ROBUST LOCALIZATION FRAMEWORK FOR WIRELESS SENSOR NETWORKS

ÖMER KARATAŞ

IŞIK UNIVERSITY

2009

ROBUST LOCALIZATION FRAMEWORK FOR WIRELESS SENSOR NETWORKS

ÖMER KARATAŞ

BS, Computer Engineering, 2006

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Engineering

IŞIK UNIVERSITY

2009

IŞIK UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

ROBUST LOCALIZATION FRAMEWORK FOR WIRELESS SENSOR NETWORKS

ÖMER KARATAŞ

Assist. Prof. Cesim ERTEN (Thesis Supervisor)	Kadir Has University	_____
Assoc. Prof. Ercan SOLAK	Işık University	_____
Assist. Prof. Hasan Fehmi ATEŞ	Işık University	_____

APPROVAL DATE 07/01/2009

ROBUST LOCALIZATION FRAMEWORK FOR WIRELESS SENSOR NETWORKS

Abstract

In this thesis we try to construct a robust localization framework for wireless sensor networks and sample algorithms which uses our robust localization framework. In order to handle noise in distance measurements, our framework tries to utilize convex constraints and confidence intervals of a random variable. At the end of the localization process nodes will be assigned a set of *feasible regions*, with corresponding probabilities.

KABLOSUZ AĞLAR İÇİN HATAYA DAYANIKLI YERELLEŞTİRME İSKELETİ

Özet

Bu tezde biz kablosuz algılayıcı ağlar için ölçüm hatalarına dayanıklı bir yerelleştirme iskeleti ve bu iskeleti kullanan iki algoritmayı örnek olarak sunuyoruz. Önerdiğimiz iskelet, algılayıcılar arasındaki mesafe ölçümlerinde ortaya çıkan hataları tolere edebilmek amacıyla dışbükey koşullar ve rassal değişkenlerin güvenilirlik aralıklarını kullanmaya çalışmaktadır. Yerelleştirme işleminin sonucunda ulaşılmak istenen, her düğüm için olası alanların ve bu alanlarla eşleşen olasılıkların bulunmuş olmasıdır.

Acknowledgements

I would like to thank to my supervisor Asst. Prof. Cesim Erten for guiding me throughout the thesis. We have been working with him for almost two years, and throughout this period he was always supportive. It was an important experience to work with him.

I would like to thank to my family for just being there, especially Tuğba for her sense of humour, that makes me laugh under any circumstances.

This would have been incomplete without Olca Arda Çakıroğlu, I have been working with him for more than two years. Throughout this time, being able to hear his opinions is an important asset to me. I am glad to work with him. All my close friends including Arda , had something in me that helps me to be here, writing these thesis.

All the instructors in Işık University, whom I taken courses, had something in this thesis. I would like to thank them all.

And I would like to thank to all the Registrar staff of Işık University together with Sevgi Dikmen and the assistants in Işık University, for their friendly attitudes and supports of any kind.

Table of Contents

List of Figures

List of Algorithms

Chapter 1
Introduction

Wireless sensor network applications may require location information of the nodes, in order to operate efficiently, such as geographic routing protocols, event detection applications. *Global Positioning System, GPS,* is the most well-known location service in use today [1], but due to its power consumption, cost, size and inability to locate with desired precision for some applications,(an inexpensive GPS receiver can locate positions within ten meters for approximately 95% percent of measurements [2]), makes GPS a last resort solution. It may be feasible to implement a GPS based solution for a small scaled ad-hoc network, but one may not consider to implement it on a large scale network of more than 100 nodes [1]. Finding global coordinates with low cost and low energy consumption, is the main motivation behind the localization problem.

In this thesis we will present a robust localization framework, which uses normally distributed distance measurement constraints to model possible areas for a given set of nodes. Our framework can be implemented to work with different types of algorithms, and does not rely on anchors to operate. Anchors are the nodes that have apriori location information, either by the use of a GPS or being placed at known positions. Without anchors, algorithm can find local coordinates, this can be useful in some applications, which relies on relative locations. To fix the localization in a global coordinate system, at least $d+1$ anchors are needed in d-dimensional space.

Our framework makes use of the normal distribution's confidence intervals, when assigning probabilities to possible location's of a given node. Our framework needs nodes to have a ranging device to make distance mea-

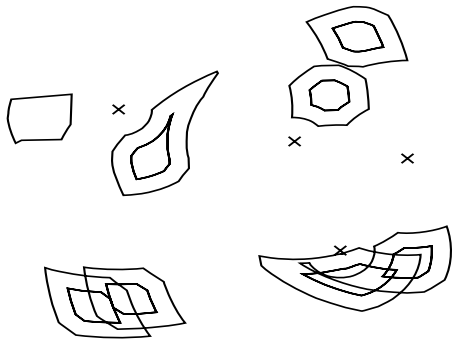


Figure 1.1: Feasible region constructed from the union of feasible regions, $F_{t'}$

surements between its neighbors and itself. The resulting assignment of a simple network consisting of 15 nodes with 4 anchors is give in figure 1.1.

Chapter 2
Previous Work

There is significant amount of previous work on wireless sensor network localization. One can classify these previous work according to some parameters to provide a clear insight.

First classification of these works can be based on where the computation takes place. Centralized algorithms achieve localization by doing most of the work in a limited number of central computers (more capable nodes). Three main approaches in centralized localization [3] are multidimensional scaling [4], linear programming [5] and stochastic optimization approaches [6]. In distributed algorithms the computation required in localization process is distributed to nodes, i.e there is no need to have a global information, related to the network. A node needs only the information from its neighbors in order to complete its localization process, in distributed approach [7, 8, 9, 10, 11]. Centralized approach can be useful where a centralized information architecture already exists. Centralized algorithms does not scale well, not feasible to be implemented for large scale networks. On the other hand distributed algorithms may need multiple iterations to converge a localization, and this may cause longer running times then centralized algorithms [3]. Multihop packet transmissions in centralized algorithms will reduce the overall energy in the network drastically, so if the number of hops to the central processor is more than the necessary number of iterations needed in the distributed approach then the distributed approach will be feasible [12].

Second of this classifications can be made according to the information needed for the localization process to take place. A class of algorithms may only need connectivity information between nodes [13, 14, 10, 8]. A sec-

ond class of algorithms can need distance information between neighboring nodes [7, 15, 11]. Algorithm that need distance information can be further divided into 4 categories based on how the distance information is gathered, time difference of arrival (TDOA) [16, 17], time of arrival (TOA) [18], received signal strength indication (RSSI) [11, 19] and by using an optical receiver [20]. Third class of algorithms can need angle of arrival information [21, 22]. Algorithms that use a combination of these measurements such as angle of arrival and distance [9] can be considered as a fourth class.

