9-2009

Management of Target-Tracking Sensor Networks

Khaled Hadi
University of Massachusetts Amherst, hadikm@hotmail.com

MANAGEMENT OF TARGET-TRACKING SENSOR NETWORKS

A Dissertation Presented

by

KHALED HADI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2009

Electrical and Computer Engineering

MANAGEMENT OF TARGET-TRACKING SENSOR NETWORKS

A Dissertation Presented

by

KHALED HADI

Approved as to style and content by:

C. Mani Krishna, Chair

Israel Koren, Member

C. Andras Moritz, Member

Nathaniel Whitaker, Member

Christopher V. Hollot, Department Head
Electrical and Computer Engineering

For my loving wife Hamdah,
And our precious ones– Ibraheem and Aisha.

ACKNOWLEDGMENTS

I am very grateful for my PhD advisor, Professor C. Mani Krishna, for giving me the opportunity to work under his guidance. I have learned a lot from him and I acknowledge his patience and kind, and giving me a lot of time and effort in order to discuss and complete this project.

I am also thankful for the assistance and advice from my committee members: Professors Israel Koren, C. Andras Moritz and Nathaniel Whitaker. Their critiques and feedback have been both invaluable and insightful, which increasingly led to improve the quality of work. Special thanks go to Dr. Zahava Koren for her helpful suggestions.

I wish to express my gratitude to Kuwait University for financing this research and giving me a scholarship to study the Master and Doctorate. Also thanks to all friends and colleagues for their support and encouragement, and for empowering my life with fun and joy. Special thanks go to my friend: Mahmoud Ben Naser for his continuing encouragement.

Last but not least, I am thankful to my family, for their dedication and the support they have given me during all of my life.

ABSTRACT

MANAGEMENT OF TARGET-TRACKING SENSOR NETWORKS

SEPTEMBER 2009

KHALED HADI

B.S., KUWAIT UNIVERSITY, KUWAIT CITY, KUWAIT, 1999

M.S., UNIVERSITY OF CALIFORNIA, IRVINE, CA, USA, 2004

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor C. Mani Krishna

Target tracking has emerged as an important application of sensor networks. There are two subproblems inherent to target tracking. The first is the initial location of the target as it enters the region being covered. The second is following its track once it has been discovered.

In this work, we outline an approach to target tracking. We present an energy-aware tracking algorithm that predicts the target track and activates nodes based on that prediction. We then discuss different energy management schemes that resolve tradeoffs between energy savings and track quality for a specified mission lifetime. Our energy management schemes perform better in terms of track quality and have an energy consumption similar to other schemes. We also consider energy harvesting in this energy management. We present a multitarget tracking algorithm; in connection with that, we present a filtering algorithm that improves the quality of tracking. We also study adaptive approaches to manage the

tracking process to the observed mobility characteristics of the target. Such adaptive approaches are shown to have noticeable performance advantages.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Advances in micro-electro-mechanical systems (MEMS) technology [8] have made possible low power, small size, and low cost wireless sensor networks (WSNs). WSNs need to gather information from the physical world and fuse it to achieve application goals [15] [21] [26] [29].

Sensor networks have many applications. Typically, a sensor network encompasses a large number of nodes deployed in the environment being sensed and controlled. Typically, each sensor node comprises wireless communication and sensors. Power, memory and computational capabilities are usually severely constrained. Sensor networks often consist of a huge number of nodes. As these nodes susceptible to failure, the topology of the network may change with time. Sensor networks may use either identical nodes or consist of a heterogeneous structure in which some nodes are much more powerful than others or have diverse resources [33].

In this work, we study an object tracking application which aims to locate a moving object, and report this location in a timely fashion to a base station. Our tracking approach is as follows: Nodes within the border region of the area being covered detect an intruder and send messages to the base station to estimate its track. Based on this estimate, the base station assigns a certain subset of interior nodes to sense the next position of the intruder. If these nodes succeed in sensing the intruder, their data is used to update the intruder's track and another set of nodes can be deputized to generate the next position sample. If these fail to find the intruder, the area of search is expanded until either the intruder is found or the search is abandoned.

1.1 Related Work

1.1.1 Tracking Schemes

Object tracking is an active research problem in sensor networks and has been addressed in previous work, e.g., [10] [18] [36] [43]. Most of these studies focus on achieving high track quality without an excessive expenditure of energy. Energy savings are obtained through a variety of sleep-wake cycling strategies.

In [18], there is only one node, called the leader node, which is awake at any given time, while the rest of the network is asleep. The leader node applies a sophisticated algorithm to estimate the target position. The leader then passes this updated belief to a node close to this estimated position [11]. This node then becomes the new leader, and the original leader goes back to sleep.

In [36], the authors show that combining a selective activation framework with prediction (SA), where a small group of nodes in a circular monitoring region is in tracking mode, and Duty Cycle Activation (DA), where the *entire* network is turned on and off at the same time, is effective in terms of tracking quality and energy efficiency.

[43] proposes a dynamic convoy tree for data collection and fusion for target tracking to achieve energy savings. A convoy tree is a moving tree of nodes that reconfigure in an attempt to track a target. Collected and fused data are collected at the root of the tree. The authors provide two schemes to track the target: 1) Conservative Scheme, where the tree expands omnidirectionally according to a given threshold equation that depends on the target speed and the current radius of the monitoring region, and 2) Prediction Scheme, where the target expands along the predicted path of the target. In this scheme, new nodes are added in the tree and some nodes are pruned. The results show that the conservative scheme achieves better tracking quality than the prediction scheme, at the price of greater energy consumption.

In [10], the authors propose a proactive waking algorithm, which involves adaptive wake-period lengthening when a target is sensed nearby. This approach can be built atop

any sleep-awake algorithm (e.g. Geographic Adaptive Fidelity GAF [40] or Probing Environment with Adaptive Sleeping PEAS [9]). The algorithm has multiple wake states of sensor nodes layered around the current target position. Results show that using the proactive waking algorithm in conjunction with a duty-cycling sleep-wake algorithm improves track quality, but has some energy cost due to extending the waking period.

In [16], the authors propose a new object tracking algorithm based on a sensor network that produces one bit value of information to indicate whether the object is approaching, or moving away from, the sensor. Using this single bit model provides inexpensive sensing and communication. This binary value provides a geometric characterization that gives good information about the direction of a moving object. The exact location of this object, however, can be determined using additional information provided by a proximity sensor. The authors assume that all sensor nodes are awake all the time.

In [39], the authors propose an optimal control policy implemented in a sensor node based on the observational history of a target, while taking into account both energy availability and tracking accuracy. The idea is to increase sensor power if the target is far away and to skimp on sensor power dissipation when the target is perceived to be close by. Using higher sensor power on faraway targets reduces the error in the position estimate; throttling back when target is nearby saves energy. An optimal policy is obtained by mean of a Markov chain analysis.

[23] weighs the positions of the sensors detecting a target, and uses a weighted average to estimate the target location. The target track is computed in a central processing node. Several weighting approached are considered and their performance evaluated. The authors do not consider energy management.

Reducing communications during target tracking is one technique to save some energy. In [45], each sensor that detects a target sends a one bit message to its cluster head, and the cluster head then selects a subset of these sensors close to the target location. Only these selected sensors then send back to the cluster head detailed information needed for target

localization. This approach also reduces network latency due to lower communication overhead.

Real-time object tracking protocols have been addressed in the literature, where end-to-end delay time from the detection of the target until the corresponding message is received by a base station is bounded. [13] studies the effect of some parameters, i.e. speed and duty cycle, on the end-to-end delay and the energy consumption per day per node. The simulation results shows that the end-to-end delay decreases when increasing target speed and sentry duty cycle, but it decreases when increasing detection delay, degree of aggregation, and number of reports. Energy consumption, however, increases upon increasing the duty cycle, and it decreases by increasing the sensing range. This work divides the sensor nodes into sentry and non-sentry subsets, where sentry nodes are active (duty cycled) and able to detect target movement. The downside of this approach is when decreasing sentry node density, we need to increase the duty cycle of these sentry nodes to maintain some coverage, and this depletes the energy source of these nodes faster.

1.1.2 Mobility Models

There are several mobility models used to study the performance of target tracking networks. The simplest of all these models has the target move along a straight track at a constant speed [10] [18]. [43] [38] [6] make use of a Random Walk model, where at the end of given time interval, the target changes its speed and direction according to some probability distribution. Where there are roads that the target has to follow, one can use the Pathway Mobility Model [18] [13], where the target tracker makes use of the fact that the target is restricted to traveling on given roads or paths. Finally, [36] uses a sinusoidal trajectory as mobility model.

1.1.3 Position Estimation Algorithms

There are different ways for a sensor network to estimate the location of a target. One simple approach is to average the position of the nodes sensing the target [36]. [23] esti-

mates a weighting scheme for sensor position that exploits the fact that if the sensor lies near to the path of the object, the detection period will be longer since the target will spend a longer time in its sensing cicle. The estimated position is the weighted sum of the detecting sensors, where the weight is a monotonically decreasing function of distance. Another, more sophisticated, approach uses cooperative signal processing and Bayes's Law [18] [17] [44]. In particular, Bayes's law is used to quantify the probability of the target position, based on its recent history

1.1.4 Track Quality

One rather obvious metric to explores the quality of a tracking algorithm is to measure the average error between the actual position of the track and the estimated one [36]. Another metric uses the fraction of successful sensing points, where the target is sensed once every T seconds [43]. Another metric uses *path exposure* to measure the quality of the tracking. Path exposure is the integral of the sensing intensity function over the track path, where the sensing intensity is inversely proportional to the Euclidian distance between the sensor and the target positions [10].

1.1.5 Energy Harvesting

A few authors have begun to consider energy harvesting issues in sensor networks, e.g., [7] [19]. In these works, the authors provide an adaptive duty-cycling mechanism that allows sensor nodes to maintain their power supply at sufficient levels (energy neutral operation) by adapting their behavior to changing environmental conditions. The technique in [19] differs from [7] by assuming a priori knowledge of the energy profile. While such approaches are reasonable in environments that exhibit low variance, it is highly inefficient in more variable scenarios, and this has been addressed in [7]. [7] solves this problem based on results from adaptive control theory.

1.1.6 Multiple targets

Multiple target tracking techniques such as multiple hypothesis tracker (MHT) [28] and the joint probabilistic data association filter (JPDAF) [5] are designed for centralized processing. Both MHT and JPDAF are based on joint data association that enumerates all possible associations between objects and observations. Both attribute data to appropriate source at every time step, thus identity updates and location are firmly interleaved. These approaches are not scalable since they grow exponentially in complexity. Therefore, they require large memory and intensive computation, as such are not suitable for sensor networks.

MHT is a multi-scan tracking algorithm that retains multiple hypotheses by associating past measurements with targets. When having an arrival of a new set of measurements, a new set of hypotheses is created from each previous hypothesis. The algorithm generates a hypothesis with the highest posterior probability as a solution. JPDAF, however, is a suboptimal single-scan approximation to the optimal Bayesian filter; it can also be seen as an assumed-density filter in which the joint state estimation is for all time a single set of tracks for a ”known” set of targets. At each time step, rather than finding a single best association between measurements and tracks, JPDAF enumerates all potential associations and calculates association probabilities.

1.2 Contributions of Dissertation

Our principal contribution in this work is as follows:

1. A detailed examination of the various tradeoffs between tracking quality and node wake-sleep cycling in the face of a limited energy budget over a given period of operation. We study the impact of energy harvesting on these tradeoffs.

2. An approach to filter false reports from malfunctioning or noisy nodes.

3. An analytical model for position estimation, and target-catch probability.

4. An adaptive approach whereby the system can learn the parameters of the intruders' mobility model and adapt the sensor network management.

5. An extension of the target-tracking algorithm handles multiple simultaneous targets. Such an extension must include methods to disambiguate between targets even if they are relatively close together. This is a topic that has been little-studied by other researchers.

6. An approach to learn the current propagation conditions and target characteristics to improve the quality of tracking.

CHAPTER 2

KEY ASSUMPTIONS AND SIMULATOR SETUP

2.1 Assumptions

The general system assumptions we are using in this work are:

- The sensor nodes are identical and distributed randomly on a given region. The sensors are denser within the border area than internal to the region (see below).

- Each sensor position can be determined sufficiently and accurately by GPS [1], or other techniques such as triangulation [25].

- The sensor nodes are divided into *border nodes*, located within a given distance of the border, and *interior nodes* (see Figure 2.1). The border nodes, between them, keep the border area under surveillance all the time. Until an intruder is discovered, only their sensor part is on: the radio communications of the border nodes are off. The border area is more densely populated with nodes to ensure that it is continuously monitored over the designated period of operation without running out of energy. The radio communications of interior nodes use a low-energy paging channel [12] [41] [38] [32]. In this setup, a very low power radio is used to monitor the channel all the time. This channel monitoring can be done for just microwatts, and the monitoring circuit is responsible for waking up the node when appropriate. Other than this radio, the sensor devices and other elements of these interior nodes are turned off except when needed.

- The location of a target is estimated by simply averaging the locations of nodes sensing this target. We show that this simple averaging approach performs as well as

a more sophisticated convex hull based algorithm [37]; we also compare it with a signal-strength approach which performs somewhat better at the price of greater complexity.

- Targets always enter the region through the border region: no targets are spontaneously created within the region under surveillance. In this work, the system can handle either a single target or multiple simultaneous targets at any time.

- A single sink (base station) is also assumed in our implementation. This is where the sensor system reports the tracking data. The base station is computationally more powerful and we assume that it is not limited in its functioning by energy constraints (e.g., it may be connected to a power outlet).

- Sensor nodes are battery powered. We consider both the simple (baseline) case of a fixed energy budget per sensor node as well as the use of energy harvesting, which allows some energy to be drawn from the operating environment.

The track movement follows the random walk model [43] [4], where the target changes its speed and direction every T seconds, forming a piecewise linear track. Each of these linear segments is at an angle with respect to its predecessor that is determined by a Uniform distribution with zero mean over a certain range. Every T seconds, the target picks a new direction generated using the Uniform distribution over a given range, with zero mean. The speed over each linear segment can either be constant or also chosen according to a Uniform distribution over a given range.

We have adopted the energy model presented in [14]. When transmitting, the radio expends energy according to the following expression:

$$E_{TX}(k, d) = E_{elec} \times k + \varepsilon_{amp} \times k \times d^2 \tag{2.1}$$

When receiving, the radio expends:

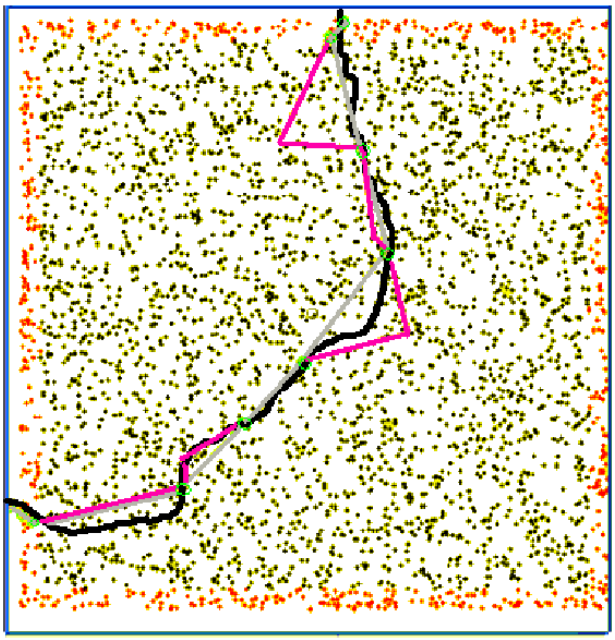


Figure 2.1. Sensor distribution, the red nodes are the border nodes, and the base station located in the center of the area

$$E_{RX}(k,d) = E_{elec} \times k \tag{2.2}$$

where E_{elec} is the radio dissipation in (nJ/bit) to run the transmitter and receiver circuitry and ε_{amp} in $(pJ/bit/m^2)$ is the energy consumed by the transmitter amplifier. d is the transmission range to transmit a k-bit message.

2.2 Simulation Setup

The simulation results we present throughout this work are based on the following simulator setup. The sensor nodes are identical and distributed uniformly over a $600m \times 600m$ rectangle with each sensor knows its location. The base station is located at the center of the sensor area. In simulation, target movement is controlled by setting different angle range as described in Section 2.1. Table I provides the simulation parameters used in all experiments.

Table 2.1. Simulation Parameters

Number of Nodes	2601, 3601
Border Width	20 m
Communication Range	80 m
Sensing Range	30 m
Power Consumption in Sensing	10 mW
Sensor Duty Cycle	10%
Power Consumption in Sleep Mode	1 uW
Transmission Rate	250 Kbps
Data Packet Size	64 bytes
Wakeup Packet Size	16 bytes
ACK Packet Size	32 bytes
Target Speeds	5 m/s, [2-8] m/s

CHAPTER 3

BASIC TRACKING ALGORITHM

3.1 The Algorithm

The tracking algorithm is summarized in the flowchart depicted in Figure 3.1. Tracking starts when a border node senses a target. It then switches on its radio communication channel and sends data information to interior nodes, which forward this message to the base station using geographic routing [20] [35] [30] [42].

For data messages initiated from border nodes only and sent directly to the base station, the base station collects these messages and computes the predicted location of the next target point using linear prediction, i.e. a linear extrapolation of the last two sensing points. The base station then sends a new data message to a sensor closest to the predicted location of the target at the next sample point. We call this sensor *a leader node*.

The leader node then wakes up sensors that are within R meters away from it: more details on the waking mechanism will be provided later in this work. R is a wakeup range parameter that can take one of several allowed values. The system starts with a small value of R and increases it as necessary to locate the target. The final R value usually involves waking up all sensor nodes in the field. The waking sensors then switch on their sensor devices, and try to detect the target. Every sensor that detects the target sends a report to its leader node. If the leader node receives reports from sensors, it computes the predicted next location of the target based on linear extrapolation of the previous and current sensing points and sends a data message to sensors close to this location. It also sends a report to the base station to update the target position. If the target is not detected after a given time

for the widest possible search area, the search is abandoned, and the target is declared to have been lost or missed.

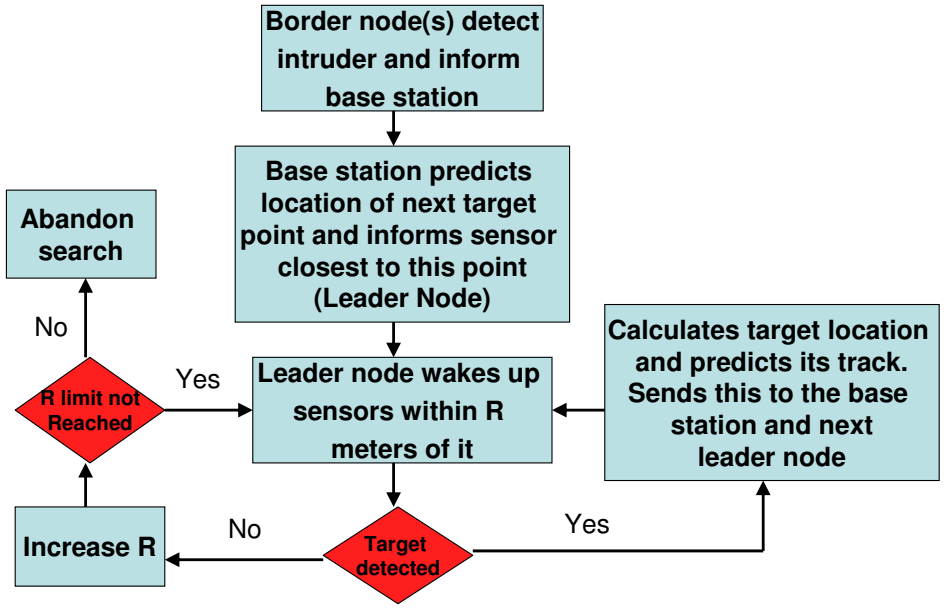


Figure 3.1. Tracking Algorithm

Key to the performance of the algorithm is the correct selection of the wakeup ranges. As a wakeup range is increased, the probability of sensing the target rises with it (and the probability of having to expand the search area due to missing the target declines). However, this comes at the cost of increased energy expenditure. The mobility model therefore greatly influences the choice of these ranges: a more deterministic track leads to smaller search areas being sufficient.

3.2 Performance of the Basic Algorithm

3.2.1 Wakeup Probability

In this analysis, we will assume that the entire sensing circle is within the wakeup circle; as long as the wakeup circle is not very small, the errors introduced by this simplifying assumption will be small.

Let the density of the live nodes (i.e., nodes which have not yet depleted their energy reserves) be Δ per square meter. If the total number of sensors is quite large (as is almost invariably the case), we can treat the scatter of the sensors as a spatial Poisson process. Let S be the radius of the sensing circle and ω be the wake probability. The probability of at least α awake sensors within the sensing circle can now be written down as:

$$\Omega(\alpha) = 1 - \sum_{i=0}^{\alpha-1} e^{-\pi S^2 \omega \Delta} \frac{(\pi S^2 \omega \Delta)^i}{i!} \tag{3.1}$$

We can now set Ω sufficiently high by controlling ω so that the position estimate is reasonably accurate. Figure 3.2 provides some numerical results. For each sensing radius value, we show the average error as a function of the wakeup probability, ω and the value of Ω.

