A Component-Based Approach to Ultrasonic Self Localization in Sensor Networks

Alriksson, Peter; Årzén, Karl-Erik

2008

Document Version:
Publisher's PDF, also known as Version of record

ISSN 0280–5316
ISRN LUTFD2/TFRT--7619--SE

A Component-Based Approach to Ultrasonic Self Localization in Sensor Networks

Peter Alriksson
Karl-Erik Årzen

Department of Automatic Control
Lund Institute of Technology
May 26, 2008

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
Internal Report

Date of issue
May 2008

Document Number
ISRN LUTFD2/TFRT--7619--SE

Author(s)
Peter Alriksson and Karl-Erik Årzén

Supervisor

Sponsoring organization

Title and subtitle
A Component-Based Approach to Ultrasonic Self Localization in Sensor Networks

Abstract
The report describes an ultrasonic-based approach to localization of mobile robots using sensor network technology. The approach is based of a number of sensor network nodes that each calculates its distance to a mobile robot using the difference in time-of-arrival of an ultrasound pulse and a radio packet, and transmits this back to the robot. The robot uses an Extended Kalman filter to fuse together the distance measures from the multiple sensor nodes with encoder signals from the robot wheels. The method has been packaged in a component framework developed within the EU FP6 IP project RUNES.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
19

Recipient's notes

Security classification

1. Introduction

Networked embedded systems play an increasingly important role and affect many aspects of our lives. By enabling embedded systems to communicate, new applications are being developed in areas such as health-care, industrial automation, power distribution, rescue operations and smart buildings. Many of these applications will result in a more efficient, accurate and cost effective solution than previous ones. The European Integrated Project Reconfigurable Ubiquitous Networked Embedded Systems (RUNES) [16] brings together 21 industrial and academic teams in an attempt to enable the creation of large scale, widely distributed, heterogeneous networked embedded systems that inter-operate and adapt to their environments. The inherent complexity of such systems must be simplified if the full potential for networked embedded systems is to be realized. The RUNES project aims to develop technologies (system architecture, middleware, networking, control etc.) to assist in this direction, primarily from a software and communications standpoint.

Networked control systems impose additional requirements that arise from the need to manipulate the environment in which the networked systems are embedded. Timing and predictability constraints inherent in control applications are difficult to meet in general, due to the variations and uncertainties introduced by the communication system: delays, jitter, data rate limitations, packet losses etc. For example, if a control loop is closed over a wireless link, it should tolerate lost packets and be able to run in open loop over periods of time. Resource limitations of wireless networks also have important implications for the control design process, since limitations such as energy constraints for network nodes need to be integrated into the design specifications. The added complexity and need for re-usability in the design of control over wireless networks suggest a modular design framework.

In this report, we propose a component-based approach to handle the software complexity of networked control systems. A general framework is presented and it is shown how it can be instantiated in the specific problem of robot self localization and control. Section 1.1 briefly presents the RUNES middleware and component architecture. In the RUNES project a motivating scenario was developed, this scenario is described in Section 1.2. The different components and their connections are presented in Section 2 and in sections 3 and 4 the algorithms used for self localization and robot control are discussed.

1.1 Middleware and Components

The RUNES middleware [11] is illustrated in Figure 1. The middleware acts as a glue between the sensor, actuator, gateway and routing devices, operating systems, network stacks, and applications. It defines standards for implementing software interfaces and functionalities that allow the development of well-defined and reusable software. The basic building block of the middleware developed in RUNES is a software component. From an abstract point of view, a component is an autonomous software module with well defined functionalities that can interact with other components only through interfaces and receptacles. Interfaces are sets of functions, variables and associated data types that are accessible by other components. Receptacles are required interfaces by a component and make explicit the inter-component dependencies. A graphical representation of a RUNES component is shown in Figure 2.

The connection of two components occurs between a single interface and a single receptacle. Such association is called binding and is shown in more

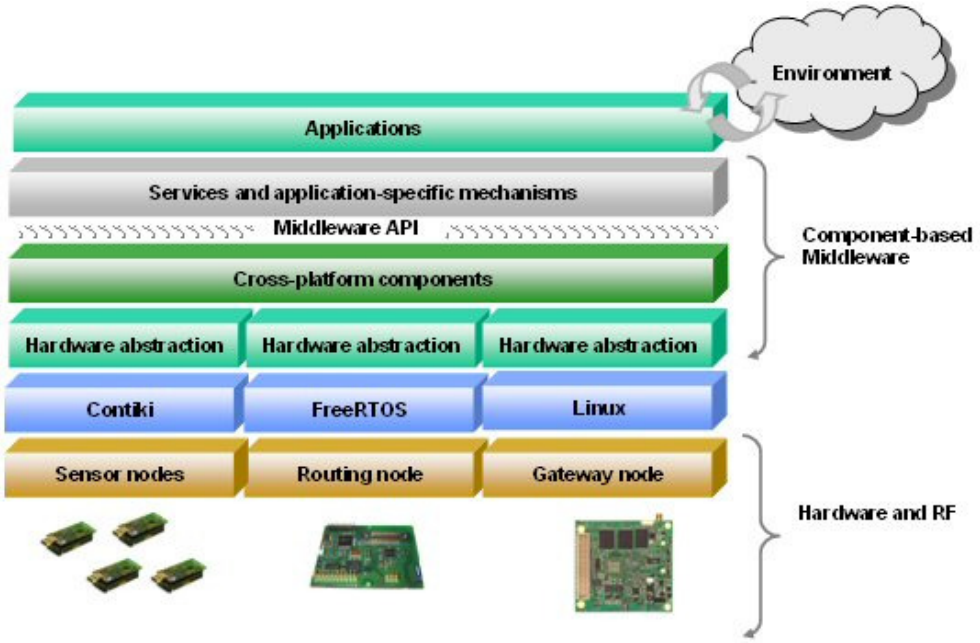

Figure 1 Overview of the RUNES middleware platform. The component-based middleware resides between the application and the operating systems of the individual network nodes.

detail in Figure 3. Part of the RUNES middleware has been demonstrated to work well together with the operating system Contiki [5], which was developed for low memory low-computation devices. The implementation of the component model for Contiki is known as the component runtime kernel (CRTK). This component framework provides for instance dynamic run-time bindings of components, i.e., during execution it allows components to be substituted with other components with the same interface.

