Long-term Vehicle Motion Prediction

Christoph Hermes*, Christian Wöhler†, Konrad Schenk† and Franz Kummert*,
*Bielefeld University, Faculty of Technology, Bielefeld, Germany
†Daimler AG Group Research, Environment Perception, Ulm, Germany

***Abstract*—Future driver assistance systems will have to cope with complex traffic situations, especially in the road crossing scenario. To detect potentially hazardous situations as early as possible, it is therefore desirable to know the position and motion of the ego-vehicle and vehicles around it for several seconds in advance. For this purpose, we propose in this study a long-term prediction approach based on a combined trajectory classification and particle filter framework. As a measure for the similarity between trajectories, we introduce the quaternion-based rotationally invariant longest common subsequence (QRLCS) metric. The trajectories are classified by a radial basis function (RBF) classifier with an architecture that is able to process trajectories of arbitrary non-uniform length. The particle filter framework simultaneously tracks and assesses a large number of motion hypotheses ($\sim 10^2$), where the class-specific probabilities estimated by the RBF classifier are used as a-priori probabilities for the hypotheses of the particle filter. The hypotheses are clustered with a mean-shift technique and are assigned a likelihood value. Motion prediction is performed based on the cluster centre with the highest likelihood. While traditional motion prediction based on curve radius and acceleration is inaccurate especially during turning manoeuvres, we show that our approach achieves a reasonable motion prediction even for long prediction intervals of 3 s for these complex motion patterns.**

I. Introduction

Future driver assistance systems will have to be able to interpret complex traffic situations, such as the road crossing scenario, and will therefore have to estimate and predict the position and motion state of the ego-vehicle and other vehicles around it over time intervals as long as several seconds. For example, such information will allow to assess the risk of an upcoming situation. This is especially important at road crossings, where potentially volatile situations often occur and have to be distinguished from normal situations.

An object prediction based on the current state (i.e. position, orientation, velocity) with the assumption of constant yaw angle and acceleration is insufficient for long prediction intervals because of different road shapes and driver behaviours. Humans are able to learn motion patterns and to predict the behaviour of traffic participants fairly accurately over time when the history of the moving objects is known. Our basic approach in this study is to adopt this capability of learning motion patterns and use them to estimate the future object position based on the motion history. For this reason we define a rotationally invariant distance metric for trajectories and use it both for an approximate direction guess provided by a radial basis function (RBF) network and for a finer prediction step where a particle filtering approach is applied. All methods are based on a set of labelled trajectories.

II. Related Work

To our knowledge, the problem of predicting a vehicle state several seconds into the future has not yet been addressed by many researchers. In [1] an explicit motion model is used to gain a stochastic threat assessment of road objects populating the environment. Similarly, in [2] a motion model is learned from given trajectories using a clustering technique to predict the object's behaviour.

In the field of body-tracking, motion prediction based on trajectories is more widely used. Echo State Networks are applied in [3] to obtain a motion prediction of a robotic system in local surroundings with main focus on real-time applications. In [4], a body-tracking system is presented where the motion is predicted with a particle filter approach based on a motion database. Similarly, a set of motion hypotheses is used in [5] to track a hand-forearm model of a worker in an industrial scenario.

A key to predict motion based on trajectories is an efficient matching technique to compare the vehicle history with the dataset. A lot of research has been done in this field, starting from different representations, e.g. polynomial descriptors [6], turning angle representation [7], and curve signatures [8]. Matching methods based on string matching techniques for raw trajectories (i.e. without change in representation) have been proven to be successful. Examples are Dynamic Time Warping [9], the Longest Common Subsequence [10], and the Levenshtein Distance on Trajectories [5]. We will further explore this idea in the following sections.

A concept strongly related to the topic of trajectory matching are classifiers applied to trajectories. Often a trajectory is converted into a feature representation, such that it can be processed by standard classifiers [11].

III. Trajectories

Motion patterns are the core issue of this study. We present them as trajectories X which consist of ordered tuples $X = ((\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N))$ combining states $\mathbf{x}_i$ with a time stamp t_i. Therefore each trajectory element $\mathbf{x}_i$ describes the current state of the tracked object over time. This representation does not depend on a specific sensor type, and parts of $\mathbf{x}_i$ may originate from different types of sensors. When applied to vehicle motion, the basic information is the object position in the 2D plane and the orientation angle relative to the ego-vehicle, here termed yaw angle. Furthermore, we found it useful to additionally encode the temporal derivatives, i.e. the velocity and the yaw rate, in the trajectory.