Algorithms utilizing distance or connectivity information can be further divided into 2 categories according to the connectivity or distance information are exchanged whether between one-hop [7] or multi-hop neighbors [10].

Using only the connectivity information may maximize the lifetime, and minimize the circuitry costs of a sensor but the localization result may be limited in precision compared with the algorithms that use more information such as distance or angle. On the other hand in order to use a distance measurement, sensors need to be equipped with at least one signal receiver/sender circuitry pair. For example algorithms that utilize TDOA technique need sensors to be equipped with both RF and ultrasound hardware. While TOA technique will need an RF hardware. A more complex hardware is needed to determine the AOA with limited error [3]. RSSI data distinguishes itself from the rest of the measurement techniques, because it does not require any additional hardware but RSSI also produces results that are limited in precision. [23] Also RSSI is used in cooperation with the access points, whose locations are known and can be used as anchors. [24]

Third classification can be based on the results of the algorithm. A class of algorithms try to find a single point for a given node at every step, called *unique localization*,[7], while a second class can assign more than one positions to a single node, called *finite localization*, [15]. Some algorithms including ours will assign possible areas to nodes during the localization

[9]. The area approach also may be subject to the previous classification, according to the assignment of the area to nodes. If the assignment of a single area is made to a single node than as an analogy to point case, we may also call this as unique localization. If a node can be assigned to more than one areas then this may also be called as finite localization. Finite localization keeps all possible location information that can be produced with the given amount of information at the time of localization, does not try to estimate a single position, and at the later stages of localization it is expected to reduce the number of possible locations with newly added information like [15]. Unlike finite localization, unique localization techniques try to estimate a single point, even when the information at the given time is insufficient, error produced in the earlier stages of the localization accumulates and the results can show unpredictable amount of difference from the ground truth. Finite localization techniques need to limit the number of possible locations, that may tend to grow exponentially.

Handling noise is an important aspect of the above mentioned algorithms. Noise can present in the system as an environmental noise and hardware noise. Hardware noise is easy to model, and simply modelled as Additive Gaussian White Noise. Contrary to this, modeling environmental noise is tricky. Some class of algorithms use *Noisy Disk* model for ultra sound and radio signal strength because it is easy to use in theoretical analysis and simulation. Noisy disk model has two parts, connectivity and noise. Connectivity component determines the maximum distance where two nodes are assumed to have a connection between themselves. Noisy disk with no noise is called as *Unit Disk* model [25]. We will use a model like noisy disk for our framework, like the one used in [7]. The parameters of the model used in [7] is borrowed from *Cricket Location Support System* [26]. In fact the noisy disk is not capable of modeling the environmental noise, but our model can be tailored to work with such a realistic model for both environmental noise and radio propagation. A novel experimental

approach proposed in [25], can be used to model noise for simulation purposes. It includes collecting the real-world distance measurements between sensors, for predetermined distance intervals and using randomly selected samples from the real-world data to model noise, which they call *Sampled Noise* and *Sampled Connectivity*.

Localization problem is related with the *Graph realization problem*. Graph realization is also critically important in molecular conformation [27]. Given a graph $G = (V, E)$ consisting of n vertices and m edges, graph realization problem is to assign vertices to coordinates such that all edge length constraints are satisfied [28]. There is also a vast amount of work on the graph realization problem. The main questions in the area are what are the conditions for unique localization of a network and what is the computational complexity of the localization. In [29] the above questions are explored in details. In graph theoretic part of the localization a concept of *rigidity* is critical, rigidity related concepts will be explained in this part, unless otherwise stated we will only talk about the two dimensional version of the problem. A realization of G is the assignment of all vertices in G to coordinates in Euclidean space. If we call this assignment as p then the combination of a graph and a realization is called as a framework and denoted as $p(G)$. The given framework $p(G)$ is said to be *flexible* if it has continuous deformations, and if the framework has only limited number deformations (discrete deformations) it is called as *rigid*. A condition to test rigidity of a given graph is described by Laman in [30]. It says that given a graph $G = (V, L)$ with n vertices and m edges is generically rigid in $\Re^2$ if and only if L contains a subset E consisting of $2n - 3$ edges with the property that for any non-empty set $E' \subset E$, the number of edges in E' can not exceed $2n' - 3$ where n' is the number of vertices of G which are endpoints of edges in E' [31]. A test to decide whether the given graph is rigid or not is proposed in [32] based on Laman's theorem , it is called as pebble game. A graph G is *redundantly rigid* if a removal of any single edge does not

destroy its rigidity. A graph $G = (V, E)$ with $n >= 4$ vertices, is said to be *globally rigid* if it is redundantly rigid and 3-connected. There are several graph construction methods to ensure that the resulting graph will be globally rigid such as Henneberg's method [33]. Global rigidity is the necessary condition for a unique graph realization. But the graph realization problem is shown to be still hard even it is known that the graph is globally rigid and it has a realization [31]. With no noise at all, localization problem with both angle(with orientation) and distance information is trivial. However in the presence of even a small amount of noise, it has been shown that the localization problem is NP-hard [9].

Finite localization is closely related with the bilateration ordering. Given $G = (V, E)$ bilateration ordering,π, of V is the ordering of nodes in V such that for every node $u \in V$, before reaching u in π we must have encountered by two of u's neighbors. And by intersecting the two edge constraints that belong to u one can only find two possible positions for u. A sample for a bilateration based algorithm can be found in [15].

Two of the algorithms that are closely related to ours are [8] and [9]. Sextant framework [8] is one of the area based localization algorithms, it uses connectivity information to extract non-convex constraints, also utilizes the negative information. Negative information can be defined as where the given node can not be. It does not assume uniform transmission radii(i.e a unit disk graph) or symmetric connectivity. Sextant uses bezier curves to represent geographic constraints. Our algorithm uses polygon approximation to represent geographic constraints, the precision of our algorithm can be adjusted, like its running-time. To have more precise results after a run of our algorithm, one need to use more vertices in circle approximation in the beginning of the algorithm. In [9] both angle and distance information is used. They approximate circular areas to polygons, extract constraints from these *"feasible regions"* and use linear programming formulations to solve localization problem.