Figure 2 Graphical representation of a RUNES component. Interfaces are sets of functions, variables and associated data types that are accessible by other components. Receptacles are required interfaces by a component.

1.2 Motivating Scenario

One of the major aims of the RUNES project is to create a component-based middleware that is capable of reducing the complexity of application construction for networked embedded systems of all types. Versions of the component runtime kernel, which forms the basis of the middleware, are available for a range of different hardware platforms. However, the task is a complex one, since the plausible set of sensing modalities, environmental conditions, and interaction patterns is very rich. To illustrate one potential application in greater detail, the project selected a disaster relief scenario, in which a fire occurs within a tunnel, much as happened in the Mont Blanc tunnel in 1999. In this, the rescue services require information about the developing scenario both before arrival and during rescue operations, and such information is provided by

a network of sensors, placed within the tunnel, on robots, and on rescue personnel themselves. We explore the scenario in more detail below, but it should be noted this is intended to be representative of a class of applications in which system robustness is important and the provision of timely information is crucial. So, for example, much the same considerations apply in the prevention of, or response to, Chemical, Biological, Radiological, Nuclear or Explosive (CBRNE) attacks; likewise, search and rescue operations, and even industrial automation systems form application domains with similar requirements for predictability of response given challenging external conditions.

In the RUNES scenario, we project what might happen in a similar situation if the vision of the US Department of Homeland Security's SAFECOM programme becomes a reality. The scenario is based around a storyline that sets out a sequence of events and the desired response of the system, part of which is as follows. Initially, traffic flows normally through the road tunnel; then an accident results in a fire. This is detected by a wired system, which is part of the tunnel infrastructure, and is reported back to the Tunnel Control Room. The emergency services are summoned by Tunnel Control Room personnel. As a result of the fire, the wired infrastructure is damaged and the link is lost between fire detection nodes (much as happened in Mont Blanc). However, using wireless communication as a backup, information from (for example) fire and smoke sensors continues to be delivered to the Tunnel Control Room seamlessly. The first response team arrives from the fire brigade and rapidly deploys search and rescue robots, following on foot behind. Each robot and firefighter carries a wireless communication gateway node, sensors for environmental temperature, chemical and smoke monitoring, and the robots carry light detectors that help them identify the seat of the blaze.

The role of the robots in this scenario is twofold: to help identify hazards and people that need attention, without exposing the firefighters to danger; and to augment the communications infrastructure to ensure that both tunnel sensor nodes and those on firefighters remain in contact with the command and control systems that the situation commander uses to make informed decisions about how best to respond. To accomplish this, the robots are moving autonomously in the tunnel taking into account information from tunnel sensors about the state of the environment, from a human controller about overall mission objectives, and from received signal strength measurements from the wireless systems of various nodes about the communication quality. The robots coordinate their activity with each other through communication over wireless links. Local backup controllers allow the robots to behave reasonably in the event that communication is lost.

2. Software Components

This section presents an overview of the components and their bindings used for navigation, self localization and control of a single autonomous robot. Components aimed at for example restoring or improving network connectivity, radio power control, coordination of multiple robots, security issues and so on are described in [1] and its companion papers.

A component-based middleware has, at least in theory, the big advantage that components can be developed and tested independently. However, when dealing with components that interact with the environment, one must be

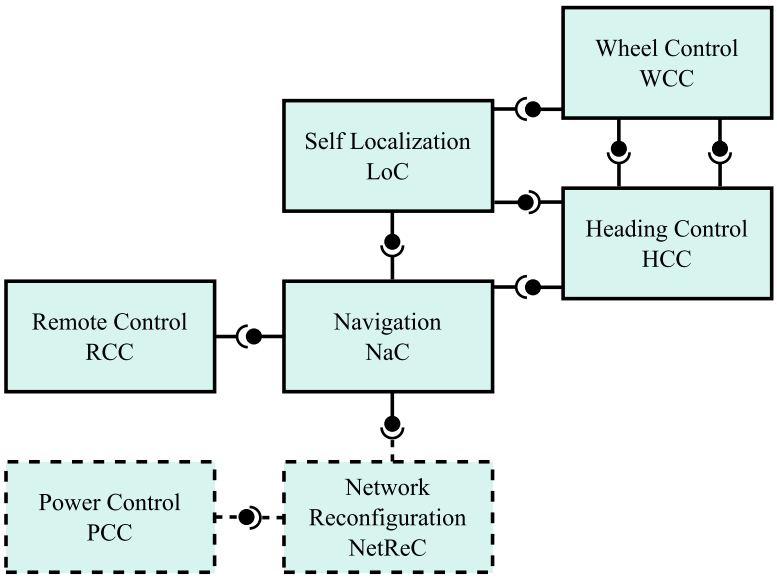

Figure 3 Overview of the component architecture used for navigation and deployment of a single autonomous robot.

aware of how the interconnection of components effect real-time performance. Thus some of the advantages with a component-based architecture can be illusive, especially in embedded systems with limited resources. Techniques like reservation-based scheduling are promising, but it is still an active research area, see for example [9] for a recent survey.

In Figure 3 an overview of the component architecture is shown. The system also contains a network communication component which provides UDP and TCP services together with Ad hoc On-Demand Vector Routing.

The interface-receptacle framework implies that components can be added in an hierarchical fashion. For example, the wheel control component (WCC), which does not require a connection to any other component and thus have no receptacles, serves as the base of the hierarchy. The self localization component (LoC) then binds to the WCC and so on.

As depicted in Figure 1 one component can reside in multiple devices, possible running different operating systems. In case of the LoC and RCC this is inherent to their function, whereas in the case of the HCC, WCC and NaC it is an implementational choice. All components except the PC part of the RCC was implemented in C code.

Next the different components together with their interfaces and receptacles will be described.