The wakeup probability has a strong impact on the energy consumption, and hence on the useful lifetime of the sensor network. Small values of ω result in a greater chance of missing the target and hence require the search area to be widened, thereby consuming more energy. Even if the target is not missed, if only one or two nodes detect the target, there can be significant error in the position estimate, which increases the prediction error for the next target sample. Such increased prediction error increases the chance of the subsequent sample not finding the target within the initial search area, which has the energy consequences mentioned above. On the other hand, very large values of ω result in too many nodes being awake in the first place, which also costs energy. This is summarized in Figure 3.3 and Figure 3.4 (the scheme for adaptive ω shown in the figure is described later in this work). Here, each node is assumed to have a fixed amount of energy and dies when

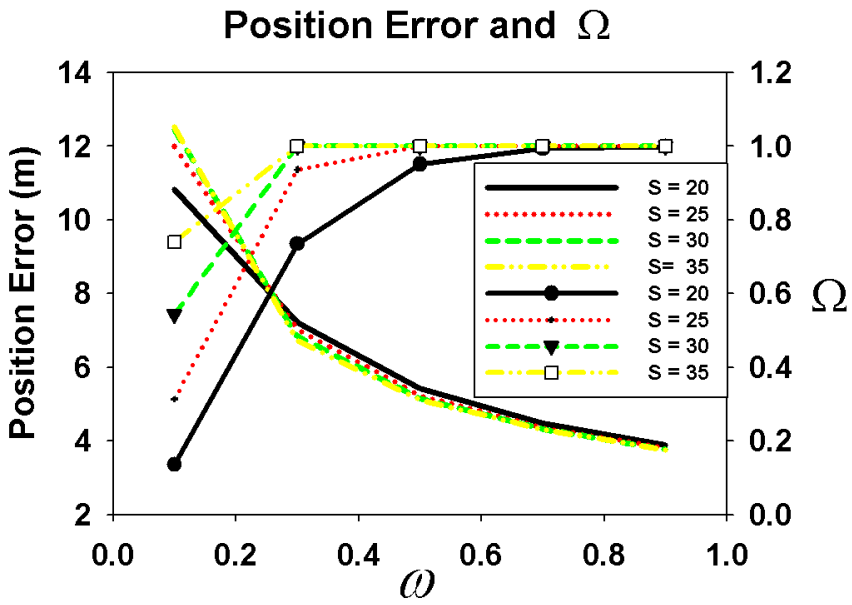


Figure 3.2. Mean Position Error and Probability of at Least 3 Awake Sensors Ω(3) as a Function of the Wakeup Probability ω for each Sensing Radius Value

that energy has been spent. It is the trends, rather than the actual numerical values, that are important.

Figure 3.5 and Figure 3.6 show the same experiment but with $R1 = 60m$. We can see that the number of dead nodes increases again when $\omega = 0.5$, and has the highest value when $\omega = 1.0$. Other ω's, however, have the same effect. Figure 3.7 shows comparison in detail between the two cases in term of dead node at track number equals 100. The number of dead nodes is high when $R1 = 30$ due to missing the target in this range, which requires the search area to be expanded and this imposes an additional energy cost. Figure 3.8 shows the corresponding track coverage of these two cases. We can see that as the number of dead nodes decreases, the track coverage increases.

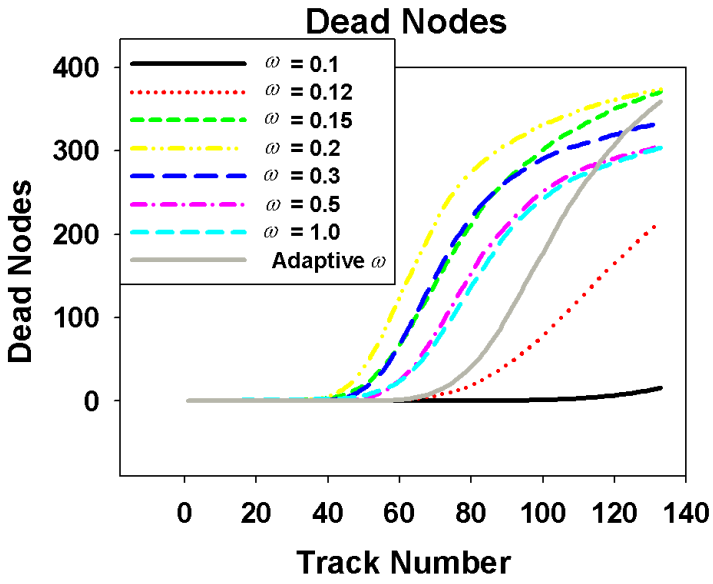


Figure 3.3. The Impact of ω on the Number of Dead Nodes ($R1 = 30$)

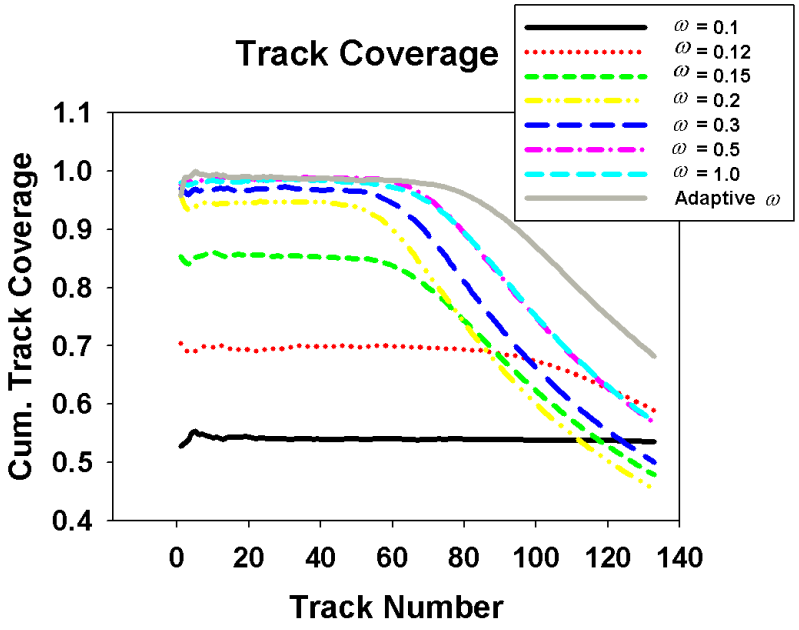


Figure 3.4. The Impact of ω on the Cumulative Track Coverage ($R1 = 30$)

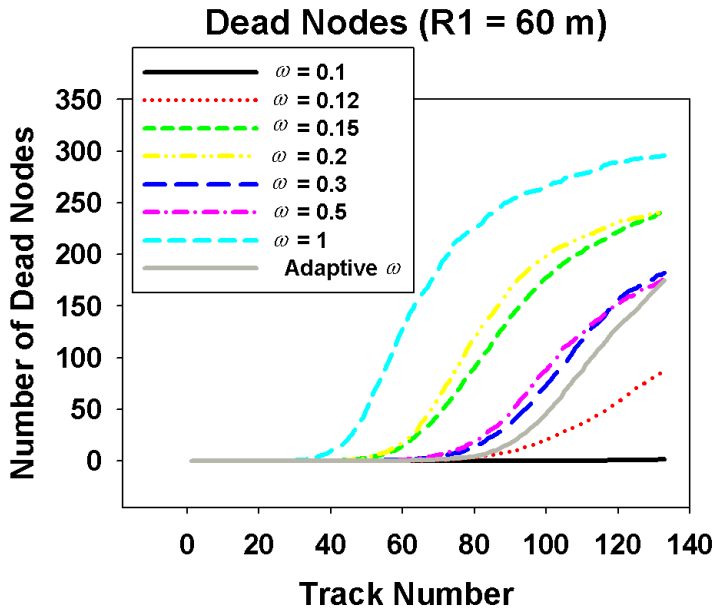


Figure 3.5. The Impact of ω on the Number of Dead Nodes ($R1 = 60$)

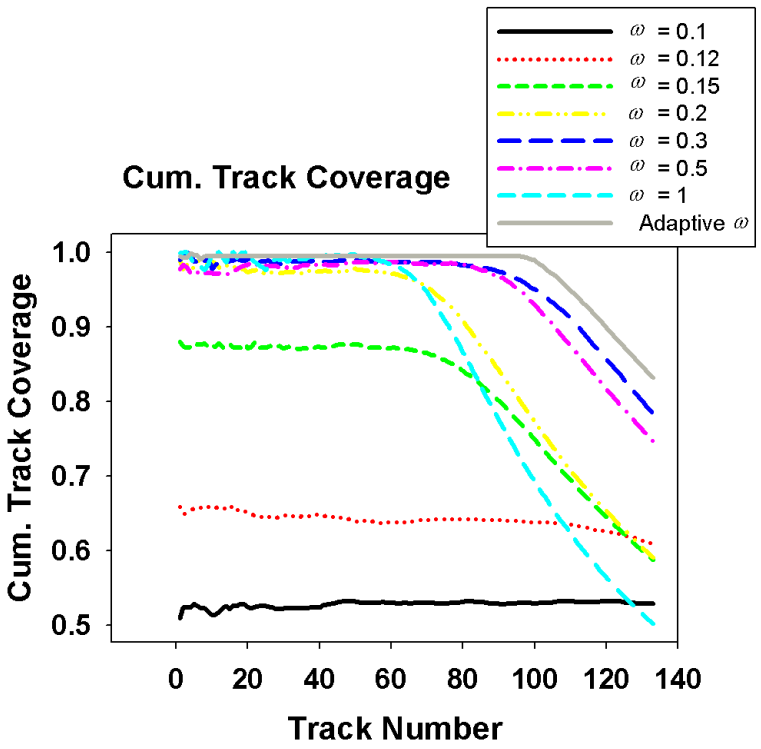


Figure 3.6. The Impact of ω on the Cumulative Track Coverage ($R1 = 60$)

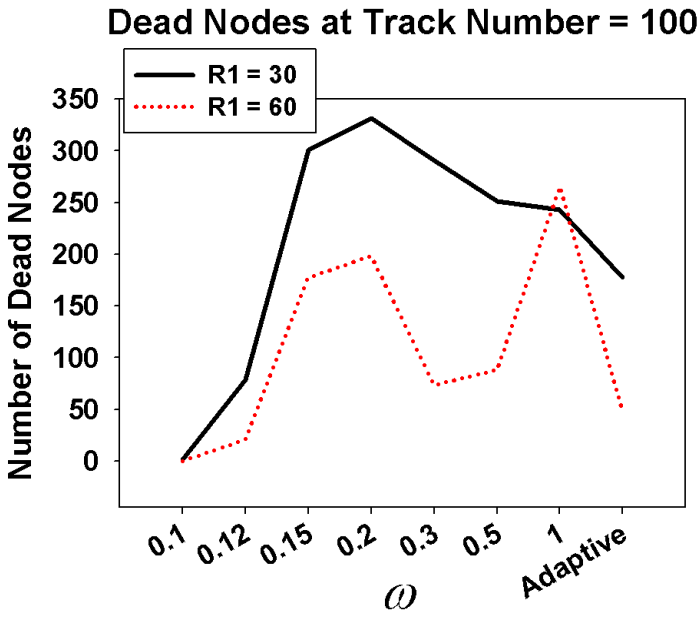


Figure 3.7. The Impact of ω on the Number of Dead Nodes

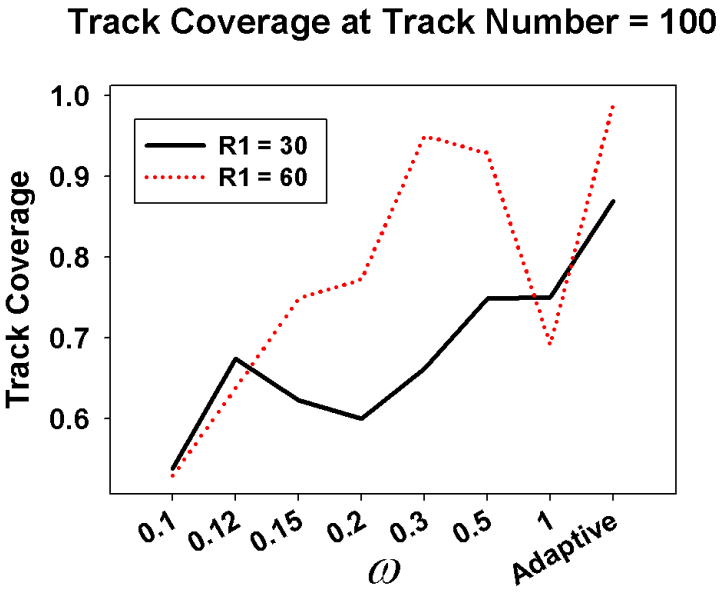


Figure 3.8. The Impact of ω on the Cumulative Track Coverage

3.2.2 Prediction Error

The prediction error depends primarily on three parameters: the accuracy of the position estimate, the mobility model followed by the target, and the sampling time step (i.e., the inverse of the frequency with which the network tries to track the target). We have already treated the accuracy of the position estimate. Here, we consider the impact of the mobility model and the sampling time step. It is rather obvious that as the target's track becomes ever more random, our ability to predict the track goes down and the prediction error increases. Similarly, the smaller the sampling time step, the better our prediction accuracy. This is due to two factors. The first is that as the time step decreases, the target has less of an opportunity to drastically change its position from what is expected. Secondly, with a reduction in time steps comes an increase in the number of position fixes that are taken, and this increased number of data points can improve accuracy. Figure 3.9 provides numerical results related to these parameters.

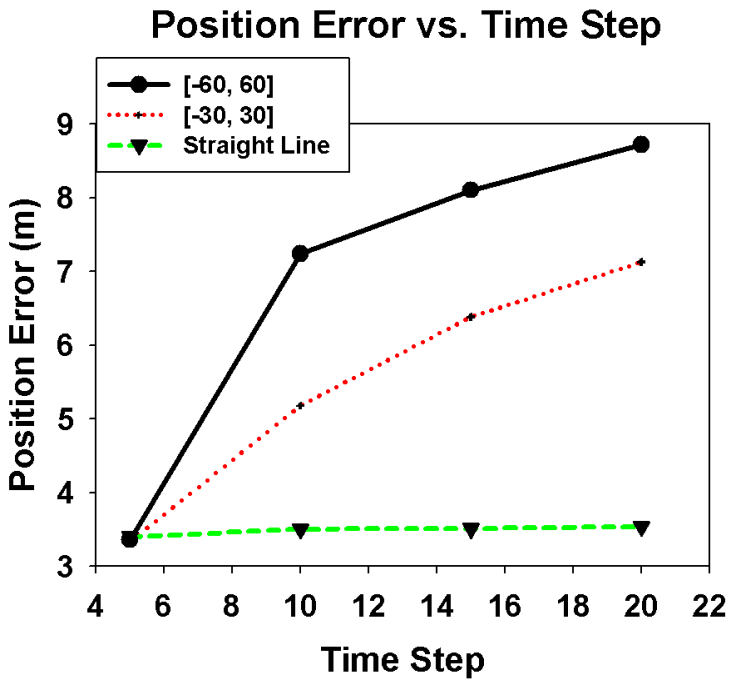


Figure 3.9. The Impact of the Mobility Model and the Sampling Time Step

3.2.3 Wakeup Range

We now consider the impact of the wakeup ranges. For simplicity, we concentrate on just the impact of just the first wakeup range, R1. The same remarks apply with respect to the other ranges.

R1 is the radius of the first circle of nodes that are woken up. If R1 is small, only a small number of sensors are awakened; as a result, only a small number of nodes expend sensing energy. However, the price to be paid is an increased chance of missing the target, in which case, this sensing step has been wasted and the search area has to be increased. Also, with an excessively small R1, even if the target is detected, the number of detecting nodes is likely to be small; this will have an impact on the accuracy of the estimate of the target position.

Figure 3.10 shows the probability of catching the target as a function of R1. As expected, this probability increases as R1 increases for a range of values, and then flattens out. Figure 3.11 shows the expected energy that is spent per sampling step. For very small values of R1, this energy is large because the probability of missing the target is high and a target miss triggers an expansion of the search area, which imposes additional energy costs. As R1 increases, the energy consumed drops, as the probability of catching the target increases. Beyond the point where any further increases in the target do not result in appreciable miss probability, the energy consumption increases because we are now waking up nodes over too wide an area. Note also the mean error in position estimate as a function of R1. For small values of R1, the probability increases that even those targets that are caught lie on the periphery of the waking circle; in such cases, the sensing circle is not entirely within the waking circle. This introduces a bias in the measurement, since the only nodes that can detect the target are those which lie in the intersection between the sensing and waking circles.

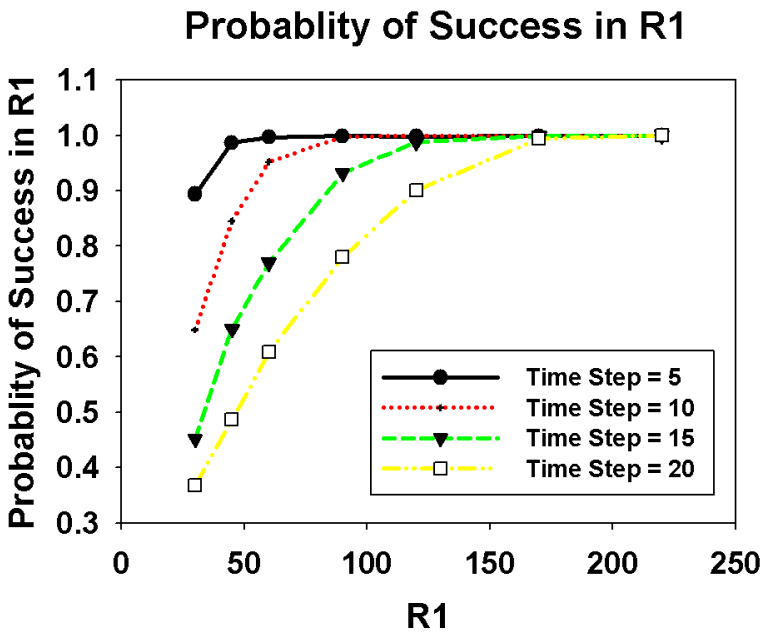


Figure 3.10. Probability of Success in R1

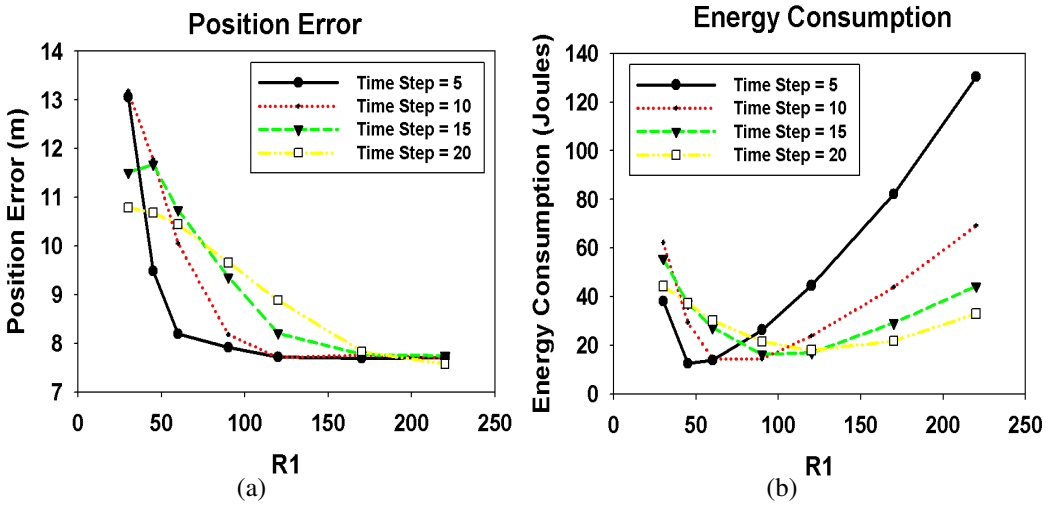
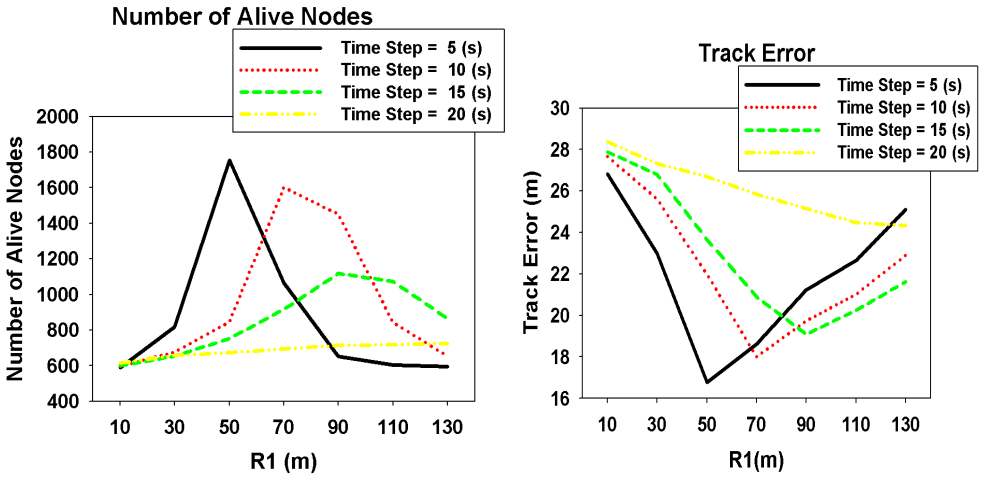


Figure 3.11. The Impact of Sampling Time Step on Track Error and Energy as a Function of R1

3.3 Impact of Sampling Interval

In this section, we study the impact of sampling time step on the number of live nodes and the track error as a function of the first wakeup range as shown in Figure 3.12. Track error is the average distance between the estimated and actual tracks. When R1 is very small, the probability of having to widen the search to the next wakeup radius is quite large. In such a case, the number of nodes that need to be awakened is greater and the drain on node energy reserves correspondingly large. As R1 increases, the need to switch to R2, R3, etc. reduces, with a corresponding reduction in energy demands on the nodes. This translates to a greater number of nodes still being alive. As R1 becomes very large, however, the number of nodes far away from the target position that are awakened every time also increases, and energy drain is high. Again, this reduces the number of nodes left alive after any given period of operation. For small time-steps, the error in the target estimate is small, and so the optimum R1 value at which the energy drain is minimized is smaller. The tracking error is also a strong function of both R1 and the time step.



