# Mobile Sensor Localization and Navigation using RF Doppler Shifts

Isaac Amundson, Xenofon Koutsoukos, and Janos Sallai
Institute for Software Integrated Systems (ISIS)
Department of Electrical Engineering and Computer Science
Vanderbilt University
Nashville, TN 37235, USA
{isaac.amundson, xenofon.koutsoukos, janos.sallai}@vanderbilt.edu

## ABSTRACT

Many wireless sensor network applications require knowledge of node placement in order to make sense of sensor data in a spatial context. Networks of mobile sensors require position updates for navigation through the sensing region. The global positioning system is able to provide localization information, however in many situations it cannot be relied on, and alternative localization methods are required. We propose a technique for the localization and navigation of a mobile robot that uses the Doppler-shift in frequency observed by stationary sensor nodes. Our experimental results show that, by using observed RF Doppler shifts, a robot is able to navigate through a sensing region with an average localization error of 1.68 meters.

## Categories and Subject Descriptors

C.2.4 [**Computer-Communications Networks**]: Distributed Systems

## General Terms

Algorithms, Experimentation, Measurement, Theory

## Keywords

Doppler Effect, Localization, Navigation, Sensor Networks

## 1. INTRODUCTION

Recently, there has been a concerted effort to integrate mobility into wireless sensor networks (WSNs). Mobile sensor nodes enable dynamic deployment and expand the sensing region [1], minimize energy consumption across the network [2], and can connect sparse networks [3]. Mote-sized platforms [4], [5], [6] have been developed, as well as larger mobile platforms [7] that can sense the environment and react to it by performing tasks throughout the sensing region.

Sensor mobility introduces many new challenges. Among these, the need for localization is perhaps the most important. Mobile robots typically employ optical encoders, sonar, or laser rangefinders that can be used for localization and navigation. As mobile devices decrease in size and cost, these types of positioning sensors become difficult to implement. Often, the Global Positioning System (GPS) is cited as the de facto method for sensor node localization. However, GPS has several pitfalls such as size and cost, and cannot be relied on in certain sensor network deployments (i.e., indoors, under dense foliage, on the moon).

Several approaches have been taken to solve the problem of mobile sensor localization [8], [9]. Typically, a set of cooperating nodes emit signals such as acoustic, ultrasound, infrared, or radio frequency. A localization algorithm then computes node positions based on signal time-of-arrival, time-difference-of-arrival, angle-of-arrival, received signal strength, or other methods. For example, RADAR [10] measures the signal strength of an RF transmission at multiple base stations to triangulate the node position. In [11], a map of 802.11 ethernet packet signal strengths is compiled a priori, and used to determine the location of a mobile entity. The Cricket location system [12] uses ultrasound signals for tracking mobile nodes. Our approach is based on the radio interferometry methodology developed in RIPS [13] and the tracking of mobile nodes using RF Doppler shifts, developed in dTrack [14].

The Radio Interferometric Positioning System (RIPS) [13] was developed for localization of low-cost stationary sensor nodes. Two nodes, a master and an assistant, transmit a pure sine wave at slightly different frequencies, which interfere, resulting in a low-envelope beat frequency whose phase can be measured by two receivers at a designated time. The difference in phase is a linear combination of the distances between the transmitters and receivers. Using RIPS, relative node positions can be determined in networks containing at least six nodes [15]. The advantage of producing a low-frequency interference signal is that its period and frequency can be determined using inexpensive commercial wireless sensor nodes, without the need for radio hardware modifications.

dTrack [14] is a system for tracking mobile nodes that also uses the same underlying principle of radio interferometry, but measures signal frequency rather than phase. Tracking is accomplished by observing the Doppler shift in frequency that occurs when the source of a transmitted signal is moving relative to an observer. Because Doppler shift is determined by the relative velocity of the transmitter and receiver, absolute translational velocity of a mobile entity can be determined using a priori knowledge of the transmitted frequency and the frequencies observed by stationary sensor nodes at known positions.

