Constrained Tracking on a Road Network[⋆]

Michał Piórkowski and Matthias Grossglauser

School of Computer and Communication Sciences,
Ecole Polytechnique Fédérale de Lausanne (EPFL),
CH-1015 Lausanne, Switzerland
`firstname.lastname@epfl.ch`

Abstract. Many applications of wireless ad hoc sensor and actuator networks (WSANs) rely on the knowledge of node locations. These are challenging to obtain when nodes are mobile and are not equipped with any specific positioning hardware. In this paper, we are interested in scenarios where there are constraints on the movement of nodes, such as with cars on a road network.

We develop and analyse a tracking algorithm called MOONwalk that explicitly takes such constraints into account in order to improve the tracking precision. Furthermore, MOONwalk does not require global knowledge of the network, and therefore lends itself well to large-scale and high-mobility applications.

We evaluate the accuracy of MOONwalk by comparing it to the optimal maximum likelihood estimator, under different radio conditions and deployment scenarios. We find that MOONwalk performs well despite its localized operation.

1 Introduction

Numerous applications of sensor and sensor-actuator networks need to track mobile objects, such as people, animals, cars, planes, etc. We are interested in scenarios where the tracked object is equipped with a communication device, but not with a positioning device, such as GPS. Such a situation may arise because of energy, cost, or radio constraints, or because the tracking system has to be operational indoors. In this case, an estimate of the location of the tracked object can be computed from channel measurements between the tracked object and a set of fixed devices. Several papers have considered this tracking problem, where the estimated position of the tracked object is unconstrained, i.e., can lie anywhere in Euclidean space [1–8].

In this paper, we study a related problem, but where the space of possible locations is constrained to a graph. More specifically, the vertices of this graph each correspond to a point in space, and each edge corresponds to a line segment

⋆ The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322. (`http://www.mics.org`)

© Springer-Verlag Berlin Heidelberg 2006

connecting its adjacent vertices. The tracked object's position always lies on this graph. This formulation of the tracking problem is inspired by the tracking of cars on the road network, where vertices model intersections and interpolation points of curved roads.

Our main application for this problem is a fully distributed system called SmartPark, which assists drivers in locating free parking spots and guides them to these spots with turn-by-turn instructions [9]. The key idea is that every parking spot is equipped with an embedded sensor, and a car is equipped with a simple guidance device. For related projects, see [10, 11].

Although it would be possible to solve this problem by relying on an unconstrained tracking algorithm and projecting the location estimate onto the road network, this may give rise to suboptimal precision and efficiency. In this paper, we therefore develop constrained tracking algorithms from first principles, i.e., where the constraint is explicitly taken into account from the outset.

We first formulate this constrained tracking problem in a maximum-likelihood (ML) framework, where the uncertainty stems from fading effects in the radio channel between the car and the sensors. Although the resulting algorithm is optimal, it is computationally demanding and requires knowledge of the full road network graph, as well as knowledge of all the sensors and their geographic positions, and it assumes a specific radio channel model.

In a system such as SmartPark, however, requiring every sensor and car to have full knowledge of the road network and of all the sensors in parking spots would be impractical. Moreover, none of the existing radio channel models fully captures the real characteristics of radio propagation in an urban environment. A realistic algorithm therefore has to be robust to channel uncertainties and has to be able to compute the car's location on the road network with only very limited local information configured in the sensors, and no knowledge in the car. This is because the set of sensors may be large (hundreds of thousands in a large city), and may change over time (e.g., because of node outages). Furthermore, it may be undesirable to configure sensors with their precise geographic position.

In this paper, we describe an algorithm called MOONwalk that relies on considerably weaker assumptions than the ML algorithm. A key feature of this algorithm is that the car's guidance device does not need any a-priori knowledge about the road network graph and about the set of sensors present in its vicinity. This is challenging, because it implies that the guidance device loses valuable information when a particular sensor is not able to communicate with the guidance device, because it is not a-priori aware of the existence of this sensor. In the ML formulation, this information is explicitly taken into account. MOONwalk relies on some very limited state information in sensors to get around this problem, which is particularly important in cases where sensor distributions are very heterogeneous in space.

Another important feature of MOONwalk concerns the a-priori knowledge in sensors, which typically has to be configured by hand or inferred through some other process. ML requires the precise geographical position of each sensor but MOONwalk only requires a set of *potential* positions on the road network (rather than in Euclidean space). This simplifies configuration and makes the tracking process more robust to small configuration errors.

This paper is organized as follows. In Section 2, we describe the models and assumptions. The ML formulation is introduced in Section 3, and the MOONwalk algorithm in Section 4. We report simulation results in Section 5, and show that MOONwalk performs well compared to the optimal ML algorithm, despite the complications cited above. In Section 6, we give a more detailed description of related work. In Section 7, we conclude the paper and describe future research.