Chapter 3
Robust Localization Framework

Let $G = (V, E)$ represent a real-world sensor network with n nodes and m edges. Nodes with apriori location information are called *anchors* and are denoted with A where $A \subset V$. Each pair of nodes $u, v \in V$ that are within a sensing range is represented with edge $(u, v) \in E$.

We assume that nodes have means for measuring distances to the neighboring nodes. Each such measurement is modelled as a gaussian random variable d where the measured distance d_m is the mean of d with the variance σ^2. We assume that σ^2 is a preset parameter of the network. We use *confidence interval* concept as a basis to represent a normally distributed measurement in our framework. Confidence interval is an interval in which a measurement or trial falls corresponding to a given probability [34]. It is possible to use a distribution other than gaussian as well. The only modification would be to use confidence intervals associated with the assumed distribution. Simulations in [7] assumes the distance measurements between sensors are gaussian and the variance of this measurements are based on the Cricket location support system [26]. The experimental approach proposed in [25] may also be used if the real-world data of sensors is provided. Sampled noise and sampled connectivity approaches in [25] makes use of real-world measurement noise and connectivity information to model noise in simulations.

Within our framework the goal of robust localization is to assign each node $u \in V$ with a set of *feasible regions*, $F_u = \{F_{u1}, F_{u2}, \ldots, F_{uk}\}$. Each feasible region $F_{ui} \in F_u$ consists of a set of simple polygons (possibly with holes) and is associated with a *confidence* c_{ui} that represents an approximate probability of u being within F_{ui} where $\sum_{i=1}^{k} c_{ui} \leq 1$. Since we assign

confidences to feasible regions, the confidence values of polygons within a feasible region are equal and $c_{ui} \neq c_{uj}$ unless $i = j$. We note that anchors may constitute a special case where the feasible region consists of a single polygon which is a point.

3.1 Constructing a Feasible Region

Let node s be the source of a measurement to a node t, where F_s is previously constructed. We construct F_t from F_s by using the measured distance $|st| = N(d_m, \sigma^2)$, where $(s,t) \in E$ and N indicates the normal distribution with mean d_m and variance σ^2. First case occurs if s is an anchor, the second is the non-anchor case.

In anchor case F_{ti} is simply a ring between the circles with $d_m - 2\sigma$ and $d_m - \sigma$ radii, with center p_s, where p_s is the location of s (see Figure 3.1). $F_{ti'}$ is the middle ring, between circles with $d_m - \sigma$ and $d_m + \sigma$ radii, with center p_s, and finally the $F_{ti''}$ is the outermost ring between circles $d_m + \sigma$ and $d_m + 2\sigma$ radii, with the same center as the other two rings. We approximate these circles with k-gons, where k is a parameter which plays an important role in the complexity of the framework.

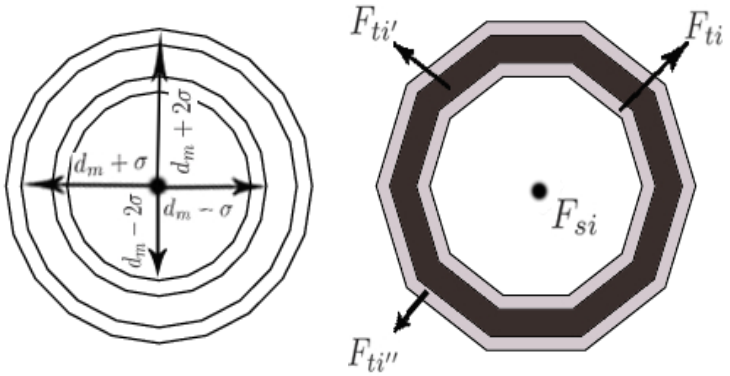
In non-anchor case, since each F_{si} is a set of polygons, before defining the construction operation we provide an *expansion operation* on polygons. Let $P = (p_1, p_2, \ldots, p_r) \in F_{si}$ be a polygon with a centroid of p_c. The expansion transformation on P is defined as $E(P,t) = (p_1 + \vec{t_1}, p_2 + \vec{t_2}, \ldots, p_r + \vec{t_r})$ where each vector t_i has a magnitude of t and is in the direction of the vector $(\overrightarrow{p_c, p_q})$. Each F_{si} gives rise to three feasible regions $F_{ti}, F_{ti'}, F_{ti''}$. We have 2 cases in feasible region construction.

$$F_{ti} = \bigcup_{\forall P \in F_{si}} (E(P, d_m - \sigma) - E(P, d_m - 2\sigma))$$

$$F_{ti'} = \bigcup_{\forall P \in F_{si}} (E(P, d_m + \sigma) - E(P, d_m - \sigma))$$

$$F_{ti''} = \bigcup_{\forall P \in F_{si}} (E(P, d_m + 2\sigma) - E(P, d_m + \sigma))$$

For the non-anchor case above equations describes the construction. The first terms are the outer bounds and second terms are the holes. An example for this construction can be seen in Figure 3.2. The confidence values associated with each feasible region are $c(F_{ti}) = c(F_{ti''}) = c(F_{si}) \times 13.6\%$ and $c(F_{ti}) = c(F_{si}) \times 68.4\%$. In anchor case $c(F_{si})$s assumed to be one.



(a) (b)

Figure 3.1: Circles and their k-gon approximations

A more complex feasible region construction is shown in Figure 3.2. F_s has three elements $F_s = (F_{s1}, F_{s2}, F_{s3})$. F_{s1} and F_{s2} has eight simple polygons in them, while F_{s3} has two. Each F_{si} has a unique confidence. Polygons within one F_{si} have same confidence values.

3.2 Intersection of Feasible Regions

Let F_t and F'_t be two feasible regions for t, created by its neighbors u and v respectively. Let $F_t = (F_{t1}, F_{t2}, \ldots, F_{tw})$, and $F'_t = (F'_{t1}, F'_{t2}, \ldots, F'_{ty})$

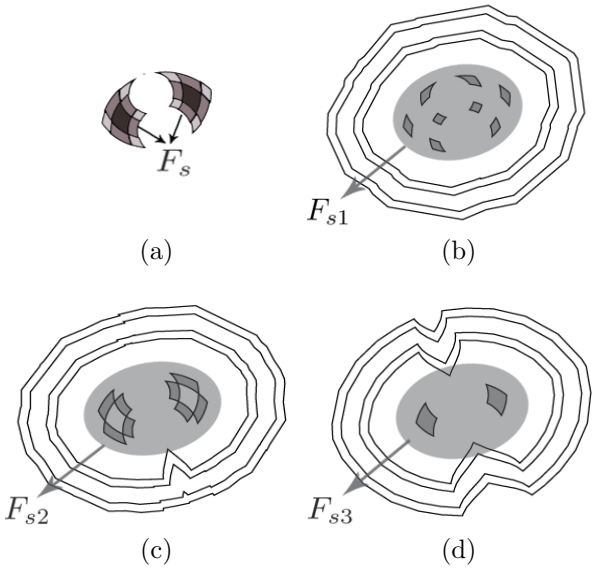


Figure 3.2: (a)F_s (b,c,d) Feasible regions constructed from F_{s1}, F_{s2}, F_{s3}