(a) Number of Live Nodes (b) Track Error

Figure 3.12. The Impact of the Sampling Time Step on **(a)** Number of Live Nodes and **(b)** Track Error as a Function of R1

3.4 Impact of Mobility Model

In this section, we take into account the impact of a finite energy reserve in the performance of the system. In particular, we count the number of misses in R1 over a period of operation for different mobility models. A target is missed when there is no awake sensor within range of the target. This may be either because of the size or position of the wakeup circle or because of sensors within range running out of energy. When sensor networks run out of energy, some or all part of the track trajectory is not covered, creating some missing sensing points. This is accounted for in the second term of Equation 3.2.

$$TotalNumberOfMissesInR1 = MissesInR1 + MissingSensingPoints \quad (3.2)$$

Figure 3.13 shows the number of misses as a function of R1. When R1 is very small, the probability of having to widen the search to the next wakeup radius is quite large. In such a case, the number of nodes that need to be awakened is greater and the drain on node energy reserves correspondingly large. In this case, both terms in Equation 3.2 contribute heavily. As R1 increases, the need to switch to R2 reduces, with a corresponding reduction in energy demands on the nodes. This translates to a greater number of nodes still being alive and fewer misses in R1. As R1 becomes very large, however, the number of nodes far away from the target position that are awakened every time also increases, and energy drain is high. Again, this reduces the number of nodes left alive after any given period of operation. In this case the total number of misses in R1 is more contributed by the second term in Equation 3.2. It is rather obvious also that as the target's track becomes ever more random, our ability to predict the track goes down and the prediction error increases, which increases the number of nodes needed to be awakened and drains node energy reserves, which affects then the number of misses in R1.

Figure 3.14 studies the impact of mission time on the number of misses in R1. For short mission lifetimes, the optimum setting for R1 is quite large, since the constraints on

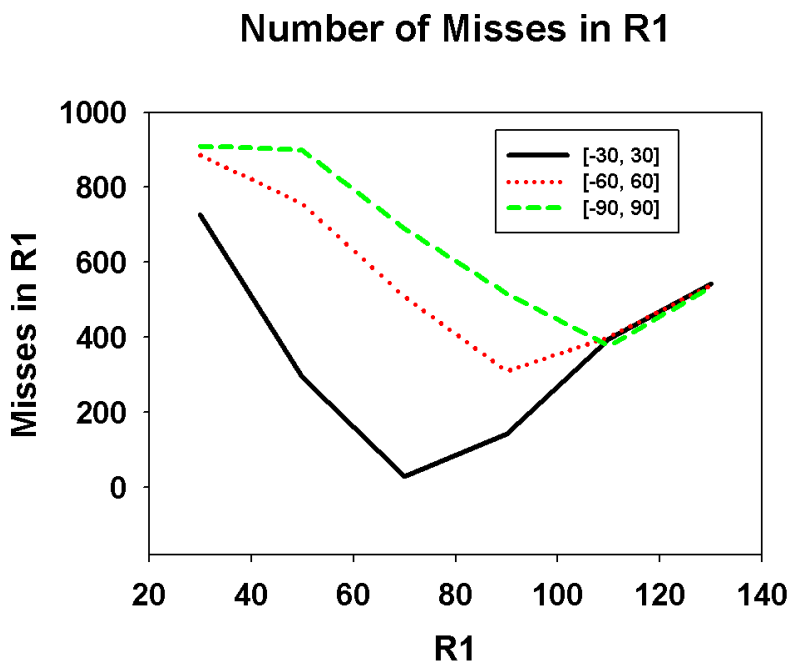


Figure 3.13. Number of Misses in R1 for 40 day mission lifetime

energy are not tight. As the mission lifetime increases, however, the chances of running out of energy increase and a less aggressive value of R1 is appropriate. Regardless of the value of R1, larger missions tend to cause a greater rate of misses since more nodes run out of energy and cause increasingly larger blind spots in the sensor network.

3.5 Assessing The Averaging Approach

Our algorithm uses simple averaging to estimate target position. In this chapter, we show that this simple approach is as good as, if not better than, more complex methods. We compare averaging against three other approaches: convex hull, linear weighting, and logarithmic weighting.

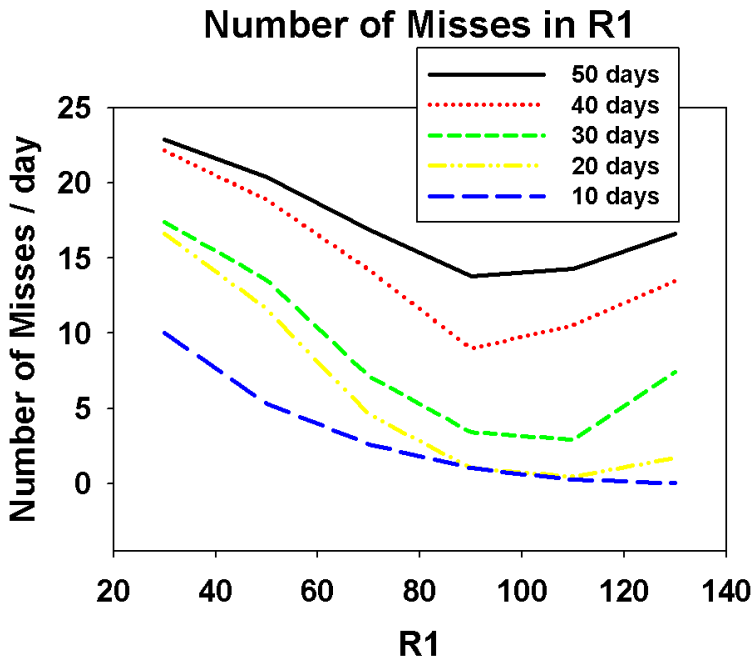


Figure 3.14. Number of Misses in R1 for 40 day mission lifetime

3.5.1 Convex Hull Approach

The convex hull of a set of points is the smallest convex area containing all these points [27]. We can calculate the convex hull of the detecting sensor nodes and estimate the target position as its centroid.

Computing the convex hull is a problem in computational geometry and several algorithms are available for computing the convex hull of a finite set of points, with various computational complexities. Computing the convex hull means that a non-ambiguous and efficient representation of the required convex shape is constructed. The complexity of the corresponding algorithms is usually estimated in terms of n, the number of input points, and h, the number of points on the convex hull [22].

We use an off-the-shelf efficient Convex Hull algorithm instead of implementing this algorithm from scratch in our simulation. To do so, a Swing java library implemented in Concord Consortium projects [2] is imported into our simulator. This library contains the Convex Hull algorithm and other useful algorithms.

Experiments were conducted to compare the convex hull approach to averaging. Sample simulation results are provided in Figure 3.15. These indicate that the convex hull algorithm has about the same performance as simple averaging.

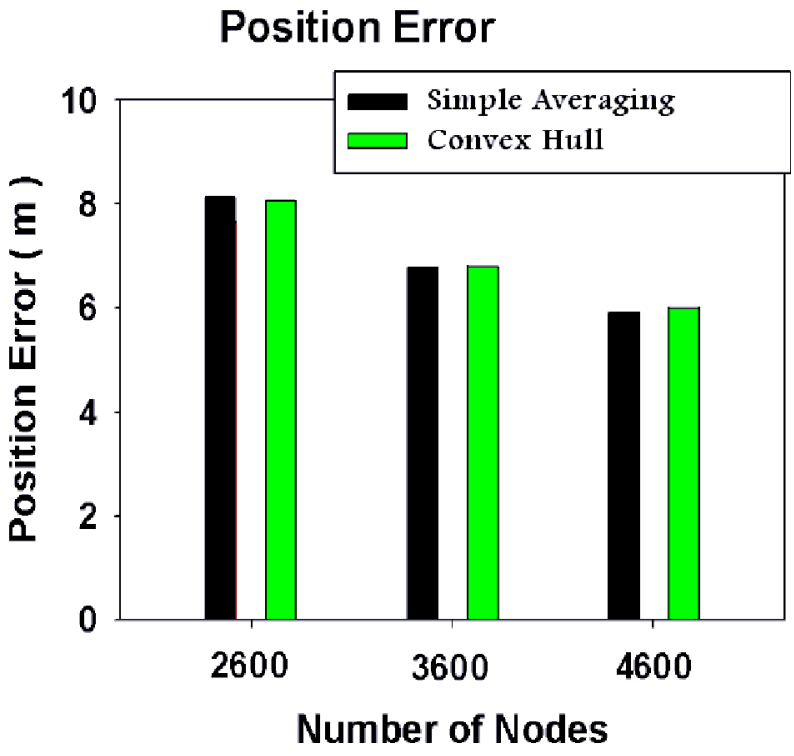


Figure 3.15. Simple Averaging vs. Complex Convex Hull

3.5.2 Linear and Logarithmic Weighting

Two interesting weighted averaging approaches have been proposed in [23]. This scheme estimates the distance of the sensor from the target and applies a weighting function that is monotonically decreasing with this distance. The distance estimate is obtained by [23] under the assumption that the target is moving in a straight line and that the sensor is always awake. In such a case, as Figure 3.16 shows, the distance between the sensor and the target can be related to the duration over which the sensor can sense the target. Proportional and logarithmic weighting apply weight that are linearly proportional to, and

a logarithm of, this duration, respectively. This weight factor is then normalized and then multiplied by sensor position to estimate the position of the target. Such a scheme cannot be exactly applied to our case, since we do not assume that sensors are always awake. As a result, a sensor cannot be sure to sense the target for the entire duration over which that target is within its sensing range (since it may have been asleep for part of that time. If we simply use the duration for which the sensor does sense the target for weighting purposes, we obtain a position error that is noticeably greater than under simple averaging. Even if we use the actual time over which the target is within the sensing circle (this is obviously not available to the sensor for reasons stated above), the quality of the position estimate (marked "Opt" in Figure 3.17) is not significantly better than that of simple averaging. NU, however, is the best in term of position error because of the increased waking nodes closer to the target.

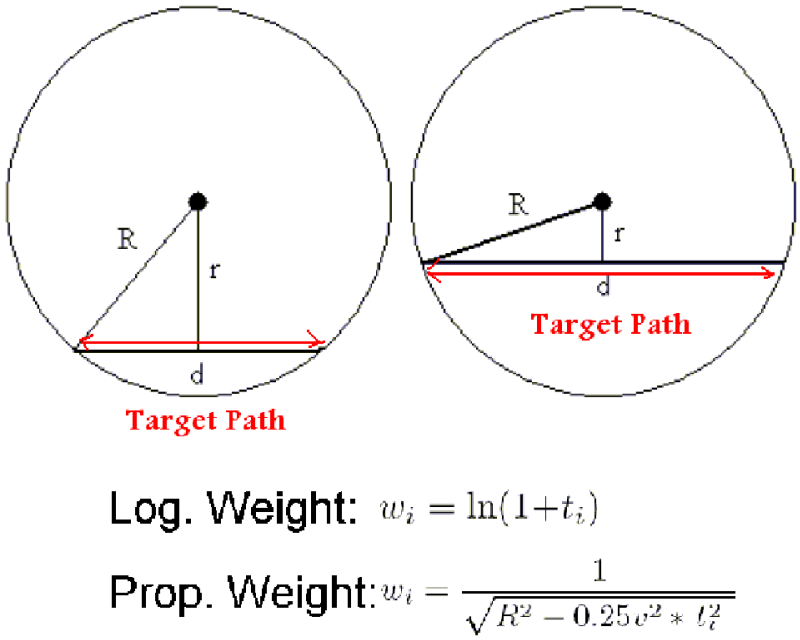


Figure 3.16. Closer Targets Spend More Time in Sensor's Field of View

Figure 3.17. Effect of Waking Schemes on Position Error

CHAPTER 4

ANALYTICAL MODELS

In this chapter, we provide analytical models for the expected and the tracking error probability of catching the target.

4.1 Impact of Node Density

We start by considering the impact of the number of detecting nodes on the error in the target position estimate. Intuitively, the greater the number of detecting nodes, the more accurate will be the position estimate. Here, we quantify this intuition.

For convenience, define the origin of the coordinate system as the object position. Picture the target as being at the center of a circle, with radius equal to the sensing range.

We start by determining the Probability Distribution Function (PDF) of the error in the estimate if there is only one active sensor over the entire sensing circle. To do so, we calculate the PDF of the x and y coordinates of this one sensor, randomly placed within the above-mentioned circle.

We start by obtaining the PDF of the x-coordinate; the derivation for the y-coordinate is similar. The first step is to calculate the area, W, of the shadowed region shown in Figure 4.1.

$$\begin{aligned} W &= 2 \times (Area \quad of \quad Sector - Area \quad of \quad Triangle) \\ &= 2\left(\frac{\theta}{2}S^2 - \frac{1}{2}\alpha S \sin(\theta)\right) \end{aligned}$$

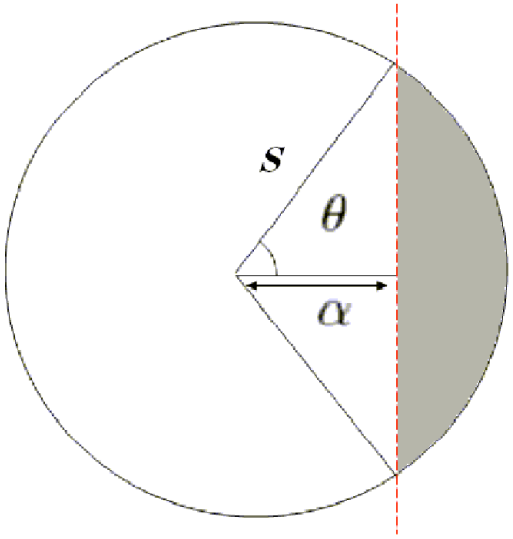


Figure 4.1. Sensing Circle

$$= S^2 cos^{-1}\left(\frac{\alpha}{S}\right) - \alpha S \sin\left(cos^{-1}\left(\frac{\alpha}{S}\right)\right) \tag{4.1}$$

The probability distribution function for the x-coordinate of the sensor, $F_X(\alpha)$, can now be derived as follows:

$$\begin{aligned} F_X(\alpha) &= 1 - P[X > \alpha] \\ F_X(\alpha) &= 1 - \frac{W}{Area \quad of \quad Circle} \\ &= 1 - \frac{S^2 cos^{-1}\left(\frac{\alpha}{S}\right) - \alpha S \sin\left(cos^{-1}\left(\frac{\alpha}{S}\right)\right)}{\pi S^2} \end{aligned} \tag{4.2}$$

Differentiating this distribution function yields the probability density function (pdf) of the x-coordinate of the single detecting sensor within the sensing circle:

$$f_{x_1}(\alpha) = \frac{2\sqrt{S^2 - \alpha^2}}{\pi S^2} \tag{4.3}$$

If we have a total of n awake sensors within the sensing circle, the estimate of the target's x-coordinate is the average of the x-coordinates of these sensors. By symmetry, the pdf of the x-coordinate of each of these sensors is the same as in Equation 4.3; furthermore, the sensor positions are assumed to be independent of one another. As a result, we can obtain the pdf of the average of the sensor positions by convolution. In particular, the pdf of the sum of the x-coordinates is

$$f_{x_1+..+x_n}(\beta) = \int_{max(\beta-S,-(n-1)S}^{min(\beta+S,(n-1)S)} f_{x_2+..+x_{n-1}}(\alpha) f_{x_1}(\beta-\alpha) d\alpha \tag{4.4}$$

The density function of the average x-coordinate can now be written as:

$$f_{\frac{x_1+..+x_n}{n}}(\beta) = n f_{x_1+..+x_n}(n\beta) \tag{4.5}$$

As mentioned before, an identical argument applies to the average y-coordinate. We can now write the pdf of the position estimate error as follows:

$$MeanPositionError = \int_{-S}^{S}\int_{-S}^{S} \sqrt{\alpha^2+\beta^2} f_{\bar{x}}(\alpha) f_{\bar{y}}(\beta) d\alpha d\beta \tag{4.6}$$

Figure 4.2 and Figure 4.3, respectively, show the density functions and mean position errors for one to up to four sensors detecting the target. There is a marked improvement in the quality of the estimate for two, as opposed to just one, detecting node. Further gains in accuracy are more limited. Our results indicate that as long as at least three nodes can detect a target, its position estimate will be quite accurate; further marginal improvements will be small. We do not therefore require a large number of sensors to be awake within the sensing circle of the target. This fact can guide us in setting some of the parameters of the tracking algorithm.

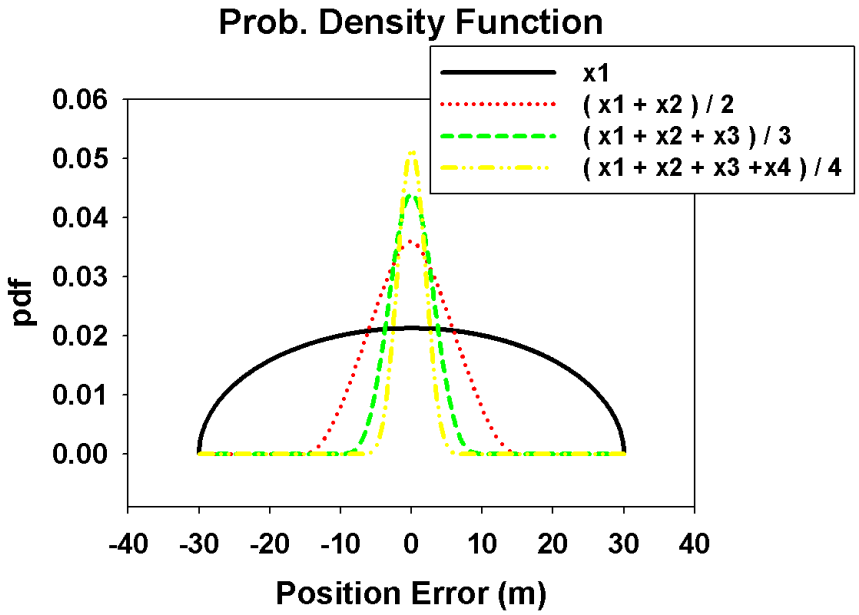


Figure 4.2. Density Function of Position Error

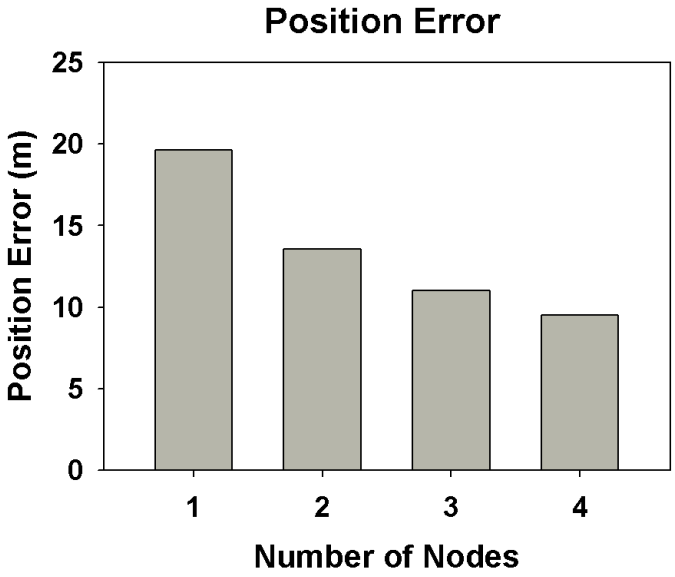


Figure 4.3. Mean Position Error for 30(m) Sensing Radius

4.2 Modeling Track Mobility

In this section, we model the probability of catching the target in the first wakeup range R1, and the expected energy consumption. As discussed earlier, our mobility model follows the Random Walk Model, where the target changes its direction every T seconds forming a piecewise linear segment as shown in Figure 4.4. We assume that the target speed s is fixed here[1]. Each such linear segment is at an angle θ with respect to its predecessor, and is uniformly distributed over a certain range.

First of all, we calculate the probability density function (pdf) of the next position of the target after n segments as a function of the previous and current sensing points.

In other words, we want to compute the projected *pdf* $\Delta x_{n+1} + \cdots + \Delta x_{2n}$ and $\Delta y_{n+1} + \cdots + \Delta y_{2n}$ as a function of the current *pdf* $\Delta x_1 + \cdots + \Delta x_n$ and $\Delta y_1 + \cdots + \Delta y_n$ as shown in Figure 4.4.

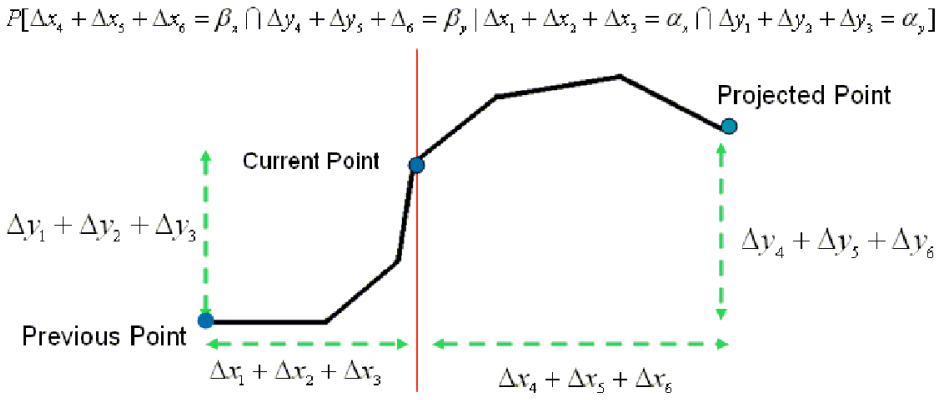


Figure 4.4. *pdf* of Projected Point Given the Current Point

We start by defining the indicator function:

[1]Extending this model by assuming randomly varying speed can be done by applying Bayes's Law and integrating over the range of allowed speed.

$$1(\gamma) = \begin{cases} 1 & \text{if } \gamma \text{ is } \textit{true} \quad (4.7a) \\ 0 & \textit{otherwise} \quad (4.7b) \end{cases}$$

Then, the probability density function of the projected position (β_x, β_y) given that the current position is (α_x, α_y) can be written as follows:

$$\begin{aligned} P[\beta_x, \beta_y | \alpha_x, \alpha_y] \quad = \quad & \frac{1}{(\phi_{up} - \phi_{lo})^n} \int_{\phi_{lo}}^{\phi_{up}} \cdots \int_{\phi_{lo}}^{\phi_{up}} 1(d\cos(A_{n+1}) + \cdots + d\cos(A_{2n}) = \beta_x \\ & AND \quad d\cos(A_0) + \cdots + d\cos(A_n) = \alpha_x) \quad 1(d\sin(A_{n+1}) + \cdots + d\sin(A_{2n}) = \beta_y \\ & AND \quad d\sin(A_0) + \cdots + d\sin(A_n) = \alpha_y) d\theta_{2n} \cdots d\theta_{n+1} \quad (4.8) \end{aligned}$$

Where $d = sT$, ϕ_{up} and ϕ_{lo} are the upper and lower bounds of the displacement angle θ_i; successive values of θ_i are independent. A_j is the absolute angle, and can be derived from the displacement angle θ_i as follows:

$$A_j = \sum_{i=0}^{j} \theta_i \quad (4.9)$$

The *pdf* of the estimated current target position can be written according to the following equation:

$$\begin{aligned} P[\alpha_x, \alpha_y] \quad = \quad & \frac{1}{(\phi_{up} - \phi_{lo})^n} \int_{\phi_{lo}}^{\phi_{up}} \cdots \int_{\phi_{lo}}^{\phi_{up}} 1(d\cos(A_0) + \cdots + d\cos(A_n) = \alpha_x) \\ & 1(d\sin(A_0) + \cdots + d\sin(A_n) = \alpha_y) d\theta_0 \cdots d\theta_n \quad (4.10) \end{aligned}$$

To find the catch probability in a given wakeup range R1, we need to compute the prediction point (x_p, y_p) based on the current target position (α_x, α_y). This can be done using a simple linear extrapolation. The probability of catching the target as a function of the current position is shown in the following equation:

$$P[Catch \quad in \quad R1|\alpha_x, \alpha_y] = \int_{-2dn}^{2dn}\int_{-2dn}^{2dn} 1(\sqrt{(\beta_x - x_p)^2 + (\beta_y - y_p)^2} \leqslant R_1) P[\beta_x, \beta_y|\alpha_x, \alpha_y]d\beta_x d\beta_y \tag{4.11}$$

Unconditioning on all possible values of current positions (α_x, α_y) we obtain the probability of finding the target in R1.