2 System Model

2.1 The Road Network Model

A vehicle is constrained to move on the road network. In the case of vehicle tracking this fact becomes an opportunity, because one has to look for the vehicle only within a road network. To model the road network we define the graph $G(V, E)$, where the set V of vertexes represents *intersections* and the set E of directed edges represents *line segments* connecting the intersections [12, 13]. The proposed model can abstract any curved road as a set of straight roads connecting virtual intersections. For the sake of simplicity we assume that G is strongly connected and E contains one-way line segments only, i.e., any two adjacent vertexes are directly connected only by one directed edge. We intend to relax this assumption in the future. Since parking sensors are deployed usually along the roads they can abstract the road topology as a G. At each sensor the information about the line segment $e = (v, w)$ and a distance from v and/or w to its geographical location can be stored. Thus we distinguish between two different types of locations of a sensor i, i.e., the *geographical location* in $\mathbb{R}^2$: X_i, and the *road network location* on G represented by so-called *location-tuple*: L_i. The location-tuple is defined by two functions $L_i = (e(X_i), \beta(X_i))$ that characterise the line segment and the distance between the actual location of a sensor X_i and the location of the preceding (or the following) intersection X_v. In order to simplify the notation we write the following $L_i = (e_i, \beta_i)$, where i represents the sensor, $e_i = (v_i, w_i) \in E$ specifies the line segment on which i is located, and $\beta_i \in [0, 1]$ specifies i's location on e_i. The β_i might be expressed as $\beta_i = \frac{||X_i - X_v||}{||X_v - X_w||}$, where $||X_i - X_v||$ is the Euclidean distance between a sensor and the preceding intersection and $||X_v - X_w||$ represents the length of e_i. However, one can use some other metrics for the β_i, e.g. a time needed to get from the preceding intersection to the actual location X_i. Without loss of generality, as a metric for the β_i we use the ratio between the Euclidean distances. In the case of the vehicle tracking on a road network one can rely on the parking sensors as they can abstract the road network topology. We believe that this approach is appropriate, since the navigation is performed based on the abstracted road topology, i.e. the driver obtains the turn-by-turn instructions in the following format: *"Drive straight until the nearest intersection, then turn right. Your parking spot is the fifth on the right side."*

2.2 The Radio Model

All sensors and wireless guidance devices use omnidirectional antennas and narrowband radio signals for communication. We investigate the probability of signal reception at a given distance from a vehicle and at a certain time, thus we use the *path loss* radio communication combined with *multipath propagation* model. The path loss attenuation between transceivers is given by $d^{-\alpha}$ [14], where α is the path loss exponent, which varies from 2 (free space) up to 6 (harsh environment), and d is the Euclidean distance between transceivers. The received power level shows rapid and deep fluctuations about the local mean with the movement of a mobile node and presence of obstacles. These fluctuations are caused by multipath propagation. They are approximated by the Rayleigh distribution [15]. In our radio model we use the Rayleigh fading model, because it is particularly appropriate when there is no direct line-of-sight between the transmitters, which is often the case in a very harsh outdoor environment. The value of the received power is in fact a random variable that depends on the actual radio propagation characteristics and the distance to the transmitter. We have verified, by means of simulations, that the proposed radio propagation model is consistent with the experimental results achieved in [16]. We do not show these results for the lack of space. Assuming that the transmitted power is 1 mW the received power at the receiver is:

$$P_i(t, d, \alpha, \gamma) = \gamma(t)^\theta d^{-\alpha} \tag{1}$$

where $\gamma(t)$ is the random number drawn according to the Rayleigh p.d.f.:

$$p(r) = \begin{cases} \frac{r}{\sigma^2} exp(-\frac{r^2}{2\sigma^2}) & \text{for } r \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where r is the envelope amplitude of received signal and σ^2 is its variance. The random variable γ is chosen in such a way that its expected value is equal to one. In that case the expected value of P_i is equal to $d^{-\alpha}$ [14]. The exponent θ represents the intensity of the fading, for example, when the multipath effect decreases the level of received signal by 20 dB, then the θ is equal to 2.

3 Maximum Likelihood Approach

In this section we consider the case where the vehicle's location estimate is obtained without any restrictions on used hardware, decentralized implementation or energy budget. We are looking for a scheme that will be optimal in terms of accuracy. We expect that it will also give us an insight into how to develop and implement an algorithm for more realistic scenarios.

We consider the following setting: the vehicle is equipped with a wireless guidance device and it moves on the road network with a constant speed. It broadcasts periodically a *hello* message. All sensors i that are in the vehicle's radio communication range are able to receive this message.

We introduce a random variable $Z_i(t)$ that describes the reception of a message by a parking sensor. A message is received when the received power level exceeds some predefined power reception threshold P_{th}:

$$Z_i(t) = \begin{cases} 1 \text{ if } P_i \geq P_{th} \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

In this method we assume that the following information is available for the vehicle explicitly at any time t:

a) road network topology: $G(V, E)$,
b) geographical locations of all sensors: X_i,
c) instantaneous radio communication conditions: α, distribution of γ; and all sensor's reception power threshold: P_{th},
d) all instantaneous sensor's message reception events: $Z_i(t)$.