After formally defining the input data, the intersection is given by :

$$F_t \cap F'_t = \bigcup_{i=1}^{w} F_{ti} \cap \bigcup_{j=1}^{y} F'_{tj}$$
$$= \bigcup_{i=1}^{w} \bigcup_{j=1}^{y} F_{ti} \cap F'_{tj}$$

A sample intersection of 3 rings $F_t = (F_{t1}, F_{t2}, F_{t3})$ with 3 rings $F'_t = (F'_{t1}, F'_{t2}, F'_{t3})$ can be seen in Figure 3.3. As described formally, F_{t1} will be intersected with all elements of F'_t, then this procedure will continue for all elements of F_t. Union of these partial intersections will give us the resulting polygons, with corresponding confidence values.

Since the result of the intersection is a feasible region, if t has more than two neighbors, we will carry out the same operation between the next neighbor and the result of intersection.

After the intersection operation we need to assign new probabilities to

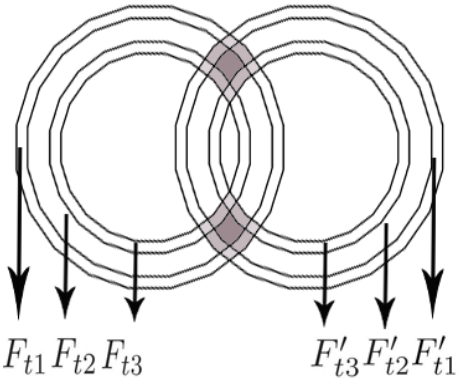


Figure 3.3: Shows the intersection of F_t and F'_t

feasible regions created as a result of an intersection. If we continue with the above intersection operation, and simplify it a little bit. Let us consider only a single element of F_t and F'_t. Let $P \in F_t$ and $P' \in F'_t$ be two feasible regions, and the problem is to assign a probability to their intersection. Let $c(P) = d$ and $c(P') = f$. Let p_t be a point in space, that corresponds to the real location of node t.

$$Pr(p_t \ in \ P \cap P') = Pr(p_t \ in \ P | p_t \ in \ P') \times Pr(p_t \ in \ P')$$

If we have the apriori information that u is in P', the probability of u being in P does not change, since two measurements are independent but the probability of u being in $P - (P \cap P')$ will be zero, i.e area of P changes, but the probability of p_t being in P does not. Then we have $c(P \cap P') = d \times f$.

3.3 Post Processing

As a result of the construction operation, there may have more than one feasible regions intersecting over a subregion. We propose methods to deal with these intersections. The aim in applying these procedures to inter-

Algorithm 1 Intersection of 2 feasible regions

```
1: procedure INTERSECTION( F_tu, F_tv )
2:     list of feasible regions L_P
3:     for all F_ti ∈ F_t do
4:         for all F'_tj ∈ F'_t do
5:             c(F_t) = c(F_ti) × c(F'_tj)
6:             if c(F_t) exists in L_P then
7:                 L_P[c(p)] = L_P[c(p)] ∪ F_t
8:             else
9:                 L_P[c(p)] = F_t
```

section of feasible regions is it will be meaningless that there will be two probabilities for the same region at the resulting assignment. Another problem is to limit the increase in the number of feasible regions. When F_s has k feasible regions, F_t will have $3k$ feasible regions, to limit the increase in number of regions in F_t we apply three more methods: *approximate reduction, discarding the regions with low probability and limiting the number of feasible regions.* A selection of these methods can be applied to resulting feasible regions, both after construction and intersection operations. A sample algorithm based on these methods are given in algorithm 2.

Algorithm 2 Sample post-processing algorithm

```
1: procedure POST-PROCESSING OF FEASIBLE REGIONS(F_t, A, α)
2:     PRESERVE UNIQUENESS(F_t)
3:     DISCARD REGIONS WITH LOW PROBABILITY(F_t, ε)
4:     for |F_t| > A do
5:         APPROXIMATE REDUCTION(F_t, α)
6:         α ← α − ε
```

Precise regions, can be checking all feasible regions F_{ti} and if any two regions F_{ti} and F_{tj} intersect between $r_1\%$ and $r_2\%$ of the smaller region, F_{tj}, then we create a new feasible region $F_{ti} \cap F_{tj}$ and this area is excluded from its parents, $c(F_{ti} \cap F_{tj})$ becomes the simple average of $c(F_{ti})$ and $c(F_{tj})$. Obviously this method increases the number of subregions while preserving the accuracy, but the increase in the number of feasible regions will grow

exponentially. This method can be applied both after construction and intersection.

Approximate reduction is the first method to reduce the number of subregions. Approximate reduction start with, $F_{si} \in F_t$, and checks all subregion pairs, (F_{si}, F_{sj}), and finds r_3, the ratio of the intersection area to the area of the smaller subregion F_{sj}. Then we find $r_4 = 1 - |c(F_{si}) - c(F_{sj})|$, r_4 is used as a measure to represent the closeness of the confidences for two feasible regions. Then if $r_3 + r_4 > \alpha$, where α is a given parameter, then we unite F_{si} and F_{sj}, $c(F_{si} \cup F_{sj})$ will be the weighted average of $c(F_{si})$ and $c(F_{sj})$, weighted with their areas. This method can be applied both after construction and intersection.

Discarding regions with low probability, is removing F_{ti}'s where $c(F_{ti}) < \epsilon$, ϵ is a predefined threshold near zero, so when a probability of a feasible region goes to zero we can discard it. This method can be applied both after construction and intersection.

Limiting the number of feasible regions, is to use appoximate reduction technique with lower r values until the number of feasible regions falls below a predetermined constant, m. An alternative to this can be to use a predefined parameter A to represent maximum number of feasible regions for any given node and if A is not reached after applying approximate reduction method, then we can divide the confidence interval values [0,100] to A partitions and unite the feasible regions that is found to be in the same partition.

