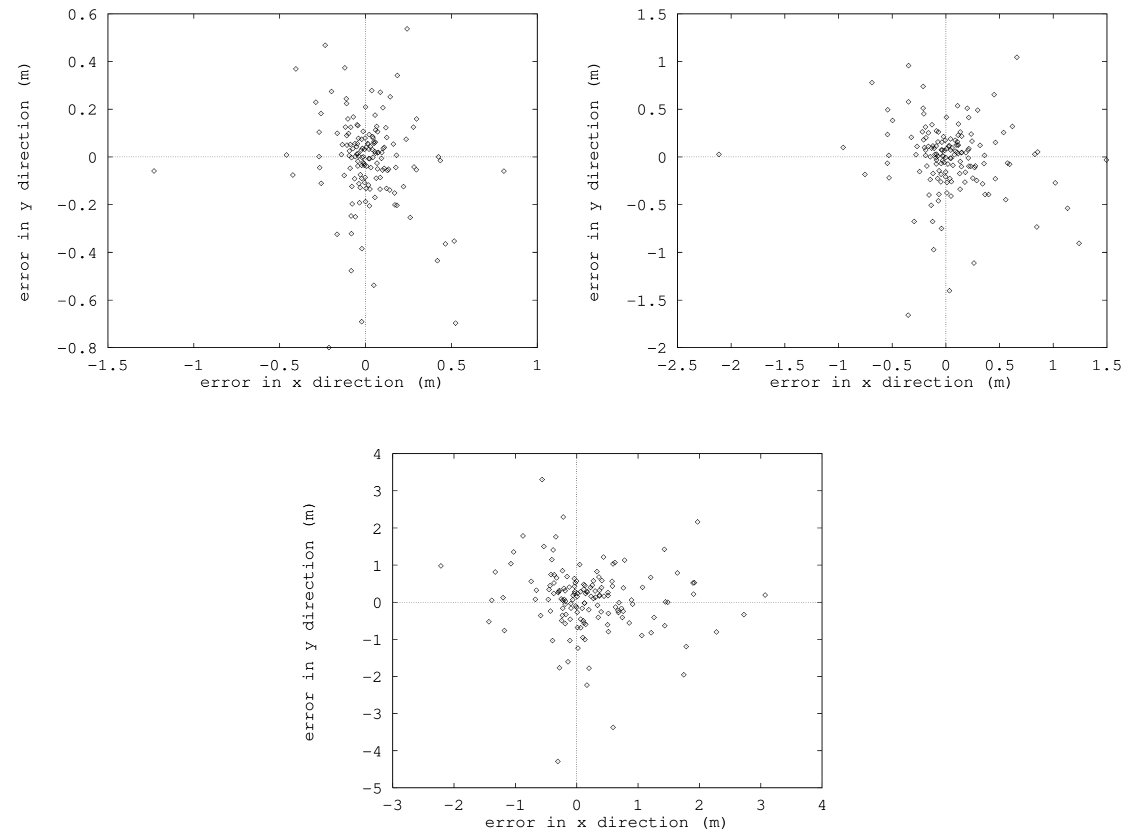

Figure 11: Computed positions of the robot in the two-dimensional environment from 200 sets of 11 landmarks for $\Delta\varphi = 1°$ (top left), $\Delta\varphi = 2°$ (top right), and $\Delta\varphi = 5°$ (bottom).

results show that the error in position has length 8.9 cm for $\Delta\varphi$ uniformly distributed within $[-1, 1]$, length 17.6 cm for $\Delta\varphi$ uniformly distributed within $[-2, 2]$, and length 42.6 cm for $\Delta\varphi$ uniformly distributed within $[-5, 5]$. The results of the experiments are summarized in Figures 12 and 13.

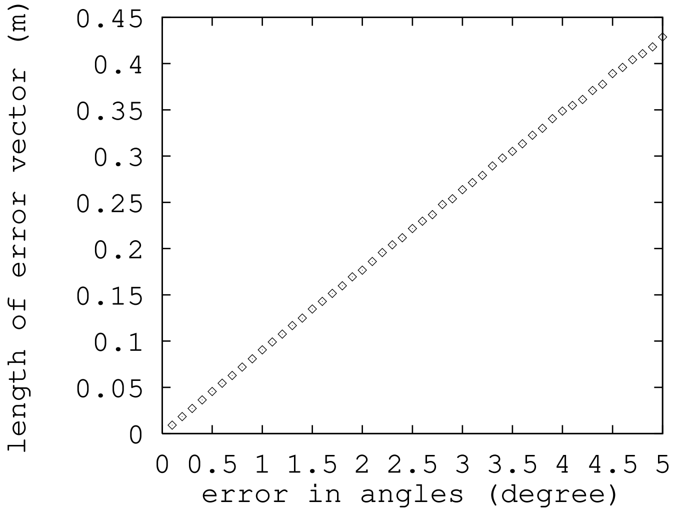

Figure 12: Simulation results using algorithm Position estimator on an input of 11 landmarks surrounding the robot and angle measurements with error uniformly distributed over $[-\Delta\varphi, \Delta\varphi]$ where $\Delta\varphi$ ranges from 0 to 5 degrees. The error $\|p_{actual} - p\|$ in meters is shown as a function of measurement errors in degrees.