978-1-4244-3504-3/09/$25.00 ©2009 IEEE

A. Distance Metric on Trajectories

In this study, trajectories are compared to infer a prediction of the motion state. In order to find an appropriate matching method, we define the following requirements to the metric:

1) Handling of different sampling rates: Different sensor types run at different frequencies.
2) Insensitivity to outliers: Noisy data is likely to occur.
3) Different lengths of trajectories: Different motion patterns are not restricted to uniform time windows.
4) Translational invariance: Similar motion patterns do not depend on the starting point.
5) Rotational invariance: Similar motion patterns do not depend on direction, and their comparison needs to be independent of the observer's viewpoint.

The longest common subsequence (LCS) metric [10] on trajectories has been shown to be an adequate metric and can handle these stated requirements, except the last two issues. We thus adopted the LCS metric and applied the property of rotational invariance by using the method in [12] which finds the optimal orthogonal transformation to superimpose two point sets based on quaternions.

B. Longest Common Subsequence (LCS)

The LCS metric originates in the field of string matching algorithms and returns the length of the longest common substring matched by two strings. To apply this technique to trajectories, a similarity matching function between two states (points) $\mathbf{a}_i$ and $\mathbf{b}_j$ from the given trajectory points $(\mathbf{a}_i, t_i^{(a)}) \in A$ and $(\mathbf{b}_j, t_j^{(b)}) \in B$ has to be defined. Vlachos et al. [10] use the minimum standard deviation $\mathrm{std}_{min}^{(d)} = \min\{\mathrm{std}(A^{(d)}), \mathrm{std}(B^{(d)})\}$ in each dimension d as a decision boundary and apply a sigmoid function to smooth the distance value in the range $[0, \mathrm{std}_{min}^{(d)}]$. In our approach it is sufficient to use a linear function to obtain the distance between $\mathbf{a}_i$ and $\mathbf{b}_j$, where $L^1(\cdot)$ denotes the L1 norm (Manhattan distance):

$$\mathrm{dist}(\mathbf{a}_i, \mathbf{b}_j) = \begin{cases} 0 \textbf{ if } \exists d \in D\colon L^1(a_i^{(d)}, b_j^{(d)}) > \mathrm{std}_{min}^{(d)} \\ \frac{1}{D}\sum_{d=1}^{D}\left(1 - \frac{L^1(a_i^{(d)}, b_j^{(d)})}{\mathrm{std}_{min}^{(d)}}\right) \textbf{ otherwise} \end{cases} \tag{1}$$

The sizes of the trajectories A and B are denoted by N_A and N_B, respectively, corresponding to the number of motion states they comprise, and the sequence $[(\mathbf{a}_1, t_1^{(a)}), \ldots, (\mathbf{a}_{N_A-1}, t_{N_A-1}^{(a)})]$ by $\mathrm{head}(A)$. We then define the LCS on trajectories as follows:

$$\mathrm{LCS}(A,B) = \begin{cases} 0 \textbf{ if } N_A = 0 \ \wedge \ N_B = 0 \\ \mathrm{LCS}(\mathrm{head}(A), \mathrm{head}(B)) + \mathrm{dist}(\mathbf{a}_{N_A}, \mathbf{b}_{N_B}) \\ \qquad \textbf{if } \mathrm{dist}(\mathbf{a}_{N_A}, \mathbf{b}_{N_B}) \neq 0 \\ \max\{\mathrm{LCS}(\mathrm{head}(A), B), \mathrm{LCS}(A, \mathrm{head}(B))\} \\ \qquad \textbf{otherwise} \end{cases} \tag{2}$$

The distance between two trajectories A and B can then be obtained by

$$\mathrm{dist}_{\mathrm{LCS}}(A,B) = 1 - \frac{\mathrm{LCS}(A,B)}{\min\{N_A, N_B\}} \tag{3}$$

with $\mathrm{dist}_{\mathrm{LCS}}(A,B) \in [0,1]$.

C. Quaternion-based Rotationally Invariant LCS (QRLCS)

The LCS can be computed by the dynamic programming algorithm in $O(n^2)$ time using tables where partial (optimal) results of the algorithm are stored. We extend these optimal solutions by rotation and translation.

The best translation of a trajectory A to a trajectory B can be obtained using the mean values $\boldsymbol{\mu}_A^{(x,y)}$ and $\boldsymbol{\mu}_B^{(x,y)}$ of the two point sets in the xy plane. For the LCS dynamic programming algorithm an incremental version can be formulated according to

$$\boldsymbol{\mu}_{t,A}^{(x,y)} = \frac{1}{t}\sum_{i=1}^{t}\mathbf{a}_i^{(x,y)} = \frac{t-1}{t}\boldsymbol{\mu}_{t-1,A}^{(x,y)} + \frac{1}{t}\mathbf{a}_t^{(x,y)}. \tag{4}$$

