On-road Trajectory Generation from GPS Data: A Particle Filtering/Smoothing Application

Michael Roth, Fredrik Gustafsson and Umut Orguner

On-road Trajectory Generation from GPS Data: A Particle Filtering/Smoothing Application

Michael Roth and Fredrik Gustafsson
Dept. of Electrical Engineering
Linköping University
Linköping, Sweden
Email: roth@isy.liu.se, fredrik.gustafsson@liu.se

Umut Orguner
Dept. of Electrical and Electronics Engineering
Middle East Technical University
Ankara, Turkey
Email: umut@eee.metu.edu.tr

***Abstract*—Many studies in target localization and tracking use GPS measurements as ground truth. These GPS locations might be in conflict with computed estimates in applications where road network information is available (and employed in the estimation procedure). This paper proposes to use particle methods to generate on-road trajectories that can be used as improved ground truth for such road constrained estimation schemes. A bootstrap particle filter and three different particle smoothers are utilized to obtain kinematic target state estimates. The particle smoothers require important adjustments for their implementation in the resulting hybrid state space. The performances of the presented methods are compared on challenging real data obtained from an urban area. Although particle filters and smoothers can be applied to general localization problems, with arbitrary sensors, we concentrate on GPS measurements, motivated by an application in cellular network systems.**

I. Introduction

There are many target tracking and localization applications where the target is constrained to a known road network. We can here mention localization in cellular networks, tracking based on ground radar or aerial vision sensors, and various ground sensor network applications. Our motivation comes from a practical need for a system manufacturer of cellular network systems, where they currently employ manual drive tests to measure system performance and optimize network parameters. This is also where our field test data come from. A related need occurs in self-optimizing networks (SON), where user equipments are commanded to transmit position related data whenever they face a problem, such as a missed or dropped call, poor signal to noise ratio, severe multipath, late hand-over *etc*. If the user equipment is concluded to be road-bound, coverage over the road network can be optimized by adjusting antenna angles and system parameters. For SON, smoothing is both possible, and also required to get sufficient accuracy of the often inaccurate network measurements.

In all of these applications, one needs to compare estimates (which are constrained to the road) with ground truth position, which has to be more accurate than the estimates. Often GPS positions are employed as ground truth. However, GPS data might be in conflict with the road information. We therefore consider the problem of generating on-road trajectories from GPS measurements. Instead of projecting the GPS data onto the closest road segment, which is problematic at road intersections, we pursue a filtering/smoothing approach. An estimation scheme that includes road network information requires state-of-the-art algorithms. A concise overview of recent developments is given in [1]. Among the vast literature on road constrained tracking we furthermore highlight the book [2], which advocates the here applied particle methods, and an article [3] that showed how improved tracking results were obtained by considering the road network.

From an estimation perspective the main challenge is that the state space has discrete components: the target's road segment. Classical (Kalman) filtering approaches are often used in an interacting multiple model (IMM) framework (with the road segment as mode state) for such a problem. Further modifications are required to obtain computationally feasible algorithms. Particle methods in turn can be applied directly. We investigate a bootstrap particle filter [4] and a number of smoothing algorithms that are based on the filter output. The involved state space model, likelihood, and transition density are developed for the problem at hand.

The on-road trajectory generation problem is formulated in the next section. Section III introduces the required particle methods while Section IV covers the application in detail. Results are shown and discussed in Section V. Concluding remarks are given in Section VI.

II. Problem formulation

We want to obtain the kinematic state of a target that is known to travel on-road by processing available GPS measurements. In our setting, road segments are considered as curves in the x-y-plane and in general the given GPS measurements do not lie on these curves. On one hand this is a result of neglecting the road widths, on the other hand the GPS measurements might be erroneous (multipath effects in urban areas, atmospheric effects, timing errors). Figure 1 shows the considered GPS measurements with time stamps and a section of the road network.

An ad-hoc procedure to obtain on-road data would be to orthogonally project each GPS position onto the closest road segment. Apart from the task of finding the closest segment,

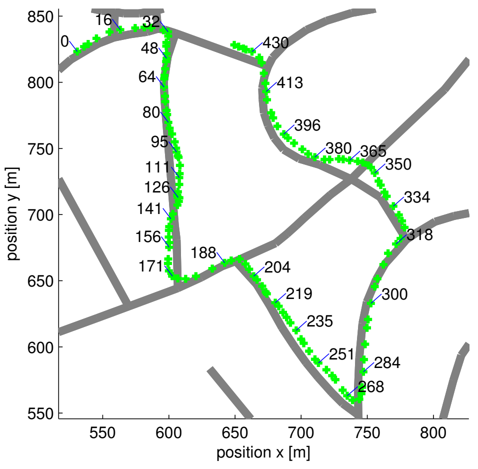

Figure 1. GPS trajectory and road network section, time stamps in seconds.

this can give erroneous results, depending on the road topology. A difficult bit can be seen in Figure 1 between 350 and 380 seconds. Clearly, a projection onto the closest segment would pick the wrong road in this example.

Alternatively, we consider a tracking problem for a target that is bound to move on-road [2]. The road segment is treated as state variable. Filtering and smoothing techniques can be used to obtain approximate probability density functions which again can be used to compute an estimate. More specifically, particle methods are chosen to cope with the hybrid state space. It is of interest to see if the inherent lookahead benefit of smoothing can yield improved performance over filtering in this off-line problem.