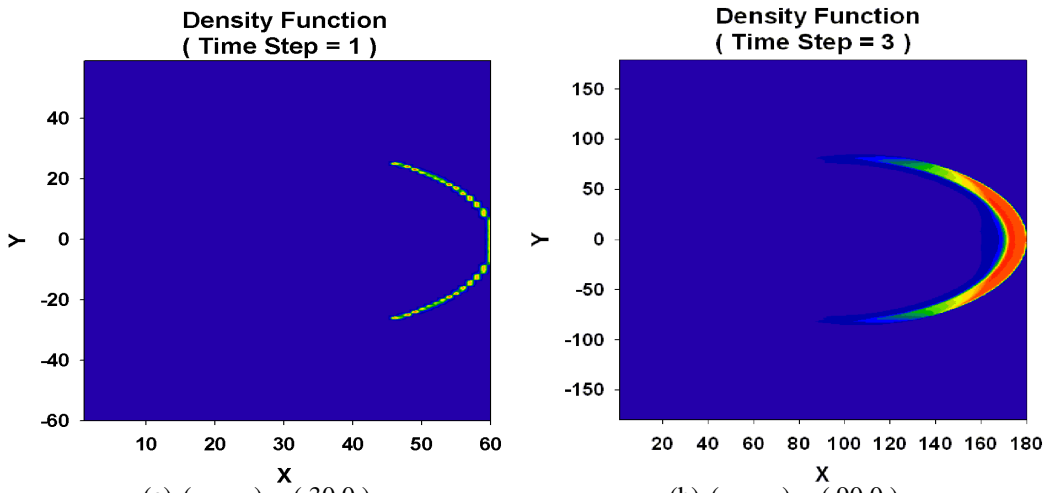
$$P_{CatchInR_1} = \int_{-dn}^{dn}\int_{-dn}^{dn} P[Catch \quad in \quad R1|\alpha_x, \alpha_y]P[\alpha_x, \alpha_y]d\alpha_x d\alpha_y \tag{4.12}$$

Finally, the expected energy can be calculated using the following equation:

$$ExpectedEnergy = (P_{CatchInR_1}R_1^2 + (1 - P_{CatchInR_1})R_{all}^2)\pi\rho E \tag{4.13}$$

where ρ is the node density and E is the energy consumption per node.

In Figure 4.5, we present results for the *pdf* of the target position after one, and three, steps, respectively. In each case, we assume that the step preceding the movement represented here was along the horizontal direction, i.e., with an absolute angle of 0. A cross-section view at different X values is depicted in Figure 4.6. This figure shows that the possibility of the target goes higher as the value of X coordinate increases, and then reduces after passing 180 m.



(a) $(\alpha_x, \alpha_y) = (30,0)$ (b) $(\alpha_x, \alpha_y) = (90,0)$

Figure 4.5. Density Function for Projected Target Position with Mobility Model [-60, 60] for Next Segments, and $d = sT = 5 \times 6 = 30m$

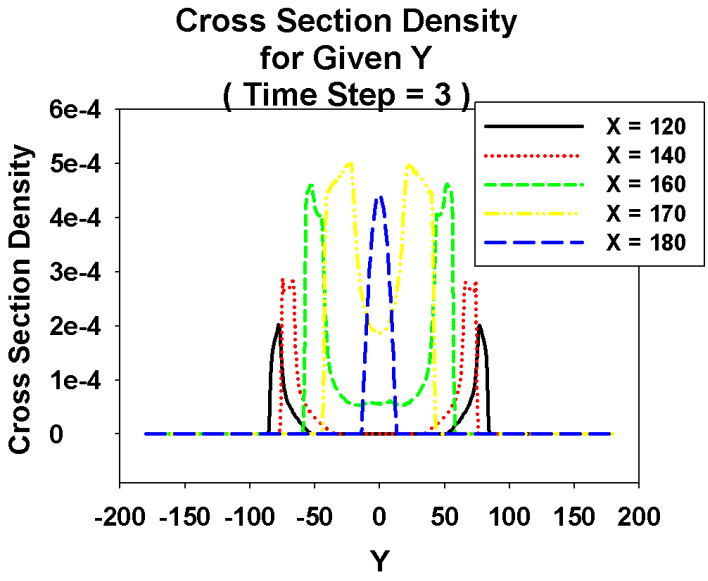


Figure 4.6. Cross Section Density

The effect of the mobility model (ϕ_{lo}, ϕ_{up}) on the catch probability in R1 and expected energy consumption is described in Figure 4.7. We can see that as the uncertainty of track increases, our ability to catch the track is reduced. As R1 becomes larger, the probability of catching also increases since the target becomes more likely to fall within. As expected, the optimum R1 (for minimum expected energy consumption) depends on the mobility model: the greater the intrinsic uncertainty of the track, the greater the optimum value of R1. The reason is quite simple to explain. When R1 is very small, the probability of having to widen the search to the next wakeup radius is quite large, and increases with the track uncertainty (which is represented in our model by the maximum angular deflections of each segment). In such a case, the number of nodes that need to be awakened is greater and the drain on node energy reserves correspondingly larger; the second term in Equation 4.13 becomes more important. As R1 increases, the need to switch to R2 reduces, with a corresponding reduction in energy demands on the nodes. As R1 becomes very large, however, the number of nodes far away from the target position that are awakened every time also increases, and energy drain is high. In this case, the first term in Equation 4.13 dominates.

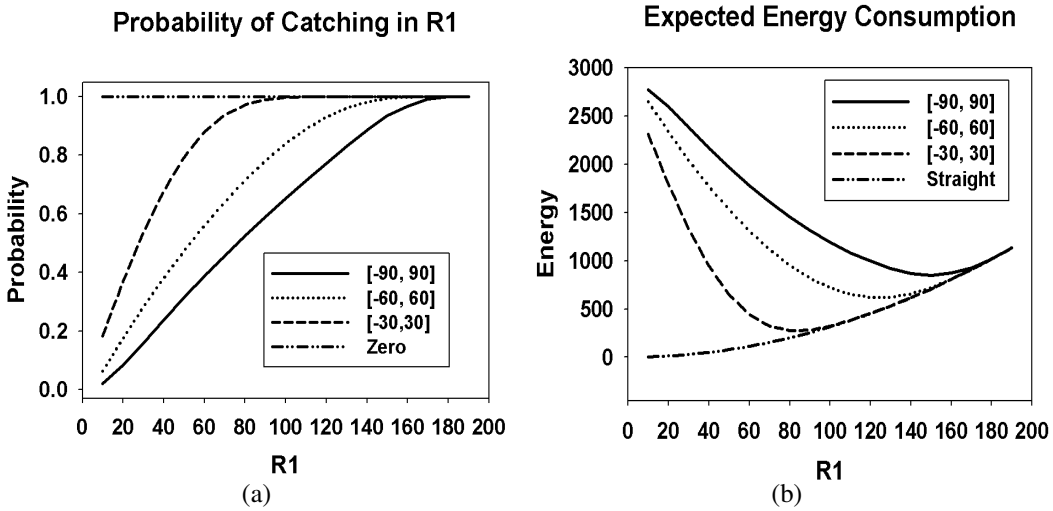


Figure 4.7. Probability of Success in R1 and Expected Energy Consumption for Different Mobility Models

We now turn to the case where there are three wakeup ranges (the third, R3, here involves waking up all the nodes in the system that still have sufficient energy to function). Figure 4.8 and Figure 4.9 provide some numerical results. Selecting the value of R1 that offers the lowest energy in the two-level case results in constant value of expected energy regardless of the value of R2 shown in Figure 4.8, this is because the catch ratio at this optimum R1 almost equals one. When we set $R1 = 70$ as shown in Figure 4.9, for example, we can have a lower energy cost for certain R2 values compared when we have the optimum R1 value. The point of Figure 4.8 is that selecting the optimum R1 does not always provide the minimum expected energy for different value of R2 as shown in Figure 4.9.

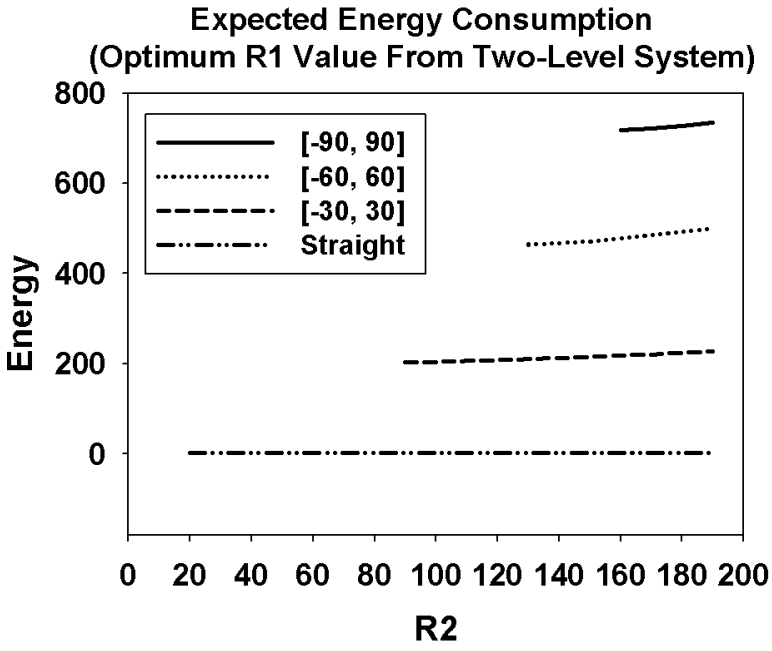


Figure 4.8. Expected Energy When R1 is Optimum

Figure 4.10 shows the catch probability in R1 for constant speed ($s = 5m/s$) and randomly varying speed over the interval [2, 8] m/s. As expected, the speed uncertainty causes a decrease in the catch probability. This change, for obvious reasons, becomes insignificant as R1 increases

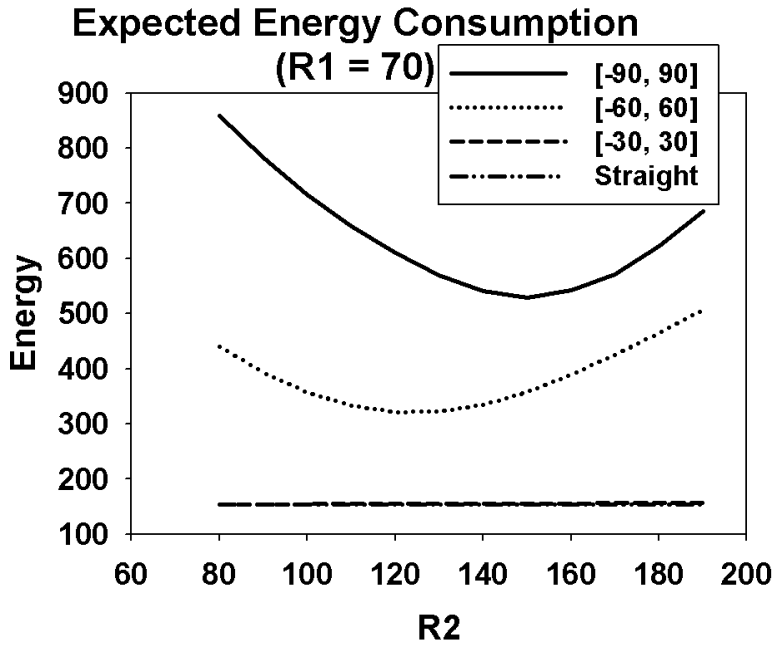


Figure 4.9. Expected Energy When R1 = 70

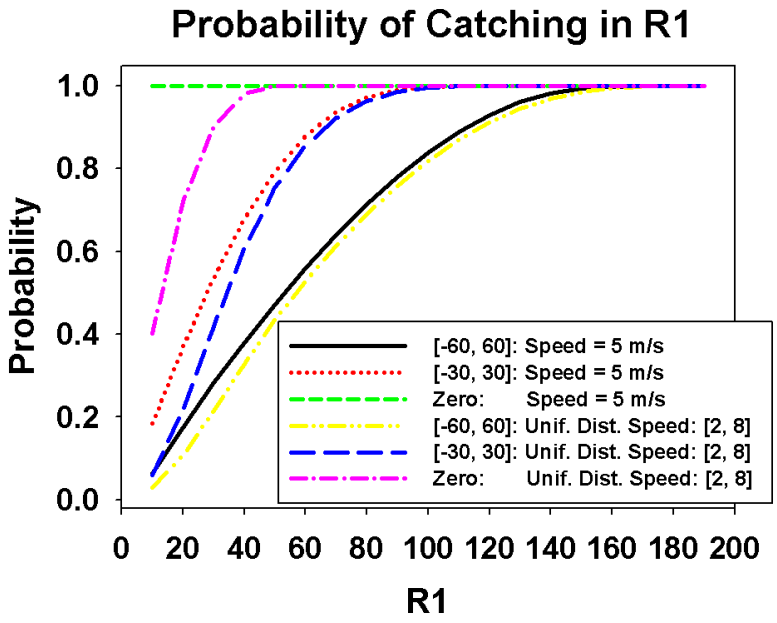


Figure 4.10. Catch Probability as Function of Constant and Uniformly Distributed Speed

CHAPTER 5

EXTENDING THE BASIC ALGORITHM

The tracking algorithm presented before can be extended to improve its performance and efficiency. In this chapter, we present some such promising extensions.

5.1 Non-Uniform Waking

Rather than wake up every node within the wakeup circle with the same probability, we can wake up more nodes closer to the anticipated location of the target, while keeping the rest of the area covered less densely. The former allows the target to be located more precisely in the likely event that the prediction error is limited; the latter ensures a lower target miss probability if the predicted point is distant from the actual target location.

Perhaps the simplest non-uniform strategy is to have the wakeup probability decline linearly as we move away from the predicted target location.

$$p(r) = \left(1 - \frac{r}{R}\right) q \tag{5.1}$$

where q is a control parameter. The expected number of waking nodes is given by

$$\begin{aligned} N(R, \rho, q) &= \int_0^R 2\pi r p(r) \rho dr \\ &= \frac{1}{3}\pi R^2 \rho q \end{aligned} \tag{5.2}$$

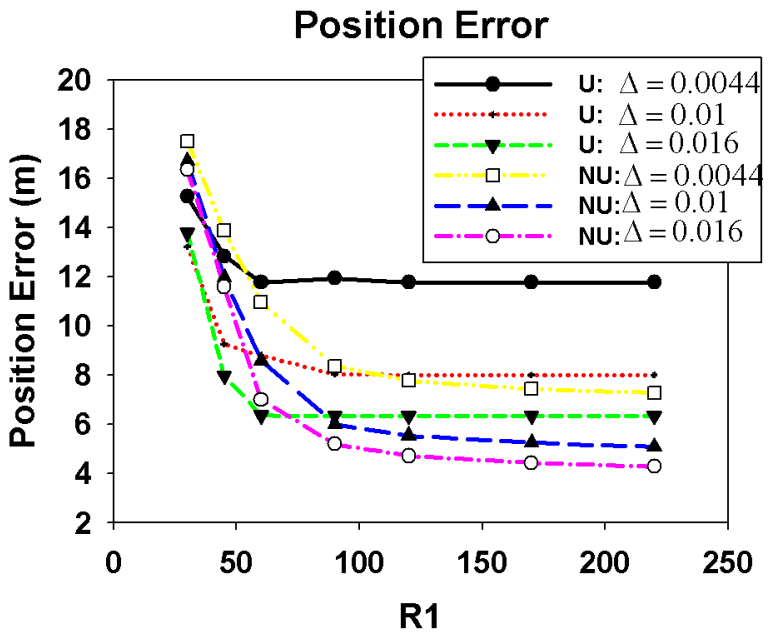


Figure 5.1. The Impact of Node Density on Position Error as a Function of R1

Figure 5.1 compares the behavior of the Non-Uniform (NU) against that of the Uniform (U) model we presented previously. To keep the comparison fair, the value of q is set so that the expected total number of waking nodes in the U and NU cases are the same.

For the same value of R1, for example, the probability of locating the target is roughly the same: this depends on R1 and not on the node density (so long as there is at least one node in the sensing area of the target). Similarly, since the expected number of waking nodes is the same, the energy consumed is about the same. However, there is a significant effect on the accuracy with which the position error is estimated, for larger values of R1. The NU approach allows an inner circle to be densely populated by awakened nodes: this is compensated for by making the outer area more sparsely covered. When R1 is large enough, the inner, denser, circle is large enough to capture most target instances and the average position error is reduced.

For a similar reason, the advantage of the NU over the U case is increased when the prediction accuracy is greater; indeed, the purpose of NU is to squeeze additional perfor-

mance out of accurate prediction. Figure 5.2 shows the estimated position accuracy as a function of the variance in the target mobility model. The lower the variance the greater the probability that the target will be in the more densely populated area near the predicted position, and the better will be the NU performance. For a purely random track, NU is not recommended.

An insufficient awakened node density is not the only reason that NU can behave poorly if there is substantial error in predicting the target position. Another is a biasing caused by the non-uniform positioning of the awakened nodes. Figure 5.3 illustrates this. Because the prediction error in this case is considerable, there are more awakened nodes on one side of the actual target position than in others. Therefore, any averaging of the positions of all those nodes which detect the target will have an inherent bias. One can correct for this bias by weighting each node position by the inverse of the awakened node density in that location and then averaging over these weighted positions.

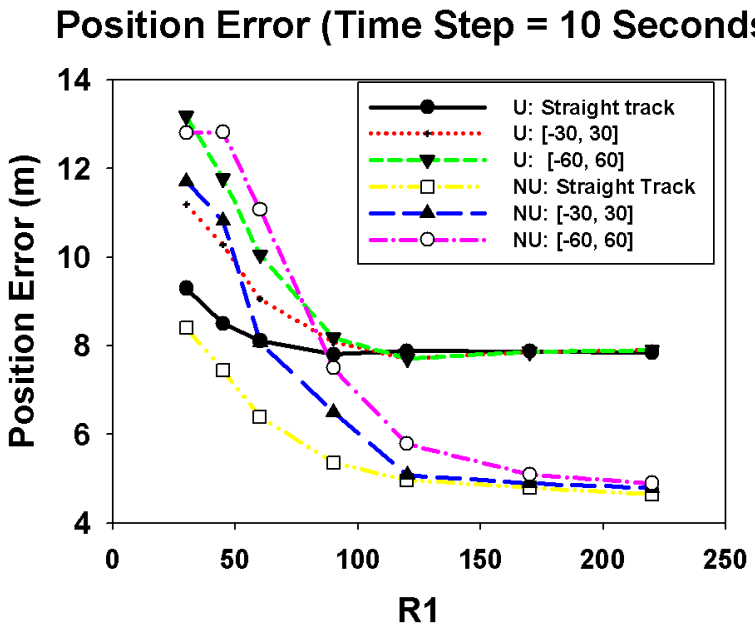


Figure 5.2. Impact of Track Uncertainty on N vs. NU Performance

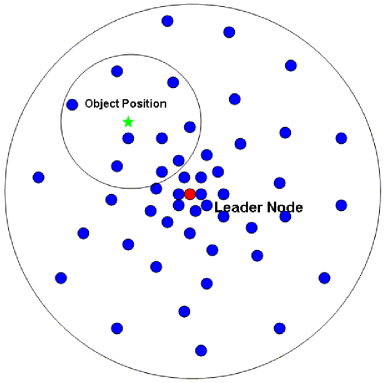


Figure 5.3. Illustrating the Bias Effect

The impact of inaccurate position estimation (and the bias effect) on the NU approach is quantified in Figure 5.4. The leader node is the one closest to the predicted target position. If this distance is small, the increased density of nodes around the actual target position renders NU more accurate than U; as this distance increases, however, NU behaves more poorly.