Kearsley [12] uses quaternions to superimpose two point sets (in that case atoms from different molecules) by finding the best rotation when the corresponding point pairs are known. This idea can be applied to trajectories in a similar way: A quaternion can be considered as a row matrix of four numbers or the combination of a scalar with a 3D Cartesian vector, $Q = [q_1, q_2, q_3, q_4] \equiv [q_1, \mathbf{q}]$. The appropriate unit quaternion $\hat{Q}$ can be used as a rotation operator on a three-dimensional vector $\mathbf{x}$ as follows, where vectors are treated as quaternions with zero scalar component:

$$[0, \mathbf{x}^R] = \hat{Q}^{-1}[0, \mathbf{x}]\hat{Q} \tag{5}$$

It should be noted that a rotation quaternion can be constructed when the rotation angle α and the three-dimensional rotation vector $\mathbf{u}$ are known:

$$Q_{rot} = [\cos(\alpha/2), \sin(\alpha/2)\,\mathbf{u}] \tag{6}$$

When rotating a trajectory point $\mathbf{a}_i^{(x,y)}$ to match another point $\mathbf{b}_j^{(x,y)}$ there will be a residual quaternion $[0, \mathbf{e}] = [0, \mathbf{b}_j^{(x,y)}, 0] - \hat{Q}^{-1}[0, \mathbf{a}_i^{(x,y)}, 0]\hat{Q}$. This can be used to formulate a residual function $\varepsilon = |\hat{Q}|^2 \sum_k |\mathbf{e}|^2$ which computes the error over all associated point pairs $\{(\mathbf{a}_i^{(x,y)}, \mathbf{b}_j^{(x,y)})\}_k$. By minimising ε with respect to a given rotation vector $\mathbf{u} = [0,0,1]$ perpendicular to the ground plane, the problem reduces to an eigenvalue problem where

$$x_{m,k} = (b_{j,k}^{(x)} - \mu_B^{(x)}) - (a_{i,k}^{(x)} - \mu_A^{(x)}) \tag{7}$$

$$x_{p,k} = (b_{j,k}^{(x)} - \mu_B^{(x)}) + (a_{i,k}^{(x)} - \mu_A^{(x)}) \tag{8}$$

with similar definitions for $y_{m,k}$ and $y_{p,k}$:

$$\begin{pmatrix} \sum_k(x_{m,k}^2 + y_{m,k}^2) & \sum_k(x_{p,k}y_{m,k} - x_{m,k}y_{p,k}) \\ \sum_k(x_{p,k}y_{m,k} - x_{m,k}y_{p,k}) & \sum_k(x_{p,k}^2 + y_{p,k}^2) \end{pmatrix} \begin{pmatrix} q_1 \\ q_4 \end{pmatrix} = \lambda \begin{pmatrix} q_1 \\ q_4 \end{pmatrix}$$

The two eigenvectors of this symmetric matrix incorporate the first and the last component of a unit quaternion $\hat{Q}_{rot} = [q_1, 0, 0, q_4]$. The smallest and largest eigenvalues of the corresponding eigenvectors represent rotations that minimise and

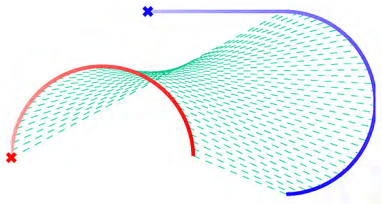


Fig. 1. Matching of two trajectories in the xy plane by QRLCS. A cross depicts the starting point of each trajectory. The red trajectory is translated and rotated to the blue trajectory.

maximise the sum of the mutual distances between all assigned trajectory points.

In the LCS dynamic programming algorithm every trajectory point pair is tested whether its distance value is below the standard deviation threshold (see Eq. 1). If this condition is met, the distance value influences the LCS value. For the rotation and translation this results in computing the complete matrix each time a new trajectory point pair is added to give the best partial solution. To tackle this problem in the dynamic programming way, we suggest storing the partial values of the matrix from the last $(K-1)$ point assignments and simply update the matrix (sub-)elements when a new point pair is found to be similar:

$$\sum_{k=1}^{K} x_{m,k}^2 = \gamma_{K-1}^{(1,x)} + (b_{j,K}^{(x)} - a_{i,K}^{(x)})^2 - K(\mu_{K,B}^{(x)} - \mu_{K,A}^{(x)})^2$$

$$\sum_{k=1}^{K} (x_{p,k}y_{m,k} - x_{m,k}y_{p,k}) = 2\gamma_{K-1}^{(2)} + 2(a_{i,K}^{(x)}b_{j,K}^{(y)} - b_{j,K}^{(x)}a_{i,K}^{(y)}) + 2K(\mu_{K,A}^{(y)}\mu_{K,B}^{(x)} - \mu_{K,B}^{(y)}\mu_{K,B}^{(x)})$$

$$\sum_{k=1}^{K} x_{p,k}^2 = \gamma_{K-1}^{(3,x)} + (b_{j,K}^{(x)} + a_{i,K}^{(x)})^2 - K(\mu_{K,B}^{(x)} + \mu_{K,A}^{(x)})^2$$