We propose a method for localization and navigation of low-cost, resource-constrained mobile robots based on the methodology of the dTrack system. Due to uneven terrain, motor degradation, and slippage, robots will often deviate from their intended trajectories. By obtaining Doppler-shifted frequency data from surrounding stationary nodes, the robot is able to estimate its actual position and velocity, and make course corrections to stay on target. In order to accurately navigate using Doppler-shifted frequency information, we require a control loop that is capable of transforming the observed frequencies into meaningful position and velocity vectors. Because we are using low-cost hardware, and due to environmental factors such as humidity and temperature, we can expect a certain degree of measurement noise. By filtering this noise, we can determine our velocity error, which can then be input to a controller to produce updated angular velocities for each wheel on the robot.

Although we use the same methodology, there are significant fundamental differences between dTrack and our research. dTrack was designed as a system for tracking mobile objects, such as packages in a warehouse. The mobile objects behave as passive mobile entities in this regard. The tracking algorithm is run on a PC base station that is not limited by the resource constraints of typical mote-sized platforms. On the other hand, our system utilizes the Doppler-shifted frequency information as feedback to *control* the mobile node. Also, our approach requires that the localization and navigation algorithms be implemented on the mobile sensor node, within the control loop. One benefit of our system over dTrack is that the mobile robot is aware of the angular velocity commands it sends to each wheel, and can use this information to arrive at a better position and velocity estimate.

We evaluate our system in a real-world outdoor setting, in which a wheeled mobile robot with differential steering is commanded to navigate along a desired trajectory. Our experimental results demonstrate that using feedback from an anchored network of sensor nodes, the robot is able to maintain its course with an average localization error of 1.68 meters. Without the navigation system, the robot is unable to make adjustments to its speed and heading, and deviates from its desired trajectory by 5.32 meters over a 30-meter distance.

The remainder of this paper is organized as follows. In Section 2, we describe our system architecture. Our experimental results are then presented in Section 3. Section 4 concludes.

## 2. SYSTEM ARCHITECTURE

In this section, we describe the design of our localization and navigation system for mobile sensors.

The sensing region consists of a set of anchored *receiver* nodes at known positions, as well as a stationary *assistant transmitter* node. The mobile node, also referred to as the *master transmitter*, moves around the sensing region, as in Figure 1. Periodically, the master synchronizes with the assistant and receiver nodes via a *SyncEvent* message [16]. The message, which also serves as the synchronization point, provides a time in the near future for the master and assistant to transmit a signal, and for the receivers to listen for the transmission. The master and assistant transmit pure sine waves at frequencies that differ slightly, which generates an interference signal upon arrival at the receiver antennas [13]. The observed frequency will be Doppler-shifted, based on the velocity of the mobile node and the relative positions of the receivers.

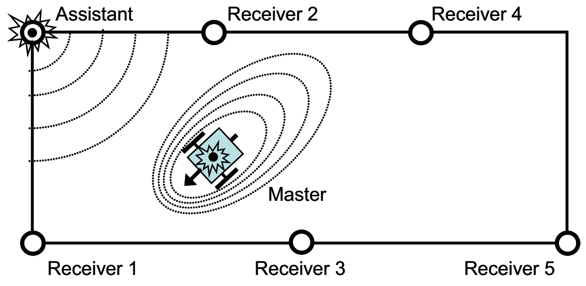

**Figure 1: Sensing region for localization and navigation. Dotted curves represent radio wave propagation. The arrow on the master node indicates direction of travel.**

The receiver nodes send the observed signal frequencies back to the master, and the observations are passed through an Extended Kalman Filter (EKF) [17] in order to arrive at an estimated robot trajectory in the presence of measurement noise. The output of the filter is the current estimate of the robot position, speed, and heading, as well as the interference frequency, which are then used to calculate the trajectory error, based on a reference speed and heading setpoint. The error is passed to the controller, which outputs updated left and right wheel angular velocities. Figure 2 illustrates the navigation system running on the mobile robot. Each component of the architecture is presented in detail below.