Based on the aforementioned information, the vehicle executes the proposed algorithm in order to determine its location on a road network. The random variable $Z_i(t)$ is characterized by the conditional probability $p(z_i(t)|X_i)$. Knowing the power reception threshold and the radio communication conditions we can express this conditional probability as follows: $p(z_i(t)|X_i) = \mathcal{P}(P_R > P_{th}) = \exp(-\frac{d^{2\alpha/\theta} P_{th}^{2/\theta}}{2\sigma^2})$. For the purpose of the ML algorithm we define a vector $\bar{z}$ of N sensor's reception events: $\bar{z}(t) = (z_1(t), z_2(t), ..., z_N(t))$, that represents values of $Z_1(t), Z_2(t), ..., Z_N(t)$. We define a likelihood function corresponding to the vehicle's location on the road network as:

$$\mathcal{J}(e, \beta) = p(\bar{z}(t)|X) \tag{4}$$

Due to the multipath propagation, when a transmitter sends a radio message, its reception by spatially distributed sensors within an urban area is independent. This implies that $Z_1(t), Z_2(t), ..., Z_N(t)$ are independent, hence the $p(\bar{z}(t)|e_i, \beta_i)$ is the product of the marginals:

$$p(\bar{z}(t)|X_i) = \prod_{i=1}^{N} p(z_i(t)|X_i) \tag{5}$$

Applying the radio model from section 2.2, we obtain:

$$\mathcal{J}(e, \beta) = \prod_{i=1}^{N} \begin{cases} \exp(-\frac{d^{2\alpha/\theta} P_{th}^{2/\theta}}{2\sigma^2}) & \text{if } z_i(t) = 1 \\ 1 - \exp(-\frac{d^{2\alpha/\theta} P_{th}^{2/\theta}}{2\sigma^2}) & \text{if } z_i(t) = 0, \end{cases} \tag{6}$$

where $d = ||X_{car} - X_i||$ is the distance between a sensor $i = 1...N$ and a hypothetic vehicle location X_{car}. The maximization is done on two sets - one discrete: E and one continuous: $\beta \in [0, 1]$. This allows us to find a value of β that maximizes (6) for all given $e \in E$ separately. Afterwards we simply select the line segment e for which the (6) takes the maximum value. Therefore we obtain the

ML estimates for both, the line segment e and the β parameter that define the location estimate of the vehicle on the road network $\widehat{L}_{car}$.

The ML tracking method is optimal in terms of accuracy and is robust to node outages. We use this method to examine the best achievable accuracy of the constrained tracking. Note that it depends on the instantaneous radio propagation conditions that are different for each transceiver in the network. Even if we knew explicitly the multipath characteristics, it would be hard to know the current path loss exponent for each transmitter. The path loss exponent depends on the specific spatial distribution of the obstacles within the area where SmartPark operates. As such, it is very difficult or even impossible to characterize the path loss exponent statistically [14].

4 MOONwalk Algorithm

4.1 From ML Towards a Realistic Scenario

In this section we develop an algorithm that, in contrast to the ML algorithm, does not require to know:

a) the whole road network topology,
b) the precise geographical location of all deployed parking sensors,
c) specific radio channel assumptions,
d) sensors that got the *hello* message and those that did not.

We assume that right after the deployment, during the so-called *warm-up* phase, all parking sensors have to discover where they are located on a road network. They can infer their location-tuples $L_i = (e, \beta)$, from a set of the vehicle's movement observations, i.e. they might obtain information about the time needed to get from a certain intersection to a given position on a certain line segment. How these location-tuples are derived, is out of scope of this paper. However, we assume that they might be imprecise, especially in the case of sensors that are close to intersections. Because of this, we allow each sensor to have more than one location tuple, so we define a *set of location-tuples* as: $\mathcal{L}_i = \{(e_i^k, \beta_i^k), ..., (e_i^l, \beta_i^l)\}$ that represents the set of possible candidates for the real location of a sensor i. We also assume that the vehicle, while estimating its road network location L_{car}, can only rely on sensors' road network locations: L_i and the *received signal strength* measurements of each received message: P_i. We consider the following scenario: the vehicle moves on the road network and broadcasts periodically a *hello* message. All sensors that can hear this message respond to the vehicle with a *reply* message. Due to the multipath effects some of the sensors will not be able to receive messages as shown in Figure 1.

The ML algorithm uses the geographical locations of all sensors to estimate the road network location of the vehicle. We seek an algorithm that will not use any information about the sensor's geographical location nor any distance measurement techniques. However, we can rely on distances between sensors and the vehicle, while estimating the location, by using the relationship between

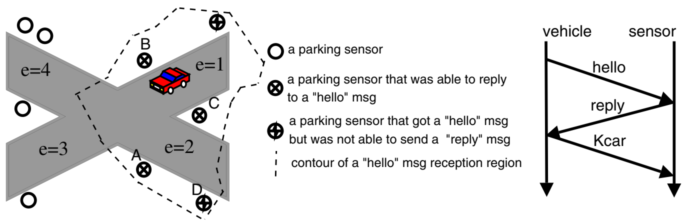

e=4
e=1
e=3
e=2
A
B
C
D
a parking sensor
a parking sensor that was able to reply to a "hello" msg
a parking sensor that got a "hello" msg but was not able to send a "reply" msg
contour of a "hello" msg reception region
vehicle
sensor
hello
reply
Kcar

Fig. 1. An example of the interaction between vehicle and parking sensors

distance and P_i. In a certain propagation direction the expected P_i is monotonically decreasing [16, 17], thus we assume that the expected P_i should be higher for sensors that are closer to the vehicle. Note that we do not make any assumptions about the correlation between absolute distance and the expected P_i. Relying only on instantaneous P_i measurements is not sufficient to design a robust method for location estimation because of the multipath effects.

