Location-Free Object Tracking on Graph Structures

Daniela Krüger, Carsten Buschmann, and Stefan Fischer

Institute of Telematics, University of Lübeck
Ratzeburger Allee 160, 23538 Lübeck, Germany
{krueger,buschmann,fischer}@itm.uni-luebeck.de
http://www.itm.uni-luebeck.de

Abstract. Using wireless sensor networks for object tracking requires ordering events with regard to time and location. In labyrinth-shaped topologies, one-dimensional ordering suffices within the different sections of the network. We present an algorithm that decomposes the network into such sections, tracks objects within using binary sensors and, if required, hands them over to the next section. We evaluate our approach through extensive simulations and show that it is robust against sensor failures and packet loss.

Keywords: In-building Object Tracking, Wireless Sensor Networks.

1 Introduction

Object tracking is a common application domain for wireless sensor networks. Most authors assume that the sensor nodes are spread out over a two dimensional, convex area. In such a setting, the typical approach of using localization algorithms to make all devices location-aware works relatively well. However, it was shown in [1] that such algorithms fail for settings where the network is non-convex, contains a large number of wholes, et cetera. A typical application where such non-convex networks occur is indoor object tracking. If the nodes are deployed in building corridors, the wireless sensor network consists of a topological structure resembling roads.

In this paper we consider the problem of tracking objects in such scenarios using binary sensors, i.e. sensors that can detect whether an object is present or not. We pick up the idea proposed in [1,2] to interpret the network structure as a graph. Corridor junctions and dead-ends are considered as graph vertices, whereas the corridor segments in-between (also called corridors) are considered as graph edges (cf. Figure 1). After network deployment, nodes determine whether they reside in a vertex or an edge area of the network. If they are part of an edge, i.e. reside in a corridor segment, they compute the one-dimensional ordering of their neighbors that is then used for object tracking.

Tracking is done on a 'per corridor' basis: Nodes within one corridor track the object until it reaches a junction, then tracking is handed over to nodes in the next corridor.

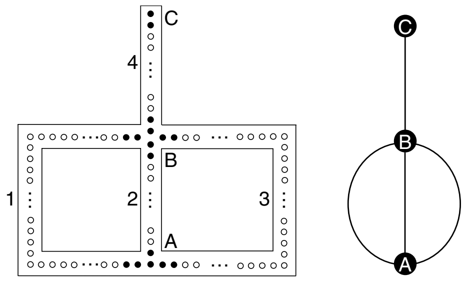


Fig. 1. Scenario

The remainder of this paper is structured as follows. In the next section we discuss related work. In Section 3 we describe our approach for one-dimensional object tracking comprising start-up and operating phase. The two phases are evaluated in Section 4, also considering the influence of packet loss as well as false and missed sensor activations. We conclude the paper in Section 5 with a summary and directions for future work.

2 Related Work

Object tracking has often been investigated for military purposes or basic surveillance applications. In contrast to our work, various approaches assume sensors that generate signals dependent on the distance from a tracked object [3] or use geographical information [4]. However, we assume that devices are equipped with a sensor providing only binary signals (like passive infrared sensors) and consider an indoor scenario without location information. Therefore, we do not review these methods further.

Other approaches that deal with binary sensors [5,6,7] consider only a single target, assume rather dense networks [6,8], or reliable sensor readings [5]. All these approaches aim at finding an object's trajectory within a two-dimensional area using particle filters. Singh et. al. propose a one-dimensional tracking algorithm called *ClusterTrack* [9]. They also track multiple targets by applying particle filters. They focus on the effectivity of their scheme rather than on the used communication protocols and hence, propose a non-distributed scheme that assumes all sensor readings being available at a central processing unit which estimates the target's trajectories. As a result, the proposed scheme is not applicable to sensor networks without adaptation.

The authors of [10] present the restriction of particles to a graph and try to infer typical motion patterns of objects within a building. However, they require that objects can be identified (which limits their approach to a single object).