Note that procedure Position estimator computes n position estimates $p_1, \ldots, p_n$ and averages them to obtain estimate p. Among $p_1, \ldots, p_n$, there may be some *outliers* that are not very close to the final position estimate p. However, in our experiments we observed 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. Figure 14 illustrates this behavior of Position estimator. Given an input of 21 landmarks $z_0, \ldots, z_{20}$, the procedure computes 20 position estimates $p_1, \ldots, p_{20}$. To model outlier errors resulting either from two noisy angle measurements or from two misidentified landmarks, we add a fixed error of $20°$ to two arbitrary angles. The other angles have an additive error uniformly distributed within $[-1°, 1°]$.

Our experiments show that the outlier landmarks produce position estimates that

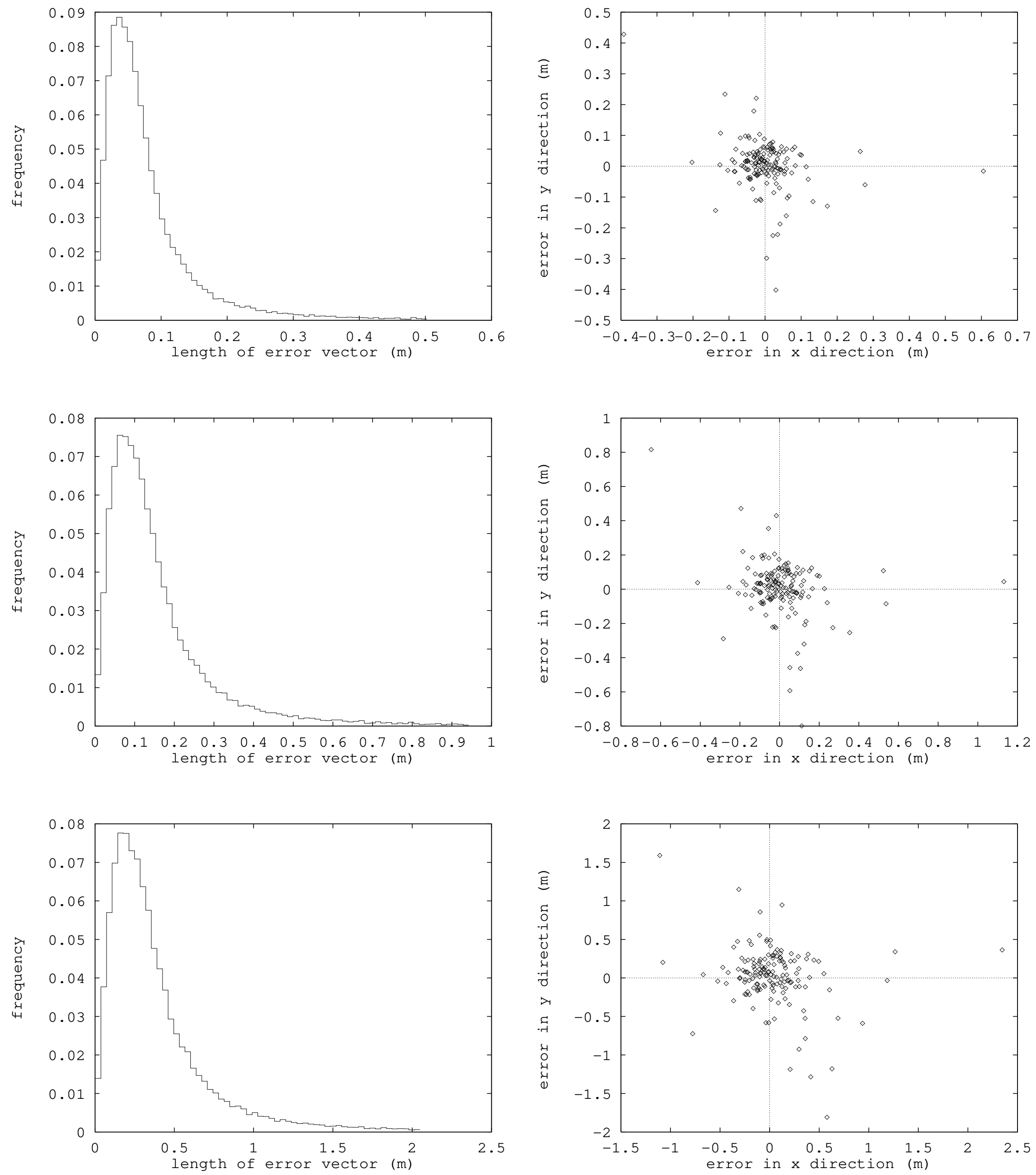