2.1 Wheel Control Component

The wheel control component is the most rudimentary component discussed here. It provides low level control of the movement of the robot. Two interfaces are provided by the WCC: one containing the current angular velocity of the two wheels and one providing a function to set the reference velocities.

2.2 Self Localization Component

The self localization component resides both in the robot and in a number of stationary anchor nodes. To reduce the computational time of the algorithm,

some computations are also distributed among two microprocessors within the robot.

The LoC provides one interface containing estimates of the robots position and heading in a predefined coordinate system. The interface also contains the covariance matrix of the estimation error which acts as a quality measure of the estimates. To enhance accuracy the LoC uses information about the movement of the robot. This is represented as a receptacle requesting wheel velocities.

2.3 Heading Control Component

The heading control component provides one interface containing functions for changing the desired heading together with the reference velocity of the robot. Measurements of the heading and wheel velocities are requested through two receptacles. The HCC also requires a way of changing the wheel reference velocities, this is represented as a receptacle requesting an interface with this functionality.

2.4 Navigation Component

The navigation component provides one interface containing a function to change the desired location of the robot. Measurements of the robots location and heading are requested through one receptacle. The NaC requires a way of changing the heading and velocity references, this requirement is also represented as a receptacle. The algorithms used for navigation are described in [13] and [6].

2.5 Remote Control Component

Naturally, also the remote control component is distributed between the controlling computer and the robot. The RCC has only one receptacle, representing a requirement of an interface that provides a function for changing the desired location of the robot. The remote control component generates this value through interaction with a remote user.

3. Self Localization

A prerequisite for navigation is self localization, i.e., the robots must know their current position and heading. Since the tunnel is assumed to be well-known, automatic map building is not considered. Instead it is assumed that the overall layout of the tunnel is known, with the exception of the position of a number of stationary obstacles, modeling, e.g., stalled vehicles.

Self localization of mobile robots can be performed with a number of techniques. In laboratory experiments it is common to use vision, e.g., a ceiling-mounted camera combined with an image-processing system. In the tunnel scenario this is not a realistic approach due to, e.g., problems with light and smoke. Another possibility is to use dead-reckoning using a high-precision inertial measurement sensor unit on-board the robot. A problem with dead reckoning-based approaches, however, is that they are open loop and that unmeasurable disturbances will cause position errors that cannot be compensated for. In an outdoor environment GPS would have been another possibility, but inside a tunnel this is less realistic.

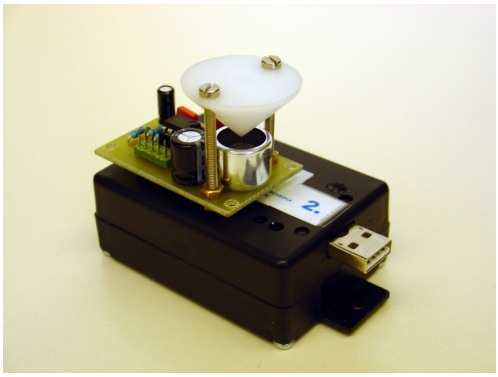

Figure 4 Stationary sensor network nodes with ultrasound receiver circuit. The nodes are packaged in a plastic box to reduce wear.

The self localization approach chosen in the RUNES project is based on measuring the distance to a number of objects with known positions, known as anchors, and then computing the position of the unknown object, in this case the robot, using these measurements. In a real road tunnel, the means of acquiring these distance measurements, would be dependent on the environment in which the system must operate. As the aim of this project was not to develop a positioning system capable of operating in the harsh environment of a burning road tunnel, but to demonstrate the benefits of a component-based design approach, a simple ultrasound based solution was chosen.

3.1 Ultrasound Distance Measurements

The basic idea is to transmit a wireless radio packet simultaneously with an ultrasound pulse from each sender node. The receiver nodes measure the difference in time of arrival between the radio packet and the ultrasound pulse and can in this way calculate their distance to the sender node.

Two main approaches exists, [17]. In an *active mobile* system the infrastructure, in this case the tunnel, has receivers at known locations, which estimate distances to a mobile device based on active transmissions from the device. Examples of this approach are the Active Badge [19], and the Ubisense [2] systems. In a *passive mobile* system, instead, the infrastructure has active beacons at known positions that periodically transmits signals to a passive mobile device. The most famous example of this is the Cricket system [14].

An advantage of the active approach is that it is more likely to perform accurate tracking than the passive approach. The passive approach, on the other hand, scales better with the number of mobile devices. Since in the tunnel scenario good tracking is important and the number of mobile robots is small, the active approach was chosen. The stationary sensor nodes in the tunnel are each equipped with an ultrasound receiver and each mobile robot is equipped with an ultrasound transmitter. The stationary sensor nodes are implemented as Tmote Sky sensor network "motes" together with a small ultrasound receiver circuit interfaced to the mote via the AD converter, see Figure 4. The mobile robots are equipped with an ultrasound transmitter circuit. Both the ultrasound transmitters and receivers are designed to be isometric, i.e., to transmit and receive in the full 360° degree plane.

A second reason for choosing ultrasound-based self localization is that it involves the use of the sensor network in closed loop. One of the objectives of the RUNES project was to investigate the possibilities and problems asso-

ciated with networked control over sensor networks. In wireless networks the lack of worst-case latency guarantees and risk of losing radio packets creates extra challenges for control. The ultrasound based location method provides a possibility for evaluating this.

3.2 State Estimation

Inferring the position (or any other quantity) of an unknown object from measurements of the distance to a number of objects with a priori known positions can be done using a variety of methods. In general the distance measurements are corrupted by noise, so any method used should involve some kind of filtering. If additional measurements such as wheel velocities and heading information is available, this information should also be incorporated in the estimate of the position. Both these requirements suggest the use of a dynamic model of the object to be positioned together with some statistical inference technique. The problem can thus be formulated as a general nonlinear state estimation problem on the form

$$\begin{aligned} x(k+1) &= f(x(k), k) + w(k) \\ y(k) &= h(x(k), k) + v(k) \end{aligned} \tag{1}$$

where $x(k) \in \mathbf{R}^n$ is the state vector to be estimated, $y(k) \in \mathbf{R}^p$ is the measured output, $w(k) \in \mathbf{R}^n$ and $v(k) \in \mathbf{R}^p$ are unknown disturbances. Note that a known input can be seen as a time dependent $f(\cdot)$.