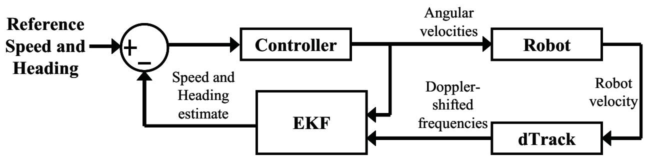
Reference Speed and Heading
+
−
Controller
Angular velocities
Robot
Robot velocity
Speed and Heading estimate
EKF
Doppler-shifted frequencies
dTrack

**Figure 2: The robot navigation system.**

### 2.1 Robot Kinematics

We use the following equations to describe the kinematic model for our two-wheeled robot with differential steering:

$$\dot{x} = \frac{r(\omega_r + \omega_l)}{2} cos\phi \quad (1)$$

$$\dot{y} = \frac{r(\omega_r + \omega_l)}{2} sin\phi \quad (2)$$

$$\dot{\phi} = \frac{r(\omega_r - \omega_l)}{2b} \quad (3)$$

where $x$ and $y$ constitute the robot position, $\phi$ is the heading, $r$ is the wheel radius, $b$ is the distance between the hub center of the driving wheel and robot axis of symmetry, and $\omega_r$ and $\omega_l$ are the right and left wheel angular velocities, respectively. The speed of the robot is the magnitude of the velocity, and in terms of angular velocity is represented as $|v| = \frac{r(\omega_r+\omega_l)}{2}$. For simplicity, this model does not take into account the effect of acceleration. For robots with sufficiently low mass, acceleration does not have a major impact on the forward kinematics of the system.

## 2.2 Controller

To arrive at the angular velocities that will keep the robot on the reference trajectory, we use a controller that takes as input the speed and heading errors, $e_{|v|}$ and $e_\phi$, respectively. Because $\phi$ wraps to 0 at $2\pi$, we shift the heading error to fall between $-\pi$ and $\pi$:

$$e_\phi = \begin{cases} e_\phi - 2\pi & \text{if } e_\phi > \pi \\ e_\phi + 2\pi & \text{if } e_\phi < -\pi \\ e_\phi & \text{otherwise} \end{cases}$$

The controller contains two PI equations, one for each error component:

$$|v| = K_p e_{|v|} + K_i \int e_{|v|} dt \quad (4)$$

$$\dot{\phi} = K_p e_\phi + K_i \int e_\phi dt \quad (5)$$

The PI equations give us the updated robot speed and angular velocity, however, the robot is commanded by specifying an angular velocity for each wheel. Consequently, we convert $|v|$ and $\dot{\phi}$ into individual wheel angular velocities, $\omega_l$ and $\omega_r$, as follows:

$$\omega_l = \frac{|v| - b\dot{\phi}}{r} \quad (6)$$

$$\omega_r = \frac{|v| + b\dot{\phi}}{r} \quad (7)$$

These angular velocities constitute the robot input, $u$.

The effect of the above transformation is that both wheels will be set with an equal base velocity to compensate for the translational speed error. If heading error exists, the robot will minimize it by turning one wheel faster than the base velocity, and the other wheel slower, which will result in the robot turning in the correct direction as it moves forward. Note that this is a simplistic controller used in the present work to demonstrate the feasibility of the navigation system, and will be replaced with a more robust control logic in future versions.

## 2.3 dTrack

A mobile node, $T$, moves through a sensing region with velocity $\overrightarrow{\text{v}}$ while transmitting a pure sine wave with frequency $f_t$ and wavelength $\lambda_t = c/f_t$, where $c$ is the speed of light. Simultaneously, a stationary node transmits a pure sine wave at a slightly lower frequency, $f_a$. A receiver node, $R_i$, observes the interference signal, with frequency $f_i$, which is Doppler-shifted. The amount of frequency shift is dependent on the relative speed of $T$ and $R_i$, and is defined by

$$f_i = \widehat{f} - v_i/\lambda_t \quad (8)$$

where $\widehat{f} = f_t - f_a$ is the interference frequency, and $v_i$ is the relative speed of $T$ with respect to $R_i$. The relative speed is the projection of the robot velocity onto the unit position vector pointing from $R_i$ to $T$, and can be expressed in the form given by

$$v_i = v_x cos(\alpha_i) + v_y sin(\alpha_i) \quad (9)$$

where $\alpha_i$ is the angle of $R_i$ from the viewpoint of $T$ with respect to the x axis, $v_x = |v|cos\phi$, and $v_y = |v|sin\phi$.