There might be different densities of sensors on different line segments; if a vehicle can know these densities, such knowledge can help it to decide on which line segment it is at the moment. This kind of information is very useful, e.g. in cases when a vehicle is close to an intersection and it can communicate with sensors that belong to different line segments as shown in Figure 1. Knowing sensors that did and did not receive a *hello* message, helps to infer their densities on line segments. In the ML algorithm the vehicle knows all the sensors that did not receive its *hello* message. A naive approach, where one could take into account only the sensors that respond to the *hello* message, would bias the results of location estimation in favour of high sensor densities, thus it is hard to design a robust real-life algorithm that may know about non-responding sensors. However, even without direct sensor-to-sensor communication, sensors can learn about all neighbouring sensors. For this purpose one could use vehicles that would propagate information about sensors they were able to communicate with recently. Based on such information, each sensor could learn over time about its neighbourhood. The reason we want to rely only on sensor-to-vehicle communication is that the radio channel between sensors might be very obstructed, as they are deployed in the ground. Moreover, the vehicle has an almost infinite resource of energy thus we can easily extend the lifetime of the whole system if we preserve sensors from direct communication between each other. The long-term sensor's observation of its neighbourhood could be sent to the passing by vehicles. Thus a vehicle would be able to find out which sensors were able or not to receive its *hello* message. Note that the accuracy of such information depends on the number of vehicles that pass by and the radio propagation conditions. The more vehicles and the better radio propagation conditions, the better approximation of the sensor's real surrounding.

4.2 How to MOONwalk

The vehicle and parking sensors perform a *three-way communication* that is needed to estimate the vehicle's road network location as shown in Figure 1.

The *hello* message contains a location query. The *reply* message contains the sensor's ID: i, the set of location-tuples: $\mathcal{L}_i$ and its *neighbour table*: N_i. The neighbour table N_i represents a sensor's knowledge of its neighbourhood. Each entry of this table: n_{ij} represents the number of occurrences of a sensor j in the neighbourhood of sensor i: $N_i = \{(j, n_{ij})\}$. Each sensor has an entry for itself in this table. Note that the N_i does not represent a *direct* observation of i's neighbourhood, since there is no sensor-to-sensor communication. After collecting all *reply* messages the vehicle finds two sets - the *reply* message set: $R = \{(i, N_i, \mathcal{L}_i)\}$ and the so-called *one-hop neighbourhood set*: K_{car}. The definition of a neighbour is the following: if a sensor i can communicate directly with the vehicle, after receiving a *hello* message, then i is called a vehicle's neighbour, thus $K_{car} = \{i : i \in R\}$. The K_{car} relies on bi-directional connectivity - if there is only a uni-directional link between two transceivers, they are not considered as neighbours. The K_{car} represents only a snapshot of vehicle's instantaneous neighbourhood at a given time. Once the K_{car} is found, the vehicle broadcasts it to all the sensors in the vicinity, so that they can update their N_i tables. Note that the n_{ii} element in N_i table corresponds to the total number of such messages. We assume that both the set of location-tuples $\mathcal{L}_i$ and the neighbour table N_i of each sensor are given, i.e., all sensors have observed their surrounding long enough.

Since the position of a node on the road network is specified by a location-tuple, we can split the algorithm into two phases. During the first phase the algorithm will identify the proper line segment on which a given vehicle is located. In the second phase the algorithm will find the position on the line segment found in the first phase.

In order to find the proper line segment we need to specify a statistic that represents the certainty of the vehicle's presence on the corresponding line segment. This statistic is a function of the vehicle's observation (K_{car}), local density of sensors that belong to the same line segments (N_is) and the P_i measurements. The proper line segment is found by comparing the statistics. Below we present the MOONwalk algorithm in more details.

Phase I. After performing the three-way communication, a vehicle creates a matrix n_{ij}^{car} using all the N_i tables it has collected:

$$n_{ij}^{car} = \begin{cases} n_{ij} & \text{if } n_{ij} \in R \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The size of this n_{ij}^{car} matrix is the number of all the sensors that are in the K_{car} and the sensors that are neighbours of the sensors from K_{car}. Each element of this matrix takes a value from a corresponding N_i table as shown in Table 1. All the elements of the n_{ij}^{car} matrix represent the number of times a sensor j appeared in the sensor i's neighbourhood. If neighbourhoods of two sensors overlap, they should *see* each other equally often, i.e. if the distance between two sensors i and j is short enough, then it is more likely that they can receive the same *hello* message from a vehicle. This shows how strong two sensors are correlated with respect to their neighbourhoods. For this reason we apply a correlation technique called Pearson's correlation, which in general shows how strong is the association between two variables:

$$\phi_{ij} = \frac{\sum_k (n_{ik} - \bar{n}_{i\cdot})(n_{jk} - \bar{n}_{j\cdot})}{\sqrt{\sum_k (n_{ik} - \bar{n}_{i\cdot})^2}\sqrt{\sum_k (n_{jk} - \bar{n}_{j\cdot})^2}} \tag{8}$$

Thus for all non-zero elements of the n_{ij}^{car} matrix the vehicle finds the Pearson's correlation coefficients matrix ϕ_{ij}^{car}.