Preserving the uniquness of confidences: If $c(F_{ti}) = c(F_{tj})$ then $F_{ti} = F_{ti} \cup F_{tj}$ and F_{tj} will be removed, this will continue since all elements in F_t will have unique reliabilities. This method can be applied both after construction and intersection.

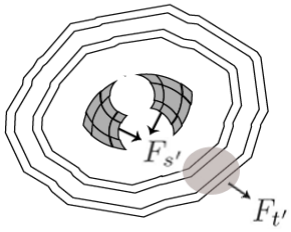


Figure 3.4: Feasible region constructed from the union of feasible regions, $F_{t'}$

Aggregate and its parts: Let $F_s = (F_{s1}, F_{s2}, F_{s3})$ and F_s defines a closed region, with no holes , for example in Figure 3.2 F_{s1}, F_{s2}, and F_{s3} will form a closed region for s when they are united. This method can be applied only after construction operation.

$$F_{s'} = \bigcup_{i=1}^{3} F_{si}$$

Let $F_{t'}$ be a feasible region of t constructed from $F_{s'}$, see Figure 3.4.

$$F_{ti} = F_{ti} \cap F_{t'}$$
$$F_{ti'} = F_{ti'} \cap F_{t'}$$
$$F_{ti''} = F_{ti''} \cap F_{t'}$$

This can be interpreted as a feasible region F_{ti} constructed from a sub-region of F_s can not be outside the $F_{t'}$, the feasible region constructed with the union of F_{si}s. In Figure 3.2, the areas are not cropped yet.

3.4 Complexity Analysis

All geometric objects used in the framework are represented as polygons, including the circles. Polygons can contain holes. Most of the algorithms on polygons depend on the number of vertices, by changing a single parameter

we can adjust the accuracy of the localization algorithm, as well as the running time. For example, a circle C is represented with a polygon P with k vertices on C, for more accuracy one can increase k, for less computing power and less accuracy k can be decreased.

In our analysis we assume the maximum number of polygons in a feasible region,F_{si}, be m and the maximum number of vertices in a single polygon be k.

Construction phase uses parameters $F_s = (F_{s1}, F_{s2}, \ldots, F_{sn})$ and d_m to construct F_t. Since we only apply a simple expansion on all polygons, the running time of the construction phase is $O(mkn)$. We limit n to a constant in our framework and k is a predefined constant, then we can omit n and k and now it becomes $O(m)$. Number of constructions needed is directly related to the number of edges in E. Each edge corresponds to a measurement that leads to a construction operation, that makes the total cost of construction $O(m|E|)$ where $|E|$ indicates the number of elements in E. If not bounded by the post-processing operations and intersection, m can grow exponentially. Since each construction creates $3m$ feasible regions where its ancestor has m feasible regions, m can be $3^{|}E|$ where the given graph is a linear order of nodes, i.e. only a single edge connects the neighboring nodes and each node has only one ancestor.

Intersection operation is applied when t has more than one neighbors. When applied, intersection decreases the total area covered by F_t, but the number of *subregions*,polygons with different confidence values, can increase. In order to reduce the growth we use the approaches proposed in sections 3.1. An intersection operation on 2 polygons having k vertices each, has the worst case running time of $O(k^2)$ [35]. We need to look for m^2 intersections and if each intersection takes $O(k^2)$ time then the worst case running time of intersection operation is $O((mk)^2)$ since k is a predefined constant it becomes $O(m^2)$. Intersection operation is also directly proportional to the number of edges, if $u \in V$ has s neighbors where $s \leq 2$ then $s - 1$

intersections will be applied, this makes the total number of intersections taken place a multiple of $|E|$. Then total running time due to intersections is $O(m^2|E|)$. Union operation needed afer the intersection results again needs point by point comparison, worst case running time of the union operation is the same as intersection. Total running time of the algorithm is $O(m^2|E| + m|E|)$ that is $O(|E|m^2)$.

Post-processing algorithms work on resulting polygons means they depend on m, and the maximum running time required by them is for comparing all pairs that takes $O(m^2)$ time.

Our framework uses three confidence intervals of normal random variable. One can increase the number of intervals used in algorithm to provide more accuracy, but after three confidence intervals adding more intervals becomes infeasible, since added intervals increase the complexity of the algorithm but do not add much to the accuracy, for example using 5 confidence intervals instead of 3, increases the accuracy only by 4.2% for a normal distribution.

Chapter 4
Sample Localization Algorithms

In this chapter we will define two sample localization algorithms based on the robust localization framework. First algorithm described throughout the section is based on an ordering such as trilateration or bilateration and the second algorithm is a distributed implementation of the framework. Since our framework does not impose any limitations beyond the measurement model and its representation, any other localization technique can work with our framework, such as distributed algorithms.

4.1 Centralized Algorithm

Each node $u \in V$ is assigned an integer label denoted with $l(u)$. This label will be used for storing an order on V. In order to limit the complexity of the resultant polygons, our algorithm assumes an ordering, $\pi = u_1, u_2, \ldots u_n$ such that for all $u \in V$, when iterating on π one will encounter at least two neighbors of u for bilateration ordering and at least three for trilateration, where the neighbors of u is denoted as $N(u)$. Ancestors of u is denoted as $anc(u) \subset N(u)$ and includes the neighbors of u which are used in the localization process of u, i.e. neighbors of u that appear before u in π. Label of nodes in V are re-assigned according to the order of appearance in π.

A node can possibly have more than one ancestors, then we construct feasible regions with respect to all ancestors, $anc(u)$, of u. For all $v \in anc(u)$ we create F_{uv}'s. Then we need to find F_u from F_{uv}'s, by intersecting them as described in the framework. We continue to apply the same procedure in algorithm 3, starting with order π then π' where π' is the inverse of π, until the improvement in feasible regions stops.

In the post-processing phase centralized algorithm uses the sample al-

gorithm 2 without its loop, which means we apply approximate reduction once with a predefined constant that will limit the number of feasible regions in an acceptable interval with no guarantee. Algorithm given as a sample limits the number of feasible regions tightly but has more running time.