A different approach dealing with graphs is described in [11]. The authors formulate the tracking problem as a hidden state estimation problem in a Markov model and then derive their tracking scheme from the Viterbi algorithm [12].

Unfortunately, they assume perfect knowledge about the number of targets and their identities, which is not available in our scenario.

The idea of decomposing networks with holes into sections or clusters and representing them as a graph was first proposed in [1]. The authors then present a method for achieving that kind of topology recognition in [2]. It is particularly targeted at border and junction detection, but is designed for extremely high network densities. Hence, the approach is not directly applicable to the scenario considered here.

3 Decentralized Object Tracking

In this section we present our location-free and robust in-building object tracking scheme for wireless sensor networks. It can be divided into the start-up phase that is executed after network deployment and the subsequent operation phase.

The basic idea of our scheme is that each node pre-computes the one-dimensional spatial ordering of the nodes within its communication range during the start-up phase. These orderings are then regarded as the expected sensor activation patterns of neighbor nodes when an object passes by.

During operation, event sequences can be compared to these patterns. In addition, the event sequences are also stored in a short history. Like this, new patterns that result from network changes due to node failures or addition of nodes can be learned over time.

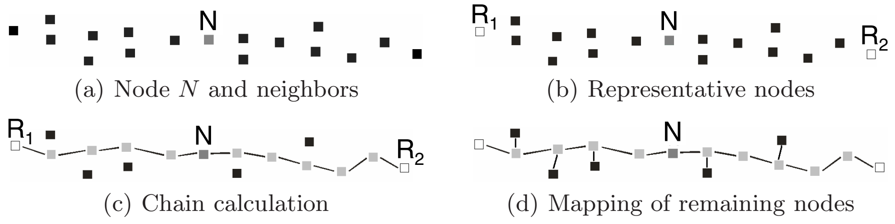
Our object tracking algorithm requires a loose time synchronization (approximately half a second as a maximum of time difference, depending on the maximum object velocity) to ensure that sensor activation events can be ordered correctly in time. Other protocols however, often impose more rigid time synchronization requirements such as duty-cycling or slotted medium access. Furthermore, we explicitly do not require that devices are location-aware. Instead, we use the distance estimation scheme described in [13]. It is based on neighborhood list comparisons instead of relying on Received Signal Strength Indication (RSSI) or Time Difference of Arrival (TDoA) measurements that are error-prone especially in indoor settings.

3.1 Start-Up Phase

The start-up phase is executed directly after network deployment. It can be repeated after times of high fluctuation or network restructuring, but this is usually not required as the algorithm works adaptively during the operating phase. During setup, the devices

1. calculate the local topology, i.e. the one-dimensional ordering of neighbor groups (usually one on each side),
2. find out whether they reside at corridor junctions or dead ends, and
3. assign corridor section memberships to their neighbors.

In the following, we discuss the start-up phase from the perspective of a node N (marked as a dark gray square in Figure 2(a)). At first, N detects the *one-dimensional ordering* of its neighbors. Details on this process are provided in [14], so we give only a short overview here. N broadcasts a 'hello' message to advertise its presence to its neighbors, and in turn collects its neighbors' messages to build up a neighborhood list.



(a) Node N and neighbors (b) Representative nodes (c) Chain calculation (d) Mapping of remaining nodes

Fig. 2. Local topology recognition

The resulting list is broadcasted again. When receiving such a list, it is compared with the local list of neighbors. A distance estimate can be derived from the fraction of common neighbors. Please refer to [13] for a detailed description of the distance estimation process. As a result, N can augment its local list with information on the distance to its neighbors.

The augmented list broadcasted again, resulting in all nodes knowing the distances of their neighbors from each other. N now sorts its neighbors by ascending distance and selects the furthest 40% of them. It then divides this subset into groups, so that neighbors that are close to each other are in same group whereas distant neighbors are in different groups. Following this, the most distant node from N in each group is chosen to be a group representative (white squares R_1 and R_2 in Figure 2(b)). Finally, so called 'chains', i.e. orderings of neighbors between the representatives and N are computed based on distance information between R_i and N (cf. Figure 2(b)). Nodes that are not part of the chains are mapped onto the nearest chain member (cf. Figure 2(d)).