Figure 13: Experiments in which the robot is surrounded by 11 landmarks. On the left, the distributions of the length of error vectors $\|p_{actual} - p\|$ for $\Delta\varphi$ uniformly distributed within $[-1, 1]$, $[-2, 2]$, and $[-5, 5]$. On the right, computed positions of the robot in the two-dimensional environment for $\Delta\varphi = 1°, 2°$ and $5°$.

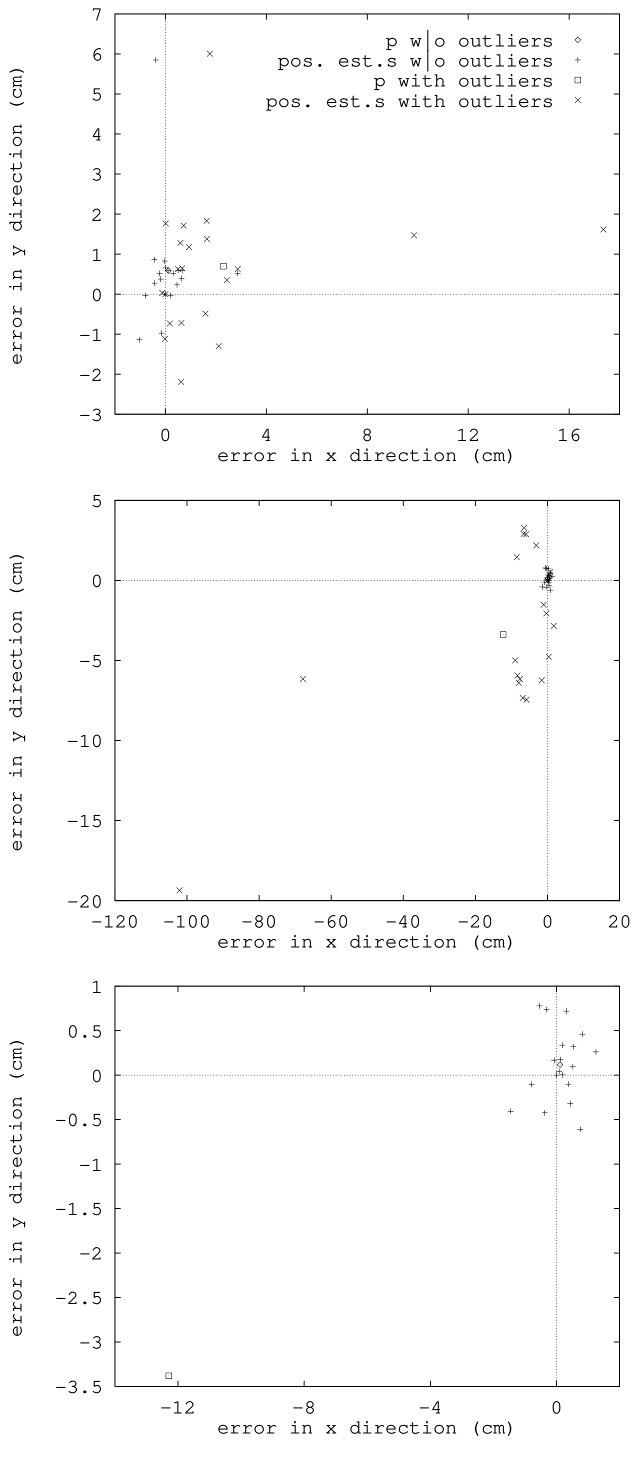

Figure 14: Two experiments with 21 landmarks including two outliers. In the first experiment (top), position estimate p computed with outliers is at $(2.3\ cm, 0.7\ cm)$ and p computed without outliers is at $(0.1\ cm, 0.6\ cm)$. In the second experiment (middle and a blowup at bottom), position estimate p computed with outliers is at $(-12.3\ cm, -3.4\ cm)$ and p computed without outliers is at $(0.1\ cm, 0.1\ cm)$.

are far from the actual position of the robot compared to the other estimates. So instead of taking the centroid of all position estimates $p_1, \ldots, p_{20}$ in line 12 of POSITION ESTIMATOR, we can obtain better results if we modify the algorithm such that we discard the outlier position estimates and calculate the centroid of the remaining points.