When solving a general nonlinear state estimation problem one must almost always resort to approximations of some sort. The by far most common approach is the Extended Kalman filter [7] where the probability distribution of $x(k)$ given all previous information is approximated using a Gaussian distribution. A Gaussian distribution is fully described by its mean and covariance, thus it is sufficient to find an approximate way of updating these quantities after a new measurement is taken and as time progresses. In the Extended Kalman Filter, or EKF for short, this is done through linearization of $f(\cdot)$ and $h(\cdot)$.

An alternative way of updating the mean and covariance is through the use of the so called unscented transform, which results in the Unscented Kalman Filter (UKF) [8]. The UKF uses a deterministic sampling approach where the mean and covariance is represented using a minimal set of carefully chosen sample points. When propagated through the true nonlinear system, these points capture the mean and covariance accurately to the 3rd order Taylor series expansion of the non-linearity.

In both the EKF and UKF the the probability density of $x(k)$ is approximated by a Gaussian distribution. If a more general distribution is needed, the use of sequential Monte Carlo methods such as the so called particle filter [3] is a common approach.

All the methods presented so far, aim at approximating the full *probability distribution* of $x(k)$ given all previous information. A different approach is the joint maximum a posteriori- or trajectory estimation method where a point estimate of the full trajectory $x(0) \dots x(k)$ is generated. Using dynamic programming a recursive procedure for generating a point estimate of the last value $x(k)$ can be derived in principal. However, in general the complexity of this recursive scheme grows with k making it impossible to implement. One common method that finds an approximation of the joint maximum a posteriori estimate is moving horizon estimation (MHE) [15]. In MHE a point

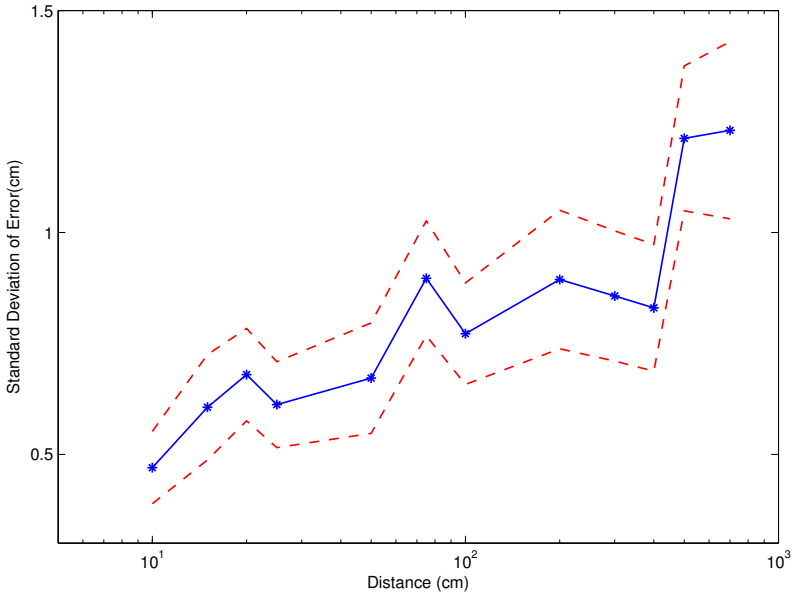

Figure 5 Standard deviation of the distance measurement error as a function of distance together with 95% confidence intervals.

estimate of $x(k-N)\dots x(k)$ for some fixed value N is generated by solving an optimization problem online.

3.3 Measurement Models

The measurement function $h(\cdot)$ is determined by if the distance measurements are preprocessed or not. If no preprocessing is done it will simply be the Euclidean distance between the object and the anchors. This setup will be referred to as the direct measurement model. In this case the unknown disturbance $v(k)$ can, at least approximately, be modeled as white Gaussian noise independent of the state. As can be seen in Figure 5 the implemented system gives an error of about 1 cm for distances up to 7 m. Also, the accuracy is fairly independent of the distance.

Trilateration One common way of preprocessing the distance measurements is called trilateration. Trilateration is a process, where three (in the plane) distance measurements together with the known positions of the anchor nodes, produces an estimate of the position of the unknown object. In this case $h(\cdot)$ reduces to an identity map for the position coordinates, but at the cost of making $v(k)$ highly dependent of $x(k)$. This dependence is in general difficult to model, due to the nonlinear characteristics of the trilateration procedure.

The basic problem is to find a solution $[\,p_x \quad p_y \quad p_z\,]^T$ to the following three nonlinear equations

$$\begin{aligned}(p_x - p_{x1})^2 + (p_y - p_{y1})^2 + (p_z - p_{z1})^2 &= d_1^2\\ (p_x - p_{x2})^2 + (p_y - p_{y2})^2 + (p_z - p_{z2})^2 &= d_2^2\\ (p_x - p_{x3})^2 + (p_y - p_{y3})^2 + (p_z - p_{z3})^2 &= d_3^2.\end{aligned}$$

where p_{xi}, p_{yi} and p_{zi} are known positions of the anchors and d_i is the distance from anchor i to the object to be positioned. The problem can be transformed

to a system of two linear equations and one quadratic equation by e.g subtracting the second and third equation from the first, see [10] for a detailed analysis.

An alternative more geometric approach was taken in [18] where the problem is solved using Cayley-Menger determinants. This approach as the benefit of a geometric interpretation of the solution in terms of volumes, areas and distances. Also the error analysis with respect to e.g distance errors is simplified.