Once the robot applies the updated angular velocity commands to each wheel motor, dTrack is used to determine the actual velocity, based on the current robot position and velocity, and the positions of a set of stationary nodes in the vicinity of the robot. For a given receiver node $i$, the expected Doppler-shifted frequency, $f_i$, is given by

$$f_i = \widehat{f} - \frac{1}{\lambda_t} \frac{|v|((x_i - x)cos\phi + (y_i - y)sin\phi)}{\sqrt{(x_i - x)^2 + (y_i - y)^2}} \quad (10)$$

In practice, we find that the reported Doppler-shifted frequencies differ from the expected frequencies. Because of low-cost hardware, we are unable to know the exact transmission frequency, which can vary by as much as 65 Hz between successive timesteps. The variance is random, and therefore difficult to approximate (see Section 3.4). We therefore treat $\widehat{f}$ as an unknown variable. The noisy measurement data is then passed through an EKF in order to arrive upon a more accurate robot position and velocity.

## 2.4 Extended Kalman Filter

The Kalman filter is a widely used technique for estimating the state of a dynamic system based on noisy measurement data. We use the *extended* Kalman filter [17] because we are dealing with the nonlinear robot dynamics (Equations 1 - 3) and the nonlinear relationship between the measured frequency and the relative velocity (Equation 10). The EKF linearizes the estimation about the current robot state by applying the partial derivatives of the process and measurement functions. These functions take the form

$$X_k = F(X_{k-1}, u_k, w_{k-1}) \quad (11)$$

$$z_k = h(X_k, v_k) \quad (12)$$

where $X_k$ is the robot state $\{x, y, |v|, \phi, \widehat{f}\}$, $z_k$ is the set of Doppler-shifted frequency measurements $f_i$, and $u_k$ is the process input (which we obtain from the controller), $w_k$ is the process noise with covariance $Q$, $v_k$ is the measurement noise with covariance $R$, and $k$ is the current timestep. The state transition function $F$ that governs the dynamics of the robot is a vector function and is given by

$$F = \begin{bmatrix} x_{k-1} + \Delta t \frac{r(\omega_r+\omega_l)}{2} cos(\phi_{k-1}) \\ y_{k-1} + \Delta t \frac{r(\omega_r+\omega_l)}{2} sin(\phi_{k-1}) \\ \frac{r(\omega_r+\omega_l)}{2} \\ \phi_{k-1} + \Delta t \frac{r(\omega_r-\omega_l)}{2b} \\ \widehat{f}_{k-1} \end{bmatrix} \quad (13)$$

where $\Delta t$ is the time elapsed since the last time step. Recall that $\omega_l$ and $\omega_r$ comprise the process input $u$.

The EKF recursively estimates the robot state in two phases. The first phase predicts the state at the next time step based on the state at the previous time step and the current process input. The second phase adjusts the prediction with actual measurement data obtained during the current time step. In addition, an error covariance matrix, $P$, is maintained, which is a measure of the accuracy of the estimated state, and is used to update the Kalman gain. Formally, these two phases are represented as

1. Prediction Phase

$$\widehat{X}_k^- = F(\widehat{X}_{k-1}, u_k, 0) \quad (14)$$

$$P_k^- = A_{k-1} P_{k-1} A_{k-1}^T + Q \quad (15)$$

where $\widehat{X}_k^-$ and $P_k^-$ are the *a priori* state and covariance estimates for the current time step $k$, $A_{k-1}$ is the Jacobian of $F$ with respect to $X_{k-1}$, and $Q$ is the covariance of the process noise.

