A Robust Localization Framework to Handle Noisy Measurements in Wireless Sensor Networks

Cesim Erten
Computer Engineering
Kadir Has University
Email: cesim@khas.edu.tr

Ömer Karataş
Computer Science and Engineering
Işık University
Email: okarat@gmail.com

***Abstract*—We construct a robust localization framework to handle noisy measurements in wireless sensor networks. Traditionally many approaches employ the distance information gathered from ranging devices of the sensor nodes to achieve localization. However the measurements of these devices may contain noise both as hardware noise and as environmental noise due to the employment conditions of the network. It is necessary to provide a general framework that handles such a noise in data and yet still be applicable within several localization algorithms. In order to handle noise in distance measurements, our framework utilizes convex constraints and confidence intervals of a random variable. At the end of the localization process nodes are assigned to a set of *feasible regions* with corresponding probabilities. The accuracy of the localization can be adjusted and the framework can easily be embedded to work within previously suggested localization algorithms.**

I. Introduction

Many applications and systems from areas such as environment and habitat monitoring, weather forecast, and health applications require use of many sensor nodes organized as a network collectively gathering useful data; see [1] for a survey. In such applications it is usually necessary to know the actual locations of the sensors. Sensor network localization is the problem of assigning geographic coordinates to each sensor node in a given network. *Global Positioning System, GPS,* is the most well-known location service in use today [2], 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 [3]), GPS usually is considered 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 [2]. Finding global coordinates of the nodes assuming limited use of GPS-embedded nodes is the main motivation behind the sensor network localization problem. Many localization algorithms assume the existence of the distance information between neighboring nodes [8], [12], [15]. Such information may be gathered by employing one of the existing techniques such as time difference of arrival (TDOA) [16], [17], time of arrival (TOA) [18], received signal strength indication (RSSI) [12], [19] or by using an optical receiver [20]. Although many of these algorithms successfully localize the given network simply employing the gathered distance information, one problem that remains unresolved is the handling of the noise in the input data. Most algorithms assume the input data is noise-free. We provide a robust localization framework, which uses normally distributed distance measurement constraints as a basis to model noisy data and provide possible areas as a result of the localization on a given set of nodes. Our framework is general in the sense that it can be implemented to work with different types of *noise-free* localization algorithms. The framework does not rely on the existence of *anchors*, nodes with *a priori* location information, either by the use of a GPS or being placed at known positions. Without anchors, algorithm can find local coordinates which prove to be useful in applications relying on relative coordinates such as geographic routing. The only requirement is that each node is equipped with a ranging device to make distance measurements to neighbor nodes.

Partially supported by TUBITAK grant 106E071

II. Previous Work

There is significant amount of previous work on wireless sensor network localization. An important distinction between the previous work lies on the model of computation. 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 [4] are multidimensional scaling [5], linear programming [6] and stochastic optimization approaches [7]. In distributed algorithms the computation required in the localization process is distributed to the nodes. 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 achieve the localization [8]–[12]. Handling noise is an important aspect of these algorithms. Noise may be present in the system as an hardware noise or environmental noise. Hardware noise is easy to model, and is usually modelled as Additive Gaussian White Noise. In contrary, modelling the environmental noise is more tricky. Some class of algorithms use the *Noisy Disk* model for ultra sound and radio signal strength as it is easy to use in theoretical analysis and simulations. 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. The disk model with no noise is called the *Unit Disk* [23].

978-1-4244-5023-7/09/$25.00 ©2009 IEEE

ISCIS September 14-16, 2009
METU
Northern Cyprus Campus

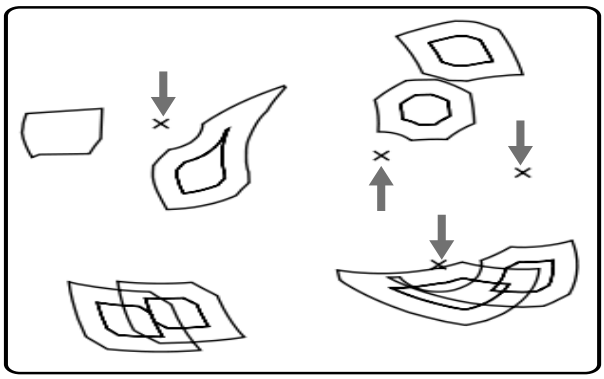

Fig. 1. Localization of a sample network. Anchors are marked with arrows.