Table 1. N_i tables for sensors A,B, and C respectively (left) and a corresponding n_{ij}^{car} matrix (right)

Sensor ID	# occur.
A	13
B	10
C	13
D	2

Sensor ID	# occur.
A	8
B	11
C	11

Sensor ID	# occur.
A	10
B	9
C	12
D	4

$\Longrightarrow$

...	**A**	**B**	**C**	**D**
A	13	8	10	0
B	10	11	9	0
C	13	11	12	0
D	2	0	4	0

After finding all correlation coefficients, for each line segment e, taken from all the received position-tuples from R, the vehicle finds corresponding S_e set that contains all the sensors from K_{car}, which belong to this line segment e: $S_e = \{i : e \in \mathcal{L}_i\} \subset K_{car}$. Note that $K_{car} = \cup_e S_e$. For each such set S_e the vehicle obtains a submatrix of coefficients from the ϕ_{ij}^{car} matrix. The size of such a submatrix is q x q, where $q = |S_e|$. In such a submatrix both rows and columns correspond to the sensors that belong to the same line segment. Suppose that the vehicle has to decide whether it is on a line segment $e = 1$ or $e = 2$ as shown in Figure 1. By inspecting the position-tuples, the vehicle knows that $S_1 = B, C$ and $S_2 = A, B, C$ as shown in Table 2.

Table 2. An example of the two submatrices of the ϕ_{ij}^{car} that contain pairwise correlation coefficients of the sensors that belong to S_1 (left) and S_2 (right)

...	**B**	**C**
B	1.0000	0.9142
C	0.9142	1.0000

...	**A**	**B**	**C**
A	1.0000	0.8889	0.9707
B	0.8889	1.0000	0.9142
C	0.9707	0.9142	1.0000

After finding these submatrices, the vehicle calculates the average correlation for each candidate line segment: $\Phi_e^i = \frac{1}{q}\sum_{j=1}^{q} \phi_{iS_e[j]}$, which approximates the sensor's belief about its line segment membership. Thus for different line segments we have different coefficients for the same sensors. We use the Φ_e^i coefficients as weights to determine the relative importance of the information about a certain location provided by a sensor. At this point we also use the vehicle's set of P_i measurements and the size of each sensor's neighborhood to define a statistic T used to find the proper line segment. By doing so we counterbalance the effect of received responses of nodes which neighbours did not respond. The T statistic takes a maximum value for a line segment for which the certainty of the vehicle's presence is highest.

Phase II. Once the estimated line segment: $\widehat{e}$ is found, the vehicle can search for the estimate $\widehat{\beta}$ of the position on $\widehat{e}$. Here the P_i measurements can be used again. We are only considering the P_i measurements of the sensors that belong to the same line segment $\widehat{e}$, found in the first phase of the algorithm. Usually the parking spots are placed along roads, so the expected P_i of each message, sent by the sensors from the parking spots, should monotonically decrease with the distance between the vehicle and a sensor. Thus we can expect that the sensor for which P_i is the maximum is the closest to the vehicle. We simply take the β parameter of such a sensor as the vehicle's estimate of its position on a certain line segment: $\widehat{\beta}$.

5 Evaluation

5.1 Methodology

In this paragraph we define the performance metric that will measure the accuracy of the proposed methods. The proposed metric is in fact a cost function that shows how much effort a driver would have to devote to get from the estimated location to the real one. Our motivation comes from the following observation: it is easier to move between the real location and the estimated one if both are on the same line segment, even if a driver would have to use the reverse gear. Whereas it is much more difficult to get from the estimated location to the real one through an intersection. Since the tracking subsystem will provide other SmartPark subsystems (e.g. navigation and reservation subsystems) with an actual vehicle's location information, we need to define a performance metric that will express the driver's effort needed to recover from misleading instructions received from SmartPark. Therefore we specify the cost function in the following way. If a driver has to move on the same line segment to get to the estimated location, she does not have to spend more than C effort. When she has to cross an intersection, she has to devote at least C effort. In our approach we express the cost function as a *road graph distance error* Δd_G as follows:

$$\Delta d_G = \begin{cases} |\beta - \widehat{\beta}| & \text{if } \widehat{e} = e \\ 2 - |\beta - \widehat{\beta}| & \text{if } \widehat{e} \neq e \end{cases} \tag{9}$$

One can notice that a driver will have to devote at most $C = 1$ effort in the first case, whereas in the second case the minimal cost will be at least $C = 1$. Therefore we expect from the proposed algorithms that their accuracy in terms of proposed error metric should be at least below 1 and close to 0 as much as possible.