2. Update Phase

$$
\begin{aligned}
K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R)^{-1} && (16)\\
\widehat{X}_k &= \widehat{X}_k^- + K_k(z_k - h(\widehat{X}_k^-, 0)) && (17)\\
P_k &= (I - K_k H_k) P_k^- && (18)
\end{aligned}
$$

where $K$ is the Kalman gain, $H$ is the Jacobian of $h$ with respect to $X$, $R$ is the covariance of the measurement noise, and $I$ is the identity matrix.

The Jacobian matrices of $F$ and $h$ with respect to $X$ are given below in Equations (19) and (20). Note that for $H_{[i,j]} = \frac{\partial h_{[i]}}{\partial X_{[j]}}$, we only need to calculate the partial derivative of $h$ once for each column, because successive rows correspond to a different receiver node and can be calculated analogously.

$$A_k = \begin{bmatrix} 1 & 0 & 0 & \Delta t \frac{-r(\omega_r+\omega_l)}{2} sin(\phi_k) & 0 \\ 0 & 1 & 0 & \Delta t \frac{r(\omega_r+\omega_l)}{2} cos(\phi_k) & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

$$
\begin{aligned}
&\frac{\partial h_i}{\partial x_k} = -\frac{1}{\lambda} \frac{|v|_k (y_i - y_k)((x_i - x_k) sin\phi_k - (y_i - y_k) cos\phi_k)}{((x_i - x_k)^2 + (y_i - y_k)^2)^{3/2}},\\
&\frac{\partial h_i}{\partial y_k} = -\frac{1}{\lambda} \frac{|v|_k (x_i - x_k)((y_i - y_k) cos\phi_k - (x_i - x_k) sin\phi_k)}{((x_i - x_k)^2 + (y_i - y_k)^2)^{3/2}},\\
&\frac{\partial h_i}{\partial |v|_k} = -\frac{1}{\lambda} \frac{(x_i - x_k) cos\phi_k + (y_i - y_k) sin\phi_k}{\sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}},\\
&\frac{\partial h_i}{\partial \phi_k} = -\frac{1}{\lambda} \frac{|v|_k (-(x_i - x_k) sin\phi_k + (y_i - y_k) cos\phi_k)}{\sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}},\\
&\frac{\partial h_i}{\partial \widehat{f}_k} = 1
\end{aligned}
\tag{20}
$$

In [14] it was observed that the EKF did not respond well to sudden maneuvers of the mobile node, and so the EKF was combined with a constrained nonlinear least squares algorithm. We are able to avoid this step because the robot is aware of any maneuvers that are made, and provides this as input ($u$) to the EKF.

## 3. EVALUATION

### 3.1 Experimental Setup

Figure 3 shows the sensor devices we use in our experiments, and Figure 4 shows our experiment setup. The robot is a MobileRobots Pioneer 3DX [7], with $r = 9.55$ cm and $b = 17.78$ cm. Note that, although the Pioneer is equipped with an onboard Linux PC, it was not powered on for these experiments. In addition, the robot has optical encoders on each wheel, however, the measurement data were not made available to the controller at runtime.

Crossbow ExScal motes (XSMs) [18] were used to control the robot, as well as for the dTrack implementation. Six motes were placed in a 20 by 30 meter sensing region (see Figure 5), elevated 1.5 meters from the ground. Five of these were receivers, and one was the assistant transmitter. The position of the assistant is not important, as long as it is stationary and the signal it transmits can be received by the participating receiver nodes. Another mote, the master transmitter, was fixed to the robot, and communicated directly to the robot microcontroller via a serial connection.

One additional mote was used to host the Kalman filter. Ideally, EKF functionality would be implemented on the same node as the controller. However, due to memory limitations (see Table 3), we made the design decision to use two nodes. We argue that this does not affect scalability of the system, and with code optimizations it would be possible to implement the EKF on the controller node. The EKF mote was mounted to the robot body and communicated with the controller node over the wireless radio interface. Tuning of the EKF was done experimentally, based on offline tests of the robot and dTrack nodes.