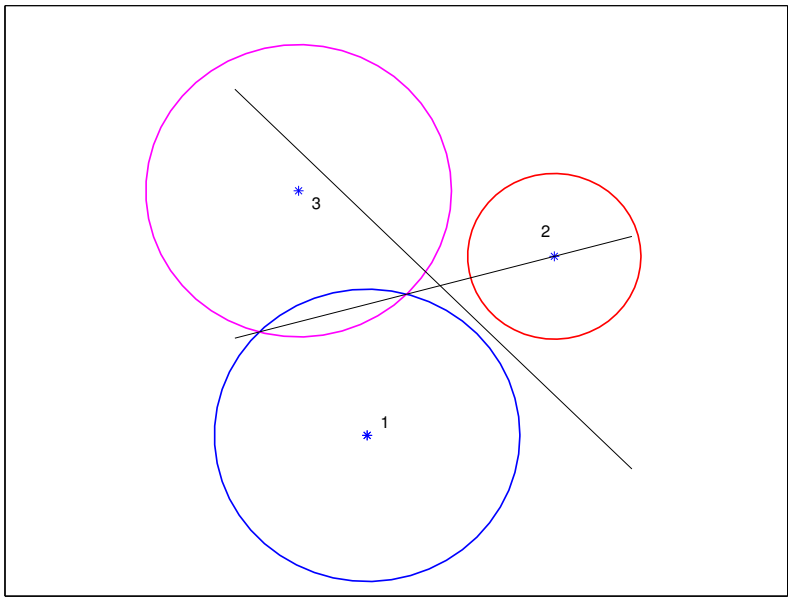

Figure 6 The intersection of two lines defining the solution to the trilateration problem. The three circles correspond to distance measurements to three anchor nodes with known positions. When measurement noise is present the circles may as shown overlap or not intersect. However, the trilateration procedure results in a reasonable result in both these cases.

As the robot is assumed to only move in the xy-plane, the problem can be reduced to a set of two linear equations as discussed above. The two linear equations will always have a solution unless all three known points are positioned on a line. The two linear equations defines two lines, see Figure 6, which can be represented as

$$a_0 y = a_1 + a_2 x \tag{2}$$
$$b_0 y = b_1 + b_2 x \tag{3}$$

where

$$\begin{aligned}
a_0 &= 2(p_{y2} - p_{y1}) \\
a_1 &= d_1^2 - d_2^2 + p_{y2}^2 - p_{y1}^2 + p_{x2}^2 - p_{x1}^2 - 2p_z(p_{z2} - p_{z1}) \\
a_2 &= 2(p_{x1} - p_{x2}) \\
b_0 &= 2(p_{y3} - p_{y1}) \\
b_1 &= d_1^2 - d_3^2 + p_{y3}^2 - p_{y1}^2 + p_{x3}^2 - p_{x1}^2 - 2p_z(p_{z3} - p_{z1}) \\
b_2 &= 2(p_{x1} - p_{x3}).
\end{aligned}$$

Note that the z-coordinate p_z of the robot is assumed to be known, as it is only moving in the xy-plane. The intersection point of these two lines constitute the trilaterated position $[\, p_x^{\text{tri}} \quad p_y^{\text{tri}} \,]^T$. Even though the three circles do not intersect in one point, the algorithm provides a reasonable result. However, for nearly singular situations like the one shown in Figure 7, the result produced need not to be very accurate. For a detailed discussion on how errors both in distance measurements and node positions influence the trilateration result, see [18] and [10].

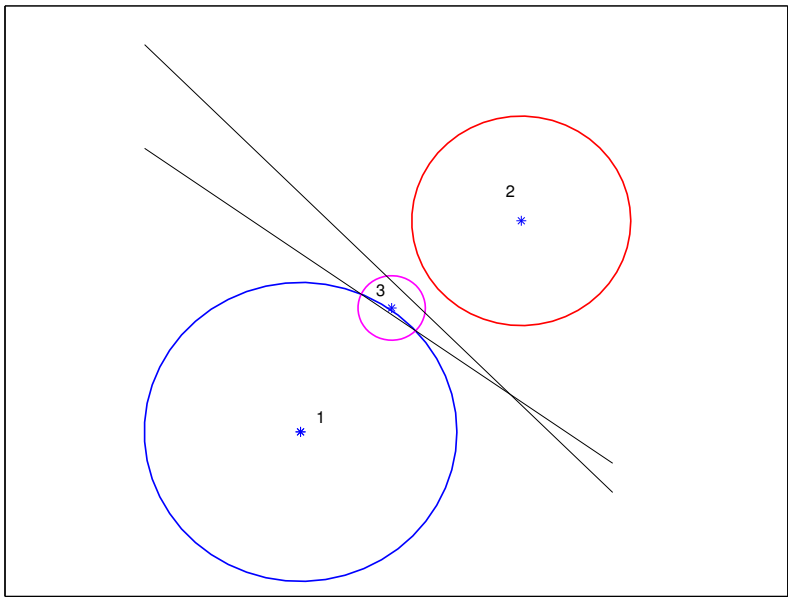

Figure 7 Nearly singular configuration where the trilateration result (intersection of the two lines) need not be very accurate. The distance measurements suggest that the object is located in the vicinity of anchor 3, but the trilateration procedure gives a result far away.

3.4 Choice of measurement model

Using the direct measurement model has two major advantages over trilateration: First of all, the direct approach can make use of only one or two measurements, whereas to perform trilateration three measurements are necessary. In a system relying on wireless communication, this is an important advantage. Secondly, modeling the state dependent noise in the trilateration case is very difficult. This results in that the algorithm is unaware of how good a measurement really is. This information is available to the direct approach as it uses a nonlinear measurement function $h(\cdot)$.

The trilateration approach has the advantage that it is conceptually simpler, as after a trilateration computation is done, an estimate of the position is directly available. In the direct measurement model, the position estimate is the result of an iterative algorithm involving other known signals and/or unknown states.

3.5 Dynamical Model

The choice of dynamical model $f(\cdot)$ should reflect both the available knowledge and which quantities that are of interest. The robot used in the RUNES project is a two wheeled dual drive robot with an unactuated support. Using this type

of robot has the big advantage that when turning, the actuated wheels are not slipping, thus a third order kinematic model can easily be derived.