We compare the performance of the two proposed methods under different conditions. The four differrent evaluation areas are illustrated in Figure 2. The

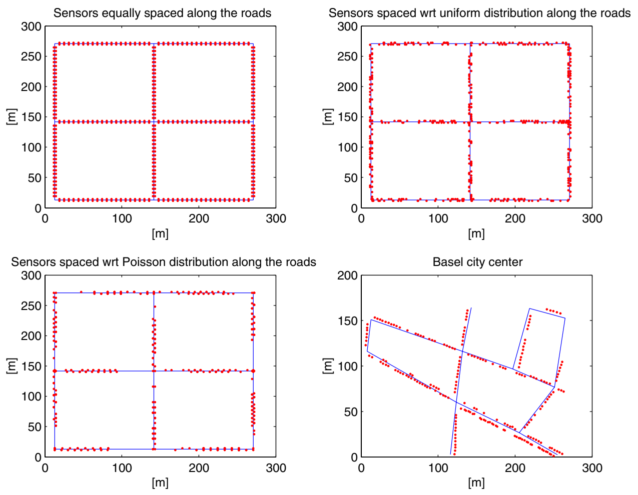


Fig. 2. Four different simulation areas used in tests

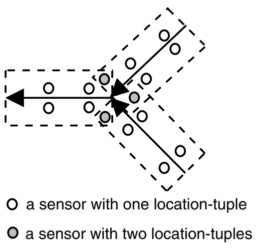
first three differ only in terms of sensors' distribution along the line segments. The fourth one is a part of a city centre that contains 220 sensors that are placed along the line segments - all parking spot coordinates and the road network characteristics were taken from the map of Basel city centre.

First we check how the performace depends on the radio conditions for four different deployments of sensors along the line segments. Next, we check the accuracy of the two algorithms with respect to different sensor densities, within three different simulation areas. For this case, we generate five different sets of sensors for each deployment scheme on the grid road network topology.

Because of the MOONwalk algorithm requirements related to the warm-up phase, we had to perform a special pre-computation that defines the road network mapping and the N_i tables for each parking sensor. For this purpose we have projected the road topology onto the parking sensors in the following way. Since all sensors are deployed along the streets, we define an area A that is a rectangle, whose symmetry axis is the given line segment e. If a sensor node is inside this rectangle A, it means that it belongs to the line segment e. It may happen that one parking sensor will be inside more than one such rectangle, which is usually the case if it is close to an intersection; then this parking sensor has more than one location-tuple that forms the set of location-tuples: $\mathcal{L}_i$. The projection technique is shown in Figure 3.

In case of the N_i tables, we specified five different intervals of occurrences i.e $< 25, 20 >$, $< 20, 15 >$, $< 15, 10 >$, $< 10, 5 >$ $< 5, 0 >$ and five corresponding radii $R_1 < R_2 < R_3 < R_4 < R_5$. The technique we used to construct the N_i for each sensor was the following - if the distance between sensors i and j was smaller than R_k then the number of occurrences of j in i's neighbourhood was uniformly chosen from the kth interval.

In our simulations we use one vehicle that is placed at a random intersection and moves randomly with the constant speed, according to pre-defined road



a sensor with one location-tuple
a sensor with two location-tuples

Fig. 3. The projection of the road network topology onto the fixed part of SmartPark

rules. We develop our own simulator written in MATLAB. Because the exact distribution of Δd_G is unknown, we evaluate the performance based on the lower quartile, the median and the upper quartile of Δd_G.

5.2 Comparison of the Algorithms

Using the same pre-defined path and radio communication conditions, we evaluate the performance of the proposed algorithms through the cumulative distribution of Δd_G, shown in Figure 4. The ML method outperforms the MOONwalk algorithm in all the cases. However the Δd_G is significantly lower than 1 for at least 80% of cases for both algorithms.

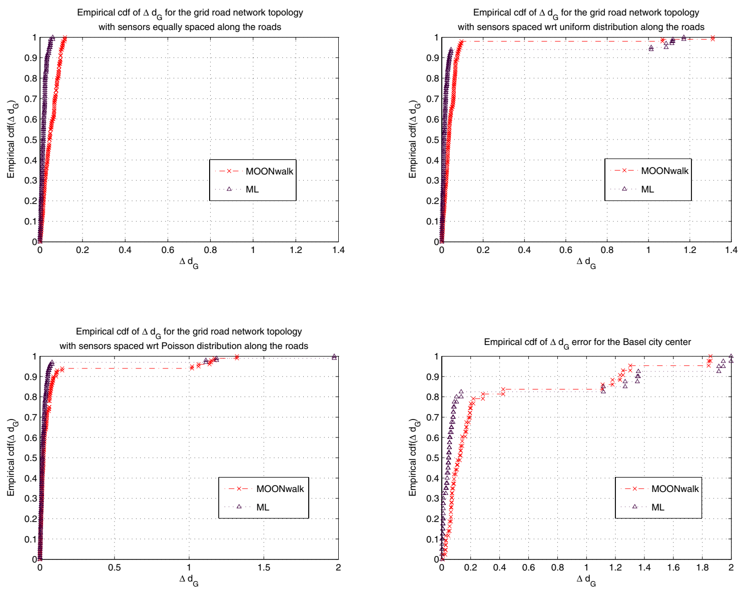


Fig. 4. Comparison of the ML and MOONwalk algorithms based of the empirical cdf of the Δd_G; simulation setup - pathloss exponent: 4, multipath fading: 20 dB, size of each journey: 9 line segments, vehicle's speed: 40 km/h