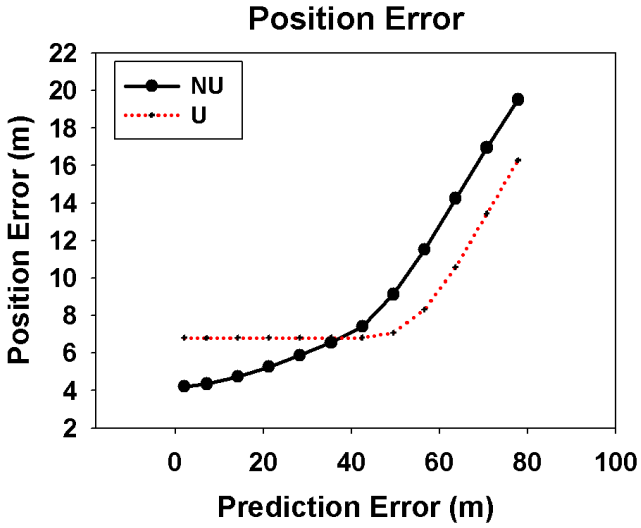


Figure 5.4. Position Error under U and NU Approaches

5.2 Filtering Out False Alarms

Nodes are susceptible to noise and noise can give rise to false reports of target acquisition. This false reporting has the potential to degrade the accuracy of the tracking if it is not filtered out. We present here a simple noise filtering algorithm (shown in Algorithm 1) that attempts to throw out noisy nodes from target position estimation.

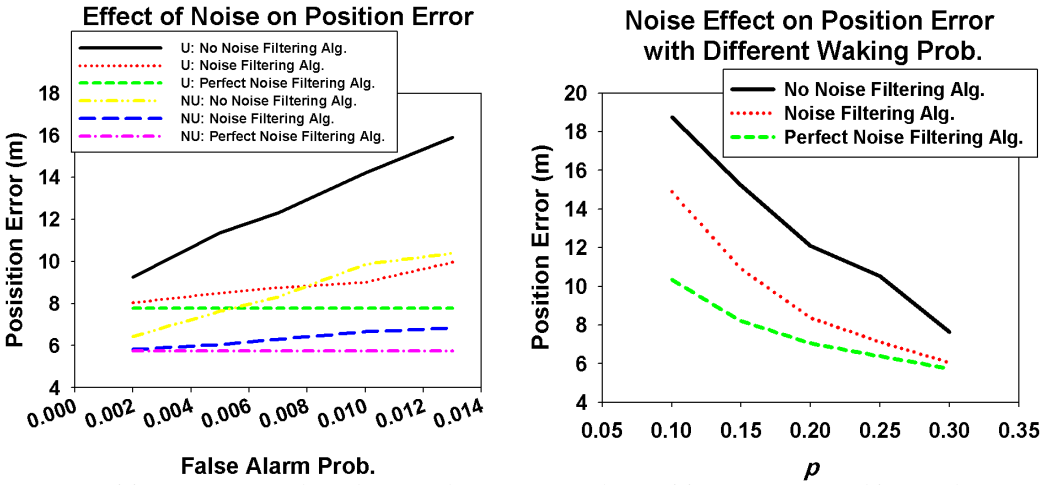
This is an iterative process, executed by the leader node. The algorithm estimates the target position based on all the reports sent by the nodes. Then, it filters out reports from nodes whose distance from that estimated target position exceeds the sensing diameter ($2 \times$ sensorRange). This results in a new estimate of target position, and the process of filtration can be iterated until no reports have to be filtered out.

Input: X, Y : Coordinate Vectors for Detecting Nodes
Output: x, y: Estimated Target Position After Filtering Noisy Nodes
$x = 0$; $y = 0$; $size = 0$;
for $i = 0$ *to Detecting Node Size* **do**
    $x = x + X[i]$; $y = y + Y[i]$; $size = size + 1$;
end
$x = x/size$; $y = y/size$;
for $i = 0$ *to Detecting Node Size* **do**
    $d = \sqrt{(X[i] - x)^2 + (Y[i] - y)^2}$;
    if $d \geq 2 \times SensorRange$ **then**
        $X[i] = NULL$; $Y[i] = NULL$;
    end
end
$x = 0$; $y = 0$; $size = 0$;
for $i = 0$ *to Detecting Node Size* **do**
    if $X[i] \neq NULL$ **then**
        $x = x + X[i]$; $y = y + Y[i]$; $size = size + 1$;
    end
end
$x = x/size$; $y = y/size$;

Algorithm 1: Noise Filtering Algorithm

Figure 5.5 shows the effect of noisy nodes on position error. It can be seen in Figure 5.5(a) that noise filtering algorithm improves position error significantly compared with the no filtering case for both U and NU waking schemes. We can see, in general, that the NU scheme performs better than the U scheme because of the higher waking density sur-

rounding the current target position. So long as the estimated target position is not far away from the actual position, the NU scheme will ensure that few nodes far away from the target will be awake. As a result, there will simply be fewer awake nodes outside the sensing range of the target. Figure 5.5(b) shows the effect of noise as we increase waking probability.



(a) Position Error vs. False Alarm Prob.

(b) Position Error vs. Waking Prob.

Figure 5.5. Noise Effect on Position Error

5.3 Adaptive Wakeup Probability

Rather than have the wakeup probability fixed, we can make it adapt to the energy level of each node. One simple approach is to set the wakeup probability equal to the ratio of the energy level at the node to its maximum energy level. This is a rather crude adaptation scheme, but tends to balance the energy expenditure among the nodes and is easy to implement and justifies itself in terms of performance (see again Figure 3.3 and Figure 3.4 in Section 3.2.1).

5.4 Energy Harvesting

Energy harvesting consists of drawing energy from the operating environment. In our energy harvesting model, we assume that each sensor node can scavenge energy from the sun. We assume that there is a harvesting period during the day. If the sky is cloudless, the harvested power rises linearly in the morning until it reaches its maximum value, at which it holds steady for some time before then going down linearly. The maximum value for any given day is generated using a Uniform distribution over a certain range. To add the effect of the clouds in this model, we assume different levels of atmospheric transparency. One can model amplitude transparency levels by means of a Markov model [24], which is represented as a state diagram with each state corresponding to a distinct level of attenuations, ranging from none (for a clear day) to almost complete darkness. The harvested power is given by Equation 5.3. Figure 5.6 shows an example of a sample path of energy generated by harvesting using Equation 5.3 during three days.

$$P_{actual}(t) = P_{cloudless}(t) \times Transparency(t) \tag{5.3}$$

Figures 5.7 shows results similar to those of Figure 3.3 and Figure 3.4; however, here, the energy reserves are periodically augmented by energy harvesting. Small values of ω result in a greater chance of missing the target and hence require the search area to be widened, thereby consuming more energy. On the other hand, very large values of ω result in too many nodes being awake in the first place, which also costs energy. We can see that the adaptive ω still the best in term of track coverage due to its adaptive behavior to balance energy consumption. In contrast to the non-adaptive scheme, a node which runs out of energy is only temporarily disabled: once its energy levels can be replenished by harvesting, it comes back to service.

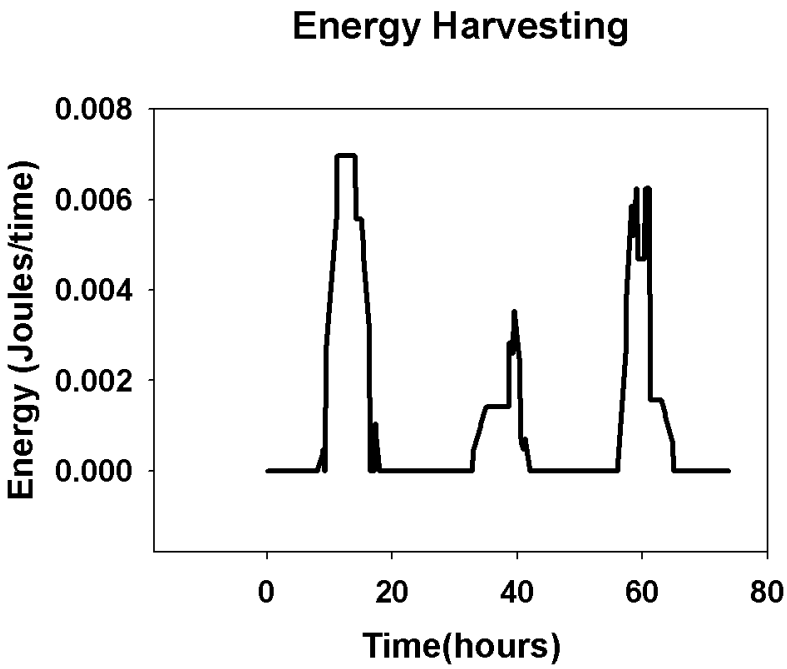
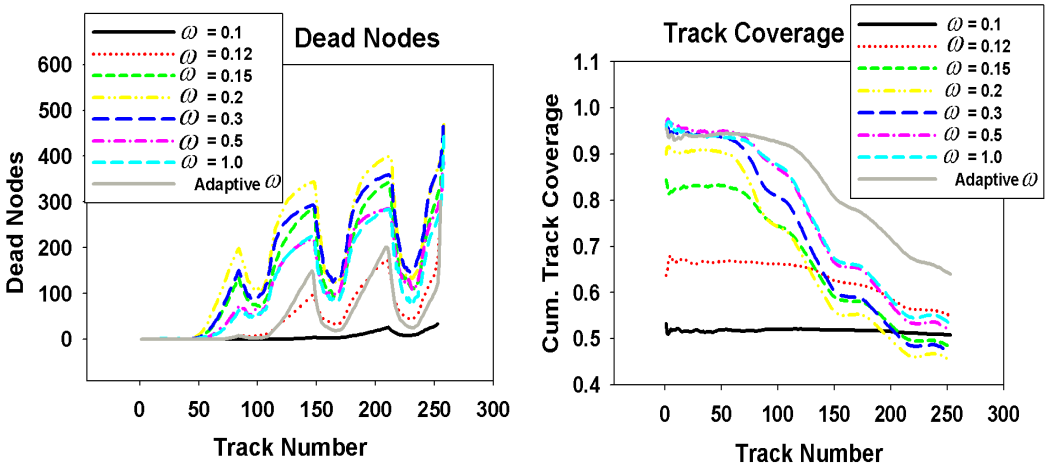


Figure 5.6. Sample Path of the Generated Energy by Harvesting



(a) Dead Nodes (b) Track Coverage

Figure 5.7. The Impact of on the Number of Dead Nodes and Cumulative Track Coverage with Energy Harvesting

5.5 Multiple Thresholds

In this approach, each node uses its available energy reserves to decide whether or not to participate in the tracking. In particular, each node constructs a linear threshold function $Threshold(t) = mt + b$, where if the current energy level of the node is greater than the current threshold value at time t, then the node is allowed to wake up and take part in sensing and communication. The slope m of the threshold function is computed based on the fraction of the start energy and the target energy level at end of mission lifetime, which usually equals zero. The fractional value of start energy depends on the current wake-up range (R). The common rule of thumb is to increase this fraction as we increase R to conserve energy, since if R is large many nodes will likely be involved in sensing. Clearly in this approach each node needs to know the mission time. On the other hand, $Threshold(t) = 0$ (baseline approach) if multiple thresholds are not used. Figure 5.8 shows an example of multiple threshold functions.

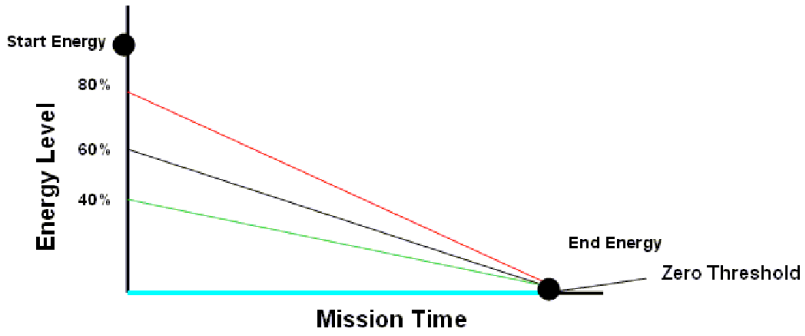


Figure 5.8. Multiple Threshold functions

In this experiment, we use multiple threshold algorithm for a given wake up range (R). Figure 5.9 shows the cumulative track coverage with respect to the track number arrived. For the baseline approach ($\omega = 1$), we have the best track accuracy in the beginning compared with the multiple threshold approaches; having the baseline starts degrading in performance very early due to the increased of number of dead nodes as shown in Fig-

ure 5.10. For multiple thresholds, having lower multiple thresholds improves track quality at the beginning of the period of operation, but it increases the chance of missing the target later on due to energy depletion in the network. Having higher multiple thresholds, however, results in low track quality in the beginning of mission time, but lower number of dead nodes. We compare the result also with the adaptive ω, we can see that it is the best in term of track accuracy but has more number of dead node.

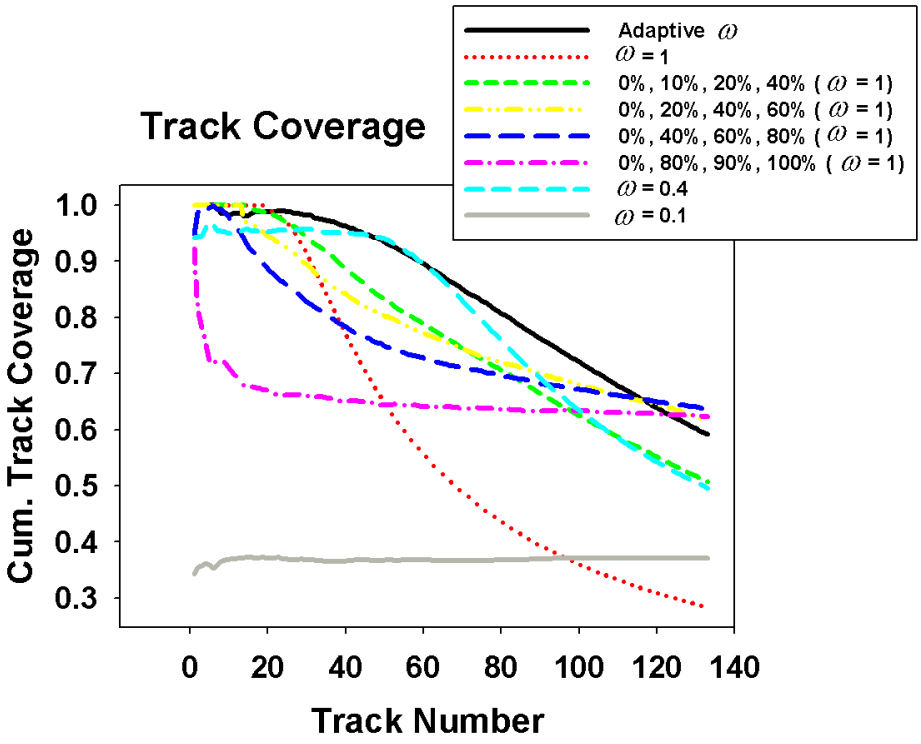


Figure 5.9. Track Coverage

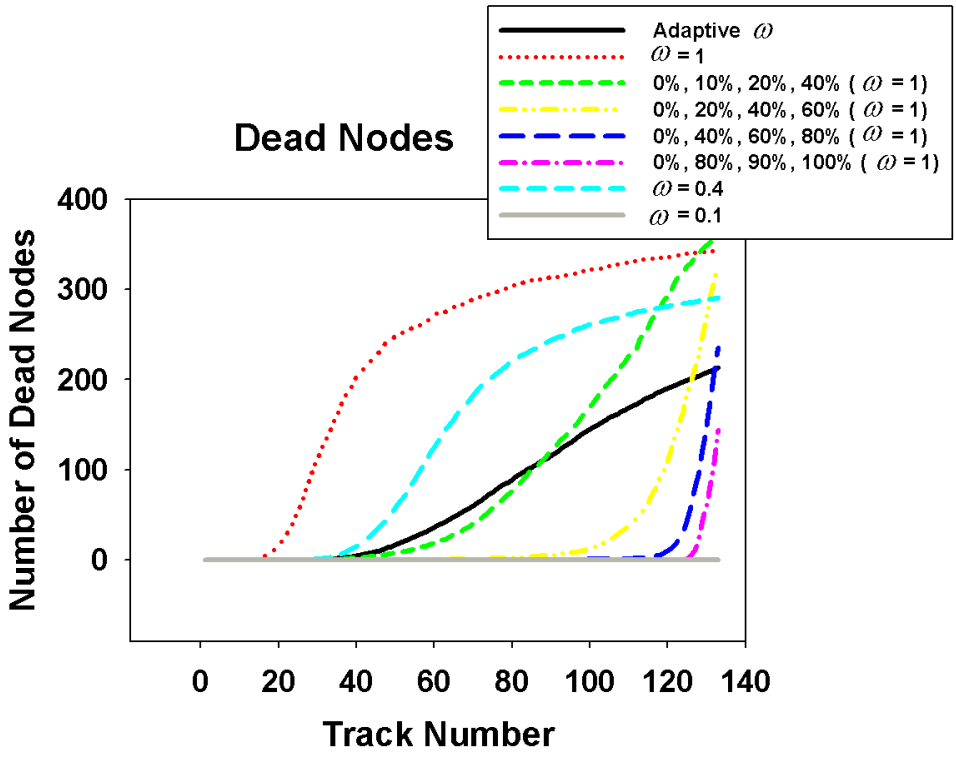


Figure 5.10. Number of Dead Nodes

CHAPTER 6

ADAPTIVE WAKEUP RANGE

We have seen before that the optimum value of the wakeup ranges is a function of the mobility model: the less predictable the track, the wider the wakeup range needs to be.

In this chapter, we consider an approach whereby the prior experience of the system concerning track characteristics can be used to adaptively choose the wakeup range. We present two approaches. The first approach is to learn the mean and standard deviation of the error between the predicted and estimated points of the track and to set R1 equal to the mean plus k times the standard deviation, where k is a control parameter. The second heuristic computes R1 based on the variation of the angle between consecutive sensing points.

6.1 Mean and Standard Deviation

As we mentioned before, this approach sets the first wakeup range R1 by learning the mean and standard deviation between the predicted and estimated points of the track as shown in the below equation:

$$R1 \quad = \quad Mean + k \times S.D \tag{6.1}$$

where k is a control parameter.

Figure 6.1 plots, however, a comparison between linear and perfect prediction under adaptive wakeup. In perfect prediction, the next location of such target is known in advance. While perfect prediction cannot be achieved in practice, it is used here as a baseline

against the linear prediction algorithm. The figure shows, as expected, the considerable impact of track uncertainty on the energy consumption associated with this algorithm. As the sampling interval increases, the number of times the target is located goes down. Everything else being equal, this would tend to reduce the energy consumption. However, with an increase in sampling interval comes a decrease in the chance of catching the target within the first wakeup radius; This tends to drive up the energy consumption by requiring a larger wakeup radius. The greater the track randomness, the shorter this sampling interval needs to be.

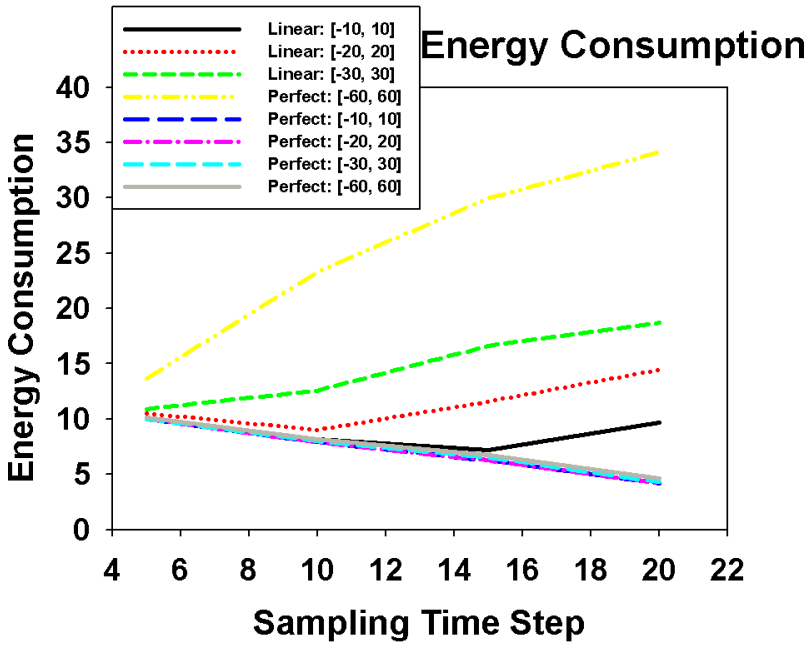


Figure 6.1. Energy Consumption in Adaptive Wakeup Algorithm

Figure 6.2 shows the probability of catching the target in R1 as function of sampling time step for different values of k. It is obvious why as we increase k, and therefore R1, the probability of catching tends to increase. Figure 6.3 shows the corresponding energy consumption for the same factors. Smaller factors result in an increase in miss probability, and hence require the search area to be expanded, and thereby expends more energy. An excessively large k, however, increases the energy consumption since this leads to a needlessly larger R1. The same behavior occurs under different mobility models as shown in Figure 6.4 and Figure 6.5.