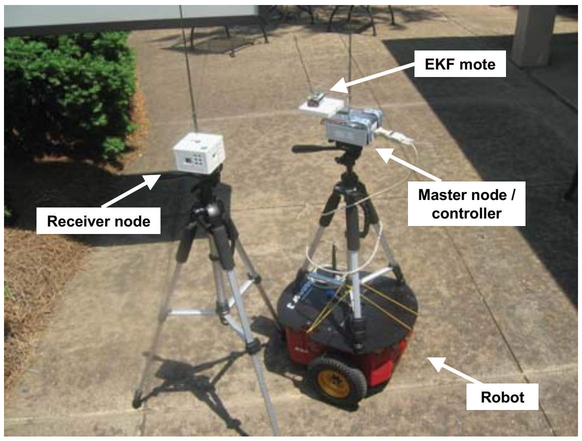

**Figure 3: Sensor hardware used in our experiments.**

**Figure 4: Experimental setup.**

### 3.2 Experimental Results

To obtain ground truth, we used measurements from the onboard optical encoders. In addition, the sensor field was recorded by video. We ran three sets of experiments. In Experiment 1, we disabled receiver node feedback, and instructed the robot to drive in a 30-meter straight line without making corrections for motor inaccuracies or wheel slippage. In Experiment 2, the robot attempted to control its speed and heading along the same straight-line trajectory based on feedback from the dTrack system. For Experiment 3, we introduced a 90° mid-course maneuver to the robot trajectory.

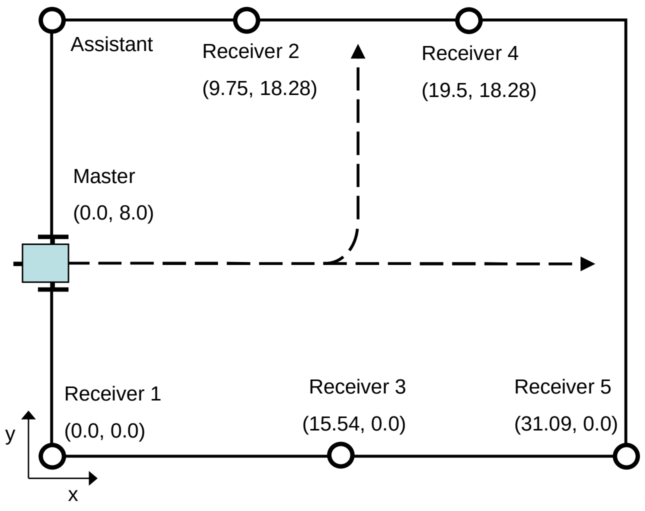

**Figure 5: Experimental sensing region. Dashed arrows denote reference trajectories.**

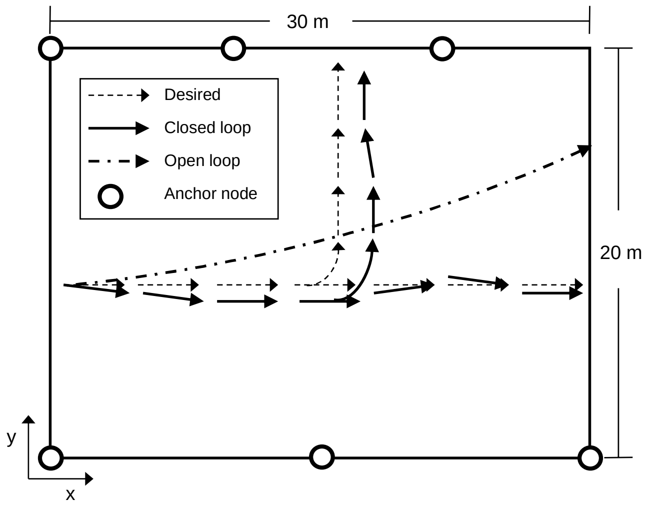

**Figure 6: Robot trajectory under both closed-loop and open-loop experiments.**

Figure 6 illustrates the desired and actual trajectories of the robot for both the open- and closed-loop sets of straight-line experiments, as well as for the 90° mid-course maneuver. With each feedback cycle, we recorded the current position, speed, and heading, and compared these with the desired values. The average errors are listed in Table 1.