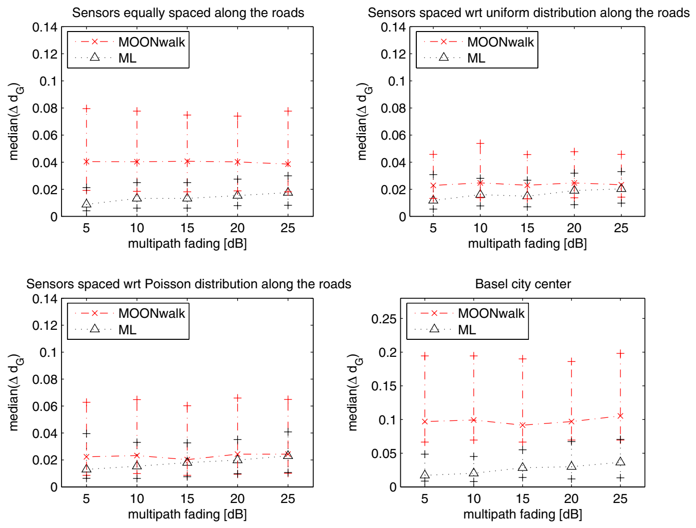


Fig. 5. Performance evaluation of ML and MOONwalk algorithms for different intensities of multipath fading on different simulation areas; simulation setup - pathloss exponent: 4, size of each journey: 70 line segments (approximately 350 Δd_Gs), vehicle's speed: 40 km/h

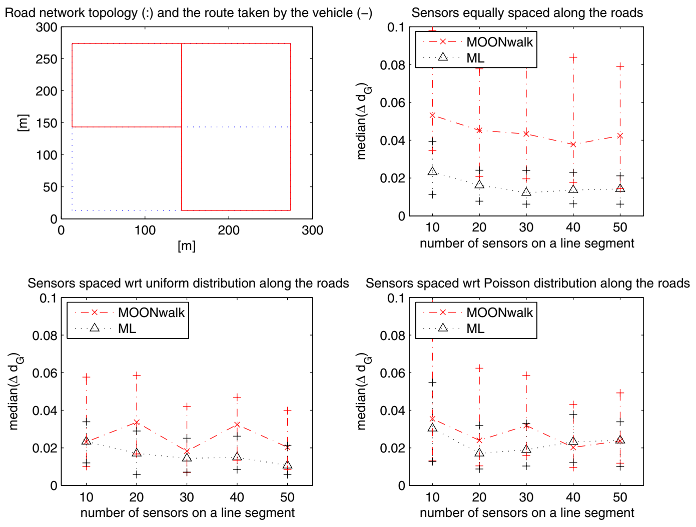


Fig. 6. Performance evaluation of ML and MOONwalk algorithms for a different number of sensors per line segment on different simulation areas; simulation setup - pathloss exponent: 4, multipath fading: 20 dB, size of each journey: 9 line segments, vehicle's speed: 40 km/h

During the second test, we let the vehicle travel through 70 line segments. We calculate the lower quartile, the median and the upper quartile of the Δd_G. We do this for different intensities of Rayleigh fading. The median value of Δd_G grows with multipath intensity for the ML algorithm, which is not the case for the MOONwalk method, as shown in Figure 5. The results of the third test are shown in Figure 6. Here we check how the number of active sensors affect the algorithms performance. The median value of Δd_G decreases with the number of sensors for both algorithms. The improvement is, however, not significant. This is a good sign because the performance of both algorithms does not suffer from sensor outages - if 80% of the sensors from a line segment cannot communicate with a vehicle at a given time, the Δd_G decreases at most of 0.0128 and 0.0123 for ML and MOONwalk algorithms, respectively.

The MOONwalk algorithm depends mostly on the proper projection of the road network topology onto the fixed part of SmartPark. However, its strong advantage is that it does not require as much input data as ML in order to achieve comparable accuracy. Moreover, its accuracy does not decrease in the case of significant multipath fading and it is much easier to implement in a real-life scenario than ML. This shows us that MOONwalk is a good candidate for the tracking subsystem of SmartPark.

6 Related Work

Extensive research has been done on localization and tracking for wireless sensor networks. A general survey on localization is found in [18]. Systems focused on the locating problem on a large geographic scale use, for example, GPS. In the past decade, there have been many different types of location-based projects that work outdoors, where the radio signal is highly disturbed by the presence of obstacles and moving objects. These systems are usually supported by networks of small spatially distributed wireless devices. Here, we focus only on localization techniques suitable for sensor networks that use only the Radio Frequency (RF) based approach.

In [7] Gupta and Das have developed a simple detection and tracking algorithm that involves only simple computation and local communication only to the nodes in the vicinity of the target. They focus only on the unconstrained tracking approach, where the target is unknown to the system and its appearance alerts all sensors along the projected path of the target. In this approach all sensors are capable of estimating the distance of the target to be tracked from the sensor readings. This requires additional effort to map the sensor readings to the distance. In order to track the target they use the triangulation method.