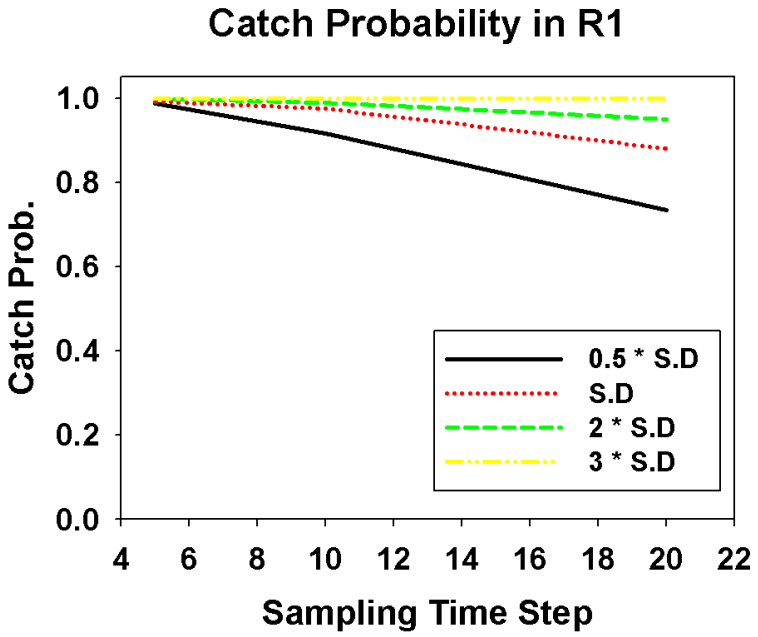


Figure 6.2. Different Factors of Standard Deviation with Mobility Mode [-60, 60]

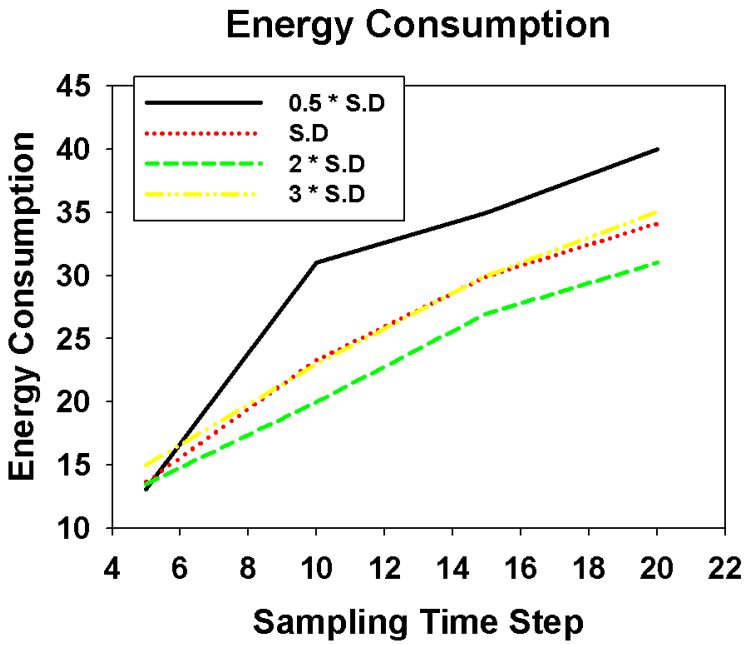


Figure 6.3. Energy Consumption for Different Factors of Standard Deviation with Mobility Mode [-60, 60]

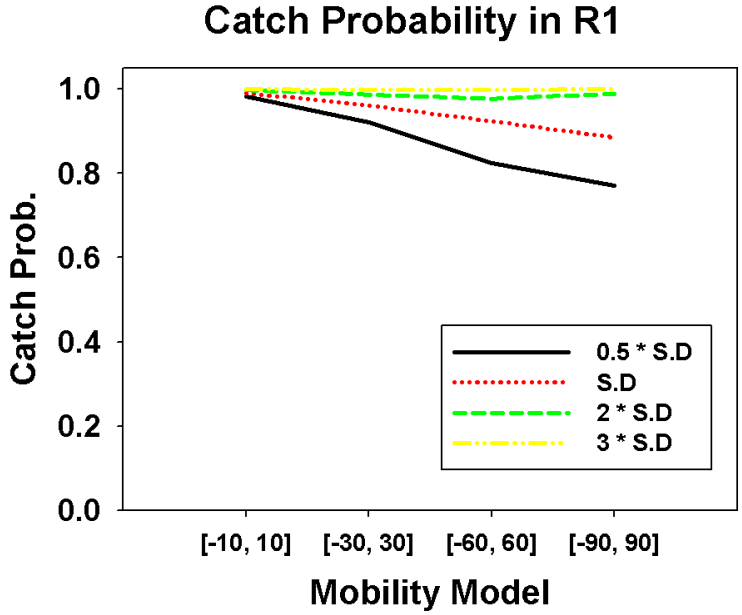


Figure 6.4. Different Factors of Standard Deviation

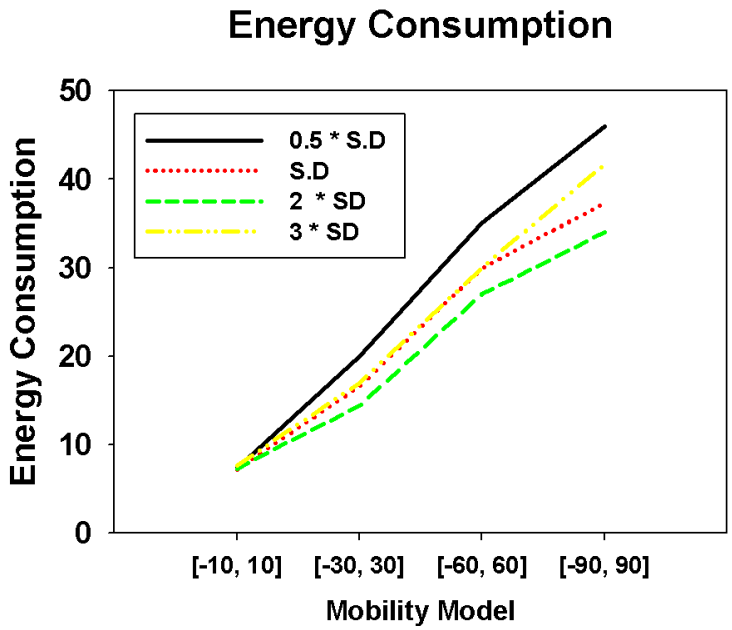


Figure 6.5. Energy Consumption for Different Factors of Standard Deviation

We study in Figure 6.6 and Figure 6.7 the impact of adaptive wakeup range with different mobility models on the system over a given mission lifetime. A similar explanation to that provided earlier explains this, and the related, results. If the sampling interval is very short we take an excessive number of samples and thereby waste energy. If the interval is not short enough, we increase the chance of missing in R1 and having to expand the search area thereby wasting energy. Clearly the more random the track the more often one needs to sample the target. Figure 6.8 shows the number of misses when using energy harvesting. Note that the shape of the curves is similar; The optimum sampling time has remained virtually unchanged. However, the absolute number of dead nodes is significantly lower.

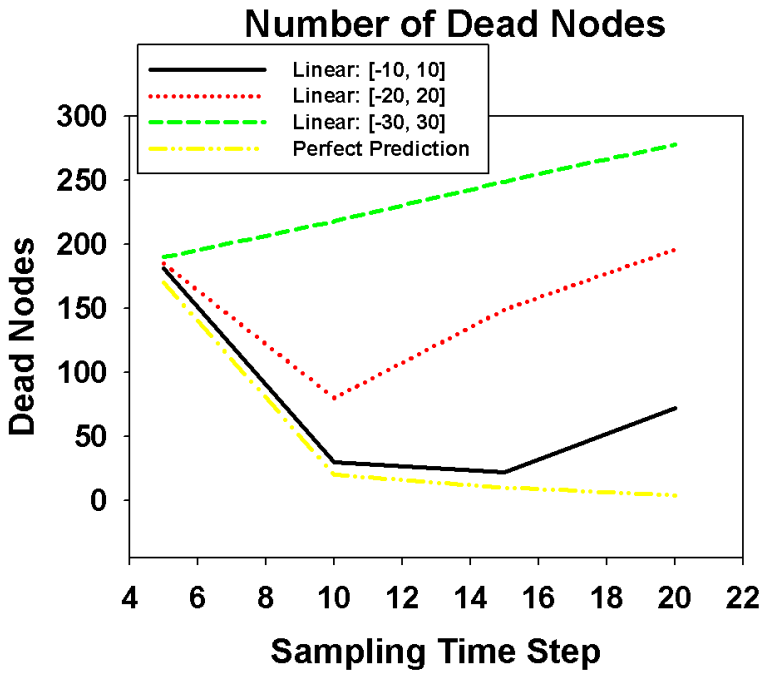


Figure 6.6. Number of Dead Nodes as a Function of Mobility Model

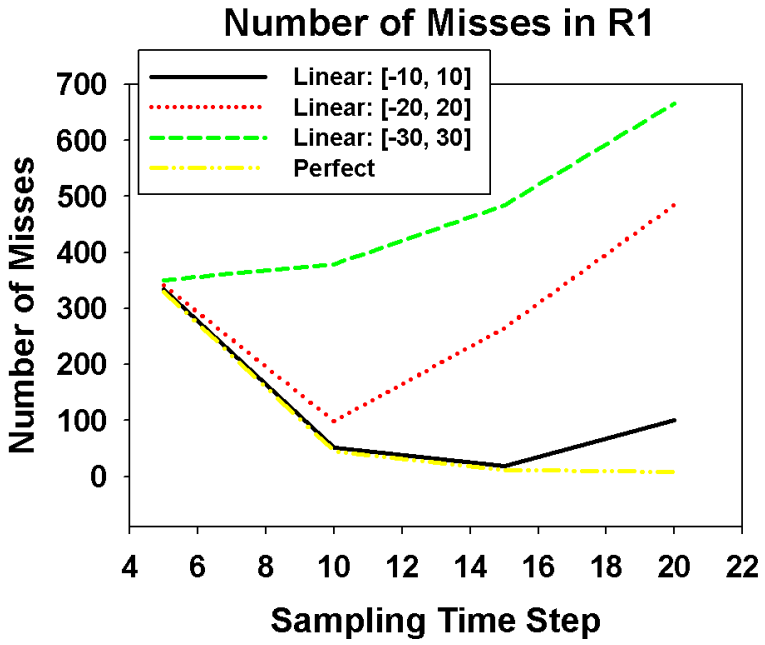


Figure 6.7. Number of Misses as Function of Mobility Model

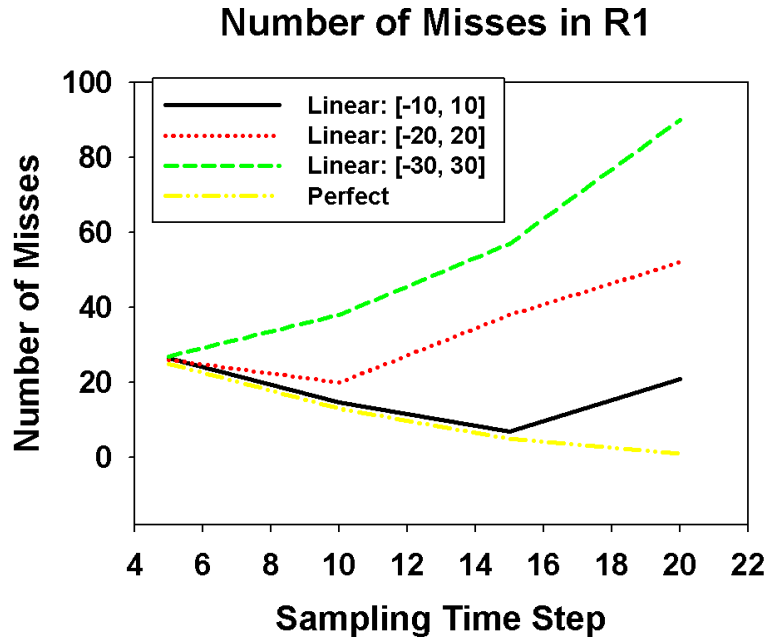


Figure 6.8. Number of Misses as a Function of Mobility Model with Energy Harvesting ($P_{cloudless}(t) = 7mW$)

6.2 Angle Variation

The second heuristic computes R1 based on the variation of the angle between consecutive sensing points. The basic idea behind this algorithm is shown in Figure 6.9. In this figure, we have the last two sensing points: s_1 and s_2 captured by the system, and from these points we are able to compute the predicted distance (d) of the next target position by means of the current distance computed between s_1 and s_2. Then R1 will be set equal to $d\tan(\theta)$, where $d = TimeStep \times TargetSpeed$. $TimeStep$ is an update time interval of the moving target position. θ is the angle between s_1 and s_2. There are two steps to calculate θ that are based on a heuristic learning method: 1) Find the standard deviation of the tangent of the angle between consecutive sensing points, and this standard deviation is updated as long as we have a new sensing points. The mean of the angle, however, is zero. 2) θ is calculated by taking the arc tangent of the newly updated standard deviation and multiplied by computed target speed and given time step to get the new updated R1 as shown in Algorithm 2. $SDA = stDev()$ is an online function that generates the standard deviation of any variable that keeps its value updated one at a time.

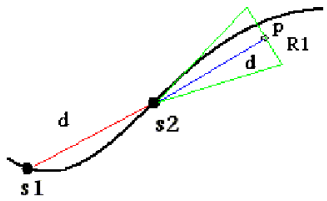


Figure 6.9. How to Compute R1 adaptively

Figure 6.10 shows also the probability of catching the target for different percentages of computed R1 size for the second adaptive algorithm. This figure shows as we reduce the percentage value, there is a corresponding reduction in catch probability in general.

Input: $x1, y1$: Coordinate of Previous Sensing Point
$x2, y2$: Coordinate of Current Sensing Point
s : Computed Target Speed
t : Time Step
Output: newly computed R_1
** Get the Newly Computed Standard Deviation * $SDA = stDev(\tan^{-1}(\frac{y2-y1}{x2-x1}))$
$R_1 = s \times t \times \tan(SDA)$

Algorithm 2: Adaptive R algorithm

Increasing the percentage beyond 100% will improve the catch probability for larger time steps, which is already high, and increase the energy consumption.

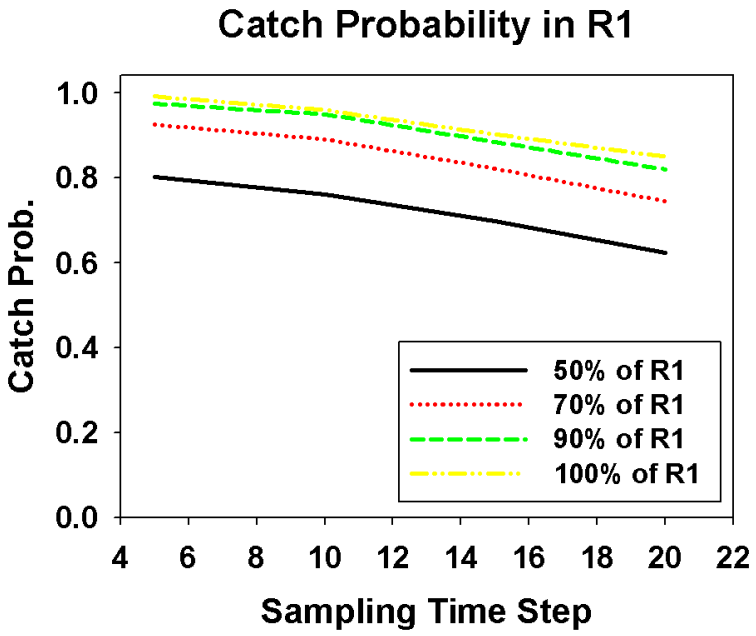


Figure 6.10. Selected Percentage of Computed R1 under [-60, 60] Mobility Model

Figure 6.11 compares the two adaptive algorithms and shows the impact of the mobility model parameters on the catch probability as a function of the sampling time step. This figure shows the effectiveness of the adaptive algorithms to estimate R1. The catch probability in R1 is very close to unity for both adaptive wakeup algorithms. Having the same catch probability results in the same energy consumption for both algorithms as shown in Figure 6.12.

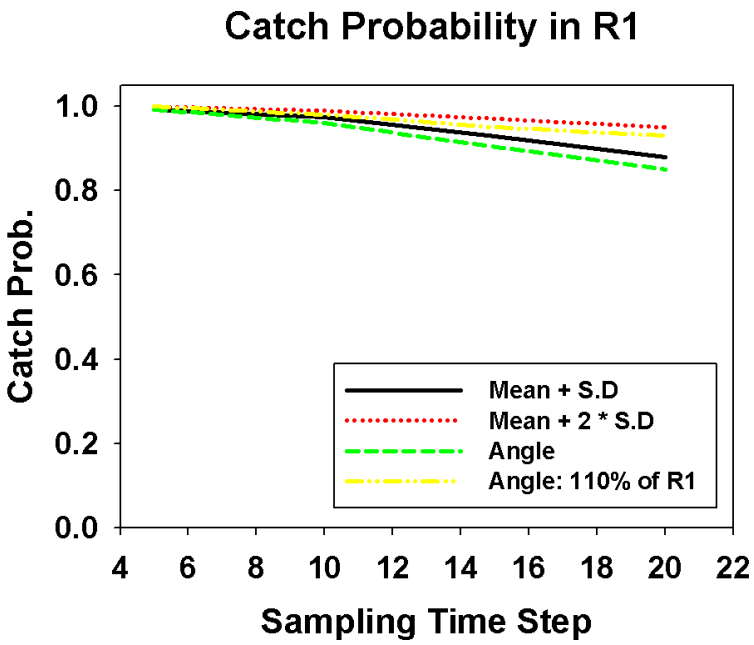


Figure 6.11. Catch Probability for both Adaptive Wakeup algorithms with Speed = 5 m/s under [-60, 60] Mobility Model

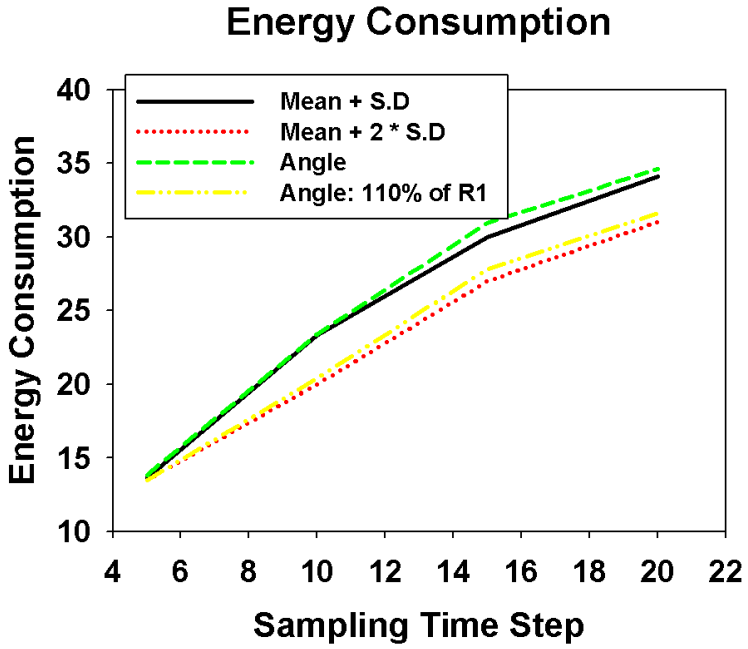


Figure 6.12. Energy Consumption for both Adaptive Wakeup algorithms under [-60, 60] Mobility Model

CHAPTER 7

MULTIPLE TARGETS

Previous works related to multiple targets [28] [5] assume that all sensor nodes are awake all the time and that there is a central processing area, where the disambiguation of targets is carried out. In our work, however, most sensor nodes are asleep and are only awakened as necessary. In this case, when targets interleave, this could lead to tracks being confused with one another. Here, we investigate how our tracking algorithm functions in the face of multiple targets and consider how to modify it to reduce ambiguation. In this case, sensor nodes cannot differentiate between individual targets or count how many targets there are within the range. Based on the sensor data received, we want to have position estimates close to the actual target positions. Additionally, maintaining an identity to which target belongs to which is important in the multiple target case. However, this confusion and mixing among targets can be reduced but cannot be avoided as explained before. The main factor associated with such loss of target identity is the mixing among targets that happens when targets become close or cross each other. We would like to study how our tracking algorithm handles multiple targets and how to achieve accuracy under different sets of system parameters.

An example of the mixing between two targets is depicted in Figure 7.1. The green circles are the wakeup circles to catch the target, and the brown and blue lines are the prediction tracks. The black lines are the actual tracks. It can be seen that increasing the wakeup range results in confusion. Figure 7.2, however, shows a case where there is no confusion among target tracks.

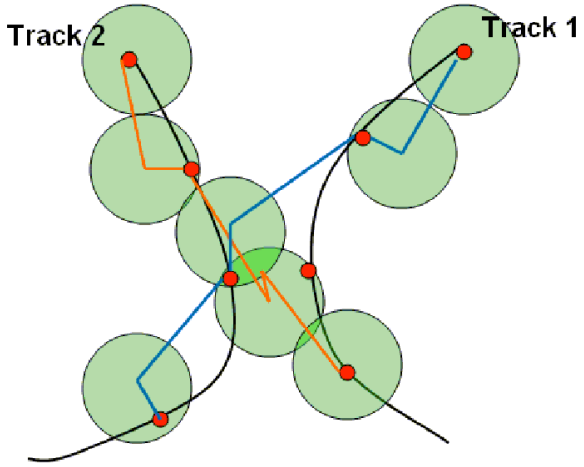


Figure 7.1. Two Targets: Confusion

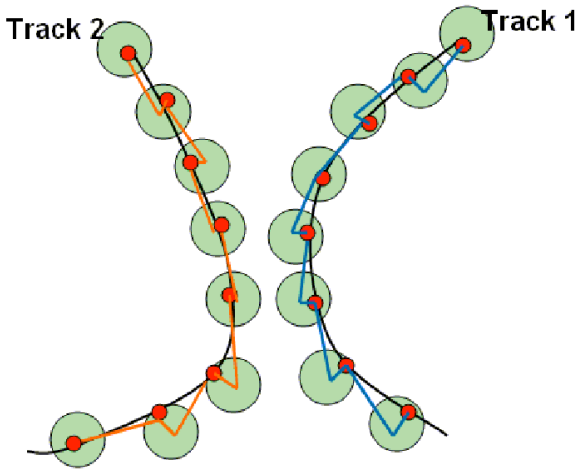


Figure 7.2. Two Targets: No Confusion

7.1 Impact of Sampling Time Step and Number of Targets on Track Coverage

In this experiment, we study the impact of the sampling time step and the number of targets on average track coverage. The track coverage is defined as the percentage of estimated sensing points that are closer to the actual target track than to other target tracks. As one would expect, the track coverage has the tendency to drop as the sampling time step and the number of targets increase: some numerical data are provided in Figure 7.3.