Based on the number of chains, nodes can *decide whether they reside at junctions or dead ends*, i.e. whether they belong to the graph's vertices. In a perfect world most nodes would have ended up with two chains because they are situated somewhere in a corridor, 'vertex nodes' near junctions or dead ends would have calculated one, three or four chains. However, influences like distance estimation errors can lead to errors here. To increase robustness, a local voting process is employed: if a node's chain number does not equal two, it broadcasts it to its neighbors. Hence, nodes learn the chain counts of their neighbors. Nodes only consider themselves as vertex nodes if at least one third of their neighbors are also vertex nodes because they calculated the same number of chains. Again for each vertex node set, a representative is elected to find the optimally located node. Nodes then calculate the average distance to the other vertex nodes.

The one with the smallest average can be expected to reside in the center of the cluster, and hence becomes its representative.

Finally, all nodes determine their neighbors' *corridor membership*. By default, all neighbors are assumed to reside in the same corridor as oneself. However, vertex representatives broadcast their chains (including the mapped nodes) to their neighbors. If a node receives such a message, it checks which chain it belongs to, and marks all its neighbors that are not in the same chain to reside in a different corridor.

3.2 Operating Phase

After start-up, the network enters its regular operation mode, that is presented in this section. Remember that the core idea is to compare sensor activation patterns with the pre-computed chains, but that nodes should be enabled to adapt to new patterns over time. Therefore the chains constitute the foundation stone for a *history* of patterns that is amended during operation.

Whenever a node's sensor activates, it broadcasts a so-called *ObjectMessage* containing

- node ID: the ID of the activated node,
- activation time,
- starting location: the ID of the representative of last passed corridor vertex,
- starting time: the time when the object entered the current corridor section (Note that starting location and time identify an object uniquely because we assume that no two objects can activate the same sensor at exactly the same time),
- activation count: the number of activations since the object entered the current corridor section,
- velocity: average inter-activation time of the object.

Nodes store incoming activation messages in their *message buffer*. Messages in the buffer are sorted by the object identification, messages concerning the same object by the activation time. The message buffer has a fixed size where newer messages replace older ones. The buffer size should ensure that messages can remain in it for at least the time it takes at most for an object to cross the sensors' communication range.

Let us assume for example that node N_{15} received 4 object messages (two from N_{12} and one from N_{13} and N_{14}). Than, its message buffer could contain two objects:

$O_{id} = (S_{Loc}),\ S_{Time}$	List of neighbors' activations
$(1, 5.5)$	$(12, 7.6)$ $(13, 9.1)$ $(14, 10.8)$
$(1, 10.8)$	$(12, 11.6)$

When a node's sensor is activated, it tries to find a sequence of activation messages in its buffer that can be correlated with a history entry. For each message sets i.e. messages that concern the same object, it constructs the according activation pattern by extracting the node ID from each message and appends itself at the end. So, if N_{15}'s sensor was activated at 12.2 in our example, there would be two resulting patterns: $12, 13, 14$ and 12.

The node then compares all these patterns to all history entries. The pattern that matches best is removed from the message buffer to conserve memory space. If exactly the same pattern already exists in the history, an associated match counter is increased; otherwise the pattern is inserted into the history. If no matching entry was found at all, a new one is created. Finally, an object message is constructed and broadcasted to the node's neighbors.

In the example let the (initial) history of N_{15} be

Match Counter	Last matching time	Pattern
0	0.0	12 13 14 15
0	0.0	18 17 16 15

Then, N_{15} finds the best matching by correlating the first message $((1, 5.5))$ with the first history entry. Hence, it increments the respective history entry counter, sets the last matching time to its activation time (12.2) and broadcasts its correlation result $((1, 5.5), (15, 12.2))$.