The robot together with coordinate definitions is shown in Figure 8. Using the two position variables p_x and p_y together with the heading θ as state variables the model can be written as

$$\begin{cases} \dot{p}_x = \dfrac{R}{2}(\omega_1 + \omega_2)\cos(\theta) \\ \dot{p}_y = \dfrac{R}{2}(\omega_1 + \omega_2)\sin(\theta) \\ \dot{\theta} = \dfrac{R}{D}(\omega_2 - \omega_1) \end{cases} \tag{4}$$

where R is the radius of the wheels and D is the distance between them. Inputs to the system are the angular velocities ω_1 and ω_2 of the two wheels. In this model, these angular velocities are assumed to be completely known as they are controlled by two PI-controllers, see Section 4.2.

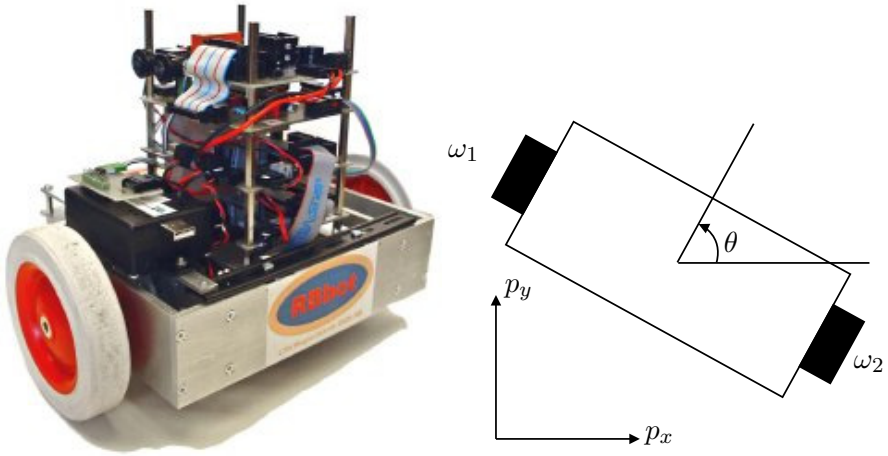

Figure 8 Definition of coordinates with respect to the robot.

To get a model on the form 1 the continuous time model 4 must be discretized. The most common choice of discretization scheme is the simple forward Euler scheme,

$$x(k+1) = x(k) + T\dot{x}(k) = f(x(k), k) \tag{5}$$

However, due to the limited computational resources in the hardware platform, the sampling interval T must be kept rather large. For large sampling intervals the forward Euler scheme can perform very badly. This motivates the use of a higher order scheme such as a second order Adams-Moulton method,

$$x(k+1) = x(k) + \frac{T}{2}\left(\dot{x}(k) + \dot{x}(k+1)\right) \tag{6}$$

In general the Adams-Mouton method is implicit, that is, we have to solve for $x(k+1)$ using some numerical method. However, for the continuous time model 4 an explicit solution can be obtained. Note that $f(x(k), k)$ will depend on the wheel velocities at time $k+1$, but this poses no problem if the model is not used for prediction.

3.6 Extended Kalman Filter

This section presents the extended Kalman filter and points out some implementational issues. The presentation is made with the direct measurement model in mind, but the same procedure applies for trilateration. Care must however be taken with how to compute the measurement noise covariance matrix R_v in the case of trilateration.

The EKF consists of two steps: time update and measurement update. In the time update step the mean $\hat{x}$ and covariance P of the Gaussian approximation of the probability density of $x(k)$ given all information up to k is updated as

$$\begin{aligned} \hat{x}(k+1|k) &= f(\hat{x}(k|k), k) \\ P(k+1|k) &= F(k)P(k|k)F^T(k) + R_w \end{aligned} \tag{7}$$

where R_w is the covariance of the process disturbance $w(k)$. The matrix $F(k)$ is the Jacobian of $f(\cdot)$ with respect to $x(k)$ at $\hat{x}(k|k)$,

$$F(k) = \left.\frac{\partial}{\partial x} f(x, k)\right|_{x=\hat{x}(k|k)} \tag{8}$$

When a measurement is available the mean and covariance is updated as

$$\begin{aligned} \hat{x}(k|k) &= \hat{x}(k|k-1) + K(k)\left(y(k) - h(\hat{x}(k|k-1), k)\right) \\ P(k|k) &= (I - K(k)H(k))P(k|k-1) \end{aligned} \tag{9}$$

where

$$\begin{aligned} K(k) &= P(k|k-1)H^T(k)S^{-1}(k) \\ S(k) &= H(k)P(k|k-1)H(k)^T + R_v \\ H(k) &= \left.\frac{\partial}{\partial x} h(x, k)\right|_{x=\hat{x}(k|k-1)} \end{aligned} \tag{10}$$

and R_v denotes the covariance of the measurement disturbance $v(k)$.

Note that if the Adams-Mouton approximation is used the time update step has to be computed at the beginning of the sampling interval due to the dependence in $f(x, k)$ of $\omega_1(k+1)$ and $\omega_2(k+1)$.

Because the number of received distance measurements may vary with time, the size of $H(k)$, $S(k)$ and $K(k)$ will also change over time. In the extreme, when no measurements are received, the measurement update step is simply ignored and the estimates are updated using the dynamical model only. This is often referred to as dead reckoning.

Perhaps both the most numerically sensitive and computationally demanding part of the above computations is the inversion of the output prediction error covariance matrix $S(k)$. This square matrix has the same dimension as the number of received measurements. For a linear Kalman filter, processing measurements with independent noise one at a time, thus reducing the matrix inversion to division with a scalar, is equivalent to processing them all at once. For a description, see for example [7] or [12] where a detailed computational analysis is done. For the EKF a similar procedure can be used, however sequential and non-sequential processing are in general not equivalent due to the nonlinear update equation. To be consistent with the structure of (9) $H(k)$ should be re-linearized after the addition of each new measurement.

3.7 Experimental Validation

To validate the accuracy of a self localization system using an EKF with the direct measurement model, a reference camera system was used. The EKF with sequential measurement processing was implemented in C code using 32 bit software emulated floating point arithmetic.

In Figure 9 the estimated position and heading is shown together with the measurements generated by the camera system. The camera system had a accuracy of about 1 cm and 6 degrees. The estimates were generated using seven anchor nodes distributed to form equilateral triangles covering the working area.