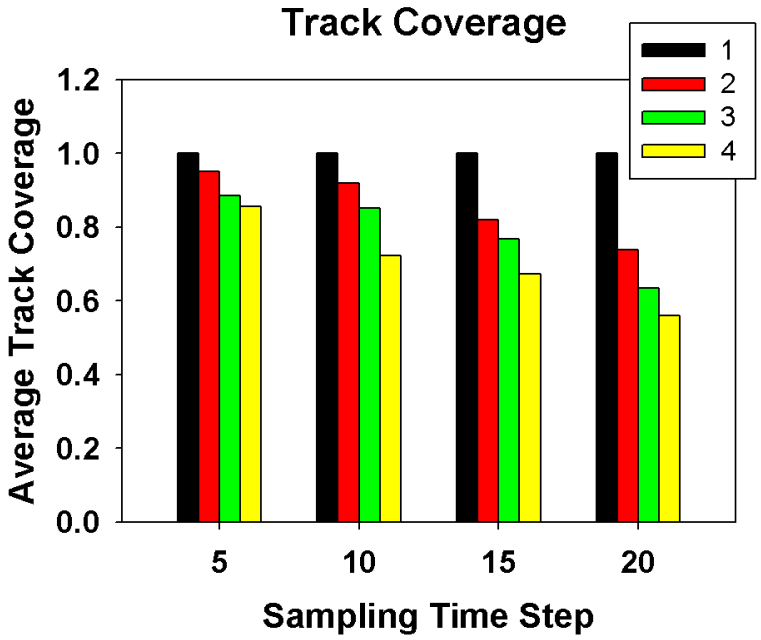


Figure 7.3. Impact of Time Step and Number of Tracks on Track Coverage [-30, 30] Mobility Model

7.2 Filtering Algorithm

A significant cause of error, when there are multiple targets, occurs when multiple targets are within the same waking circle. Our goal is to reduce this confusion by adopting a filtering mechanism that casts away reports from sensors that are far away from the projected target location.

The filter step is shown in Algorithm 3. We filter out all reports received at the leader node from nodes far away from the predicted point. The algorithm starts by binning the distances of reporting sensors from the current leader node based on the ratio between this distance and sensing radius. If the ratio, for example, is less than one, then the sensor is located within the sensing radius. Figure 7.4 shows an example of filtering algorithm and no filtering if two targets are captured in the wakeup circle of radius R1 (the big circle). T1 and T2 (blue stars) stand for the presence of two targets at this moment of time, and L1 represents the leader node (the green dot) supposed to track T1 while the rest of the nodes are denoted by the black dots. The small circles around T1 and T2 are the sensing circles while the other small circle centered at L1 denotes the range of acceptable sensor readings at L1. The red dots constitute the acceptable reporting sensors at L1. It is obvious that in this example that no filtering results in larger position error than filtering algorithm since in no filtering the L1 accepts all reports from sensor nodes.

Figure 7.5 shows that filtering is very successful in removing ambiguity when the sensing region is too large (either because R1 was too large to begin with or R1 was too small, which led to a larger R2 circle being awakened). The position error is rendered largely insensitive to the sensing radius: it is only affected by the sampling time step.

We consider the impact of the mobility model and the sampling time step as depicted on Figure 7.6. It is rather obvious that as the target's track becomes ever more random, our ability to predict the track goes down and the prediction error increases. And this increases the confusion among targets which is translated in greater error in position estimate. Similarly, the smaller the sampling time-step, the better our prediction accuracy. This is due

Input: All data collected at Leader Node at this moment of time
Output: closed data reading to leader node
for *All data collected at the Leader Node* **do**
  for *i = 0; i < List.size; i++* **do**
    distance = the distance between a given coordinate stored in each data packet and the Leader Node
    if $\frac{distance}{SensingRange} \leq (i + 1)$ **then**
      list[i].add(data); break;
    end
  end
end
for *i = 0; i < List.size; i++* **do**
  if *list[i] not empty* **then**
    return list[i];
  end
end

Algorithm 3: Filtering Algorithm

to the fact that as the time-step decreases, the target has less of an opportunity to drastically change its position from what is expected. A longer time-step, however, increases the prediction error and confusion among tracks since the target could closer to the other track than to its actual track, and hence averaging the detecting sensor positions leads into larger position error. We can also see how the filtering algorithm improves the position estimation accuracy.

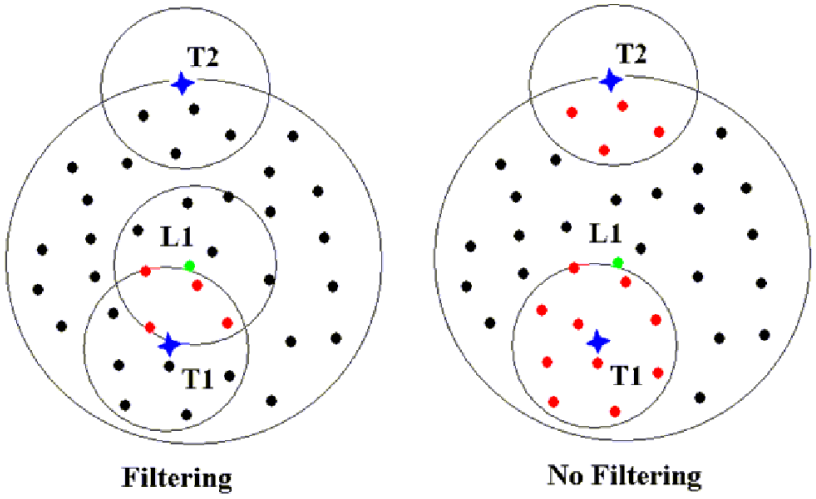


Figure 7.4. Filtering and No Filtering

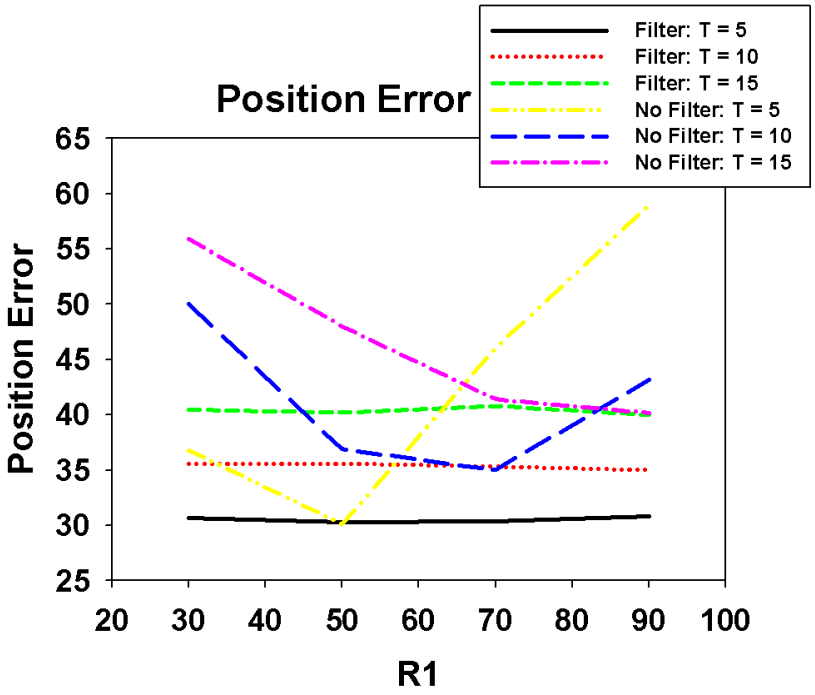


Figure 7.5. Impact of Time Step on Position Error as Function of R1

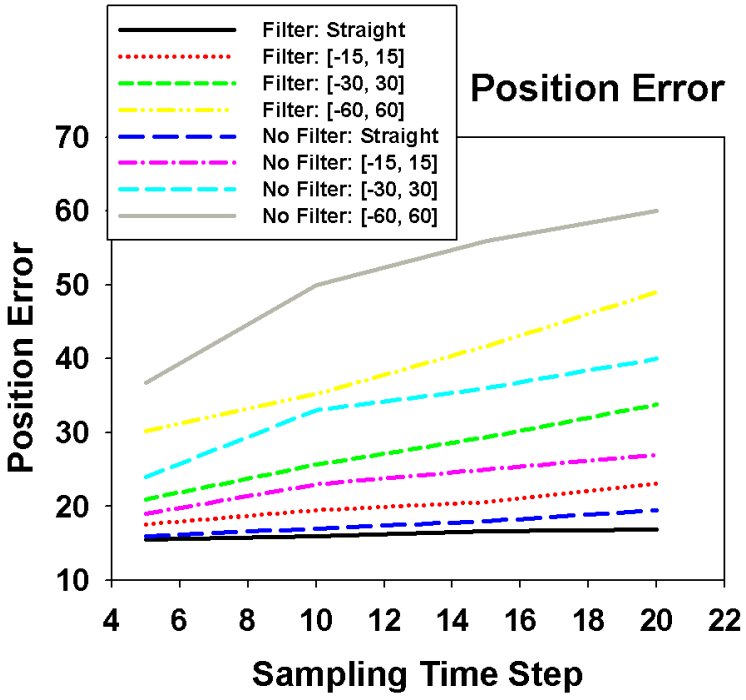


Figure 7.6. Impact of Mobility Model on Position Error

CHAPTER 8

USING SIGNAL STRENGTH

In our original implementation, the target position is estimated by simply averaging sensor locations of detecting nodes. This requires us only to determine which sensors have detected the target and not what the amplitude was at the sensor node. Such an approach has the virtue of simplicity; however, the question arises whether the use of signal strength information would allow for improved accuracy.

Generally, when all targets present in the sensor fields, they emit some types of signals (acoustic, vibration, light, etc.) [31]. These signals will be attenuated as moving away from target source with a decay intensity ζ. Equation 8.1 shows the signal strength model used for the distance sensing model [18]. Where s_i is the received signal strength at sensor i, a is the amplitude of signal strength, ζ is a coefficient that depends on the nature of the device and weather. The typical value of ζ is 2 [18], but it can range depending on the aforementioned factors. r_i is the Euclidean range distance from sensor i and current target position.

$$s_i = a r_i^{-\zeta} \tag{8.1}$$

8.1 Position Estimation

We use linearization and minimum mean square estimate to estimate the position of the target using the measured distance range described in Equation 8.1 [3]. Ideally, we would like the error to be 0.

$$f_i = r_i - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} = 0 \tag{8.2}$$

Rearranging the previous equation, we get:

$$(x_0^2 + y_0^2) + x_0(-2x_i) + y_0(-2y_i) - r_i^2 = -x_i^2 - y_i^2 \tag{8.3}$$

Subtract the last equation from the previous one to get rid of quadratic terms:

$$2x_0(x_k - x_i) + 2y_0(y_k - y_i) = r_i^2 - r_k^2 - x_i^2 - y_i^2 + x_k^2 + y_k^2 \tag{8.4}$$

Note that this is linear. In general, we have an over-constrained linear system

$$Ax = b \tag{8.5}$$

where

$$b = \begin{bmatrix} r_1^2 - r_k^2 - x_1^2 - y_1^2 + x_k^2 + y_k^2 \\ r_2^2 - r_k^2 - x_2^2 - y_2^2 + x_k^2 + y_k^2 \\ \vdots \\ r_{k-1}^2 - r_k^2 - x_{k-1}^2 - y_{k-1}^2 + x_k^2 + y_k^2 \end{bmatrix} \tag{8.6}$$

$$A = \begin{bmatrix} 2(x_k - x_1) & 2(y_k - y_1) \\ 2(x_k - x_2) & 2(y_k - y_2) \\ \vdots & \vdots \\ 2(x_k - x_{k-1}) & 2(y_k - y_{k-1}) \end{bmatrix} \tag{8.7}$$

$$x = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{8.8}$$

Now we can use the least squares equation to compute estimation.

$$x = (A^T A)^{-1} A^T b \tag{8.9}$$

8.2 Learning ζ and a

In our tracking algorithm, learning the amplitude of signal strength a under unknown target signature can be deduced by the received signal strength received at the leader node. The maximum received signal strength value will be considered as a. We learn the decay factor ζ by calculating the radius of the convex hull area of detecting nodes. Algorithm 4 shows the algorithm in detail. The first part of the algorithm finds the maximum received signal strength from detecting nodes. Then it finds the convex hull points, and uses these points to compute the radius of the convex hull, which approximately constitutes the radius of the sensing circle as depicted in Figure 8.1. Finally, using this radius and maximum signal strength, we are able to calculate the decay factor, ζ. Figure 8.2 provides some numerical results showing the rate of learning in term of position accuracy. We get the initial values of position accuracy using 0/1 model. As the target moves across the sensor field, the values of a and ζ are updated based on the signal received at the sensor nodes. We also consider the case where ζ is known; in such a case, there is greater position accuracy. The figure also shows that the corresponding errors for the 0/1 model are much higher than those obtained with the distance sensing model: we can see that signal strength sensing yields notably higher accuracy (in exchange for increased complexity).

Figure 8.3 shows the impact of the mobility model on position error while using distance sensing model. The target mobility parameters have an effect on the rate of learning of ζ and a, which eventually affect the position estimate. Initially, the first sample point uses the 0/1 model since there is no information about ζ , successive points apply the updated value of ζ for the position measurement. Increasing the randomness of the track tends to increase the prediction error, where the target becomes far away from the predicted point. This results in not having the sensing circle entirely within the waking circle. This affects the convex hull calculation to compute ζ and finding the maximum signal strength value to be used as the amplitude of the signal strength, a.

Input: All data collected at Leader Node at this moment of time
Output: Get computed decay factor ζ and amplitude of signal strength *a*
//For all data collected at a Leader Node
for *i = 0; i <data.size; i++* **do**
  if *data[i].signal > maxSignal* **then**
    maxSignal = data[i].signal;
  end
end
// Get convex hull points, convex hull algorithm is hidden here
points = **getCovexHullPoints**(data);
// Compute the central point of convex hull points, where points are sensor positions
for *i = 0; i < points.size; i++* **do**
  xc = xc + point[i].x; yc = yc + point[i].y; size = size + 1;
end
xc = xc / size; yx = yc / size;
// Compute the radius of convex hull
for *i = 0; i < points.size; i++* **do**
  d = d + $\sqrt{(point[i].x - xc)^2 + (point[i].y - yc)^2}$;
end
d = d / size;
$\zeta = \frac{\log(maxSignal)}{\log(d)}$;

Algorithm 4: Computing Decay Factor and Signal Strength Amplitude

Figure 8.4 compares the signal strength learning model with 0/1 model. it shows that 0/1 model with high value of ω is better at the beginning of sampling time numbers, but the learning with small value of ω outperforms the 0/1 model in later sampling point numbers. Figure 8.5 shows that the energy consumption of both the learning method and 0/1 model, where it is recommended to use smaller value of ω to save energy and retain high accuracy at the same time for later sampling point numbers.

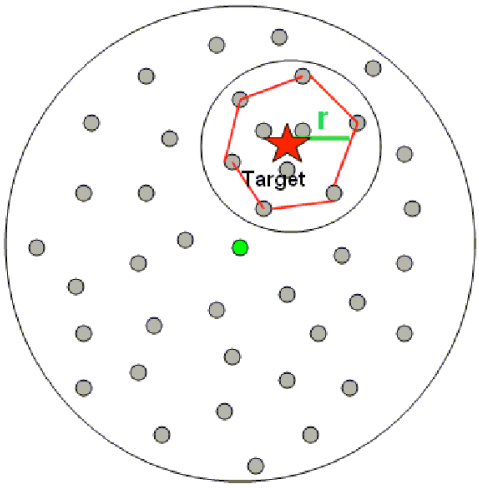


Figure 8.1. The Convex Hull Points Approximate the Sensing Circle

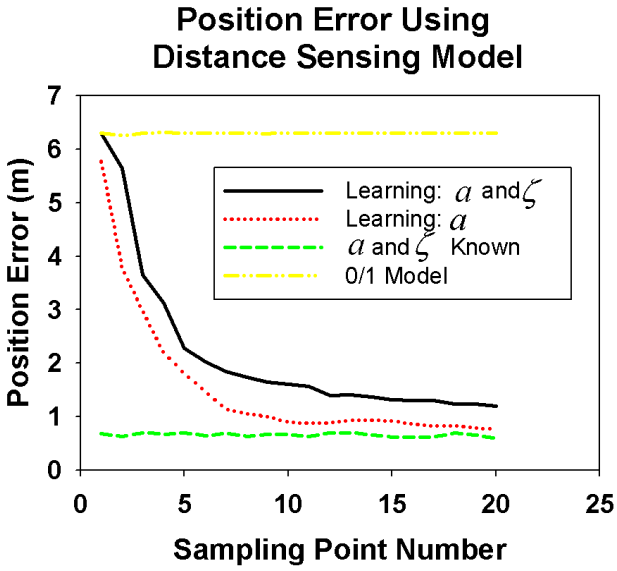


Figure 8.2. The Impact of Learning ζ and a on Position Error as a Function of Sampling Point Number (set a =10, and ζ = 2: known by the simulation)

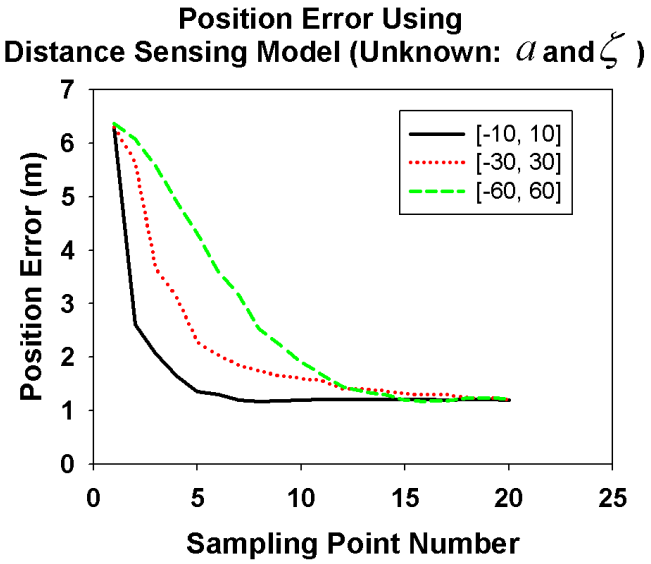


Figure 8.3. The Impact of Mobility Model on Position Error While Learning ζ and *a* a Function of Sampling Point Number

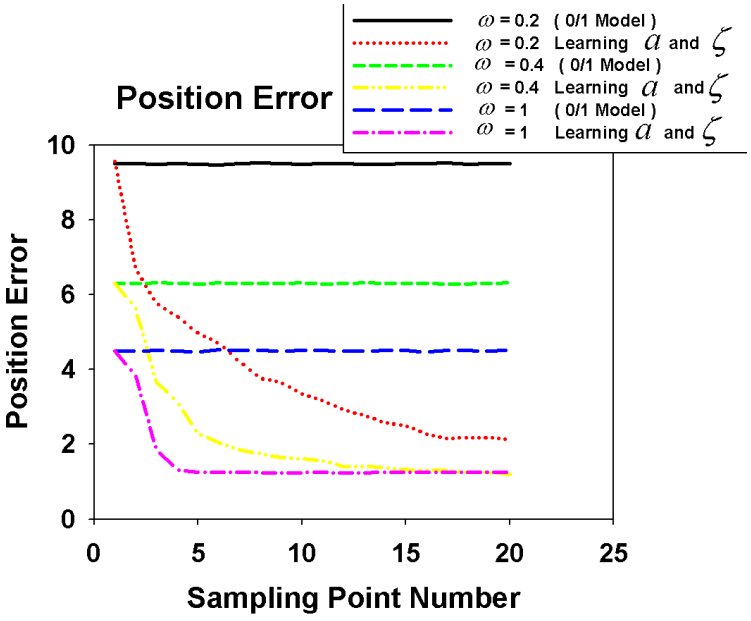


Figure 8.4. Learning vs. 0/1 Model

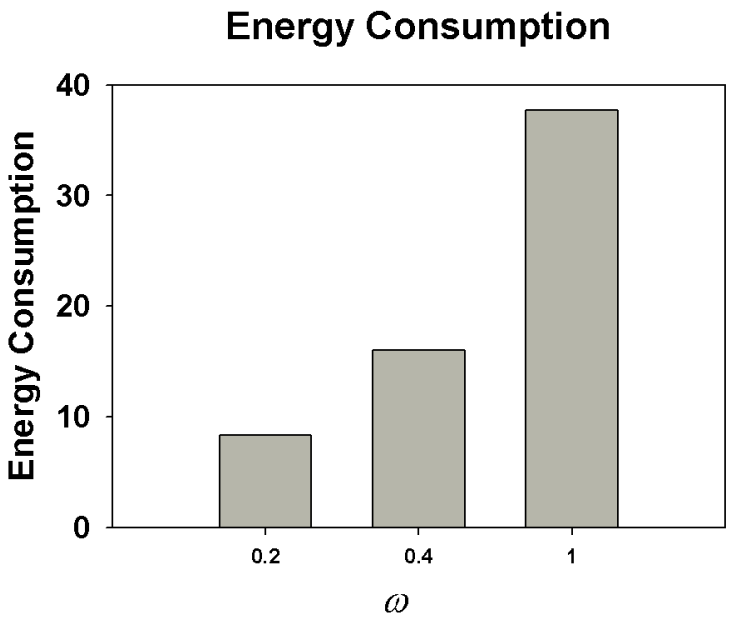


Figure 8.5. Energy Consumption for both Learning and 0/1 Model

8.3 Recalibration of Sensor Drift

Sensor miscalibration leads to an increased error in the target position estimate. This miscalibration may occur both due to an initial miscalibration at manufacture and to sensors drifting over time. In this section, we present simple heuristics to mutually recalibrate the sensors by using correlated information from neighboring sensors.

Generally, the process of device calibration involves forcing a device to match a specified input/output mapping. This is often made by adjusting the device externally by passing the device's output through a calibration function that maps the actual device response to the corrected response [34].

In this section, we study the case where the sensor drifs from its original value over a period of operation. We create a drift model where the calibration factor, k varies with time, t. k is defined as the ratio between measured and computed signal strengths. We assume that for each sensor, the calibration factor is generated according to the following:

$$k_i(t) = a_i + b_i \times t \tag{8.10}$$

where a_i and b_i are constants, which are randomly picked for each sensor i. (That is, each sensor will have its own a_i and b_i values; these values will stay fixed for the lifetime of that sensor). This is one of the simplest drift models one can think of.