III. Particle filtering and smoothing theory

In this section we briefly introduce the required particle methods. We consider a discrete time state space model

$$x_{k+1} = f_k(x_k, v_k), \tag{1}$$
$$y_k = h_k(x_k, e_k). \tag{2}$$

Subsequent states x_{k+1} are related to current states x_k and process noise v_k by the (possibly nonlinear and possibly time-varying) function f_k. A measurement y_k is generated by x_k via the function h_k and measurement noise e_k. Both v_k and e_k are assumed to have known probability distributions. In case of additive noise e_k and a continuous measurement space, the likelihood $p(y_k|x_k)$ can be computed by evaluating the probability density function of e_k at $y_k - h(x_k)$.

A. Particle Filtering (PF)

An introduction to the subject can be found in e.g. several tutorial papers [5]–[7] or recent books [2], [8]. Gordon et al. [4] is often cited as first particle filter (PF) and indeed we will stick to the proposed method.

In the filtering problem one seeks the posterior distribution of a state x_k given all the measurements up to time step k: $p(x_k|y_{1:k})$. Although a conceptual solution to this problem is given using Bayes' law, the resulting posterior cannot be described by a finite number of parameters in general (except for the well known special cases). The PF therefore approximates the posterior by a set of samples (particles) and weights $\{x_k^{(i)}, w_k^{(i)}\}_{i=1}^N$. Actually, the algorithm provides a particle representation of the joint smoothing density $p(x_{0:k}|y_{1:k})$ that is sequentially computed based on the concept of importance sampling. At each step k, N samples $x_k^{(i)}$ are drawn from proposal distributions $q(x_k|x_{0:k-1}^{(i)}, y_{1:k})$ and appended to the existing trajectories to form $\{x_{0:k}^{(i)}\}_{i=1}^N$. The corresponding weights are then computed according to

$$w_k^{(i)} = \frac{p(x_k^{(i)}|x_{k-1}^{(i)})p(y_k|x_k^{(i)})}{q(x_k^{(i)}|x_{0:k-1}^{(i)}, y_{1:k})} \tag{3}$$

and subsequently normalized. In case the proposal is chosen as prior $p(x_k|x_{k-1}^{(i)})$, as in [4], each particle (trajectory) gets weighted by its likelihood only. Samples from the prior can be drawn by predicting the particle with an independent realization of v_k according to (1).

Problematic is that this sequential importance sampling eventually leads to so called weight degeneracy. That is, all but one weights will turn to zero. In [4] a crucial resampling step, in which particles are discarded or duplicated according to their weights, was introduced to counteract this problem. The degree of degeneracy can be assessed by the effective number of particles (approximately) given by

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^N (w^{(i)})^2}. \tag{4}$$

Resampling can be applied whenever N_{eff} falls below a threshold.

An approximate marginal filtering density can be obtained by extracting $\{x_k^{(i)}, w_k^{(i)}\}_{i=1}^N$ from $\{x_{0:k}^{(i)}, w_k^{(i)}\}_{i=1}^N$. The algorithm is initialized with appropriate particles and weights $\{x_0^{(i)}, w_0^{(i)}\}_{i=1}^N$.

B. Particle Smoothing

Particle smoothers have only recently been developed. There exist different variants of which three will be explained in this section. Further details can be found in the overview [6], the thesis [9], and the references below.

We seek the marginal smoothing density $p(x_k|y_{1:K})$ where $y_{1:K}$ is a batch of measurements and k ranges from 0 to K. Again, a conceptual solution can be derived but needs to be approximated by sets of particles. The presented algorithms rely on the output of a previously run PF and do not generate new particles. Therefore, it is necessary that regions of interest in the state space are explored thoroughly in the filtering procedure. For bootstrap filters this means that the process noise should not be too small.

1) Fixed lag smoothing distribution from filter output (FL): In Section III-A we stated that the particle filter actually gives an approximation of the joint smoothing density $p(x_{0:k}|y_{1:k})$. In the same way as we extracted marginal densities at time k we could also obtain a particle representation of a fixed lag smoothing density $p(x_{k-L}|y_k)$ by pairing the corresponding weights and samples $\{x_{k-L}^{(i)}, w_k^{(i)}\}_{i=1}^N$. It should be noted that due to the resampling process many particles at step k share common ancestors. For this reason a depleted sampling representation is expected for large lags L.

2) Forward filter backward smoother (FFBSm): The method proposed in [10] targets the marginal smoothing distribution. It is sometimes referred to as forward filter backward smoother. The particles from the filtering step are kept but assigned new weights. Initialized with the filtering weights $\{w_K^{(i)}\}_{i=1}^N$ the following update is performed for $k = K-1$ to $k = 0$:

$$\bar{w}_k^{(i)} = \sum_{j=1}^{N} \bar{w}_{k+1}^{(j)} \frac{w_k^{(i)} p(x_{k+1}^{(j)}|x_k^{(i)})}{\sum_{l=1}^N w_k^{(l)} p(x_{k+1}^{(j)}|x_k^{(l)})}. \tag{5}$$

The computational complexity for smoothing a trajectory this way is of order KN^2 as the transition density needs to be evaluated for all pairings of particles at k and $k+1$.