where

$$\gamma_K^{(1,x)} = \sum_{k=1}^{K} (b_{j,K}^{(x)} - a_{i,K}^{(x)})^2 \tag{9}$$

$$\gamma_K^{(2)} = \sum_{k=1}^{K} (a_{i,K}^{(x)}b_{j,K}^{(y)} - b_{j,K}^{(x)}a_{i,K}^{(y)}) \tag{10}$$

$$\gamma_K^{(3,x)} = \sum_{k=1}^{K} (b_{j,K}^{(x)} + a_{i,K}^{(x)})^2 \tag{11}$$

The matrix sub-elements $\sum_{k=1}^{K} y_{m,k}^2$ and $\sum_{k=1}^{K} y_{p,k}^2$ can be computed in a similar way based on the corresponding values $\gamma_K^{(1,y)}$ and $\gamma_K^{(3,y)}$, respectively.

Fig. 1 shows an example with two test trajectories. The assignments are recorded during QRLCS computation. At each assignment test in the dynamic programming algorithm, the values of $\boldsymbol{\mu}_{K,A}^{(x,y)}$, $\boldsymbol{\mu}_{K,B}^{(x,y)}$, $\gamma_K^{(1,\{x,y\})}$, $\gamma_K^{(2)}$, and $\gamma_K^{(3,\{x,y\})}$ are stored to obtain the best partial solution, similar to the partial LCS values.

IV. Trajectory Classification

A straightforward approach to the classification of trajectories consists of treating the first (i.e. most significant) n coefficients of the Chebyshev decomposition [13] of the velocity and yaw angle components of each trajectory as a feature vector of length n. These vectors are used as input to a standard classifier architecture. For this purpose, we applied the well-known second-order polynomial classifier [14] with ideal output vectors $\boldsymbol{\tau} = [1, 0]$ for trajectories of class "turn left" and $\boldsymbol{\tau} = [0, 1]$ for trajectories of class "straight on". This straightforward feature extraction and classification approach, however, turned out to be of limited value since a turn-left manoeuvre is often not recognised as such before it has almost been completed (cf. Section VI).

An alternative classification approach is to directly utilise the QRLCS distance metric within the framework of a radial basis function (RBF) network classifier [14]. The RBFs are radially symmetric Gaussians according to

$$\omega_p(A) = \exp\left[-\eta_p \left(\mathrm{dist}_{\mathrm{LCS}}(A, C_p)\right)^2\right] \tag{12}$$

with $\mathrm{dist}_{\mathrm{LCS}}(\cdot)$ as the QRLCS distance metric defined in Eq. 3, C_p as the p-th prototype of the RBF network, and η_p as the width parameter of the p-th RBF. The number of classes is given by N_c, the overall number of prototypes and thus RBFs by $N_P = cN_c$, and each class is represented by the same number c of prototypes. The output vector $\boldsymbol{\delta}(A)$ of the RBF network for an input pattern A is then given by

$$\boldsymbol{\delta}(A) = W \cdot \boldsymbol{\Omega}(A) \tag{13}$$

with $\boldsymbol{\Omega}(A)$ as the vector of length N_P made up by the RBF values $\omega_p(A)$ and W as a $N_c \times N_P$ weight matrix. We assume a training set consisting of M samples. Each sample has an associated desired output vector $\boldsymbol{\tau}_m$ as defined above for the polynomial classifier and a corresponding RBF output vector $\boldsymbol{\delta}_m$. The RBF network is trained by minimising the error function

$$E_{\mathrm{RBF}} = \sum_{m=1}^{M} \|\boldsymbol{\delta}_m - \boldsymbol{\tau}_m\|^2 \tag{14}$$

with respect to the elements W_{kp} of the weight matrix W, the width parameters η_p, and the prototypes C_p. Minimisation with respect to W_{kp} and η_p is performed based on the gradient descent technique. Generally each training sample has a different length, such that no feature space of a given dimension exists in which each point may in principle represent a RBF prototype, and the prototypes cannot be determined based on a standard continuous optimisation technique. Hence, we restrict the space of possible prototypes to the training samples themselves, determining the set of N_P prototypes that minimise the error function E_{RBF} according to Eq. 14 by an exhaustive search over the training set after each gradient descent step for W_{kp} and η_p.

V. Motion Prediction

In this study, we implement motion prediction as a probabilistic tracking framework. Given a history of measurements, i.e. a trajectory $M_{1:t}$ up to a current time t, we intend to predict the object state $\boldsymbol{\phi}_T$ at a specific point in time T in the

future. The uncertainty of this prediction can be formulated as a distribution $p(\phi_T|M_{1:t})$, which is rewritten as

$$p(\phi_T|M_{1:t}) = p(\phi_T|\Psi_t)\, p(\Psi_t|M_{1:t}), \tag{15}$$

where we have incorporated the current object state Ψ_t.