The core comparison mechanism is derived from the so-called *local alignment* [15] that was intended for finding similar sequences in DNA strings. It can tolerate entry permutations or missing entries to a certain degree, and delivers a degree of similarity. We adapted it slightly to our problem.

$$\text{Let } s : \mathbb{N}^n \times \mathbb{N}^m \to \mathbb{N}^{n \times m}, s_{i,j} = \begin{cases} 0 : a_i \neq b_j \\ 2 : a_i = b_j \wedge \|i - j\| > 2 \\ 4 : \mathit{otherwise} \end{cases}$$

be the equality matrix of two sequences $a = \{a_1, ..., a_n\}$ and $b = \{b_1, ..., b_m\}$, let d (with $d < 0$) be the penalty measure for a deletion and let F be the matrix for the computation of the local alignment with

$$F(0,0) = 0$$
$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_i) \\ F(i-1, j) + d \\ F(i, j-1) + d \\ 0 \end{cases} .$$

The matrix F is filled line-by-line, its maximum value $\max\limits_{0 \leqslant i,j \leqslant m,n} F(i,j)$ indicates the degree of similarity, where higher values represent more similar patterns. As we do not consider dense networks, the quadratic time complexity should not be a problem. In particular is the size of the patterns limited to half of the number of neighbors. To avoid the quadratic space complexity the implementation uses only two vectors of size n for the computation. This is possible as it suffices to store the previous raw and the maximum value.

In addition to the local alignment, the comparison includes a number of sanity checks that can degrade the matching. They include the time since the last activation (derived from the message buffer), whether the velocity and last activation time fit the current activation and whether previous nodes reside in the same corridor. To adapt to new nodes or new orders of neighbors unknown patterns are added to the history and old history entries (depending on the last matching time) are removed from the history.

4 Evaluation

To evaluate our approach, we ran an extensive set of simulations with the scenario shown in Figure 1. The simulation area covers 50 by 50 meters. For all simulations 140 nodes equipped with a sensor monitoring its corridor sector over a length of 1 meter were placed on the wall of all corridors about every 1.5 meters.

The scenario includes four corridor segments, two junctions and one dead-end, allowing to evaluate a large number of different settings. In addition, two of the corridors feature two right angle corners. These areas are particularly difficult, as the distance estimation scheme tends to yield erroneous results here. Apart from wrong orderings, this may lead to false vertex detections.

We decided not to simulate the whole communication stack because we wanted to lay the focus of our work on effects that directly influence the object tracking rather then on networking aspects. For this reason we chose SHAWN [16] for our simulations. It models the effects of a MAC layer (e.g., packet loss, corruption and delay), but does not perform a complete simulation of the stack. It can simulate message exchange and measurements very efficiently and was developed for algorithmic simulations.

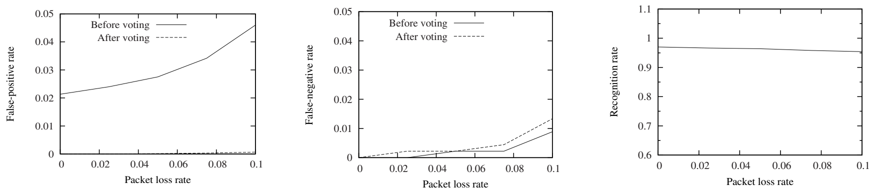
Instead of using the rather unrealistic unit disk graph radio propagation model, we employed the Radio Irregularity Model (RIM) [17]. It was developed to model radio properties of wireless sensor nodes and is a well established nowadays. Rather than using a fixed radio range, the maximum communication distance of a node also depends on the angle between sender and receiver. This in particular leads to a certain percentage of unidirectional links.

To achieve a reasonable network density, the average communication range was set to 9 meters. As a result, nodes on average had 6 neighbors on each side. While nodes within the corridor segments thus had about 12 neighbors, nodes had up to 20 neighbors at junctions and down to 6 neighbors in the dead-end. We consider this network density as a reasonable choice as wireless networks of smaller densities tend to partition [18,19].