3) Forward filter backward simulator (FFBSi): Another approach, suggested in [11], is to systematically pick M samples $\{\tilde{x}_k\}_{i=1}^M$ from $\{x_k^{(i)}\}_{i=1}^N$ backwards in time for each k and to append them to previously assembled trajectories $\{\tilde{x}_{k+1:K}\}_{i=1}^M$. These M "simulated" trajectories (hence the name) form an equally weighted particle approximation of the joint smoothing density. We present how a single trajectory is drawn. At K, one sample $\tilde{x}_K = x_K^{(i)}$ is chosen with probability $w_K^{(i)}$. For each k from $K-1$ to 0, N normalized weights are computed according to

$$\tilde{w}_k^{(i)} = \frac{w_k^{(i)} p(\tilde{x}_{k+1}|x_k^{(i)})}{\sum_{l=1}^N w_k^{(l)} p(\tilde{x}_{k+1}|x_k^{(l)})} \tag{6}$$

and again used to select $\tilde{x}_k = x_k^{(i)}$ with probability $\tilde{w}_k^{(i)}$. Implementing this procedure requires KMN evaluations of the transition density. We can hence use $N > M$ particles in the filter and subsequently select fewer particles for the smoother to decrease the number of operations. An even more efficient variant that scales as KN has been suggested in [12].

For algorithmic details of all presented smoothing algorithms and the fast FFBSi implementation [12], the reader is referred to [9].

C. Computing an estimate from a cloud of particles

In most applications a state estimate $\hat{x}_k$ and an indication of its confidence (e.g. a covariance matrix) are required. In a continuous state space the minimum mean squared error (MMSE) estimate is a popular choice. It is merely given by a weighted average of particles $\hat{x}_k = \frac{1}{N}\sum_{i=1}^N w_k^{(i)} x_k^{(i)}$. Alternatively, a maximum a posteriori (MAP) estimate is desired in a filtering application. This is not given by the particle with the highest weight. A method to compute MAP estimates from particles can be found in [13]. The suggested algorithm involves evaluation of the transition density and can be used to incorporate more information into the filtering process while keeping a simple bootstrap filter thats draws samples from the prior. Similar to a MAP estimate for filtering a maximum smoothing pdf estimate could be derived.

In a state space with discrete components MMSE estimates cannot be computed by simple averaging. Also, constraints on the continuous part of the state space are challenging to handle. We provide a solution for our specific problem in Section IV-G.

IV. Application to the on-road trajectory generation problem

We will next develop the required functions to apply the algorithms of the previous section.

A. Road network information

There are several ways how a road network can be described, as seen in e.g. [1], [14]. The choice of description in turn leads to a certain type of road coordinates.

We focus on a point wise description which represents the network by labeled way points (nodes) that are stored along their position and the labels of all connected nodes. (Example: node 4 has position $\mathsf{x} = 3$ and $\mathsf{y} = 204$ and is connected to nodes 3, 8, and 149.) The nodes are assumed to be connected by straight road segments and the segment lengths can be readily computed. Also one-way roads can be represented in this format. For the time being, we do not consider road widths, speed limits, stop signs and traffic lights, which among other information could be included in more informative road network data.

The description relates to the field of graph theory and node indices and segment lengths can be seen as a weighted directed graph [15]. Consequently we can obtain the shortest paths between all pairs of nodes and also find out about the number of intersections on these shortest paths. For each node pair i, j we store the distance on the shortest path and a factor f_{ij} that is computed according to following algorithm:

```
1: set $\mathsf{f}_{ij} = 1$
2: **for all** nodes $l$ on the shortest path from $i$ to $j$ **do**
3:   **if** node $l$ is an intersection **then**
4:     set $\mathsf{n}_l$ = number of nodes attached to $l$
5:     $\mathsf{f}_{ij} = \mathsf{f}_{ij}/(\mathsf{n}_l - 1)$
6:   **end if**
7: **end for**
```

Each factor f_{ij} is the probability to be at node j when, starting at node i, a distance as long as the shortest path between i and j is traveled. Here the assumption is made that all other connecting paths can be neglected and only one initial direction is considered. The factor relates to the motion model of Section IV-C and will be used in the evaluation of transition densities in Section IV-E.

We symbolically denote the entire road network by $\mathcal{R}$.

B. Road and global coordinates

Given the road network $\mathcal{R}$, we now turn to the state representation of a road bound target. Each segment of $\mathcal{R}$ can be identified by two node index integers n_1 and n_2. These form the discrete part of the state space.

Let d be the target position (a distance) on the current road segment, by convention measured from n_1. Obviously this distance has to obey $0 < \mathsf{d} < \mathsf{l}$ where l is the segment length. Let s be the speed on the current road segment. By convention a target with $\mathsf{s} > 0$ is traveling from n_1 towards n_2.

A kinematic state of an on-road target is given by

$$\begin{aligned} x_k &= \begin{pmatrix} \mathsf{d}_k, & \mathsf{s}_k, & \mathsf{n}_{1k}, & \mathsf{n}_{2k} \end{pmatrix}^T \\ &\triangleq \begin{pmatrix} \mathsf{l}_k - \mathsf{d}_k, & -\mathsf{s}_k, & \mathsf{n}_{2k}, & \mathsf{n}_{1k} \end{pmatrix}^T . \end{aligned} \tag{7}$$

The second row shows that the same state has two different representations.