When a target is detected by multiple sensors, their signal strength measurements can be collected and used for sensor recalibration. Each detecting node then calculates the drift factor k_i according to its signal strength reading and the computed signal strength value based on the distance of this node from the estimated target position. Here, we study different target position estimation techniques: 1)using the 0/1 model, 2)weighted averaging, where we give more weight to the sensor that has higher signal strength value as shown in Equations 8.11 and 8.12, and 3) least square estimate described in the previous section.

$$xw_i = x_i \times \frac{\text{Signal Strength at Sensor i}}{\text{Sum of Signal Strengths for all Detecting Nodes}} \tag{8.11}$$

$$yw_i = y_i \times \frac{\text{Signal Strength at Sensor i}}{\text{Sum of Signal Strengths for all Detecting Nodes}} \tag{8.12}$$

We then use k_i as a correction factor. This calibration process is shown in Algorithm 5. Also, we maintain a list of the last window w of values of k_i's for each sensor. We use the average value of these as the estimate for the deviation factor, k_i.

Input: X, Y : Coordinate Vectors for Detecting Nodes
x, y : Estimated Target Position before Calibration Using either:
1. Average Position Estimate
2. Weighted Average Estimate
3. Least Square Estimate
Output: Recalibrate sensor
for $i = 0$ *to Detecting Node Size* **do**
    /** Get the distance from estimated average position and each node */
    $d_i = \sqrt{(X_i - x)^2 + (Y_i - y)^2}$;
    /** Get the calibration factor for node i, k_i */
    $k_i = \frac{S_i}{ad_i^{-\zeta}}$;
end

Algorithm 5: Recalibration Algorithm

In the following experiments, we quantify how the sensor drift affects position accuracy, and how our recalibration algorithm improves this accuracy. Throughout, we assume that the sensors are initially well-calibrated (i.e. $a_i = 1$) and it is the drift over time that needs to be corrected. The value of b_i in Equation 8.10 is picked at random for each sensor from a given interval. For obvious reasons, the calibration factor has upper and lower bounds. The value of $k_i(t)$ is then given by

$$k_i(t) = \begin{cases} k_{min} & \text{if } a_i + b_i t < k_{min} \quad (8.13a) \\ k_{max} & \text{if } a_i + b_i t > k_{max} \quad (8.13b) \\ a_i + b_i t & otherwise \quad (8.13c) \end{cases}$$

Figure 8.6 shows the impact of the drift rate range on the position estimate. We use these different approaches to making the initial position estimate that drives the estimate of $k_i(t)$: simple averaging of the position of the detecting nodes, weighted averaging based on signal strengths, and the method of least squares. Since the last two of these depend on signal strength values, we use the latest available calibration factors to correct them before use.

Once $k_i(t)$ has been estimated, least squares is used to obtain the corrected position estimate.

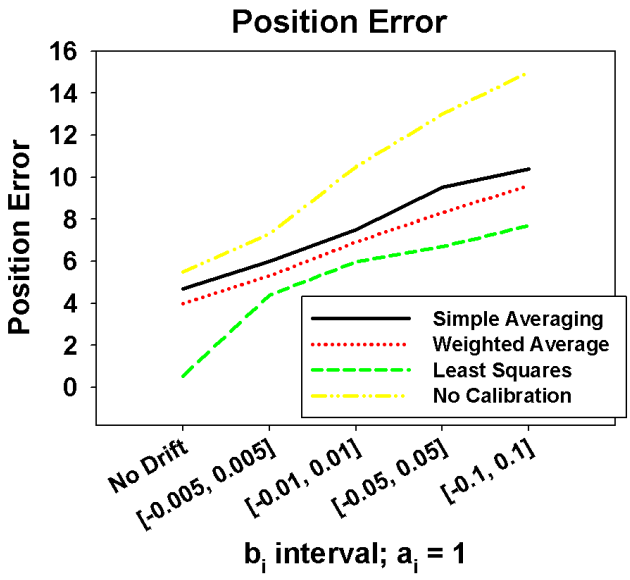


Figure 8.6. Position Estimate with Node Density, $\Delta = 0.0044$, $w = 4$, $[k_{min}, k_{max}] = [0.1, 2]$, and $t = 40$

Figure 8.7 shows the impact of the t value on position estimate. Obviously, as t increases, the position estimate degrades due to the effect of drift.

Figure 8.8 shows the impact of the window size on the accuracy. As the window size increases, we average over a larger number of readings, thereby potentially increasing accuracy by reducing the impact of random effects. However, as window size is increased, we also use older data as part of the computation, which potentially reduces accuracy.

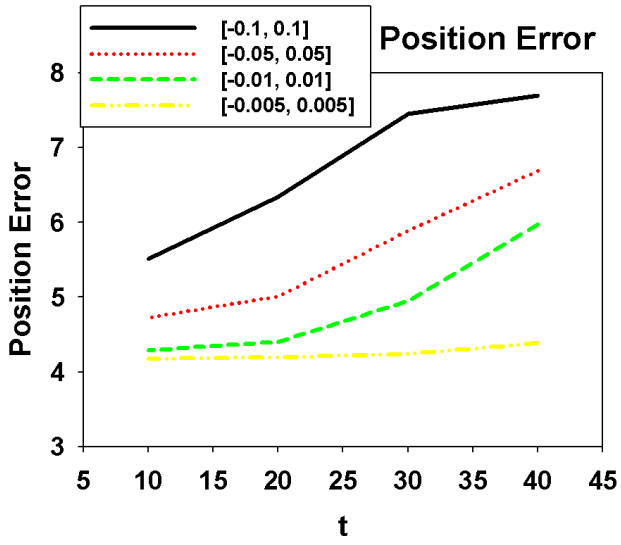


Figure 8.7. Position Estimate with Node Density, $\Delta = 0.0044$, $w = 4$

Figure 8.9 shows the impact of node density Δ on position accuracy for a given drift function. We can see that the position estimate with high density is improved compared with lower node density case due to the increased accuracy of estimating the k_i value due to a potentially larger number of nodes available to detect the target.

The impact of the mobility model on error in position estimate is quantified in Figure 8.10. It is rather obvious that as the target becomes more random, the number of detecting nodes is reduced due to increase in prediction error, which increases the error of the k_i calculation, which eventually increases the position error.

Figure 8.11 shows in detail how the drift affects position estimate over a given mission lifetime. At the beginning of the mission, the position estimate starts the same for all given drift ranges, the estimates diverge from each other as time passes by, and then level off as the k_{min} and k_{max} bounds are reached. Increasing node density improves in general the estimated value of k_i, and thus the position estimate.

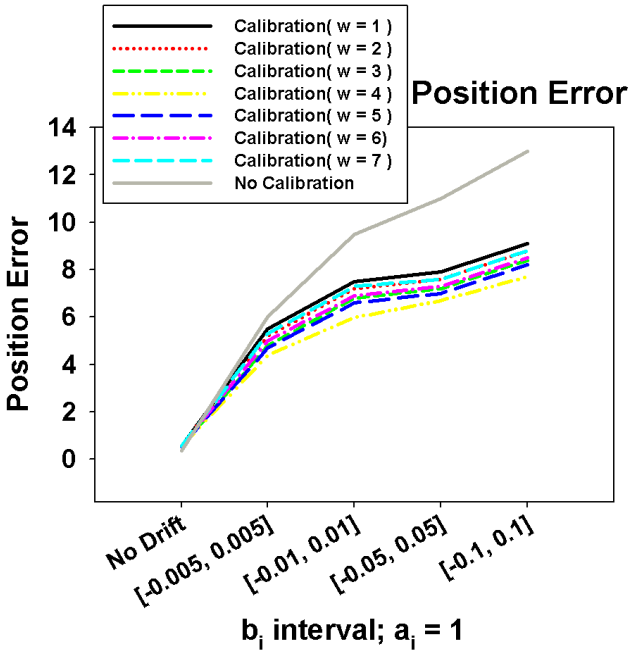


Figure 8.8. Position Estimate Using No Calibration and Calibration with Node Density, $\Delta = 0.0044$

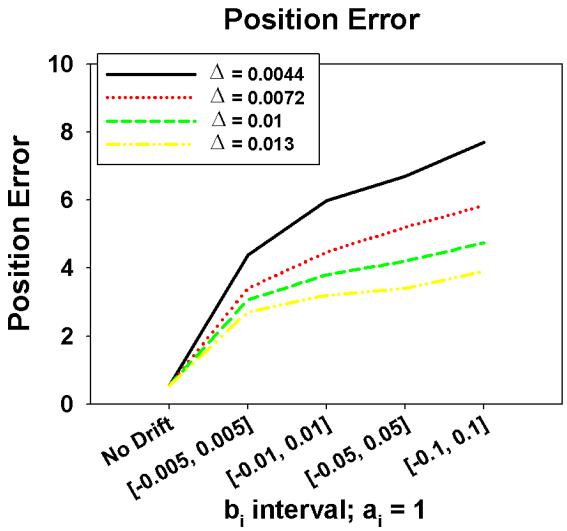


Figure 8.9. The Impact of Node Density on Position Estimate Using Calibration with $w = 4$

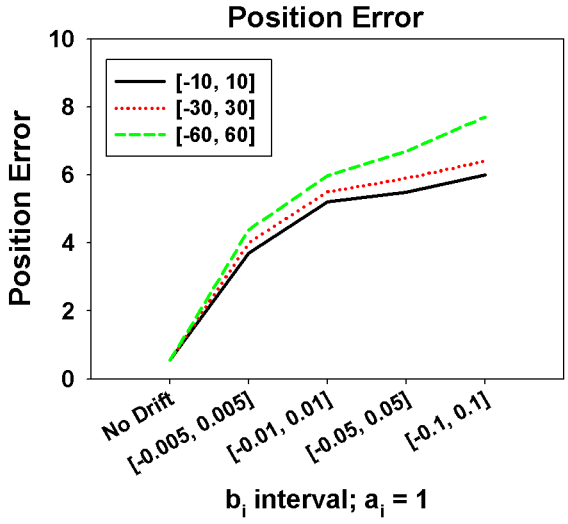


Figure 8.10. Position Estimate Using Calibration given a mobility model with $w = 4$ and $\Delta = 0.0044$

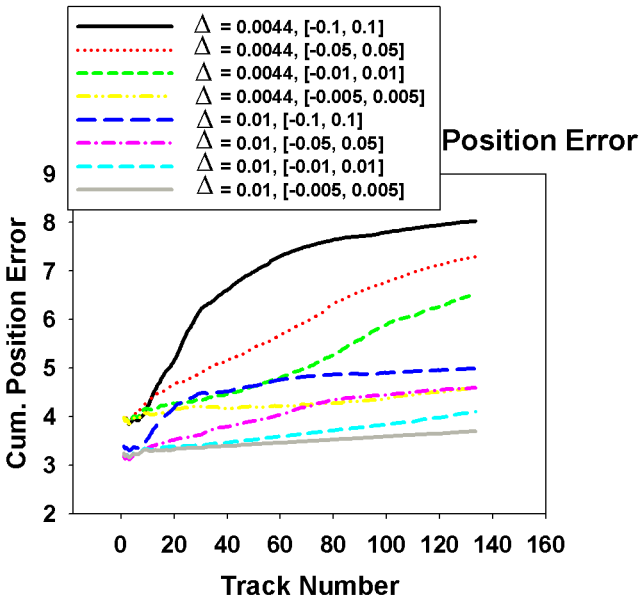


Figure 8.11. Position Estimate with Calibration over a Mission Lifetime with $w = 4$

CHAPTER 9

DISCUSSION

In this work, we have studied the problem of target tracking by means of a sensor network. Our basic approach is that of the selective awakening of nodes, based on a prediction of the target track. Our baseline algorithm awakens nodes within a given radius of the predicted target position and estimates that position by simply averaging the position of the nodes which detect the target. If the target cannot be detected within a given wakeup circle, the wakeup circle is successively increased until either the target has been detected or the search is abandoned.

We present a simple analytical model to evaluate the impact of the number of detecting sensor nodes on the accuracy of the position estimate. We show that relatively few nodes are sufficient to allow an accurate estimate; indeed, beyond three detecting nodes, the increased accuracy per every additional awakened node drops rather dramatically.

We consider several extensions of this basic algorithm. We show that there are conditions under which a non-uniform waking of nodes provides superior target location. In this approach, nodes close to the predicted target position are awakened with a greater probability; if -as is generally the case - the prediction is reasonably accurate, this allows for a greater number of nodes to detect the target. If the prediction is not correct, a sparser node field is still available to carry out target detection farther away from the predicted target position.

We present an energy-adaptive node awakening scheme, which seeks to balance the energy level at the various nodes. In this approach, nodes with lower energy levels are less

likely to be awakened. We show that this approach is the best in term of track coverage and energy savings compared with non-adaptive scheme.

We introduce a filtering algorithm for two purposes: to remove the effects of false reporting and to reduce the problem of confounding multiple targets. The fundamental idea here is to ignore reports from sensor nodes that seem to be far away from the target position. We show that the filtering algorithm always improves target estimate significantly compared with no filtering case. Comparative little has been published on multiple target detection, and results show that our filtering approach works well in improving tracking accuracy and reducing the probability of confounding targets.

We consider energy harvesting and its impact on the performance of target tracking. In energy harvesting, the network can obtain some, or most, of its operating energy from the environment (e.g., from the sun or from vibration). Energy harvesting allows us to be more aggressive in our expenditure of energy, since the system can rely on replenishing its energy resources. We introduce and use a simple energy harvesting model in the context of target tracking.

Most of our work involves sensors using a 0/1 detection model. In other words, a sensor reports that it has detected the target; it does not report the signal strength. Obviously, such information has the potential for improving our target position estimates. We provide results in this work to quantify the extent of such improvement.

Adaptive approaches, which allow the system to react to perceived parameter values in the operating environment, have obvious potential for improving system performance. We study the advantages of learning the intruders' mobility model parameters and then adapting the tracking parameters appropriately. We are considering problems associated with sensor miscalibration or drift when signal strength information is used for target estimation. To counter the inaccuracies that then result, we are developing approaches which seek to correlate inputs from neighboring sensors and to carry out mutual recalibration.

Finally, we are working on extending our analytical model to handle non-uniform sensor distributions.

BIBLIOGRAPHY

[1] Us naval observatory (usno) gps operations. In *http://tycho.usno.navy.mil/gps.html* (April 2001).

[2] Concord consortium projects. In *http://source.concord.org/softwaredocs/swing/index.html* (2005).

[3] Bachrach, Jonathan, and Taylor, Christopher. Localization in sensor networks: Handbook of sensor networks. In *Wiley Series on Parallel and Distributed Computing* (2005).

[4] Bai, Fan, and Helmy, Ahmed. A survey of mobility models in wireless adhoc networks. 129.

[5] Bar-Shalom, Y., and Fortmann, T.E. Tracking and data association. In *Academic Press* (1988).

[6] Chen, Wei-Peng, Hou, Jennifer C., and Sha, Lui. Dynamic clustering for acoustic target tracking in wireless sensor networks. *IEEE Transactions on Mobile Computing 3*, 3 (2004), 258–271.

[7] Christopher M. Vigorito, Deepak Ganesan, and Barto, Andrew G. Adaptive control of duty cycling in energy-harvesting wireless sensor networks. In *4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks* (June 2007), pp. 21–30.

[8] Eloy, JC. Status of the mems industry. In *Yole Development* (May 2007), p. 306.

[9] F. Ye, G. Zhong, J. Cheng S. Lu, and Zhang, L. Peas: A robust energy conserving protocol for long-lived sensor networks. In *International Conference on Distributed Computing Systems* (2003), vol. 23, IEEE Computer Society, pp. 28–37.

[10] Gui, Chao, and Mohapatra, Prasant. Power conservation and quality of surveillance in target tracking sensor networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking (MOBICOM)* (2004), pp. 129–143.

[11] Guiasu, Silviu. Information theory with applications. In *McGraw-Hill* (1977), New York.

[12] Guo, C., Zhong, L.C., and Rabaey, J.M. Low power distributed mac for ad hoc sensor radio networks. In *Global Telecommunications Conference* (2001), pp. 2944 – 2948 vol.5.

[13] He, Tian, Vicaire, Pascal A., Yan, Ting, Luo, Liqian, Gu, Lin, Zhou, Gang, Stoleru, Radu, Cao, Qing, Stankovic, John A., and Abdelzaher, Tarek. Achieving real-time target tracking using wireless sensor networks.

[14] Heinzelman, Wendi Rabiner, Chandrakasan, Anantha, and Balakrishnan, Hari. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS)* (2000).

[15] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci. Wireless sensor networks: A survey. In *Computer Networks* (March 2002), vol. 38.

[16] Javed Aslam, Zack Butler, Florin Constantin Valentino Crespi George Cybenko, and Rus, Daniela. Tracking a moving object with a binary sensor network. In *Proceedings of the First International Conference on Embedded Networked Sensor Systems* (November 2003), ACM Press, pp. 150–161.

[17] Juan Liu, Feng Zhao, and Petrovic, Dragan. Information-directed routing in ad hoc sensor networks. In *the 2nd ACM international conference on Wireless sensor networks and applications* (2003), pp. 88 – 97.

[18] Juan Liu, James Reich, and Zhao, Feng. Collaborative in-network processing for target tracking. In *EURASIP JASP, , Journal on Applied Signal Processing (JASP)* (2003), pp. 378–391.

[19] Kansal, Aman, Hsu, Jason, Zahedi, Sadaf, and Srivastava, Mani B. Power management in energy harvesting sensor networks. *ACM Trans. Embedded Comput. Syst. 6*, 4 (2007).

[20] Karp, B., and Kung, H. Greedy perimeter stateless routing. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000)* (2000).

[21] Klein, L. Sensor and data fusion concepts and applications. In *In Society of Photo-Optical Instrumentation Engineers (SPIE) Optical Engr Press* (1993).

[22] M. de Berg, M. van Kreveld, Mark Overmars O. Schwarzkopf. Computational geometry, algorithms and applications. In *Springer* (2000).

[23] Mechitov, Kirill, Sundresh, Sameer, Kwon, YoungMin, and Agha, Gul. Cooperative tracking with binary-detection sensor networks. In *SenSys* (2003), pp. 332–333.

[24] Meyn, S.P., and Tweedie, R.L. Markov chains and stochastic stability. In *Springer-Verlag, London. Available at probability.ca/MT* (1993).

[25] N. Bulusu, J. Heidemann, and Estrin, D. Gps-less low cost outdoor location for very small devices. In *IEEE Personal Communication, Special Issue on "Smart Space and Environments"* (October 2000).

[26] Pottie, G. J. Wireless sensor networks. In *In IEEE Information. Theory Workshop Proceedings (ITW)* (June 1998).

[27] Preparata, Franco P., and Hong, S.J. Convex hulls of finite sets of points in two and three dimensions. In *Communication ACM* (1977), vol. 20, ACM, p. 87–93.

[28] Reid, D.B. An algorithm for tracking multiple targets. In *IEEE Transaction on Automatic Control* (December 1979), pp. 843–854.

[29] S. Tilak, N. Abu-Ghazaleh, and Heinzelman, W. A taxonomy ofwirelessmicrosensor network models. In *ACM SIGMobile Computing and Communications Review (MC2R)* (April 2002), vol. 6.

[30] Santos, R. A., Edwards, A., Alvarez, O., Gonzalez, A, and Verduzco, A. A geographic routing algorithm for wireless sensor networks. In *Electronics, Robotics and Automotive Mechanics Conference (CERMA'06)* (2006), pp. 64–69.

[31] Seapahn Meguerdichian, Farinaz Koushanfar, Gang Qu, and Potkonjak, Miodrag. Exposure in wireless ad-hoc sensor networks. In *International Conference on Mobile Computing and Networking* (2001), pp. 139 – 150.

[32] Song, L. Hatzinakos, D. A cross-layer architecture of wireless sensor networks for target tracking. In *IEEE ACM TRANSACTIONS ON NETWORKING* (2007), vol. 15, THE IEEE COMMUNICATIONS SOCIETY, pp. 145–158.

[33] Stankovic, John A., Lu, Chenyang, Sha, Lui, Abdelzaher, Tarek, and Hou, Jennifer. Real-time communication and coordination in embedded sensor networks. In *Proceedings of the IEEE* (2003), pp. 1002–1022.

[34] Stum, Karl. Sensor accuracy and calibration theory and practical application. In *National Conference on Building Commissioning* (2006).

[35] Subramanian, S. Shakkottai, S. Gupta P. On optimal geographic routing in wireless networks with holes and non-uniform traffic. In *26th IEEE International Conference on Computer Communications. IEEE (INFOCOM)* (May 2007), pp. 1019–1027.

[36] Sundeep Pattem, Sameera Poduri, and Krishnamachari, Bhaskar. Energy-quality tradeoffs for target tracking in wireless sensor networks. In *Information Processing in Sensor Networks (ISPN)* (2003), p. 553.

[37] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Stein, Clifford. In *Introduction to Algorithms* (2001), vol. Second Edition, MIT Press and McGraw-Hill, p. 947–957.

[38] V.S. Tseng, K.W. Lin, and Hsieh, Ming-Hua. Energy efficient object tracking in sensor networks by mining temporal moving patterns. In *Ubiquitous and Trustworthy Computing* (June 2008), IEEE International Conference on Sensor Networks, pp. 170–176.

[39] Wagner, Jean-Paul, and Cristescu, Razvan. Power control for target tracking in sensor networks. In *Conference on Information Sciences ans Systems* (2005).

[40] Xu, Ya, Heidemann, John S., and Estrin, Deborah. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)* (2001), pp. 70–84.

[41] Xu, Yingqi, Winter, Julian, and Lee, Wang-Chien. Prediction-based strategies for energy saving in object tracking sensor networks. In *Mobile Data Management* (2004), pp. 346–357.

[42] Yu, Y., Govindan, R., and Estrin, D. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks, 2001.

[43] Zhang, W., and Cao, G. Dctc: Dynamic convoy tree-based collaboration for target tracking in sensor networks, 2004.

[44] Zhao, F. Liu, J. Liu J. Guibas L. Reich J. Collaborative signal and information processing: An information-directed approach. In *IEEE INSTITUTE OF ELECTRICAL AND ELECTRONICS* (2003), vol. 91, pp. 1199–1209.

[45] Zou, Yi, and Chakrabarty, Krishnendu. Energy-aware target localization in wireless sensor networks. In *PerCom* (2003), pp. 60–.