We ran the object tracking scheme and analyzed the results of both start-up and operating phase. To obtain statistically sound results, we averaged the results of 100 simulations using the same parameter set but different random seeds.

4.1 Topology Recognition Evaluation

First, we evaluated the recognition of vertex nodes at junctions and dead-ends. For the subsequent object tracking, the absence of false vertex nodes within the



(a) False vertex recognitions (b) Missed vertex recognitions (c) Corridor membership detection

Fig. 3. Quality of start-up phase

corridors is of the same importance as the presence of vertex nodes in junction areas.

Figure 3(a) shows the effectiveness of the voting process against false positives, i.e. nodes that reside in corridor segments but determined themselves as vertex nodes. While the pre-voting error rate rises from 2% to about 4.5% with increasing packet loss rate, the post-voting error rate is zero and literally does not increase at all.

As the voting successfully prevents false positives by reducing the number of vertex nodes, it must be ensured that it does not inhibit vertex detection at junctions.Figure 3(b) depicts the fraction of nodes that spuriously did not consider themselves as vertex nodes over different packet loss rates. While more or less all vertices are recognized for less than 7% packet loss, the rate of false negatives slightly rises for higher loss rates. However, it is also visible that the voting process has hardly any negative influence here. Nevertheless, it might be beneficial to resend messages to guarantee a good cluster recognition if extremely high packet loss rates are expected.

Finally, we evaluated the recognition of the corridor membership. Figure 3(c) indicates the fraction of neighbors that were assigned to the correct corridor section. It is obvious that only a small fraction of about 3% of the neighbors is miscategorized regardless of the packet loss rate. These errors are negligible for the object tracking scheme and can hence be tolerated.

4.2 Object Tracking Evaluation

We then evaluated the second part of our scheme, the operating phase.

We applied our scheme to various object movement patterns with different numbers of objects moving at different velocities. The patterns comprise objects passing by, following each other as well as overtaking each other.

In this section we present results for five selected movement patterns:

1. *1after2*: one object moves from C to A through corridors 4 and 1, then a second obejct takes the same way, and finally a third object moves the opposite way afterwards.
2. *Succeeding*: Two objects move from A to B through 1 and 4 following each other with a distance of 2 meters.

3. *Overtaking*: Two objects move from A to C through 1 and 4, one overtaking the other.
4. *1towards4*: Four objects move from A to C through 1 and 4 following each other with a distance of 5 meters, while at the same time a fifth object moves from C to A through the same corridor segments.
5. *3Ways*: Three objects move from A to C, one object through corridor 1, one through 2 and one through 3,at speeds that ensure their simultaneous arrival.

Influence of Activation Time Variation and Packet Loss. It must be assumed that not all sensors are adjusted perfectly with regard to orientation and position. To account for such effects, we added a random error to the activation times to obtain realistic simulations. Additionally, this compensates for the unnaturally perfect deployment of nodes in the simulations which may not be possible in real buildings.

In our simulation model the perfect activation occurs when the object reaches the point closest to the sensor (given that the object intrudes the detection range at all). We added a random offset to that point in time.

Precipitated and delayed activations cause a certain fraction of swapped activations which aggravates the object correlation significantly. In addition, the estimated velocity becomes extremely error-prone, impeding the differentiation between overtaking and closely succeeding objects.

We obtained the random error values from a Gaussian distribution with a standard deviation of $\sigma = 0.45$. Apart from the false activation time itself, this causes 1.7% of the activations to swap. Figure 4 and Figure 5 show the results of these simulations.

First of all one can see in Figure 4 that all objects are correctly tracked if neither the activation times are error-afflicted nor packets are lost. Correcty tracked means that for each corridor segment the object's start and end locations and times have been correctly recognized by a one vertex node. We then increased the packet loss rate up to 10%. The rate of correctly tracked objects decreases slowly for the more difficult movement patterns like *1towards4* and *3Ways*. Given a packet loss rate of 10% here, about 95% of the objects are correctly tracked. The moving pattern *3Ways* is difficult because of the objects simultaneously leaving from and arriving at a junction. If two objects arrive at a junction at the same time and a node's sensor detects the first one, the node must decide which previous activations in the neighborhood correlate better, but as both objects could continue their way, both objects are contemplable to be the reason for this activation.