Algorithm 3 Localization procedure for a centralized algorithm

```
1: procedure LOCALIZATION(G = (V, E))
2:     A ← anchors
3:     order ← an order on V
4:     for all u ∈ order and u ∉ A do
5:         anc(u) ← 1 hop ancestors of u
6:         if size of anc(u) > 1 then
7:             for all w ∈ anc(u) do
8:                 e ← the edge between w and u
9:                 F_uw ← feasible region constructed with e and F_w
10:                Append F_uw to list of feasible regions Lf
11:            loc(u) ← Intersection of feasible regions in Lf
```

4.2 Distributed Algorithm

In the distributed implementation of the framework, all nodes will start by measuring distances to neighboring nodes. If none of u's neighbors have a location information yet, then u will wait until a positional update occurs in one of its neighbors. When an update occurs in $v \in N(u)$, v will notify all its neighbors, including u. u will update its position according to the newly updated feasible region of v and the distance measured between u and v. Notification is a messege containing the feasible region of the sender.

Algorithm 4 Distributed algorithm

1: **procedure** INITIALIZATION OF A SIMPLE NODE
2: Measure the distances to neighboring nodes
3: Wait for a notification
4: **procedure** INITIALIZATION OF AN ANCHOR
5: Send notification to all neighbors
6: **procedure** NOTIFICATION HANDLER FOR NODE t (F_s)
7: Construct $F_{t'}$ according to F_s and (s, t)
8: If there exists F_t intersect it with $F_{t'}$
9: Notify neighbors

Chapter 5
Experiments

Our implementation is coded in C++, using LEDA [36] (Library of Efficient Data Types and Algorithms). We implemented the centralized algorithm based on the robust localization framework.

5.1 Experiment Parameters

One of the parameters that we need to inspect is k, the number of vertices on the $k-gon$ that we used to approximate a circle. Certainly k will effect the feasible regions produced at the later stages of the localization.

Second parameter is the variance and mean of $d = N(d_m, \sigma)$, the normal random variable modelling distance measurements between sensor nodes. As σ increases the area of the feasible regions can grow accordingly.

We will test our algorithm that uses robust localization framework in randomly generated graphs with varying average degrees. This will help us to find at which average connectivity level our framework works effectively. This makes average degree a parameter to be inspected. In addition to this, number of anchor nodes in a given network, a, directly affects the outcome of the localization process.

We will use a number of performance measures. The first one is the number of nodes that are localized, i.e. a feasible region assigned to that node. The second one is the number of nodes whose real locations are found to be in its feasible region. Third one is the average area of the feasible regions.

5.2 Experimental Results and Discussion

All random graphs in our experiments are generated in a 450x450 unit square area, with 30 nodes. A point in Figure indicates an average of the results after applying the algorithm to 10 different random graphs. All results plotted are obtained after a single run of the centralized algorithm, in fact a loop until no more improvement can be seen in the area of feasible regions can be preferred to obtain better results with more running time.

In the first of our experiments plotted in Figure 5.1, we generated random graphs with varying average degrees. The parameters other than average degree are fixed $k = 20$, $\sigma = 20$, $a = 4$.

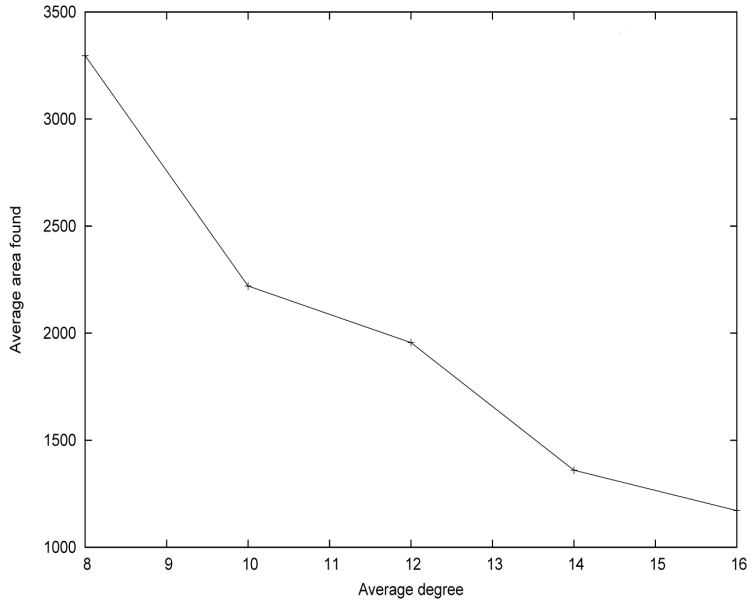


Figure 5.1: Average area assigned when average degree changes from 8 to 16

In almost all instances the real location of the nodes are found to be in their respective feasible regions assigned by the algorithm. Figure 5.1 shows the area of feasible regions decrease with the increase in average degree. There are only 12 instances in 50 runs used to create the Figure 5.1 that the algorithm failed to localize a node or two.

In second experiment plotted in Figure 5.2 we will fix the average degree to 14 depending on the results obtained from the first experiment and change a to see how it effects the outcomes.

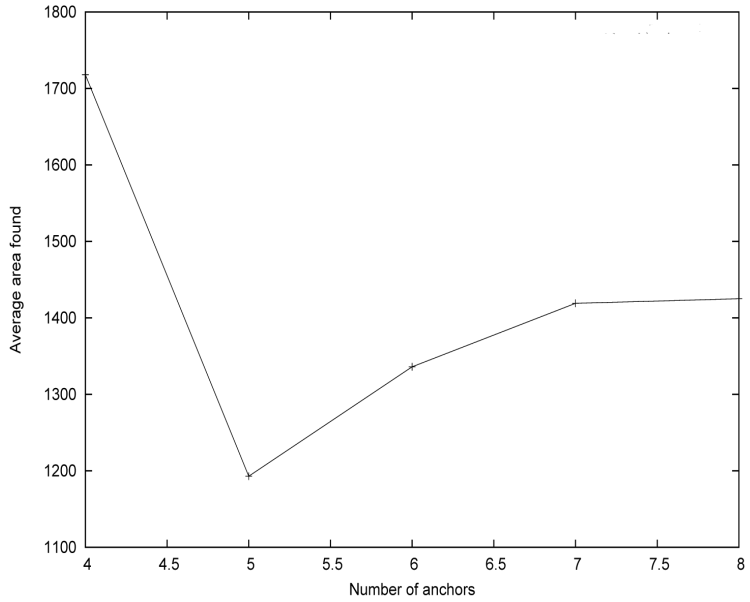


Figure 5.2: Average area assigned when a changes from 4 to 8

There are again 12 instances in all 50 runs plotted in Figure 5.2 that has a node or two that are not localized by the algorithm. Since the number of nodes in network is fixed to 30, then an increase in a will decrease the average area of feasible regions.