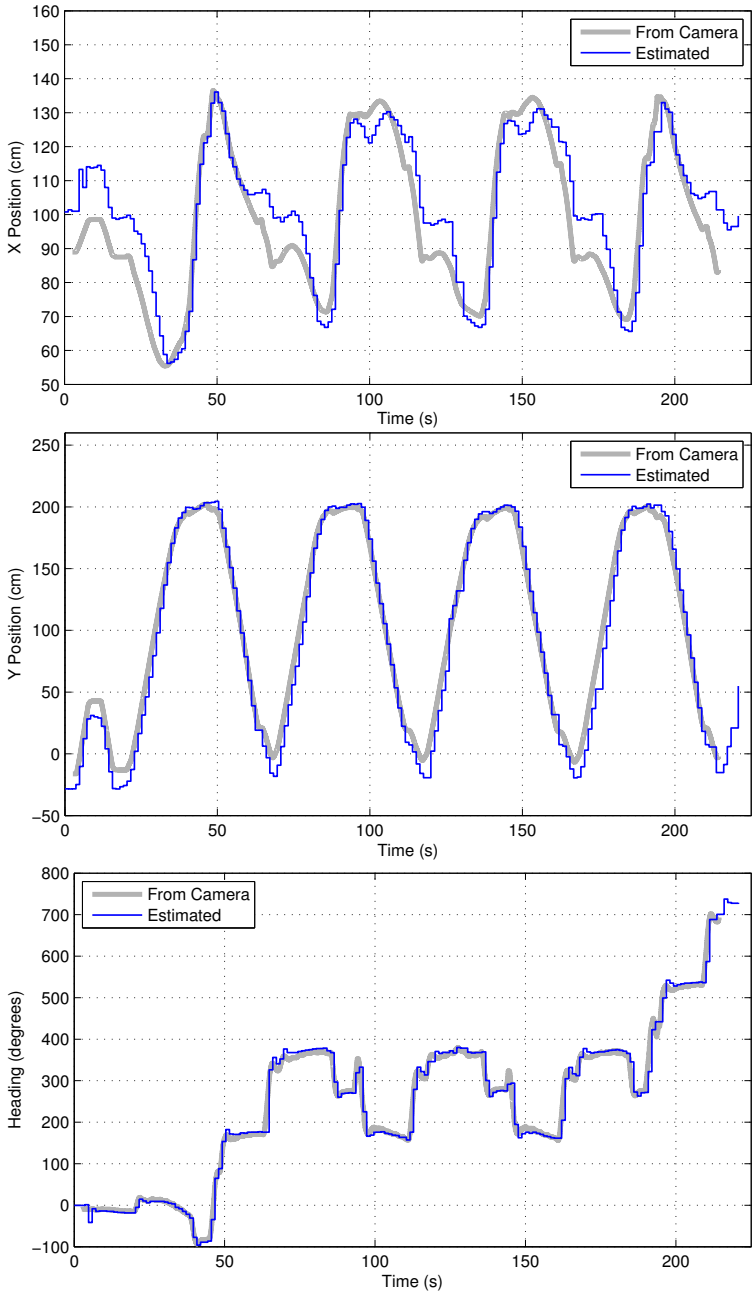

Figure 9 Estimates generated by a self localization system using the EKF together with the direct measurement model together with measurements generated by a reference camera system.

4. Robot Control

Many robot navigation algorithms produce heading- and speed references as their output. These reference values must be turned into actual control signals for the two motors driving the robot. In Figure 10 a hierarchical control system for thus purpose is shown. First the heading- and speed references are transformed into wheel speed references. The two wheels are then controlled individually to these references.

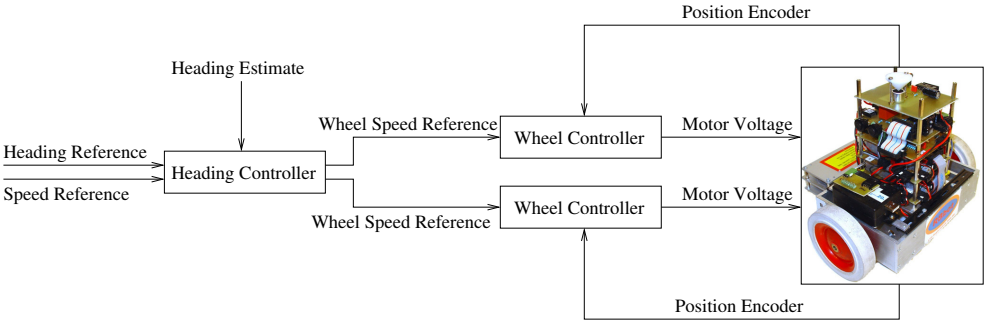

Figure 10 Hierarchical control system used to transform heading- and speed references into actual control signals for the two motors.

4.1 Heading Control

Using the same dynamical model of the heading θ as in the self localization algorithm and assuming that the wheel speeds are well controlled, a model suitable for control can be written as

$$\frac{d}{dt}\begin{bmatrix}\theta \\ \Delta\omega\end{bmatrix} = \begin{bmatrix}0 & \frac{R}{D} \\ 0 & -\frac{1}{T_\omega}\end{bmatrix}\begin{bmatrix}\theta \\ \Delta\omega\end{bmatrix} + \begin{bmatrix}0 \\ \frac{1}{T_\omega}\end{bmatrix}\Delta\omega_{\text{ref}} \tag{11}$$

Here $\Delta\omega = \omega_2 - \omega_1$ and $T_\omega \approx 0.4$ s is the time constant of the closed loop wheel control systems. Sampling the system with a sampling interval of 400 ms and designing an LQ-controller that minimizes

$$\sum_{k=0}^{\infty}(\theta(k) - \theta_{\text{ref}}(k))^2 + R^2\Delta\omega^2(k) + 10R^2\Delta\omega_{\text{ref}}^2(k) \tag{12}$$

results in a controller on the form

$$\Delta\omega_{\text{ref}} = -L_1(\theta - \theta_{\text{ref}}) - L_2\Delta\omega \tag{13}$$

Because $\dot{\theta}$ is proportional to $\Delta\omega$, the control law can be interpreted as a PD-controller. The two wheel speed references are then computed as

$$\begin{aligned}\omega_{\text{ref},1} &= \frac{v_{\text{ref}}}{R} - \frac{1}{2}\Delta\omega_{\text{ref}} \\ \omega_{\text{ref},2} &= \frac{v_{\text{ref}}}{R} + \frac{1}{2}\Delta\omega_{\text{ref}}\end{aligned} \tag{14}$$

where v_{ref} is the speed reference of the robot in m/s.