As depicted in Figure 5 the results with the time-varied activations are nearly as good as the results without error. This shows the robustness of our tracking scheme.

Influence of Missed Activations. As hardware errors may lead to the oversight of moving objects by a sensor, we added activation miss rates of up to 10%. Figure 6 shows the result tracking rates without and Figure 7 shows result

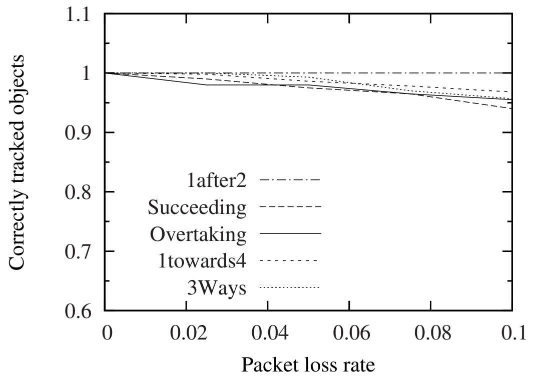


Fig. 4. Standard deviation $\sigma = 0$

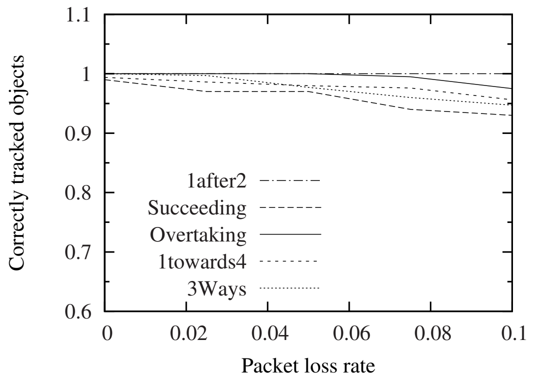


Fig. 5. Standard deviation $\sigma = 0.45$

tracking rates with the sensor activation time variance from the last section. For these simulations, the packet loss rate was set to 0.

For activation failure rates of up to 0.05 the results closely resemble those without sensor failure but with a corresponding packet loss rate. This is hardly surprising because for a node's neighbors it does not make any difference whether it failed to activate or whether the resulting packet was lost; it only make a difference for the directly affected node. If there are e.g., two objects closely succeeding each other and the sensor activates caused by the second object, having missed the first object may result in the wrong correlation decision which is not the case for packet loss. This difference is responsible for the slightly lower tracking rates at high failure rates compared to the results of the previous section.

However, experiments with real passive infrared sensors show that these tend to overreact rather than to miss events. Hence, we also evaluated influence of false positive activations in the following section.