We next relate the state to the GPS measurements which are given in a Cartesian coordinate frame. Let x and y denote the corresponding two dimensional target position, $\dot{\mathsf{x}}$ and $\dot{\mathsf{y}}$ Cartesian velocities. Using the road network information, each state (7) can be transformed to Cartesian coordinates by simple geometric operations. In particular, we denote the mapping of road bound to Cartesian position by $\mathsf{M}(\mathsf{d}, \mathsf{n}_1, \mathsf{n}_2, \mathcal{R})$ and obtain the following nonlinear measurement equation (2):

$$y_k = \begin{pmatrix} \mathsf{x}_k^{\mathsf{GPS}}, & \mathsf{y}_k^{\mathsf{GPS}} \end{pmatrix}^T = \underbrace{\mathsf{M}(\mathsf{d}_k, \mathsf{n}_{1k}, \mathsf{n}_{2k}, \mathcal{R})}_{h(x_k)} + e_k. \tag{8}$$

The distribution of the two dimensional noise e_k will be described in Section IV-D.

C. State transition function

In order to draw independent samples from the prior distribution $p(x_{k+1}|x_k)$ in a particle filter, we predict each particle with an independent process noise realization. Here we develop the required state transition function (1).

The two state components d_k and s_k are first treated without (segment length) constraints and subsequently adjusted if necessary. As the available measurements are sampled non-uniformly at times t_k we define $\mathsf{T}_k = \mathsf{t}_{k+1} - \mathsf{t}_k$ to be the sampling intervals. Of several available one-dimensional motion models [8], [16] we pick a (nearly) constant velocity type:

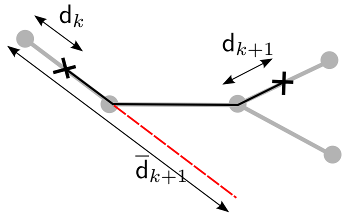

Figure 2. Prediction of on-road distances; here, a road segment is skipped.

$$\begin{pmatrix} \bar{\mathsf{d}}_{k+1} \\ \mathsf{s}_{k+1} \end{pmatrix} = \begin{pmatrix} 1 & \mathsf{T}_k \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathsf{d}_k \\ \mathsf{s}_k \end{pmatrix} + \mathsf{v}_k. \tag{9}$$

The process noise v_k affects the target's maneuverability. We specify its covariance matrix

$$\mathsf{Q}_k = \operatorname{cov} \mathsf{v}_k = \begin{pmatrix} \mathsf{T}_k^3/3 & \mathsf{T}_k^2/2 \\ \mathsf{T}_k^2/2 & \mathsf{T}_k \end{pmatrix} \mathsf{q} \tag{10}$$

but do not restrict ourselves to the commonly chosen Gaussian distribution. The model (9) together with (10) can be derived by discretization of a continuous-time constant velocity model with white acceleration noise (power spectral density q) input [16]. The matrix (10) has full rank in contrast to its more often used alternative $[(\frac{\mathsf{T}_k^2}{2}, \mathsf{T}_k)\, \mathsf{q}\, (\frac{\mathsf{T}_k^2}{2}, \mathsf{T}_k)^T]$ and thus, any expressions involving its inverse, e.g. densities, can be evaluated.

The particle filtering framework merely requires that samples from the process noise can be generated. This in turn opens up for the use of wider tailed distributions than the Gaussian that might be beneficial in modeling a wider range of maneuvers, for instance Student's *t*-distribution [17]. Such process noise choices can be used to complement IMM approaches [3] and might reduce the number of required modes.

The linear update (9) might yield a $\bar{\mathsf{d}}_{k+1}$ that extends the current road segment. Using the road network information we project the excess distance onto a following segment [18] and alter the node indices accordingly. Of course there might be several candidates so whenever an intersection is involved we pick one of the alternatives with equal probability (without reconsidering the segments we came from). Also, road segments can be skipped in one prediction, especially for a dense road network with many short segments. An example is illustrated in Figure 2. Symbolically, we define the projection P and write

$$(\mathsf{d}_{k+1}, \mathsf{n}_{1k+1}, \mathsf{n}_{2k+1})^T = \mathsf{P}(\bar{\mathsf{d}}_{k+1}, \mathsf{n}_{1k}, \mathsf{n}_{2k}, \mathsf{c}_k, \mathcal{R}) \tag{11}$$

where c_k represents the random choices during the projection.

D. Likelihood

The two dimensional real valued measurement noise e_k can be obtained from Equation (8). As it enters additively, the likelihood $p(y_k|x_k)$ can be computed by evaluating the (yet to be determined) noise pdf. We will pursue a practical approach and base the likelihood choice on its function in the PF.

In the utilized bootstrap filter each particle is weighted by its likelihood. By inspecting Figure 1 it is reasonable to have a likelihood that 1) does not decay too fast with Euclidean distance from 0 (such that the filter works even though the measurement is far from the road), but also that 2) does not assign disproportionately high weights to particles that are close to measurements. The latter demand is to avoid being tricked by ambiguous situations as between 350 and 380 seconds. In summary, the likelihood should not have a high peak close to 0 but instead be "less informative".

Choices for $p(y_k|x_k)$ include 1) uniform on a circle around the origin, i.e. all particles in a defined vicinity of the measurement get assigned equal weights. The circle radius should be larger than the maximum distance between measurement and the "true road". Here, particles that are too far away from the measurement will be discarded immediately in the resampling step. 2) Gaussian with large covariance matrix. Here, each particle gets assigned an individual weight.