The distribution $p(\phi_T|\Psi_t)$ is the likelihood of observing the predicted state ϕ_T when the current object state Ψ_t is given. This can be roughly approximated by the output of a classifier which returns the mapped class contributions as a likelihood of being a turn-left or a straight-on trajectory when the sub-trajectory is given.

In Eq. 15 the current state Ψ_t is taken into account. In the context of trajectories, it represents a sequence of trajectory points (a sub-trajectory) including the position at the current time t and its history over a given travelled distance d. We choose a distance window instead of a time window because the characteristics of vehicle motion are represented by the travelled distance, while the motion history especially of objects which are standing or moving slowly may be less meaningful when integrated over a uniform time interval.

Applying the Bayes rule on the remaining distribution $p(\Psi_t|M_{1:t})$ results in an estimation of the current state based on the current measurement and the previous states according to

$$p(\phi_T|M_{1:t}) = \eta\, p(\phi_T|\Psi_t)\, p(M_{1:t}|\Psi_t) \int p(\Psi_t|\Psi_{t-1})\, p(\Psi_{t-1}|M_{1:t-1})\, d\Psi_{t-1} \tag{16}$$

with η as a normalisation constant. This distribution is represented by a set of samples or particles $\{\Psi_t^{(s)}\}_S$, which are propagated in time using a particle filter [15]. Therefore, each particle $\Psi_t^{(s)}$ represents a sub-trajectory for the current state. The distribution $p(M_{1:t}|\Psi_t)$ represents the likelihood that the measurement trajectory $M_{1:t}$ can be observed when the model trajectory is given; it can be obtained by the QRLCS metric. According to [4], it is sufficient to sample the particles from the distribution $p(\Psi_t|\Psi_{t-1})$ from a motion database as follows, resulting in an efficient implicit probabilistic motion model. In a first step, the trajectory database is structured by creating samples with overlapping windows of equal travelled distances d. Because this procedure creates sub-trajectories with different numbers of points, we applied the Chebyshev decomposition [13] to the velocity and yaw angle components of the trajectories to obtain a vector of Chebyshev coefficients $[\mathbf{c}_v, \mathbf{c}_a]$ for the velocity and the yaw angle, respectively. Following the approach in [4], a dimensionality reduction is performed using principal component analysis (PCA) [14]. The particles are also transformed to this low-dimensional coefficient space.

The database of samples is then structured into a binary tree using the previously determined coefficients. The top node in the tree corresponds to the coefficient that captures the dimension of largest variance in the database, where lower levels capture the finer motion structure. At each level l a sub-trajectory i is assigned to the left sub-tree when its coefficient $c_{i,l} < 0$ and assigned to the right one if $c_{i,l} \geq 0$. Each of the leaf nodes contains an index into the motion database. This concept is depicted in Fig. 2. As noted in [4], a balanced tree is constructed.

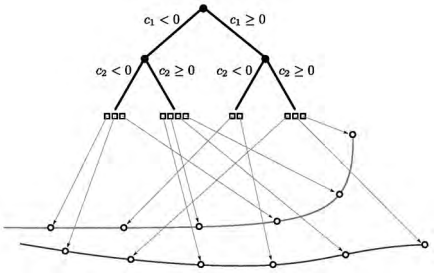


Fig. 2. Binary tree construction out of two sample trajectories. Each leaf has a list of pointers to the appropriate location on the trajectory. These samples overlap to obtain a finer scanning.

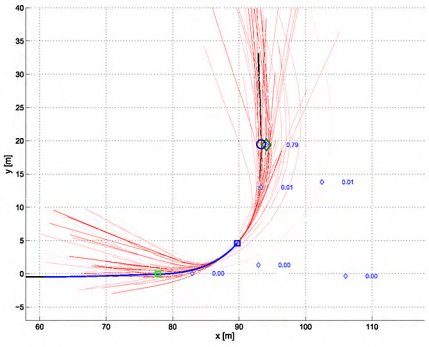


Fig. 3. Predicted states: The blue line depicts the motion history with the estimated current object position (blue square). The green square indicates the point on the trajectory where the curve radius gets below 200 m. The red lines denote the trajectories consistent with the motion history, where the strength of the colour corresponds to the weight value. The resulting hypotheses (two seconds prediction interval) obtained by the mean-shift method are shown as diamonds, where the largest symbol denotes the most probable position. The ground truth is shown as a circle.