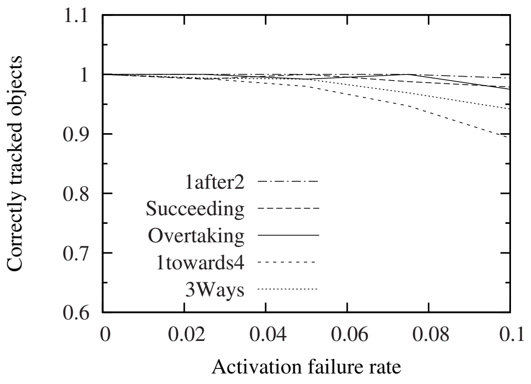


Fig. 6. Standard deviation $\sigma = 0$

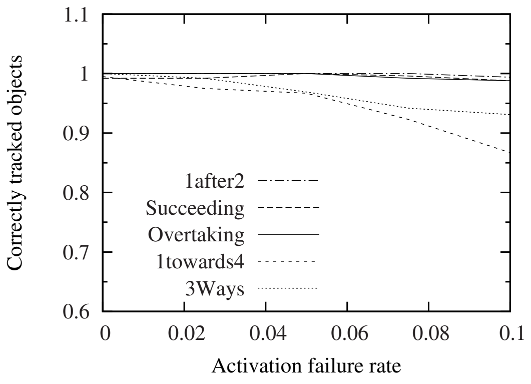
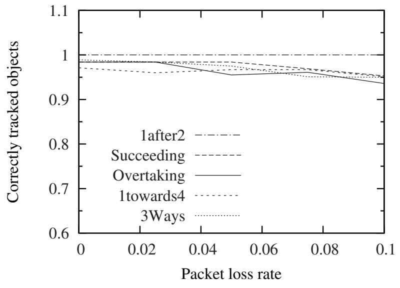


Fig. 7. Standard deviation $\sigma = 0.45$

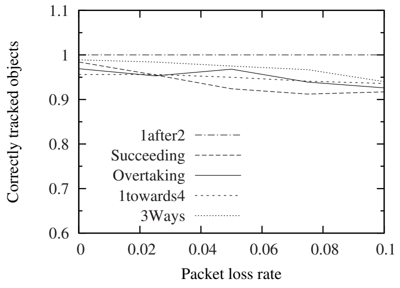
Influence of False Positive Activations. Finally, we investigated the influence of additional false activations caused by hardware errors, temperature variation and air draft. We considered two cases where we generated 29 respectively 58 random additional activations per node and day. These rates correspond to about 15% and 30% of false activations given each node activates one time during each simulation.

Figure 8 shows the resulting rates of correctly tracked objects for different activation time variances and packet loss rates. It becomes clear that additional activations hardly influence the object tracking scheme: While the error increases by about 2% for low packet loss rates, the ghost activation influence even decreases for higher packet loss rates. If every tenth packet is lost, the false positive activations lose their influence completely.

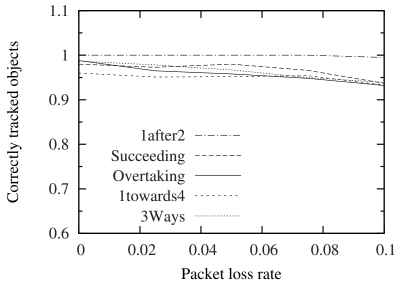
However, it can be noted that different movement patterns suffer to a varying extent. Obviously, ghost activations have most influence when multiple objects are close to each other: The additional activations make it more difficult to assign sensor activations to the correct objects.



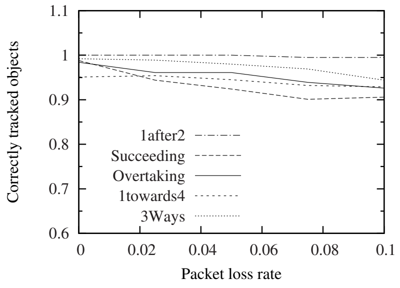
(a) 29 error activations, $\sigma = 0$



(b) 29 error activations, $\sigma = 0.45$



(c) 58 error activations, $\sigma = 0$



(d) 58 error activations, $\sigma = 0.45$

Fig. 8. Detection rate over packet loss with additional false activations

5 Conclusion

In this paper we presented a new approach for location-free object tracking in complex network topologies. Its core idea is to decompose the network into different areas where the tracking problem can be reduced to one dimension. Consequently, it becomes possible to do object tracking without location-aware nodes that are required by most other schemes.

By running an extensive set of simulations we showed that our scheme can not only handle a wide range of complex movement patterns, but is also robust against packet loss and sensor failures.

As future work, we plan to implement the algorithm on real hardware to enable further evaluation under real-life conditions.