In third experiment plotted in Figure 5.3 we will fix a to 4 and average degree to 7, depending on the results of first two experiments and change k to see how it effects the results of the localization.

An increase in the number of vertices in the initial approximation of a circle will help us to cover the circle with more accuracy and increase the area covered in a circle. This explains the increase in the area of feasible regions when k increases (Figure 5.3).

The next experiment plotted in Figure 5.4 is to test how σ effects the outcome of the experiments, while the other parameters are fixed, $a = 4, k = 20, averagedegree = 7$.

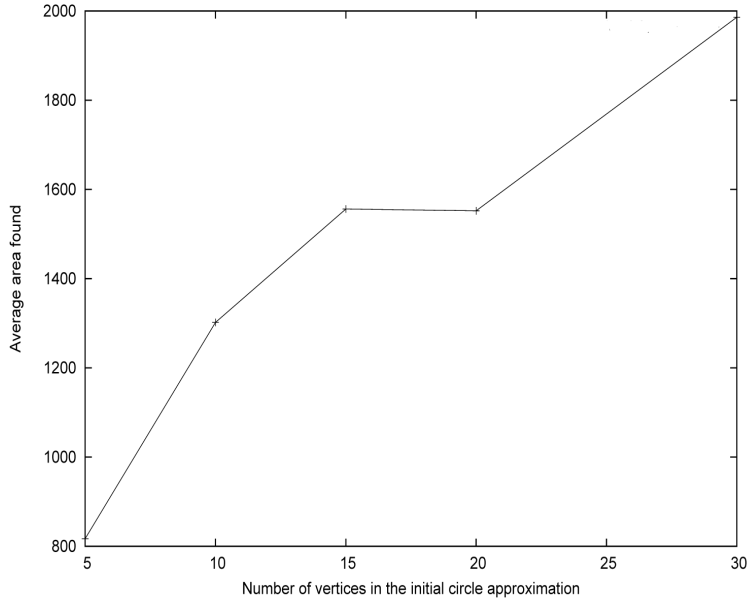


Figure 5.3: Average area assigned when k changes from 5 to 30

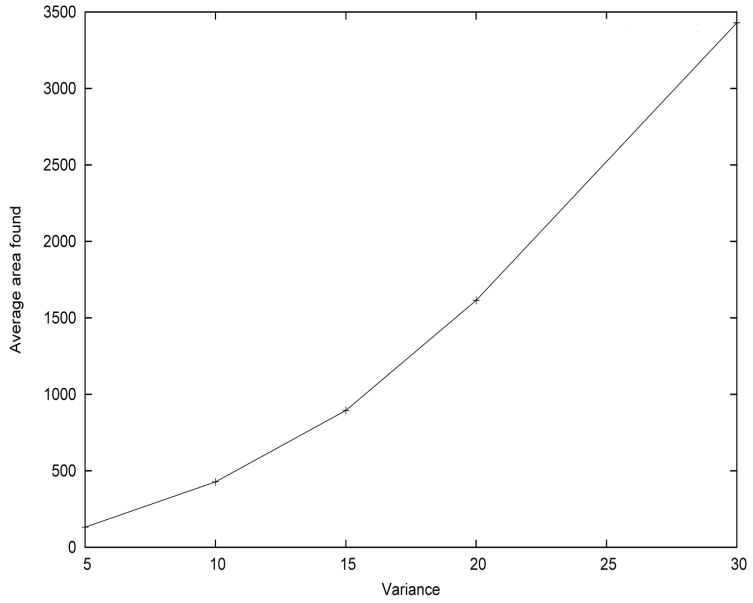


Figure 5.4: Average area assigned when variance changes from 5 to 30

As σ increases the area of the feasible regions increases, since an increase in σ means more inaccuracy for distance measurements.

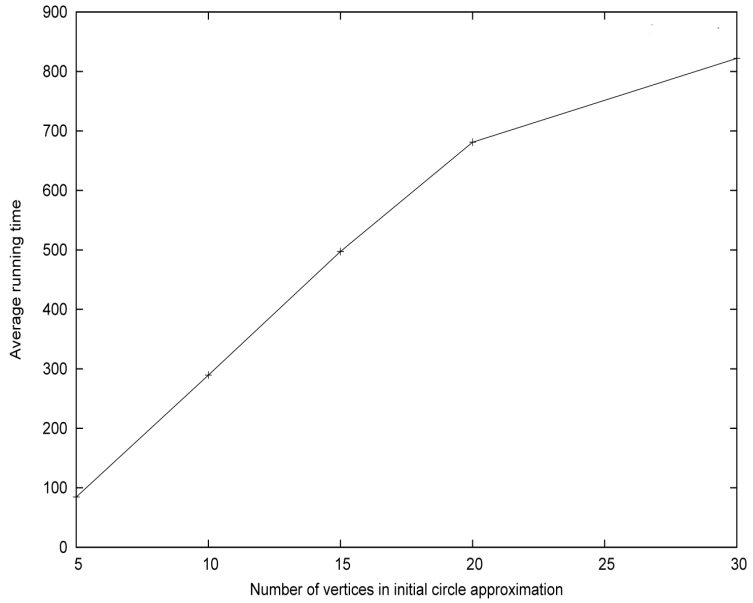


Figure 5.5: Average running times when k changes from 5 to 30

The running time plot in Figure 5.5 for varying k values given for a network of 20 nodes, $a = 4$, average degree = 7, and $\sigma = 15$. A single point in this plot is the average time(seconds) needed to process a random graph by the algorithm. The increase in k has a significant effect on the running time of the algorithm. It may be prefarable to use smaller k values with simple sensor configurations.

Chapter 6
Conclusion and Future Work

We proposed a framework to be used in localization of wireless sensor networks, and simulated it with a centralized algorithm that utilizes the framework. Our framework can be used with variety of algorithms and measurement models, which means it can be used as a basis for many localization approaches. A more comprehensive simulation can be carried out to find out that the framework is realistic enough to be used in real world applications. In order to do this frameworks that try to find possible areas as a result of localization process need to be run under more realistic noise models.