Sidenbladh et al. [4] argue that sampling particles from the state transition distribution $p(\Psi_t|\Psi_{t-1})$ can be approximated by a probabilistic search in the database. When a particle reaches a leaf, the prediction step is performed by shifting the particle (i.e. the sub-trajectory) with the appropriate time over the trajectory to which the leaf points. The probabilistic search depends on the particle represented by its PCA-transformed Chebyshev coefficients $\mathbf{c}_i$. At each level l in the binary tree it is decided with the probability

$$p_{\text{right}} = p(c_{t,l} \geq 0 | c_{i,l}) = \frac{1}{\sqrt{2\pi\beta}\sigma_l} \int_{z=-\infty}^{c_{i,l}} e^{-\frac{z^2}{2\beta\sigma_l^2}}\, dz \tag{17}$$

whether the particle is moved to the right subtree, otherwise the left one is chosen. The value β is a temperature parameter describing the spreading deviation around each particle $\mathbf{c}_i$. The higher the value of β, the more likely it is that new regions of interest are explored. The variances σ_l^2 are normalisation factors and correspond to the eigenvalues of the covariance matrix computed for determining the PCA of the Chebyshev coefficients.

Since the distribution of the predicted states $p(\phi_T|M_{1:t})$ is approximated by means of the particle filter, the estimated

states $\boldsymbol{\phi}_T^{(s)}$ can be obtained by looking ahead for a specific time interval ΔT from the current object states $\Psi_t^{(s)}$ on the associated trajectories. This results in many hypotheses which often lie closely together. To condense this set into a small number of hypotheses, we apply a mean shift method [16], [17]. The key idea is to estimate the local densities of the predicted states $\boldsymbol{\phi}_T^{(s)}$ by constructing a kernel over each state and then to shift the states iteratively towards higher densities with the mean shift vector

$$\mathbf{m}_{h,g}(\boldsymbol{\phi}_T^{(s)}) = \frac{\sum_{i=1}^{S} \boldsymbol{\phi}_T^{(i)} w^{(s)}\, g\left(\left\|\frac{\boldsymbol{\phi}_T^{(s)}-\boldsymbol{\phi}_T^{(i)}}{h}\right\|^2\right)}{\sum_{i=1}^{S} w^{(s)}\, g\left(\left\|\frac{\boldsymbol{\phi}_T^{(s)}-\boldsymbol{\phi}_T^{(i)}}{h}\right\|^2\right)} - \boldsymbol{\phi}_T^{(s)}, \tag{18}$$

where $w^{(s)} = p(\boldsymbol{\phi}_T^{(s)}|M_{1:t})$ denotes the weight of a particle, $g(x) = -G'(x)$ is the derivative of a kernel function $G(x)$, and h is the kernel width. In this work we use a Gaussian kernel $G(x) = \exp\left[-\frac{1}{2}x^2\right]$. The shifting procedure is repeated until no particle moves any longer. Then the inferred cluster centres are reweighted according to the particle weights of their members. The result of this method are several weighted hypotheses of the predicted state for each time step. The hypothesis with the highest weight is taken as the final prediction state. An example is shown in Fig. 3.

VI. Results

For evaluation of the proposed method, we use a data set consisting of vehicle odometry data, where the trajectories are given as velocity and yaw rate values at discrete time steps. In order to retrieve the time-dependent position these values are integrated over time. To examine the ability of the system to distinguish between different vehicle manoeuvres, a data set containing turn-left manoeuvres and straight-on drives performed by eight different drivers on five different road crossings was recorded, resulting in 938 trajectories.

For comparison with the proposed method, we simultaneously apply a standard extrapolation technique assuming constant acceleration and curve radius with respect to the current vehicle state. All prediction results are compared to the ground truth provided by the odometry data. Besides the position error, we also state the errors for the velocity, yaw angle, and yaw rate, as they represent important characteristics of the predicted vehicle state.

Training of the polynomial and the RBF classifier as well as the construction of the trajectory database for particle filtering is performed using the trajectories obtained on four road crossings, leaving one crossing out for testing. For the polynomial classifier, the rate of correctly classified trajectories is higher than 99% on the test set. However, all "turn left" trajectories in the training set and the test set display full turning manoeuvres, which reduces the distinction between them and the "straight on" trajectories to a fairly easy problem. Keeping in mind the actual goal of motion prediction, an early recognition of the turning manoeuvre is highly desirable. In this context, we found that if a "turn left" trajectory that develops over time is

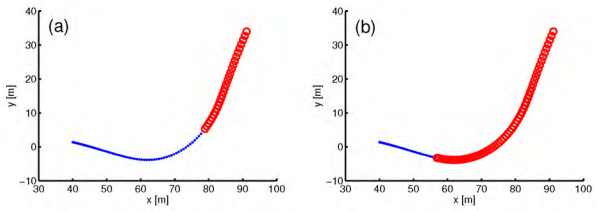


Fig. 4. Example of trajectory classification (left: polynomial classifier on Chebyshev coefficients; right: RBF classifier) with two classes "straight on" (blue dots) and "turn left" (red circles).

analysed by the polynomial classifier at each time step, it is not recognised as such before reaching the maximal yaw rate, i.e. when the turning manoeuvre has almost been completed (cf. Fig. 4a). Adding partial trajectories of the "turn left" class to the training set, displaying a straight-on phase and the beginning of a turning manoeuvre, leads to contradictory labels for similar feature vectors and thus to the degradation of classifier performance.