References

1. Buschmann, C., Fekete, S.P., Fischer, S., Kröller, A., Pfisterer, D.: Koordinatenfreies Lokationsbewusstsein (Localization without Coordinates). IT - Information Technology, Themenheft Sensornetze 47(4) (April 2005)
2. Kröller, A., Fekete, S.P., Pfisterer, D., Fischer, S.: Deterministic boundary recognition and topology extraction for large sensor networks. In: Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006) (2006)

3. Chellappa, R., Qian, G., Zheng, Q.: Vehicle detection and tracking using acoustic and video sensors. In: 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing, May 17-21, 2004, vol. 3, pp. 793–796. IEEE, Los Alamitos (2004)
4. Pahalawatta, P.V., Depalov, D., Pappas, T.N., Katsaggelos, A.K.: Detection, classification, and collaborative tracking of multiple targets using video sensors. In: Zhao, F., Guibas, L.J. (eds.) IPSN 2003. LNCS, vol. 2634, pp. 529–544. Springer, Heidelberg (2003)
5. Aslam, J.A., Butler, Z.J., Constantin, F., Crespi, V., Cybenko, G., Rus, D.: Tracking a moving object with a binary sensor network. In: Akyildiz, I.F., Estrin, D., Culler, D.E., Srivastava, M.B. (eds.) SenSys., pp. 150–161. ACM, New York (2003)
6. Kim, W., Mechitov, K., Choi, J.Y., Ham, S.K.: On target tracking with binary proximity sensors. In: IPSN, pp. 301–308. IEEE, Los Alamitos (2005)
7. Shrivastava, N., Mudumbai, R., Madhow, U., Suri, S.: Target tracking with binary proximity sensors: fundamental limits, minimal descriptions, and algorithms. In: Campbell, A.T., Bonnet, P., Heidemann, J.S. (eds.) SenSys, pp. 251–264. ACM, New York (2006)
8. Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., Mittal, V., Cao, H., Demirbas, M., Gouda, M.G., ri Choi, Y., Herman, T., Kulkarni, S.S., Arumugam, U., Nesterenko, M., Vora, A., Miyashita, M.: A line in the sand: a wireless sensor network for target detection, classification, and tracking. Computer Networks 46(5), 605–634 (2004)
9. Singh, J., Madhow, U., Kumar, R., Suri, S., Cagley, R.: Tracking multiple targets using binary proximity sensors. In: Abdelzaher, T.F., Guibas, L.J., Welsh, M. (eds.) IPSN, pp. 529–538. ACM, New York (2007)
10. Liao, L., Fox, D., Hightower, J., Kautz, H., Schulz, D.: Voronoi tracking: Location estimation using sparse and noisy sensor data (2003)
11. Oh, S., Sastry, S.: Tracking on a graph. In: IPSN, pp. 195–202. IEEE, Los Alamitos (2005)
12. Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory 13(2), 260–296 (1967)
13. Buschmann, C., Hellbrück, H., Fischer, S., Kröller, A., Fekete, S.: Radio propagation-aware distance estimation based on neighborhood comparison. In: Proceedings of the 14th European conference on Wireless Sensor Networks (EWSN 2007), Delft, The Netherlands (January 2007)
14. Blinded
15. Setubal, J., Meidanis, J.: Introduction to computational molecular biology. PWS Publishing Company, Setubal (1997)
16. Kröller, A., Pfisterer, D., Buschmann, C., Fekete, S.P., Fischer, S.: Shawn: A new approach to simulating wireless sensor networks. In: Design, Analysis, and Simulation of Distributed Systems 2005, Part of the SpringSim 2005 (April 2005)
17. Zhou, G., He, T., Krishnamurthy, S., Stankovic, J.A.: Impact of radio irregularity on wireless sensor networks. In: MobiSys 2004: Proceedings of the 2nd international conference on Mobile systems, applications, and services, pp. 125–138. ACM Press, New York (2004)
18. Hellbrück, H., Fischer, S.: Towards analysis and simulation of ad-hoc networks. In: Proceedings of the 2002 International Conference on Wireless Networks (ICWN 2002), Las Vegas, Nevada, USA, pp. 69–75. IEEE Computer Society Press, Los Alamitos (2002)
19. Xue, F., Kumar, P.R.: The number of neighbours needed for connectivity of wireless networks. IEEE Wireless Networks 10(2), 169–181 (2004)