We use a model similar to the noisy disk described in [8]. The parameters of the model used in [8] is borrowed from *Cricket Location Support System* [24]. In fact the noisy disk is not capable of modelling the environmental noise, but our model can be tailored to work with such a realistic model for both environmental noise and the radio propagation. The novel experimental approach proposed in [23] 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*. Two of the algorithms that are closely related to ours are [9] and [10]. Sextant framework [9] is one of the area based localization algorithms. It uses connectivity information to extract non-convex constraints and utilizes the negative information. 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. In [10] both angle and distance information is used. Circular areas are approximated to polygons and the resulting constraints are solved using linear programming formulations.

III. Robust Localization Framework

Let $G = (V, E)$ represent a real-world sensor network with n nodes and m edges. Nodes with a priori 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 variance σ^2. We assume that σ^2 is a preset parameter of the network. We use the *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 distribution [25]. 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 [8] assume the distance measurements between sensors are gaussian and the variance of these measurements is based on the Cricket location support system [24]. The experimental approach proposed in [23] may also be used if the real-world data of sensors is provided. Sampled noise and sampled connectivity approaches in [23] make 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*, $F_u = \{F_{u1}, F_{u2}, \ldots, F_{ur}\}$. The resulting feasible region assignment of a simple network consisting of 15 nodes with 4 anchors is given in Figure 1. 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 $\sum_{i=1}^{r} c_{ui} \leq 1$. Since we assign confidences to feasible regions, the confidence values of polygons within a feasible region are all equal and $c_{ui} \neq c_{uj}$ unless $i = j$. We note that anchors constitute a special case where the feasible region consists of a single polygon which is a point. The confidence of the polygon is set to 1 for this special case.

A. Constructing Feasible Regions

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 not-an-anchor case. In the anchor case F_t consists of three feasible regions, $F_t = \{F_{t1}, F_{t2}, F_{t3}\}$. Let p_s be the location of the anchor s. F_{t1} is the innermost ring between the circles centered at p_s with $d_m - 2\sigma$ and $d_m - \sigma$ radii. F_{t2} is the middle ring between the circles centered at p_s with $d_m - \sigma$ and $d_m + \sigma$ radii. Finally, F_{t3} is the outermost ring between the circles with $d_m + \sigma$ and $d_m + 2\sigma$ radii; see Figure 2. Simulating the confidence intervals of the normal distribution, the confidence values associated with each feasible region are assigned as $c(F_{t1}) = c(F_{t3}) = 13.6\%$ and $c(F_{t2}) = 68.4\%$. We note that the circles under discussion are approximated with regular k-gons, where k is a preset parameter that plays an important role in the complexity of the framework. In the not-an-anchor case, since each F_{si} is a set of polygons, before defining the construction operation we provide an *expansion operation* on polygons. Let $P = (p_1, p_2, \ldots, p_q) \in F_{si}$ be a polygon with a centroid of p_c. The expansion transformation on P is defined as $E(P, t) = (p_1 + \vec{t_1}, p_2 + \vec{t_2}, \ldots, p_q + \vec{t_q})$ where each vector t_i has a magnitude of t and is in the direction of the vector $(\overrightarrow{p_c, p_i})$. Each F_{si} gives rise to three feasible regions $F_{ti}, F_{ti'}, F_{ti''}$:

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

The first terms are the outer boundaries and second terms are the holes. The confidence values associated with each feasible