E. Transition density

In order apply the algorithms of Section III-B, we need to be able to evaluate transition densities $p(x_{k+1}|x_k)$. The key idea is to reverse the mapping (11) to reconstruct the intermediate $\bar{\mathsf{d}}_{k+1}$ from (9). With certain rearrangements we can then proceed as if x_{k+1} and x_k were on the same road segment.

Let us first consider the trivial case of x_{k+1} and x_k on the same road segment: For interchanged node indices ($\mathsf{n1}_{k+1} = \mathsf{n2}_k$) we need to alter one of the states in the same way as shown in (7). If $\mathsf{n1}_{k+1} = \mathsf{n1}_k$ we can simply derive the process noise term v_k from (9). Evaluating the pdf of v_k gives the desired transition density under the assumption that the target moved on the shortest path from x_k to x_{k+1}.

For the case of x_{k+1} and x_k on different road segments we again assume the target to have taken the shortest path. From the available states we know that it traveled on one of four different routes (via $\mathsf{n1}_k$ and $\mathsf{n1}_{k+1}$, $\mathsf{n1}_k$ and $\mathsf{n2}_{k+1}$, $\mathsf{n2}_k$ and $\mathsf{n1}_{k+1}$, or $\mathsf{n2}_k$ and $\mathsf{n2}_{k+1}$). Obviously, these paths will have different lengths which we can compute using the road network information. Let

$$\tilde{\mathsf{d}}_{k+1} = \bar{\mathsf{d}}_{k+1} - \mathsf{d}_k \tag{12}$$

be an increment in on-road distance obtained by the prediction step. Then the shortest path is the smallest increment among the candidates

$$\begin{aligned}
\tilde{\mathsf{d}}^1_{k+1} &= \mathsf{d}_{k+1} + \mathsf{D}(\mathsf{n1}_{k+1}, \mathsf{n2}_k, \mathcal{R}) + (\mathsf{l}_k - \mathsf{d}_k),\\
\tilde{\mathsf{d}}^2_{k+1} &= \mathsf{d}_{k+1} + \mathsf{D}(\mathsf{n1}_{k+1}, \mathsf{n1}_k, \mathcal{R}) + \mathsf{d}_k,\\
\tilde{\mathsf{d}}^3_{k+1} &= (\mathsf{l}_{k+1} - \mathsf{d}_{k+1}) + \mathsf{D}(\mathsf{n2}_{k+1}, \mathsf{n2}_k, \mathcal{R}) + (\mathsf{l}_k - \mathsf{d}_k),\\
\tilde{\mathsf{d}}^4_{k+1} &= (\mathsf{l}_{k+1} - \mathsf{d}_{k+1}) + \mathsf{D}(\mathsf{n2}_{k+1}, \mathsf{n1}_k, \mathcal{R}) + \mathsf{d}_k.
\end{aligned}$$

The utilized function D returns the distance of two nodes on the shortest connecting path (which is basically a look up from $\mathcal{R}$). For instance $\mathsf{D}(\mathsf{n1}_{k+1}, \mathsf{n2}_k, \mathcal{R})$ gives the distance on the shortest path from $\mathsf{n2}_k$ to $\mathsf{n1}_{k+1}$. As each increment corresponds to one of the four alternatives, we can (with careful rearrangements) find $\bar{\mathsf{d}}_{k+1}$ and s_{k+1}. Care needs to be taken with the sign of s_{k+1}. With the recovered quantities we can again evaluate the pdf of v_k from (9).

For different road segments $p(x_{k+1}|x_k)$ is not necessarily given by the pdf of v_k because there could have been intersections on the shortest path. Therefore we multiply it by the correcting factor $\mathsf{f} \leq 1$ (Section IV-A) that accounts for the random choices (Section IV-C) on the assumed shortest path.

F. Filter and smoother initialization

The filtering algorithm depends on its initial conditions, that is the set of particles at $k = 0$. For the on-road trajectory generation problem we suggest to manually select appropriate on-road states $x_0^{(i)}$ (especially the road segment), based on the corresponding GPS measurement. Similarly, we can replace the smoother initial particles (approximation of $p(x_K|y_{1:K})$) by a manually selected set. Here, it might be useful to spread out the particles a bit further such that the evaluated $p(x_K^{(j)}|x_{K-1}^{(i)})$ differs from zero for sufficiently many pairs of particles.

G. Computing an estimate

At this stage we should remind ourselves of the actual aim of on-road trajectory generation. We want to obtain accurate kinematic states that can be used as ground truth for other tracking algorithms. The desired trajectory should be given in a global coordinate frame instead of road coordinates. Of course any state x in road coordinates can be transformed to global coordinates by simple geometric relations. We introduce $\xi = \begin{pmatrix}\mathsf{x}, & \mathsf{y}, & \dot{\mathsf{x}}, & \dot{\mathsf{y}}\end{pmatrix}^T$ as such a transformed x and proceed by considering estimates that can be obtained from weighted particles $\{\xi^{(i)}, w^{(i)}\}_{i=1}^N$.