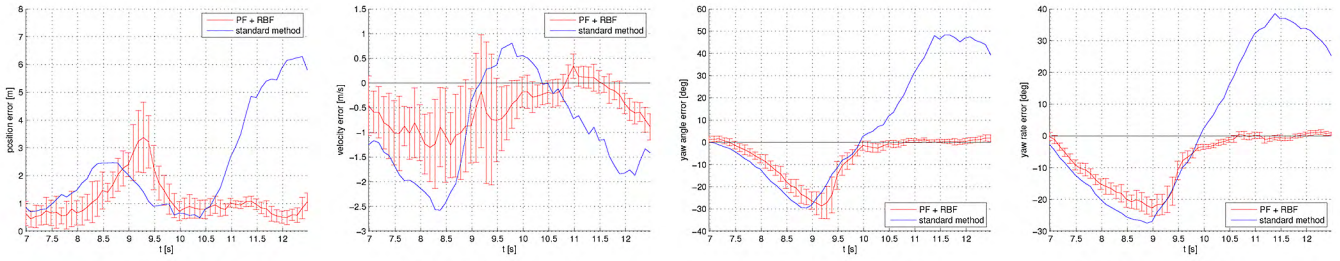
We found that the RBF classifier in combination with the QRLCS metric recognises a turning manoeuvre much earlier than the quadratic polynomial classifier applied to the Chebyshev coefficients (cf. Fig. 4b) for the result of a RBF architecture with $c = 3$ prototypes per class). At the same time, the rate of correctly classified trajectories remains higher than 99% on the test set.

In the particle filter system we use $S = 200$ particles, 50 Chebyshev coefficients in each dimension, $c = 3$ RBF prototypes per class, a temperature parameter of $\beta = 0.8$, a travelled distance of $d = 30$ m and a mean-shift kernel width of $h = 4.0$ m.

Since the particle filter is a probabilistic approach, each run on the same trajectory will give slightly different results. To examine if the standard deviation of the prediction result remains reasonably small, the turning part of a single test trajectory (cf. Fig. 3) was regarded and the particle filter system was run 30 times for a prediction interval of 2 s. The resulting average prediction error and its standard deviation are shown in Fig. 5 along with the prediction error of the standard model. The prediction behaviour of the particle filter approach is superior during the turning manoeuvre, where the prediction of the standard model displays gross errors especially for the yaw angle and the yaw rate.

Furthermore, we have evaluated the results for prediction intervals of 1, 2, and 3 s for the particle filter system and the reference prediction model. The corresponding root-mean-square prediction errors are given in Table I. The prediction accuracy of the particle filter approach is significantly higher than that of the standard model especially for the yaw angle and the yaw rate even for a short prediction interval of 1 s.

Table II shows the RMSE on a set of 24 test trajectories acquired by eight different drivers. Results are given for the standard prediction model (M) and the particle filter without classifier (PF), with polynomial classifier (PF+PC), and with RBF classifier (PF+RBF), where the prediction interval is 3 s.



(a) Position error (b) Velocity error (c) Yaw angle error (d) Yaw rate error

Fig. 5. Errors and standard deviations over time for a turning manoeuvre at a single test trajectory (cf. Fig. 3), prediction interval 2 s, 30 runs of the particle filter with RBF classifier.

TABLE I
RMSE FOR DIFFERENT PREDICTION INTERVALS FOR THE TRAJECTORY AND THE PERIOD OF TIME REGARDED IN FIG. 5. PF+RBF: PARTICLE FILTER COMBINED WITH RBF CLASSIFIER; M: STANDARD MODEL.

		position [m]	velocity [m/s]	yaw angle [deg]	yaw rate [deg/s]
1 s	PF+RBF	0.7±0.0	0.4±0.0	3.6±0.4	6.5±0.4
	M	0.4	0.5	6.7	12.3
2 s	PF+RBF	1.4±0.1	0.9±0.1	11.5±1.1	11.1±0.7
	M	3.0	1.4	27.1	23.7
3 s	PF+RBF	5.0±0.6	2.4±0.2	27.2±1.6	15.8±0.5
	M	6.3	2.2	45.6	27.4

TABLE II
RMSE FOR 24 TEST TRAJECTORIES ACQUIRED BY EIGHT DIFFERENT DRIVERS AT A CROSSING NOT CONTAINED IN THE TRAINING SET.