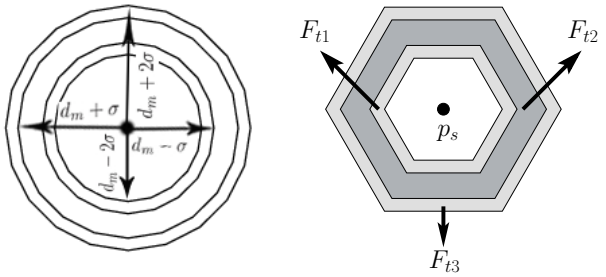

Fig. 2. Feasible region construction when s is an anchor located at p_s. (a) Confidence intervals; (b) Feasible regions of t and approximating the circles with a k-gon when $k = 6$.

region are $c(F_{ti}) = c(F_{ti''}) = c(F_{si}) \times 13.6\%$ and $c(F_{ti}) = c(F_{si}) \times 68.4\%$. An example for this construction can be seen in Figure 3 where $F_s = \{F_{s1}, F_{s2}, F_{s3}\}$. The feasible regions F_{s1}, F_{s2} consist of eight simple polygons each whereas F_{s3} contains only two.

B. Updating Feasible Regions via Intersections

Using the construction method described in the previous subsection we can deduce a feasible region for each edge $(s,t) \in E$. If there are multiple edges incident on node t, the feasible regions arising from all the neighbors' measurements must be updated for each edge. We describe the update procedure inductively. Let d_t denote the degree of node t, i.e. the number of neighbors within its sensing range. If $d_t = 1$, the feasible region can simply be computed using the described construction method. Now for $d_t > 1$, assume the set of feasible regions F_t for $d_t - 1$ neighbors is already computed. Let F'_t be the set of feasible regions constructed according to the measurements of the d_t^{th} neighbor. Let $F_t = (F_{t1}, F_{t2}, \ldots, F_{tw})$ and $F'_t = (F'_{t1}, F'_{t2}, \ldots, F'_{ty})$. The new feasible region after the update becomes:

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

Each $F_{ti} \in F_t$ is intersected with all regions of F'_t. Union of these partial intersections gives us the resulting polygons. After the intersection operation we need to assign new probabilities to feasible regions created as a result of an intersection. Consider only a single element of F_t and F'_t. Let $F_{ti} \in F_t$ and $F'_{tj} \in F'_t$ be two feasible regions. Let p_t be a point in space that corresponds to the real location of node t.

$$Pr(p_t \in F_{ti} \cap F'_{tj}) = Pr(p_t \in F_{ti} | p_t \in F'_{tj}) \times Pr(p_t \in F'_{tj})$$

If we have the *a priori* information that $u \in F'_{tj}$, the probability of u being in F_{ti} does not change, since the two measurements are independent. However the probability of u being in $F_{ti} - F'_{tj}$ is zero. In other words, the area where p_t may be located in F_{ti} changes, but the probability does not and we have $c(F_{ti} \cap F'_{tj}) = c(F_{ti}) \times c(F'_{tj})$.