For the open-loop case, the average speed is very close to the desired, because the translational velocity is set once at the beginning of the run, and the wheel motors are able to maintain a constant angular velocity fairly well. However, wheel slippage and uneven terrain caused the robot to gradually veer off course, which contributed to the large position and heading error. The closed-loop case has much lower position and heading error than the open-loop, however its average speed error is higher. This is because the robot places a higher priority on correcting its heading than it does speed. We believe we can reduce speed error in the future by tightening the cycle time of the control loop and designing a more robust controller. For the maneuver, rather than having the robot come to a stop, rotate, then resume moving in the new direction at the desired speed, we instructed the robot to maintain its target speed while performing the turn. This resulted in a 1.23 meter overshoot of the desired turning point.

| Experiment | Position error (m) | Speed error (m/s) | Heading error (rad) |
|---|---|---|---|
| 1 | 5.32 | 0.05 | 0.167 |
| 2 | 1.68 | 1.31 | 0.13 |
| 3 | 1.93 | 3.91 | 0.21 |

**Table 1: Average position, speed, and heading error over 66 measurements for open-loop (OL) and closed-loop (CL) experiments.**

## 3.3 Implementation Benchmarking

In order to ensure an accurate controller response, our implementation was required to iteratively run within a bounded timeframe. Table 2 displays the average and maximum execution time observed over 100 feedback cycles. From these execution times we see that we can provide feedback to the robot controller at a rate of approximately 1 Hz.

| Component | Average (ms) | Maximum (ms) |
|---|---|---|
| Radio interferometry | 360 | 361 |
| Frequency calculation | 61 | 116 |
| Routing | 266 | 580 |
| EKF | 230 | 241 |
| PI update | 1.3 | 1.4 |
| Robot control | 56 | 133 |
| **Total** | **974.3** | **1432.4** |

**Table 2: Execution time of each component.**

Another implementation benchmark is code size. Table 3 lists the memory requirements for the major components in our system. For the XSM, we were limited to 4 kB RAM and 512 kB ROM, resulting in our decision to host the EKF on a separate mote.

| Component | RAM (kB) | ROM (kB) |
|---|---|---|
| Radio interferometry | 3.1 | 45.9 |
| EKF | 1.2 | 22.1 |
| Robot control | 0.5 | 6.7 |
| **Total** | **4.8** | **74.8** |

**Table 3: Memory requirements of system components.**

## 3.4 Discussion

These preliminary results show that our technique works well for localization and navigation of mobile sensor nodes. Because we are using low-cost hardware, one problem we encounter is that we are unable to know the exact interference frequency, $\widehat{f}$, which can vary between successive timesteps by as much as 65 Hz. The variance is random, and therefore difficult to approximate. We see the effects of a variable $\widehat{f}$ in Figure 7, in which we plot the Doppler-shifted frequency error between what was observed by the receiver nodes and what was expected based on the current robot state and the interference frequency as estimated by the EKF. The figure illustrates the degree to which we are unable to estimate the interference frequency, with error ranging from about zero to 65 Hz, and an average of 26.41 Hz.

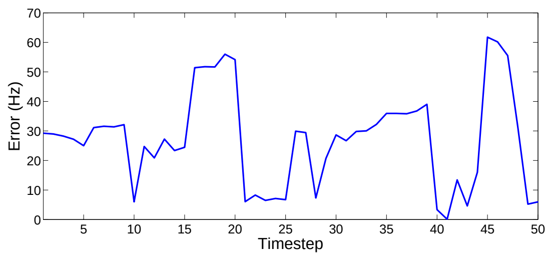

**Figure 7: Doppler shifted frequency error between observed and expected values.**

Our proof-of-concept experiments were conducted in an open environment in order to minimize multipath effects. Adding large obstacles to the sensing region will affect the observed Doppler shift at the receivers. The effect of multipath has been explored in [15], however, the extent of the multipath impact on our navigation system must still be evaluated.