		position [m]	velocity [m/s]	yaw angle [deg]	yaw rate [deg/s]
PF	straight	7.5±2.6	3.0±1.1	28.4±12.7	18.3±6.7
	turn	7.6±2.0	1.8±0.7	30.2±11.7	10.3±2.7
PF+PC	straight	10.9±3.8	4.5±1.3	29.9±14.1	19.1±6.7
	turn	12.1±5.1	3.6±1.7	47.4±14.9	12.1±2.6
PF+RBF	straight	7.6±3.0	3.0±1.3	27.8±11.0	18.3±6.9
	turn	7.7±2.1	1.8±0.8	30.0±11.9	10.3±3.0
M	straight	4.3±1.3	2.3±0.9	28.6±10.7	18.9±6.8
	turn	10.2±2.7	4.0±1.6	63.3±23.2	27.3±10.7

The label "turn" refers to the turning manoeuvre phase of each trajectory, which we defined to begin when the curve radius gets below 200 m. Using PF+PC leads to an unfavourable prediction performance. In the straight parts of the trajectories, the position and velocity errors of the standard model are slightly lower than those of the PF configurations, while the yaw angle and yaw rate errors are similar. During the turning manoeuvres, the prediction performance of PF+RBF is strongly superior to that of the standard model. It is similar to that of the PF alone, but specific information about the moment in time when a turning manoeuvre begins is provided.

VII. CONCLUSION

In this study we have presented a method that predicts the motion state of vehicles several seconds into the future based on trajectory classification and simultaneous tracking of multiple hypotheses with a particle filter framework. The classifier output is used as a probabilistic filter for the particle filter based prediction part. Compared to a traditional constant acceleration and curve radius prediction model, the accuracy of the proposed particle filter approach is superior during turning manoeuvres displaying complex motion patterns.

REFERENCES

[1] A. Eidehall and L. Petersson, "Statistical threat assessment for general road scenes using monte carlo sampling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 137–147, 2008.

[2] F. Large, D. A. Vasquez Govea, T. Fraichard, and C. Laugier, "Avoiding cars and pedestrians using velocity obstacles and motion prediction," in *Proc. of the IEEE Intelligent Vehicle Symp.*, June 2004.

[3] S. Hellbach, S. Strauss, J. Eggert, E. Körner, and H.-M. Gross, "Echo state networks for online prediction of movement data - comparing investigations." in *ICANN (1)*, ser. Lecture Notes in Comput. Sci., vol. 5163. Springer, 2008, pp. 710–719.

[4] H. Sidenbladh, M. J. Black, and L. Sigal, "Implicit probabilistic models of human motion for synthesis and tracking," in *ECCV*. London, UK: Springer-Verlag, 2002, pp. 784–800.

[5] M. Hahn, L. Krüger, and C. Wöhler, "3d action recognition and long-term prediction of human motion." in *ICVS*, ser. Lecture Notes in Computer Science, A. Gasteratos, M. Vincze, and J. K. Tsotsos, Eds., vol. 5008. Springer, 2008, pp. 23–32.

[6] A. Naftel and S. Khalid, "Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space," *Multimedia Systems*, vol. 12, no. 3, pp. 227–238, December 2006.

[7] S. D. Cohen and L. J. Guibas, "Partial matching of planar polylines under similarity transformations," in *Symp. on Discr. Algorithms*, 1997.

[8] A. Croitoru, P. Agouris, and A. Stefanidis, "3d trajectory matching by pose normalization," in *Proc. of the 13th annual ACM intl. workshop on GIS*. New York, NY, USA: ACM Press, 2005, pp. 153–162.

[9] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. of KDD*, Seattle, Washington, Jul. 1994, pp. 359–370.

[10] M. Vlachos, G. Kollios, and D. Gunopulos, "Elastic translation invariant matching of trajectories," *M.. Learn.*, vol. 58, no. 2-3, pp. 301–334, '05.

[11] X. Li, W. Hu, and W. Hu, "A coarse-to-fine strategy for vehicle motion trajectory clustering," in *ICPR: Proc. of the 18th ICPR*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 591–594.

[12] S. K. Kearsley, "On the orthogonal transformation used for structural comparisons," *Acta Cryst.*, vol. A45, pp. 208–210, 1989.

[13] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge University Press, 1992, ch. 5.8 Chebyshev Approximation, pp. 190–194.

[14] J. Schürmann, *Pattern classification: a unified view of statistical and neural approaches*. NY, USA: John Wiley & Sons, Inc., 1996.

[15] M. J. Black and A. D. Jepson, "A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions," in *ECCV*. London, UK: Springer, 1998, pp. 909–924.

[16] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on PAMI*, vol. 24, no. 5, pp. 603–619, 2002.

[17] J. Schmidt, J. Fritsch, and B. Kwolek, "Kernel particle filter for real-time 3d body tracking in monocular color images," in *Proc. of the 7th Intl. Conf. on Automatic Face and Gesture Recognition*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 567–572.