In [17] Youssef et. all present a method for inferring the location of a device based on FM radio signal strengths. The proposed method is robust to measurement differences between devices by basing the inferences on rankings of radio stations rather than on their absolute received signal strength. The method does not require any manual survey of received signal strength as a function of location. However, it requires the usage of the simulated radio maps designed for a given area, validation of simulated received signal strength maps and an additional pre-processing needed to reduce the number of rank vectors that correspond to given locations.

Our approach also uses a RF-based method needed to perform the localization of a mobile node within a wireless sensor network. However, in contrast to the above-mentioned systems where the tracking is unconstrained, our work focuses on the problem of how to build the mobile node's movement constraints into the tracking algorithm in order to meet the application needs. Given the fact that we use a decentralized architecture of small devices that are relatively inexpensive, our constrained tracking approach, especially MOONwalk with the option where no extra hardware is needed, represents an attractive and powerful solution to the vehicle tracking on a road network problem.

7 Conclusion and Future Work

The MOONwalk algorithm is a good candidate for the tracking subsystem of SmartPark mainly because it does not require any additional range-measurement hardware. Its accuracy is comparable with that achieved by the optimal approach specified by ML algorithm. We believe there is still place for improvement. If we introduce more vehicles we can rely not only on the sensor-to-vehicle, but also on the vehicle-to-vehicle communication. We currently are investigating this approach called *collaborative tracking* in more detail. Another possible improvement is to let the vehicle send a *hello* message with the maximum RF power and to let the parking sensor send back several *reply* messages, using different RF power levels. One could combine this approach with the collaborative tracking. By this it might be possible to achieve a sufficient trade-off between communication overhead and accuracy. We are also working on the dynamic version of both methods where we are using the dead reckoning techniques. In order to improve the accuracy of the location estimation, we apply the past location estimates of a vehicle. Preliminary results show that the improvement is significant. We are also investigating the problem of the road network topology projection onto the fixed wireless network of parking sensors. We are currently evaluating the MOONwalk algorithm on a test-bed in a real-life scenario.

References

1. J. Heidemann N. Bulusu and D. Estrin. GPS-less Low-Cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications*, 7:28–34, October 2000.
2. D. Niculescu and B. Nath. DV Based Positioning in Ad Hoc Networks. *Telecommunication Systems*, pages 267–280, January-April 2003.
3. H.T. Kung and D. Vlah. Efficient Location Tracking Using Sensor Networks. In *IEEE Wireless Communications and Networking WCNC2003*, volume 3, pages 1954–1961, March 2003.
4. T. He, C. Huang, B.M. Blum, J.A. Stankovic, and T. Abdelzaher. Range-free Localization Schemes for Large Scale Sensor Networks. In *The 9th Annual International Conference on Mobile Computing and Networking*, pages 81–95, September 2003.

5. D. Estrin N. Bulusu, V. Bychkovskiy and J. Heidemann. Scalable, Ad Hoc Deployable RF-based Localization. In *The Grace Hopper Celebration of Women in Computing Conference*, October 2002.
6. A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha. Tracking Moving Devices with the Cricket Location System. In *MobiSYS '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 190–202, New York, NY, USA, 2004. ACM Press.
7. R. Gupta and S.R. Das. Tracking Moving Targets in a Smart Sensor Network. In *IEEE Vehicular Technology Conference VTC2003-Fall*, volume 5, pages 3035–3039, October 2003.
8. 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. ACM Press, 2004.
9. SmartPark project website. http://smartpark.epfl.ch.
10. Vehicle Information and Communication System. `http://www.vics.or.jp/english/index.html`.
11. www.roadtraffic-technology.com : The web site for the road traffic industry. http://www.roadtraffic-technology.com/contractors/parking/.
12. J. Tian, L. Han, K. Rothermel, and C. Cseh. Spatially aware packet routing for mobile ad hoc inter-vehicle radio networks. In *Proceedings of the IEEE 6th International Conference on Intelligent Transportation Systems (ITSC '03)*, October 2003.
13. A. Leonhardi, Ch. Nicu, and K. Rothermel. A Map-Based Dead-Reckoning Protocol for Updating Location Information. In *IPDPS '02: Proceedings of the 16th International Parallel and Distributed Processing Symposium*, page 15, Washington, DC, USA, 2002. IEEE Computer Society.
14. W.C. Jakes. *Microwave Mobile Communications*. Wiley-IEEE Press, May 1994.
15. B. Sklar. Rayleigh Fading Channels in Mobile Digital Communication Systems .I. Characterization. *IEEE Communications Magazine*, 35:90–100, July 1997.
16. D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks, February 2002.
17. A. Youssef, J. Krumm, E. Miller, G. Cermak, and E. Horvitz. Computing Location from Ambient FM Radio Signals. In *IEEE Wireless Communications and Networking Conference (WCNC 2005)*, March 2005.
18. J. Hightower and G. Borriello. Location Systems for Ubiquitous Computing. In *IEEE Computer*, volume 34, pages 57 – 66, August 2001.