The classical MMSE estimate given by $\hat{\xi}_k^{\mathsf{MMSE}} = \sum_i w_k^{(i)} \xi^{(i)}$ is generally off-road and hence not applicable to our problem. However, if the estimate is chosen as minimizer of some cost function among all particles, that is $\xi^{(i)}$ with

$$i = \arg\min_i C(\xi^{(i)}), \tag{13}$$

the resulting estimate will be on-road. It can be computed by evaluating the cost function at all particles. Minimization of

$$C(\xi) = \sum_j w^{(j)} (\xi - \xi^{(j)})^T (\xi - \xi^{(j)}) \tag{14}$$

yields the true expected value for many particles in the absence of road constraints. With road constraints present, $\xi^{(i)}$ found by (13) provides a meaningful on-road expected value estimate and can be seen as constrained extension of the standard MMSE scheme. We denote the obtained estimate by $\hat{\xi}^{\mathsf{CMMSE}}$. Moreover, the cost function in (13) can be chosen arbitrarily and facilitates computation of alternative estimates, for instance based on $C(\xi) = \sum_j w^{(j)} \|\xi - \xi^{(j)}\|$.

V. Results and discussion

A. Road network and trajectory analysis

The considered road network consists of 830 nodes which are connected by 890 road segments. It covers an area of about two square kilometers.

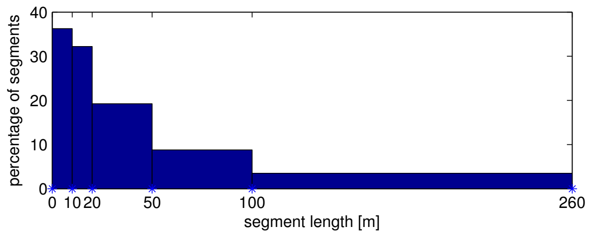

Figure 3. Road network data: normalized segment length histogram.

Figure 3 shows a normalized histogram of the segment lengths. It can be seen that about 36 per cent are less than 10 meters, about 32 per cent are between 10 and 20 meters in length. These short segments are likely to be skipped in case of a target that travels at higher speeds or large intervals between consecutive measurements. On the other end of the scale we see that more than 3 per cent of the segments are longer than 100 meters. These few long segments, however, amount to more than 4 kilometers of road. Turning to the amount of intersections in the road network we analyze the number of adjacent segments to each node. About 15 per cent of the nodes are intersections with 3 (12 per cent) and 4 (3 per cent) attached roads. A considerably large percentage of the nodes (5 per cent) are dead ends. This can be explained by boundary effects of the regarded network sector. Appropriate reactions to such a dead end road must be accounted for in the motion model to avoid strange behavior while filtering. The majority of nodes (80 per cent) has two road segments attached.

We next analyze the provided GPS trajectory which consists of $K = 140$ measurements that have been taken from a larger data set. From Figure 1 we saw already that it deviates from the road network. Figure 4 contains two normalized histograms of which the first displays the Euclidean distances between consecutive measurements. It can be seen that the increments are rather short with a mean of about 5 meters. The second histogram shows the times between consecutive sampling instants which range from 2.6 to 3.9 seconds. The target speed is low with a mean value of 2 meters per second.

B. Filtering and smoothing results

The filtering and smoothing algorithms of Section III have been implemented and tested with different settings. We present results that were obtained with $N = 500$ and $M = 100$ particles. Resampling in the PF was carried out whenever $N_{\mathsf{eff}} < \frac{1}{2}N$. The likelihood was chosen to be uniform on a circle with 25 meter radius. The driving noise parameter of (10) was set to $\mathsf{q} = 0.1$, and the corresponding v were chosen from a Student's t distribution [17] with degrees of freedom parameter $\nu = 3$. The fixed lag (FL) results have been obtained from the PF output with $L = 3$.

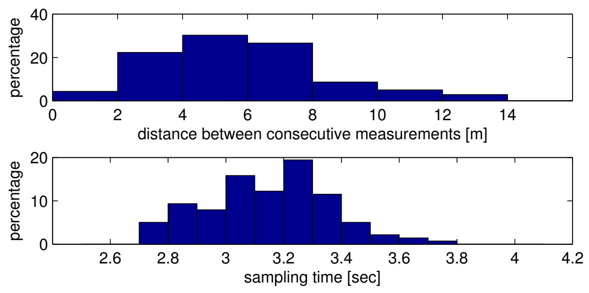

Figure 4. GPS trajectory: normalized histograms of distances between consecutive measurements, sampling times.

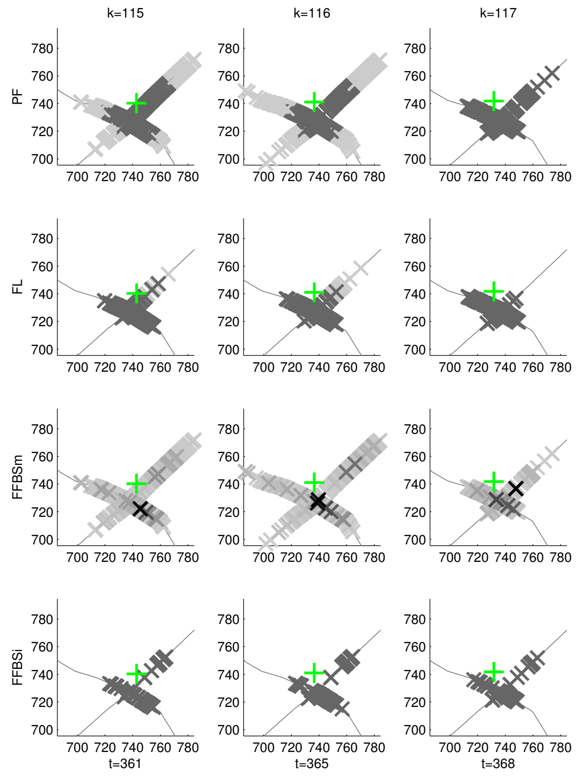

Figure 5. Particle clouds for the applied algorithms at a challenging road map sector; GPS measurement marked by green +; x and y in meter, times t in seconds.