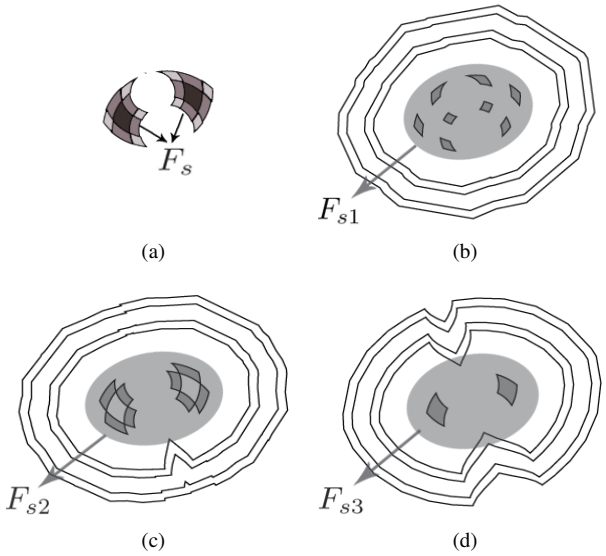

Fig. 3. (a)F_s (b,c,d) Feasible regions constructed from F_{s1}, F_{s2}, F_{s3}

C. Post-Processing

As we approximate the confidence intervals using polygonal regions some degeneracies may arise after the construction or the update operations. Note that each feasible region is assigned a unique confidence. Thus a degeneracy occurs if two or more feasible regions overlap at a subregion in the plane. This is undesirable as it would lead to an output set of points with multiple confidence values. A second type of a degeneracy occurs if more than one feasible region is assigned the same confidence value. Note that by definition each feasible region is assigned a unique confidence. We provide a post-processing algorithm not only to handle these two degenerate cases but also to limit the growth of the output number of feasible regions which is exponential in the worst case. The post-processing procedure, shown in Algorithm 1, is applied after each of the construction and update operations.

a) (Type-1 degeneracy) Check overlaps: We check every pair of feasible regions F_{ti}, F_{tj}. If the area of the intersection is between $r_1\%$ and $r_2\%$ of the smaller region, then we create a new feasible region $F_{ti} \cap F_{tj}$ with confidence equal to the average of $c(F_{ti})$ and $c(F_{tj})$, and the new region is excluded from the parent feasible regions.

b) (Type-2 degeneracy) Preserve uniqueness: For each confidence value c associated with an existing feasible region, we go through the list of feasible regions and create a list L of regions with confidence c. We create a new feasible region equal to $\bigcup_{\forall F_{ti} \in L}(F_{ti})$ with confidence $c \times |L|$. We then remove every $F_{ti} \in L$ from the set of feasible regions.

c) Limit growth: We first discard regions with low probability by removing each F_{ti} with $c(F_{ti}) < \epsilon$ for some predefined ϵ close to zero. Next we apply an *approximate reduction*. For each pair of feasible regions F_{ti}, F_{tj}, if $|c(F_{ti}) - c(F_{tj})| <$

Algorithm 1 Post-processing of feasible regions

```
1: procedure POST-PROCESS(F_t, D, α)
2:     HANDLE DEGENERACIES(r_1, r_2) //if any
3:     //Limit Growth
4:     DISCARD REGIONS WITH LOW PROBABILITY(F_t, ε)
5:     while |F_t| > D do
6:         HANDLE DEGENERACIES(r_1, r_2) //if any
7:         APPROXIMATE REDUCTION(F_t, α)
8:         α ← α − ε
```

α, where α is predefined, we construct a new region $F_{ti} \cup F_{tj}$ with confidence equal to $c(F_{ti}) + c(F_{tj})$. We continue the reduction in iterations each time with a lower α value until D, the desired number of feasible regions, is reached.

D. Complexity Analysis

All geometric objects, including the circles, used in the framework are represented as polygons. Polygons may contain holes. For the construction operation both the expansion and the difference operations can be done in time linear in the size of the input which is determined by the number of vertices of all polygons in the input feasible region F_s. Let m be the maximum number of polygons in a feasible region and k be the maximum number of vertices in a single polygon. Then the running time of a construction applied to F_s is $O(mkn)$, where n is the number of feasible regions in F_s. We limit n to a constant within our framework and k is a predefined constant. This implies that a single construction requires time $O(m)$. The number of committed construction operations is $O(|V|)$ as for each node $u \in V$ in the network, only one incident edge is processed via the construction and the rest of the edges incident on u are handled via the updates. Therefore all constructions throughout the network require $O(m \times |V|)$ total time. We note that if not bounded by the post-processing operations and the succeeding updates the size of the input feasible region may grow exponentially. Since each construction creates $3n$ feasible regions where its ancestor has n feasible regions, the number of feasible regions may grow to be of order $3^{|V|}$. As for the updates, each update operation decreases the total area covered by a feasible region, but the number of *subregions*, i.e. polygons with different confidence values, may increase. In order to reduce the growth we use the approaches proposed in the previous subsection. Each update entails an intersection between two arbitrary polygons, possibly with holes. Given two polygons with k vertices each, the running time of the intersection is $O(k^2 + k \times k')$, where k' is the number of intersection points between the two polygons [26]. Since we assume k and k' are constant the running time of all intersections required by an update is $O(m^2)$. The same holds for the union operations committed within an update. Finally the number of updates is $O(|E|)$. Therefore the total time required by all updates is $O(m^2 \times |E|^2)$. We note that the same bound applies to the post-processing algorithm as well.