## 4. CONCLUSION

We have developed a technique for localization and navigation for mobile sensors by using the principles of Doppler shift and radio interferometry. With feedback at approximately 1 Hz, we were able to direct a mobile robot along a 30 meter trajectory with an average position error of 1.68 meters. Future work involves waypoint navigation through a sensor field using Doppler-shifted frequencies, as well as exploration of our technique in multipath environments.

**Acknowledgements.** This work was supported in part by ARO MURI grant W911NF-06-1-0076, NSF CAREER award CNS-0347440, NSF grant CNS-0721604, and a Vanderbilt University Discovery grant. The authors also wish to thank Branislav Kusy, Peter Volgyesi, and Akos Ledeczi for their help with this project.

## 5. REFERENCES

[1] G. Wang, G. Cao, T. Porta, and W. Zhang, “Sensor relocation in mobile sensor networks,” in *IEEE INFOCOM*, 2005.

[2] S. Gandham, M. Dawande, R. Prakash, and S. Venkatesan, “Energy efficient schemes for wireless sensor networks with multiple mobile base stations,” in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2003.

[3] R. Shah, S. Roy, S. Jain, and W. Brunette, “Data mules: modeling a three-tier architecture for sparse sensor networks,” in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[4] S. Bergbreiter and K. S. J. Pister, “CotsBots: An off-the-shelf platform for distributed robotics,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[5] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. S. Sukhatme, “Robomote: enabling mobility in sensor networks,” in *The Fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

[6] J. Friedman, D. C. Lee, I. Tsigkogiannis, S. Wong, D. Chao, D. Levin, W. J. Kaisera, and M. B. Srivastava, “Ragobot: A new platform for wireless mobile sensor networks,” in *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2005.

[7] Pioneer P3-DX. MobileRobots. [Online]. Available: http://www.activrobots.com/ROBOTS/p2dx.html

[8] J. Hightower and G. Borriello, “Location systems for ubiquitous computing,” *Computer*, vol. 34, no. 8, pp. 57–66, 2001.

[9] F. Gustafsson and F. Gunnarsson, “Mobile positioning using wireless networks,” *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 41–53, 2005.

[10] P. Bahl and V. N. Padmanabhan, “RADAR: An in-building RF-based user location and tracking system,” in *IEEE INFOCOM*, 2000.

[11] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, “Practical robust localization over large-scale 802.11 wireless networks,” in *ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 2004.

[12] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha, “Tracking moving devices with the Cricket location system,” *In Proc. of MobiSys*, 2004.

[13] M. Maroti, B. Kusy, G. Balogh, P. Volgyesi, A. Nadas, K. Molnar, S. Dora, and A. Ledeczi, “Radio interferometric geolocation,” in *The 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[14] B. Kusy, A. Ledeczi, and X. Koutsoukos, “Tracking mobile nodes using RF doppler shifts,” in *The 5th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2007.

[15] A. Ledeczi, P. Volgyesi, J. Sallai, B. Kusy, X. Koutsoukos, and M. Maroti, “Towards precise indoor RF localization,” in *HotEmNets*, 2008.

[16] B. Kusy, P. Dutta, P. Levis, M. Maroti, A. Ledeczi, and D. Culler, “Elapsed time on arrival: A simple and versatile primitive for time synchronization services,” *International Journal of Ad hoc and Ubiquitous Computing*, vol. 2, no. 1, 2006.

[17] G. Welch and G. Bishop, “An introduction to the Kalman filter,” Department of Computer Science, University of North Carolina at Chapel Hill, Tech. Rep. TR 95-041, 2004.

[18] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, “Design of a wireless sensor network platform for detecting rare, random, and ephemeral events,” in *The Fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.