The subplots in Figure 5 illustrate clouds of particles for three consecutive time steps at a challenging road map sector, see also Figure 1. Only the position components $\mathrm{x}_k^{(i)}$ and $\mathrm{y}_k^{(i)}$ are displayed as gray crosses. The color intensity corresponds to the particle weights. In all plots the GPS measurement is illustrated with a green +-marker, and clearly off-road. Time steps and corresponding times are provided for each column. The first row shows PF samples. As a result of the uniform likelihood, all particles in the 25 meter circle around the measurement have the same weight. In the last scan of the first row all samples have the same weight so resampling must have taken place. It can furthermore be seen how particles spread out on the correct and wrong road. The second row displays fixed lag smoothing (FL) samples which appear more focused around the measurement. Even for a small lag $L = 3$ fewer particles are found on wrong roads. The third row presents similar scans for the FFBSm algorithm. Relatively few samples are assigned significant weights (the darkest crosses), among them some on the wrong segment (e.g. third column). This can be explained by the fact that the algorithm computes the weight not only based on the illustrated position but also on the particle speed. FFBSi is illustrated in the last row. Here, only $M = 100$ samples are used. Although all particles carry the same weight, some of them are duplicates. Formed clusters in each scan reveal multi modality in the provided smoothing density.

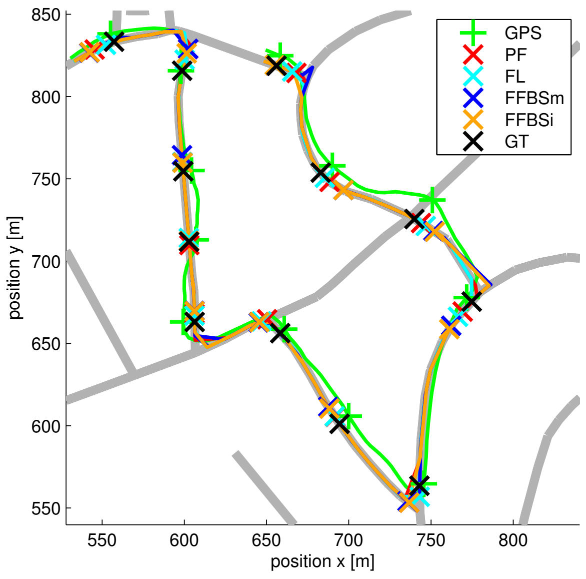

Figure 6. On-road trajectories and point estimates for a number of steps.

Position estimates were computed using the CMMSE scheme of Section IV-G. In Figure 6 they are plotted as solid lines, obviously aligned to the road map except for few cut corners. For selected k the estimates, along the GPS measurement and a manually projected position (ground truth GT), are given as markers. All algorithms managed to resolve the ambiguous situation that was highlighted in Section II. FFBSm and FFBSi briefly pick the wrong road segment in the rightmost turn. Also PF is subject to such erratic estimates, in other runs even to a larger extend. The manually projected data are generally ahead of all estimates and closest to them is the FL smoother. FFBSm and FFBSi lie often close to one another. The manually projected data (GT) were used to compute RMS

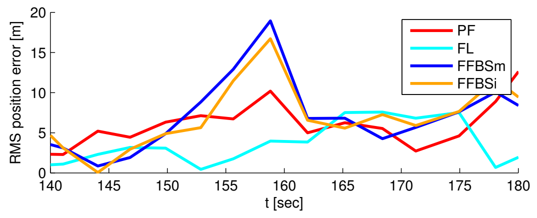

Figure 7. RMS position error computed from manually projected position data.

position errors which are illustrated in Figure 7. Averaged (over the entire trajectory) RMS position errors (in meter) are PF 7.9, FL 6.4, FFBSm 11.7, FFBSi 11.3.

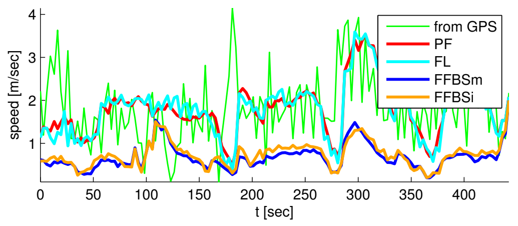

Figure 8. Speed computed from velocity estimates and differenced GPS data.

Velocity estimates ($\dot{\mathrm{x}}$ and $\dot{\mathrm{y}}$) are not constrained to the road map and can hence be computed as weighted particle means. Figure 8 displays the associated speed $\sqrt{\dot{\mathrm{x}}^2 + \dot{\mathrm{y}}^2}$ over time, including a signal that has been obtained by differencing the GPS data. The GPS speed is noise corrupted and thus likely to be higher than how fast the target actually traveled. PF and FL closely follow the GPS signal – their velocity is overestimated due to noise effects. FFBSm and FFBSi both provide slower velocity estimates which appear more realistic.