Our framework uses three confidence intervals of the normal random variable. One may increase the number of intervals used in the algorithm to provide more accuracy. However after three confidence intervals, adding more intervals becomes infeasible as each extra interval increases the complexity of the algorithm while contributing very little to the accuracy. For instance using 5 confidence intervals rather than 3, increases the accuracy only by 4.2% for a normal distribution.

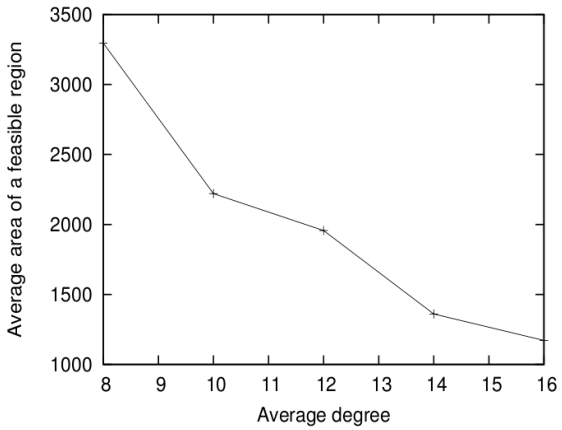

Fig. 4. Average area assigned when average degree changes from 8 to 16

IV. Experimental Evaluation

A. A Sample Localization Algorithm

In order to evaluate the proposed localization framework to handle noisy measurements, we describe a sample localization algorithm that embeds our proposed framework. The algorithm employs the centralized model of computation and is similar in essence to the order-based algorithms of [15], [27]. We note that since our framework does not impose any limitations beyond the measurement model and its representation, any other localization technique such as distributed algorithms similar to that of [10] could also be tailored to work with our framework. In an order-based localization algorithm an ordering $\pi = u_1, u_2, \ldots u_n$ on the nodes in V is assumed. Each $u_i \in V$ has at least t neighbors that come before it in π. For a trilateration order $t = 3$ [27], whereas for bilaterations $t = 2$ [15]. Each node is taken in this order and localized based on the localizations of its preceding neighbors in π. We note that for a trilateration ordering if no noise in measurements is assumed then a unique localization of every node in the network is guaranteed with one traversal. Our localization algorithm is similar. As we go through the nodes in order, we take the next node u, apply a construction on u operation with reference to its first left neighbor, and apply updates referencing each of the rest of its preceding neighbors. After each construction and update we apply the post-processing procedure. For a single traversal of the complete network the complexity analysis provided previously applies directly. Once the traversal of π finishes, we start a *reverse traversal* where we traverse the inverse of π applying the same procedures. These back-and-forth traversals continue until no further improvements on the feasible regions are possible.