However, it is even better to call POSITION ESTIMATOR on an input of 19 landmarks (without the outlier landmarks). Figure 14 shows the results of our computation on two scenarios in which the length of the error vector is computed. For each of the scenarios, we show position estimates that are computed with and without the outlier landmarks. In the first scenario, the length of the error vector is 24.0 *cm* when computed with outlier landmarks, and 6.0 *cm* when computed without the outlier landmarks. In the second scenario, the length of the error vector is 127 *cm* when computed with outlier landmarks, and 1.6 *cm* when computed without the outlier landmarks.

6 Conclusions

We described an algorithm for estimating the position and orientation of a mobile robot given noisy input data. The key advantages of our POSITION ESTIMATOR over the approach described in Section 4 is that (1) the algorithm runs in time linear in the number of landmarks, (2) the algorithm provides a position estimate which is very close to the actual robot position and orientation, and (3) 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. Procedure POSITION ESTIMATOR could be incorporated into a navigation algorithm that keeps track of this information. One common approach is Kalman filtering.

References

[1] Arthur Albert. *Regression and the Moore-Penrose Pseudoinverse.* Academic Press, New York, 1972.

[2] David Avis and Hiroshi Imai. Locating a robot with angle measurements. *Journal of Symbolic Computation*, 10:311–326, 1990.

[3] Rodney A. Brooks. Aspects of mobile robot visual map making. In Hideo Hanafusa and Hirochika Inoue, editors, *Robotics Research.* MIT Press, 1985.

[4] Philippe Burlina, Daniel DeMenthon, and Larry S. Davis. Navigation with uncertainty: I. Reaching a goal in a high collision risk region. Technical Report CAR-TR-565, CS-TR-2717, University of Maryland at College Park, June 1991.

[5] R. Chatila and J. Laumond. Position referencing and consistent world modeling for mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 138–145, 1985.

[6] Frédéric Chenavier and James L. Crowley. Position estimation for a mobile robot using vision and odometry. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 2588–2593, Nice, France, 1992.

[7] Ruel V. Churchill. *Introduction to complex variables and applications.* McGraw-Hill, 1948.

[8] Simon Cooper and Hugh Durrant-Whyte. A Kalman filter model for GPS navigation of land vehicles. In *IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, pages 157–163, September 1994.

[9] J. J. Craig. *Introduction to Robotics: Mechanics and Control.* Addison-Wesley, Reading, Massachusetts, 1989.

[10] 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.

[11] Michael Drumheller. Mobile robot localization using sonar. *IEEE transactions on pattern analysis and machine intelligence*, PAMI-9(2):325–332, March 1987. Also published as MIT AI Memo 826, 1985.

[12] Alberto Elfes. A sonar-based mapping and navigation system. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, February 1986.

[13] Gellert, Küster, Hellwich, and Kästner, editors. *Mathematik*. VEB Verlag Enzykopädie, Leipzig, 1967.

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

[15] Leonid Gurvits and Margrit Betke. Robot navigation using landmarks. In preparation, 1995.

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

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

[18] Stephen Judd. Private communication. 1993.

[19] Lindsay Kleeman. Optimal estimation of position and heading for mobile robots using ultrasonic beacons and dead-reckoning. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 2582–2587, Nice, France, 1992.

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

[21] E. Krotov. Mobile robot localization using a single image. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 978–983, 1989.

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

[23] John J. Leonard and Hugh F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IEEE International Conference on Intelligent Robot Systems*, Osaka, Japan, Nov 1991.

[24] 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.

[25] T.S. Levitt, D. T. Lawton, D. M. Chelberg, and P. C. Nelson. Qualitative navigation. In *Proc. DARPA Image Understanding Workshop*, pages 447–465, 1987.

[26] Maja J. Mataric. A distributed model for mobile robot environment-learning and navigation. Technical Report AI-TR-1228, MIT, 1990.

[27] Larry Matthies and Steven A. Shafer. Error modelling in stereo navigation. Technical Report CMU/CS/86-140, CMU, 1986.

[28] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986.

[29] Gilbert Strang. *Linear Algebra and its Applications.* Academic Press, New York, 1976.

[30] Kokichi Sugihara. Some location problems for robot navigation using a single camera. *Computer Vision, Graphics, and Image Processing*, 42:112–129, 1988.

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

[32] 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.