4.2 Wheel Speed Control

As mentioned in Section 3.5 the angular velocity of the two wheels are controlled by two PI-controllers operating at 100 Hz. The angular velocity is estimated from position encoders by differentiating the position signal and then using a low pass filter with a time constant of 100 ms.

In the heading control model it was assumed that the closed loop wheel speed control system has a time constant of $T_w \approx 0.4$ s. This is achieved by prefiltering the reference signal with a first order low pass filter. The time constant T_w had to be chosen large enough to prevent the wheels from slipping, which would violate the assumptions made when deriving the kinematic model (4).

5. Conclusions

The self localization and robot control system was part of a large scale demo involving a variety of hardware platforms, wireless radio technologies and operating systems. This situation very much captures what might be expected in a real world scenario. In the light of the experience draw from the demo, a number of areas, some where further development is needed, can be pointed out:

Co-existence of different wireless radio technologies need to be further investigated. In the RUNES project IEEE 802.15.4, IEEE 802.15.1 (Bluetooth) and IEEE 802.15.11 (WLAN), all operating in virtually the same frequency band, where used simultaneously. This together with the complicated indoor radio environment created by multipath propagation and the presence of people, gave rise to time variations which were virtually impossible to predict. How to develop closed loop control systems capable of handling this environment still remains a very challenging task.

When constructing distributed control/estimation systems operating on severely resource constrained platforms the lack of distributed debugging and monitoring tools becomes evident. Normally straight forward tasks such as logging of measured signals become a problem. Using wired logging is not possible if the network covers a large geographical area and wireless logging will consume bandwidth and CPU time, thus effecting the system under study. Debugging also becomes cumbersome as many problems will only be detected after deployment. Thus, there is great potential for development of tools resolving these issues. Another option is of course simulation, but simulating wireless radios, dynamical systems and software all at the same time is no simple task.

In heterogeneous environments such as the one described above, the use of a common network technology is very important. In the RUNES project a IP based solution was used. This allowed for example sensor nodes to communicate with desktop PCs without the use of adaptor functions, that would have been necessary if for example ZigBee would have been used. The key technology that allowed this is the uIP stack [4] which is capable of running on systems with severe resource constraints.

Acknowledgement

Many thanks to Jerker Nord and Jan Edhner for their efforts in implementing, testing and debugging the software. We also thank Rolf Braun for developing the hardware. This work was supported by RUNES Integrated Project contract IST-2004-004536 and the Swedish research council.

References

[1] Karl-Erik Årzén, Antonio Bicchi, Gianluca Dini, Stephen Hailes, Karl Henrik Johansson, John Lygeros, and Anthony Tzes. A component-based approach to the design of networked control systems. *European Journal of Control*, 13(2-3), June 2007.

[2] J. Cadman. Deploying commercial location-aware systems. In *Proc. Fifth International Conference on Ubiquitous Computing*, October 2003.

[3] Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo methods in practice*. Statistics for engineering and information science. Springer, 2001.

[4] Adam Dunkels. Full TCP/IP for 8 Bit Architectures. In *Proceedings of the First ACM/Usenix International Conference on Mobile Systems, Applications and Services (MobiSys 2003)*, San Francisco, May 2003. USENIX.

[5] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, November 2004.

[6] Jan Edhner. Obstacle avoidance for mobile robots. Master's Thesis ISRN LUTFD2/TFRT--5803--SE, Department of Automatic Control, Lund University, Sweden, October 2007.

[7] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[8] S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*, 1997.

[9] Mikael Lindberg. A survey of reservation-based scheduling. Technical Report ISRN LUTFD2/TFRT--7618--SE, Department of Automatic Control, Lund University, Sweden, October 2007.

[10] Dimitris E. Manolakis. Efficient solution and performance analysis of 3-d position estimation by trilateration. *IEEE Transactions on Aerospace and Electronic Systems*, 32(4):1239–1249, 1996.

[11] Cecilia Mascolo, Stefanos Zachariadis, Gian Pietro Picco, Paolo Costa, Gordon Blair, Nelly Bencomo, Geoff Coulson, Paul Okanda, and Thirunavukkarasu Sivaharan. D5.2 RUNES middleware architecture. RUNES deliverable, 2005.
`http://www.ist-runes.org/publicdeliverables.html`.

[12] J. Mendel. Computational requirements for a discrete Kalman filter. *IEEE Transactions on Automatic Control*, 16(6):748–758, Dec 1971.

[13] Jerker Nordh. Ultrasound-based navigation for mobile robots. Master's Thesis ISRN LUTFD2/TFRT--5789--SE, Department of Automatic Control, Lund University, Sweden, February 2007.

[14] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. In *Proc. Sixth ACM MOBICOM Conf.*, Boston, MA, August 2000.

[15] Christopher V. Rao. *Moving Horizon Strategies for the Constrained Monitoring and Control of Nonlinear Discrete-Time Systems*. PhD thesis, University of Wisconsin-Madison, 2000.

[16] RUNES. Reconfigurable ubiquitous networked embedded systems., 2007. `http://www.ist-runes.org`.

[17] Adam Smith, Hari Balakrishnan, Michel Goraczko, and Nissanka Bodhi Priyantha. Tracking Moving Devices with the Cricket Location System. In *2nd International Conference on Mobile Systems, Applications and Services (Mobisys 2004)*, Boston, MA, June 2004.

[18] Federico Thomas and Llus Ros. Revisiting trilateration for robot localization. *IEEE Transactions on Robotics*, 21(1):93–102, 2005.

[19] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, 1992.