Motivated by the velocity results, we investigate FFBSm and FFBSi further and turn to computational aspects. The configuration $N = 500$ led to largely increased computation times for FFBSm (KN^2 operations). A lower N could be used, but this resulted in a depleted particle representation as too few weights turned out to be of significant size. Even for $N = 500$ the effective number of particles (4) is low for FFBSm, as illustrated in Figure 9. Furthermore, numerical problems occurred during FFBSm runs whenever the denominator in (5) turned out to be zero. Even for only few such occurrences the algorithm diverged. An adjustment in the likelihood and transition density implementations could circumvent these issues. Also, the use of wider tailed distributions for v reduced such effects. Turning to the FFBSi, an advantage is that the number of backwards particles M can be flexibly adjusted to reduce computational load (KNM operations, about KN in its fast implementation). For $M = 100$, the FFBSi computations were

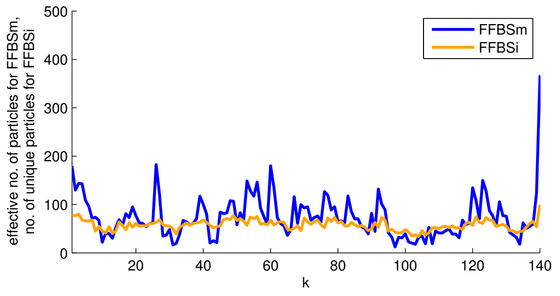

Figure 9. Effective number of particles for FFBSm, $N = 500$; number of unique particles for FFBSi, $N = 500$, $M = 100$.

much quicker than those of FFBSm and the performance as good, even in terms of particle diversity. A quantity that gives an indication of the diversity of unweighted samples in FFBSi is the number of unique particles and also illustrated in Figure 9.

The experiments show that the lookahead provided by smoothing can help resolve ambiguous situations. On average, however, no larger improvements of the PF output was achieved by smoothing. This is because PF already predicts its samples based on the motion model and inclusion of the transition densities provides little extra information for simple scenarios as for instance motion on a straight road. When compared to manually projected position data, both FFBSm and FFBSi are outperformed by a simple FL algorithm that, compared to PF, comes at no extra computational load. The manually projected data are, however, not necessarily the true trajectory that has been actually traveled – the error performance should thus be interpreted with care. If a smoothed velocity estimate is desired, the advanced smoothers present a more realistic picture as PF and FL tend to overestimate the target speed. From a computational point of view, FFBSi is to be preferred over FFBSm.

VI. Conclusion

We have shown how particle methods can be applied to obtain accurate position estimates comparable to ground truth data. We developed an advanced on-road motion model that can easily be extended to account for complex road network topologies. Furthermore, a way how the corresponding state transition density can be evaluated has been shown. GPS data were processed using a particle filter and smoothing algorithms were subsequently applied to the filter output. In ambiguous road topologies smoothing yields improved results. A fixed lag smoother was shown to be sufficient for position estimation, but overestimates the target speed. Improved velocity estimates were obtained by employing advanced smoothing algorithms (FFBSm and FFBSi). Among the two schemes, the particle forward filter backward simulator (FFBSi) seems to be the better choice justified by its computational complexity and flexibility (M not necessarily equal to N). The algorithms were successfully applied to real data.

Acknowledgment

This work has been supported by the MC Impulse (Monte Carlo based Innovative Management and Processing for an Unrivaled Leap in Sensor Exploitation) project, a European Commission FP7 Marie Curie Initial Training Network.

References

[1] F. Gustafsson, U. Orguner, T. B. Schon, P. Skoglar, and R. Karlsson, "Navigation and tracking of road-bound vehicles," in *Handbook of Intelligent Vehicles*, A. Eskandarian, Ed. Springer, Mar. 2012.

[2] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, Jan. 2004.

[3] U. Orguner, T. B. Schon, and F. Gustafsson, "Improved target tracking with road network information," in *2009 IEEE Aerospace conference*. IEEE, Mar. 2009, pp. 1–11.

[4] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, no. 2, pp. 107–113, Apr. 1993.

[5] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[6] O. Cappe, S. J. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 899–924, May 2007.

[7] F. Gustafsson, "Particle filter theory and practice with positioning applications," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 25, no. 7, pp. 53–82, 2010.

[8] ——, *Statistical Sensor Fusion*. Studentlitteratur AB, Mar. 2010.

[9] F. Lindsten, "Rao-Blackwellised particle methods for inference and identification," Licentiate Thesis no. 1480, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, Jun. 2011.

[10] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[11] S. J. Godsill, A. Doucet, and M. West, "Monte Carlo smoothing for nonlinear time series," *Journal of the American Statistical Association*, vol. 99, no. 465, pp. 156–168, 2004.

[12] R. Douc, "Sequential Monte Carlo smoothing for general state space hidden markov models," *The Annals of Applied Probability*, vol. 21, no. 6, pp. 2109–2145, Dec. 2011.

[13] H. Driessen and Y. Boers, "MAP estimation in particle filter tracking," in *2008 IET Seminar on Target Tracking and Data Fusion: Algorithms and Applications*. IET, Apr. 2008, pp. 41–45.

[14] M. Ulmke and W. Koch, "Road-map assisted ground moving target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 4, pp. 1264–1274, Oct. 2006.

[15] A. Bondy and U. Murty, *Graph Theory*, 3rd ed. Springer, Aug. 2008.

[16] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, 1st ed. Wiley-Interscience, Jun. 2001.

[17] S. Kotz and S. Nadarajah, *Multivariate t Distributions and their Applications*. Cambridge University Press, Feb. 2004.

[18] D. Streller, "Road map assisted ground target tracking," in *2008 11th International Conference on Information Fusion*. IEEE, Jul. 2008.