References

[1] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Comput. Netw.*, 43(4):499–518, 2003. [cited at p. 1]

[2] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, 2001. [cited at p. 1]

[3] G. Mao, B. Fidan, and B. D. O. Anderson. Wireless sensor network localization techniques. *Comput. Netw.*, 51(10):2529–2553, 2007. [cited at p. 3, 4]

[4] C. Wu, W. Sheng, and Y. Zhang. Mobile sensor networks self localization based on multi-dimensional scaling. *Robotics and Automation, 2007 IEEE International Conference on*, pages 4038–4043, April 2007. [cited at p. 3]

[5] P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. *Dept. of Computer Science, Stanford University*, (69), 2003. [cited at p. 3]

[6] L. Hu and D. Evans. Localization for mobile sensor networks. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 45–57, New York, NY, USA, 2004. ACM. [cited at p. 3]

[7] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 50–61, New York, NY, USA, 2004. ACM. [cited at p. 3, 4, 5, 8]

[8] S. Guha, R: Murty, and E.G. Sirer. Sextant: A unified node and event localization framework using non-convex constraints. In *Proceedings of The International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*. The International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc), May 2005. [cited at p. 3, 7]

[9] A. Basu, J. Gao, J. S. B. Mitchell, and G. Sabhnani. Distributed localization using noisy distance and angle information. In *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 262–273, New York, NY, USA, 2006. ACM. [cited at p. 3, 4, 5, 7]

[10] D. Niculescu and B. Nath. Ad hoc positioning system (aps) using aoa. In *In INFOCOM 03*, 2003. [cited at p. 3, 4]

[11] P. Bahl, V. N. Padmanabhan, and A. Balachandran. Enhancements to the radar user location and tracking system. Technical report, 2000. [cited at p. 3, 4]

[12] M. Rabbat and R. Nowak. Distributed optimization in sensor networks. In *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*, pages 20–27, New York, NY, USA, 2004. ACM. [cited at p. 3]

[13] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. Localization from mere connectivity. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 201–212, New York, NY, USA, 2003. ACM. [cited at p. 3]

[14] B. Blum J. A. Stankovic T. He, C. Huang and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *9th International Conference on Mobile Computing and Networking (MobiCom*, 2003. [cited at p. 3]

[15] D. K. Goldenberg, P. Bihler, M. Cao, J. Fang, B. D. O. Anderson, A. S. Morse, and Y. R. Yang. Localization in sparse networks using sweeps. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 110–121, New York, NY, USA, 2006. ACM. [cited at p. 4, 5, 7]

[16] S.R. Drake and K. Dogancay. Geolocation by time difference of arrival using hyperbolic asymptotes. *Acoustics, Speech, and Signal Processing, 2004.*

Proceedings. (ICASSP '04). IEEE International Conference on, 2:ii–361–4 vol.2, May 2004. [cited at p. 4]

[17] J. Xiao, L. Ren, and J. Tan. Research of tdoa based self-localization approach in wireless sensor network. *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2035–2040, Oct. 2006. [cited at p. 4]

[18] C. Chong I. Guvenc and F. Watanabe. Joint toa estimation and localization technique for uwb sensor network applications. *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, pages 1574–1578, April 2007. [cited at p. 4]

[19] X. Li, H. Shi, and Y. Shang. A sorted rssi quantization based algorithm for sensor network localization. In *ICPADS '05: Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, pages 557–563, Washington, DC, USA, 2005. IEEE Computer Society. [cited at p. 4]

[20] K. Römer. The lighthouse location system for smart dust. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 15–30, New York, NY, USA, 2003. ACM. [cited at p. 4]

[21] W. Whiteley T. Eren and P.N. Belhumeur. Using angle of arrival (bearing) information in network localization. *Decision and Control, 2006 45th IEEE Conference on*, pages 4676–4681, Dec. 2006. [cited at p. 4]

[22] P.Rong and M.L. Sichitiu. Angle of arrival localization for wireless sensor networks. *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, 1:374–382, Sept. 2006. [cited at p. 4]

[23] X. Li E. Elnahrawy and R. P.Martin. The limits of localization using signal strength: a comparative study. In *Proceedings of IEEE Sensor and Ad Hoc Communications and Networks (SECON 2004)*, pages 406–414, 2004. [cited at p. 4]

[24] P. Bahl and V. N. Padmanabhan. Radar: An in-building rf-based user location and tracking. In *System, 19 th Anual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM), Tel Aviv*, 2000. [cited at p. 4]

[25] K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler. The effects of ranging noise on multihop localization: an empirical study. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 10, Piscataway, NJ, USA, 2005. IEEE Press. [cited at p. 5, 6, 8]

[26] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. pages 32–43, 2000. [cited at p. 5, 8]

[27] B. Hendrickson. The molecule problem: Exploiting structure in global optimization. *SIAM Journal of Optimization*, 5(4):835–857, 1995. [cited at p. 6]

[28] B. Hendrickson. Conditions for unique graph realizations. *SIAM J. Comput*, (21):65–84, 1992. [cited at p. 6]

[29] T. Eren, D. K. Goldenberg, W. Whiteley, and Y. R. Yang. Rigidity, computation, and randomization in network localization. In *In Proceedings of IEEE INFOCOM 04, Hong Kong*, pages 2673–2684, 2004. [cited at p. 6]

[30] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4:331–340, 1970. [cited at p. 6]

[31] W. Whitley Y. Yang A. Morse B. Anderson T. Eren, D. Goldenberg and P. Belheumer. Rigidity, computation, and randomization in network localization. In *in Proceedings of IEEE Infocom 2004*, 2004. [cited at p. 6, 7]

[32] D. Jacobs and B. Hendrickson. An algorithm for two-dimensional rigidity percolation: the pebble game. *J. Comput. Phys*, (137):346–365, 1997. [cited at p. 6]

[33] L. Henneberg. *Die Graphische Statik der Starren Systeme*. Johnson Reprint, Liepzig, 1911. [cited at p. 7]

[34] E. W. Weisstein
. Confidence interval. http://mathworld.wolfram.com/ConfidenceInterval.html. From MathWorld–A Wolfram Web Resource. [cited at p. 8]

[35] S. Z. Bao M. Gombosi Y. K. Liu, X. Q. Wang and B. Zalik. An algorithm for polygon clipping and for determining polygon intersections and unions. *Computers & Geosciences*, 33:589–598, 2007/5. [cited at p. 16]

[36] K. Mehlhorn and S. Nher. Leda: A platform for combinatorial and geometric computing, 2000. [cited at p. 21]

Curriculum Vitae

Publications :

[1] Olca A. Çakiroglu, Cesim Erten, Ömer Karatas, Melih Sözdinler *Crossing Minimization in Weighted Bipartite Graphs.* WEA,2007, 122-135

[2] Olca A. Çakiroglu, Cesim Erten, Ömer Karatas, Melih Sözdinler *Crossing Minimization in Weighted Bipartite Graphs* to appear in Journal of Discrete Algorithms