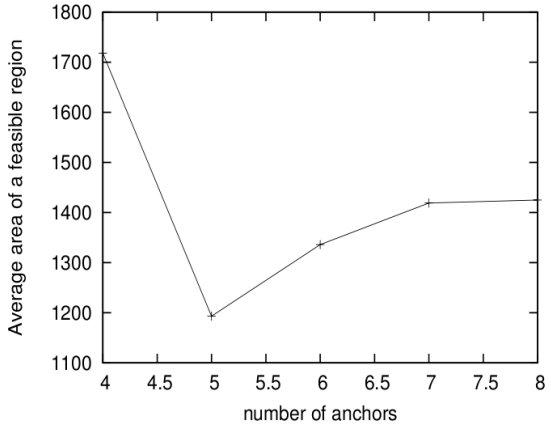

Fig. 5. Average area assigned when a changes from 4 to 8

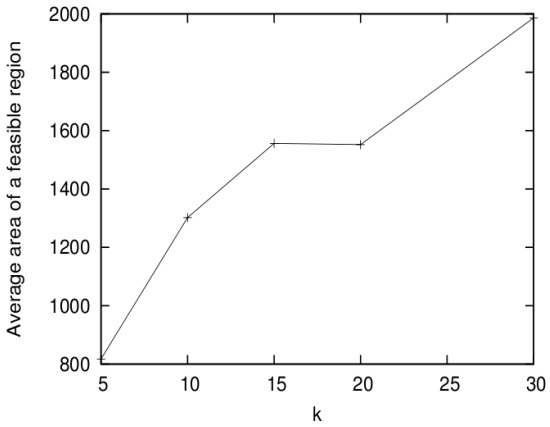

Fig. 6. Average area assigned when k changes from 5 to 30

B. Experimental Results

We implemented the proposed robust localization framework to handle noisy measurements. We also implemented the described localization algorithm that embeds our framework. The implementations are coded in C++, using the LEDA library [28]. All the implementations and the experiments are freely available at http://hacivat.khas.edu.tr/~cesim/handleNoise.rar. We inspect the affects of several parameters within the proposed localization framework. One parameter is k, the number of vertices on the $k-gon$ that is used to approximate a circle. Second parameter is the variance and the 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. For our experiments we use randomly generated graphs with varying average degrees. Thus average degree is another parameter of our experimentation. Finally, the number of anchor nodes, a, in a given network, directly affects the outcome of the localization process. We use a number of performance measures: The number of nodes that are localized, i.e. a feasible region assigned to that node, the number of nodes whose real locations are found to be in their feasible regions, and the average area of the feasible regions.

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. In the first of our experiments plotted in Figure 4, we generated random graphs with varying average degrees. The parameters other than average degree are fixed to $k = 20$, $\sigma = 20$, $a = 4$. In almost all instances the real locations of the nodes are found to be in their respective feasible regions assigned by the algorithm. Figure 4 shows that 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 4 that the algorithm failed to localize a node or two. In the second experiment plotted in Figure 5 we 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. There are again 12 instances in all 50 runs plotted in Figure 5 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 decreases the average area of feasible regions.

In the third experiment plotted in Figure 6 we fix a to 4 and the average degree to 7, depending on the results of first two experiments. We 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 helps localization cover the circle with more accuracy and increase the area covered in a circle. The next experiment plotted in Figure 7 is designed to evaluate how σ effects the outcome of localization, while the other parameters are fixed, $a = 4, k = 20$, average degree=7. As σ increases the area of the feasible regions increases, since an increase in σ implies more inaccuracy in distance measurements. The running time plot in Figure 8 for varying k values is given for a network of 20 nodes where $a = 4$, average degree = 7, and $\sigma = 15$. A single point in this plot is the average time in 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 preferable to use smaller k values with simple sensor configurations.

V. Conclusion

We provide a robust localization framework to handle noisy measurements in wireless sensor networks. The provided framework is general in the sense that it is independent of the employed localization algorithm and can easily be tailored to work with many different types of algorithms be it centralized or distributed. An important direction for future work is to embed the proposed framework within a distributed paradigm of localization and evaluate the results.

Acknowledgment

The authors would like to thank Alon Efrat for fruitful discussions and valuable comments.

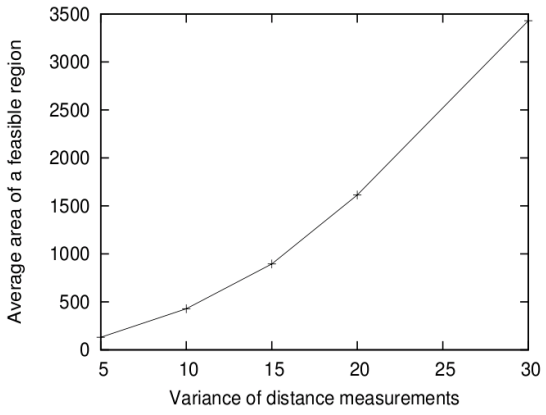

Fig. 7. Average area assigned when variance changes from 5 to 30

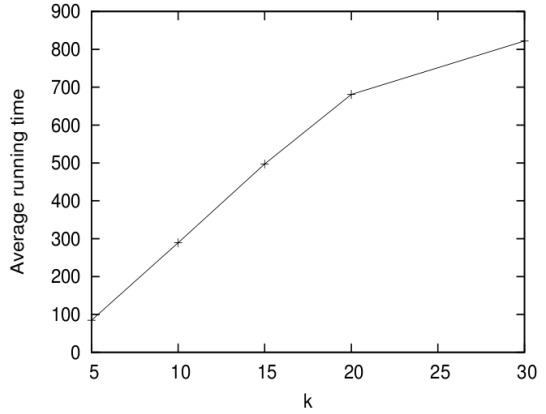

Fig. 8. Average running times when k changes from 5 to 30

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[2] K. Langendoen and N. Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison," *Comput. Netw.*, vol. 43, no. 4, pp. 499–518, 2003.

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

[4] G. Mao, B. Fidan, and B. D. O. Anderson, "Wireless sensor network localization techniques," *Comput. Netw.*, vol. 51, no. 10, pp. 2529–2553, 2007.

[5] C.-H. Wu, W. Sheng, and Y. Zhang, "Mobile sensor networks self localization based on multi-dimensional scaling," *Robotics and Automation, 2007 IEEE International Conference on*, pp. 4038–4043, April 2007.

[6] P. Biswas and Y. Ye, "Semidefinite programming for ad hoc wireless sensor network localization," in *IPSN '04: Proceedings of the 3rd international symposium on Information processing in sensor networks*, 2004, pp. 46–54.

[7] 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*. New York, NY, USA: ACM, 2004, pp. 45–57.

[8] 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*. New York, NY, USA: ACM, 2004, pp. 50–61.

[9] 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.

[10] 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*. New York, NY, USA: ACM, 2006, pp. 262–273.

[11] D. Niculescu and B. Nath, "Ad hoc positioning system (aps) using aoa," in *In INFOCOM03*, 2003.

[12] P. Bahl, V. N. Padmanabhan, and A. Balachandran, "Enhancements to the radar user location and tracking system," Tech. Rep., 2000.

[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*. New York, NY, USA: ACM, 2003, pp. 201–212.

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

[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*. New York, NY, USA: ACM, 2006, pp. 110–121.

[16] S. 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*, vol. 2, pp. ii–361–4 vol.2, May 2004.

[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*, pp. 2035–2040, Oct. 2006.

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

[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)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 557–563.

[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*. New York, NY, USA: ACM, 2003, pp. 15–30.

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

[22] P. Rong and M. 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*, vol. 1, pp. 374–382, Sept. 2006.

[23] 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*. Piscataway, NJ, USA: IEEE Press, 2005, p. 10.

[24] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," 2000, pp. 32–43.

[25] A. I. Mekhannikov, "Algorithm for the confidence interval of a combined measurement error," *Measurement Techniques*, vol. 46, pp. 635–638, 2003.

[26] Y. Peng, J. Yong, W. Dong, H. Zhang, and J. Sun, "A new algorithm for boolean operations on general polygons," *Computers and Graphics*, vol. 29, pp. 57–70, 2005.

[27] 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*, 2004, pp. 2673–2684.

[28] K. Mehlhorn and S. Nher, "Leda: A platform for combinatorial and geometric computing," 2000.