A PARTICLE FILTER BASED SEQUENTIAL TRAJECTORY CLASSIFIER FOR BEHAVIOR ANALYSIS IN VIDEO SURVEILLANCE

Vahid Bastani, Lucio Marcenaro, Carlo Regazzoni

Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture (DITEN), University of Genova, Genoa - Italy

ABSTRACT
The problem of behavior assessment in video surveillance is approached using trajectory classification. Lagrangian state dynamic is used for probabilistic modeling of trajectory patterns and an off-line parameter learning method for the model is proposed. For classification purpose, an on-line sequential maximum a posterior trajectory classifier is introduced based on particle filter. Finally, the performance of this method is evaluated using a traffic video data set.

Index Terms— On-line Trajectory Classification, Behavior Analysis, Video Surveillance

1. INTRODUCTION
In surveillance applications, actions of objects are represented by their motion pattern. Thus it is essential for an intelligent surveillance system to classify trajectory patterns in order to detect events and discriminate between different behaviors [1]. Hence, an increasing amount of research has been devoted recently to trajectory modeling and analysis [2], [3], [4], [5]. This problem is often divided in three sub-problems: trajectory representation, distance measurement and clustering [6] which may be combined in a single model [7].

Recently, the Lagrangian function (LF) has been proposed [8] to model generalized state motion of a cognitive dynamic system (CDS). In this paper, the LF state dynamic model is used as trajectory descriptor. This is justified by the fact that the objects which are under observation in video surveillance are usually of the CDS type, since they are often either human or vehicles driven by human operator. In Sec.2 the idea of using LFs to model trajectories is introduced and in Sec.3 a method for learning the parameters of this model is described.

In [9], [10] and [11] a Bayesian method for supervised learning and classification of behavior using trajectory analysis has been proposed. The method is based on the division of the environment in zones and learning trajectories as transitions between these zones. Despite reasonable performance, this technique is not able to sequentially assess the trajectory while the object is in one zone. To overcome this limitation, in this paper a method is proposed that can sequentially improve the belief about the true trajectory class as new measurement is received from sensors (e.g. video camera). This is done using a dynamic Bayesian network model and a particle filter based algorithm for multilevel Bayesian processing that is proposed in Sec.4.

2. LAGRANGIAN STATE DYNAMICS OF COGNITIVE DYNAMIC SYSTEM
Lagrangian mechanics is an alternative formulation of classical equations of motion where the entire behavior of the system is summarized in a single LF, which is the difference of kinetic and potential energy [12] of the system. Taking inspiration from the more general approach proposed in [8] to analyze neural inference processes, in this paper, we aim to model the motion of a cognitive object using LFs.

For a single point-like unit-mass object moving in Cartesian plane, the state at the time \( t \) in the phase space is defined as \( x(t) = [x_1(t), \dot{x}_1(t), x_2(t), \dot{x}_2(t)]^T \) where \( x_1 \) and \( x_2 \) are positions and \( \dot{x}_1 \) and \( \dot{x}_2 \) are velocities in the first and second axes respectively. We also introduce the position-only state \( x_p(t) = [x_1(t), x_2(t)]^T \) for further use. We may drop the time variable in some equations for simplicity.

The object at each time instance may chooses between \( K \) different possible behaviors indicated by the variable \( r(t) \in \mathcal{K} = \{1, \cdots, K\} \). Using Lagrangian dynamics, the motion of its state in discrete time is

\[
x(t + \delta) = x(t) + (Q - \Gamma)\nabla L_r(t)(x(t)),
\]

where \( \delta \) is the time difference, \( \nabla \) is the gradient operator, \( L_r(x) \) is the LF corresponding to behavior index \( r \), \( Q \) is an antisymmetric matrix that is defined by the physics of the problem and \( \Gamma \) is a matrix that accounts for fluctuations due to non-conservative forces, which are in our problem defined as

\[
Q = \begin{bmatrix}
0 & \delta \\
-\delta & 0
\end{bmatrix}, \quad \Gamma = \begin{bmatrix}
0 & \kappa \delta^2/2 \\
0 & \kappa \delta
\end{bmatrix}
\]

using a friction constant \( \kappa \). For the rest of the paper we use \( t \) as discrete time variable which corresponds to time of observation or frame index in the video sequence.
The kinetic energy part of the LF in Eq.1 depends on the velocity of the object. Thus, the LFs are characterized by their corresponding potential function (PF), \( L_r(x) = \frac{1}{2}(\dot{x}_1^2 + \dot{x}_2^2) \), where the PF \( \phi_r(x_p) \) is a function of position. The sequence \( r(t) \), which represent the internal will or motivation of the moving cognitive object, is a Markov chain independent of \( x(t) \). Therefore, for an object traversing trajectory pattern \( m \in M = \{1, \cdots, M\} \) the dynamic of \( r(t) \) is characterized by Markov model transition matrix \( \Pi^m \) whose elements are the probabilities of jump from one LF to another at successive time instances.

Overall, in this framework the trajectory patterns are identified using the set of PFS \( \{\phi_r(.)\}_{r=1}^K \) and set of motivation transition matrices \( \{\Pi^m\}_{m=1}^M \). However, in order to perform sequential MAP classification of trajectories the set of initial state probability distributions of each trajectory pattern \( \{p(x(0)|m)\}_{m=1}^M \) and prior probabilities \( \{P(m)\}_{m=1}^M \) should also be known. In the next section methods to approximate the PFS and learning other required parameters are described.

3. POTENTIAL FUNCTIONS APPROXIMATION AND TRAJECTORY LEARNING

3.1. Local Motion Models

We use piecewise affine functions to approximate the PFS. Let \( \{\mathcal{P}_n\}_{n=1}^N \) be a set of mutually exclusive and exhaustive partition of the position space such that in each partition the shape of PFS are approximately linear and the segments of all PFS that fall in same partition are called local motion models (LMM). In addition, we take advantage of local similarities of PFS to reduce the number of LMMs that should be learned. The \( k \)th LMM exists in partition \( \mathcal{P}_n \) is represented by

\[
\varphi_{k_n}(x_p) = a_{k_n}^T x_p + c_{k_n},
\]

where \( a_{k_n} \) is the slope and \( c_{k_n} \) is the inception. Note that we are only concerned by \( a_{k_n} \) because \( c_{k_n} \) will disappear due to the gradient operation in Eq.1. Then, the approximation of a PF in terms of LMMs is

\[
\phi_r(x_p) \approx \sum_{n=1}^N \lambda_n(x_p) \varphi_{k_n}(r)(x_p),
\]

where \( \lambda_n(x_p) \) is a window function in position space that equals one when \( x_p \in \mathcal{P}_n \) and zero otherwise. \( l_n(r) \) is a mapping that corresponds appropriate LMM in partition \( \mathcal{P}_n \) to PF index \( r \). It is worth noting that the slope vector in Eq.4 is exactly the acceleration of the object while it is in the partition \( \mathcal{P}_n \) and it moving according to The \( k \)th LMM of that partition.

For learning purpose, the existing LMMs inside each zone are modeled by a Gaussian mixture model on the slope vector

\[
a_{k_n} \sim \sum_{c=1}^{K_c} \rho_c^n \mathcal{N}(\mu_c^n, \Sigma_c^n)
\]

where the number of mixture components \( K_c \), mixture weight \( \rho_c^n \), mean vector \( \mu_c^n \) and covariance matrix \( \Sigma_c^n \) of components are learned using a Dirichlet process mixture model (DPMM) [13]. As a result, it is not necessary to explicitly define the number of LMMs because it is estimated within the learning algorithm of the DPMM.

The training data is given as a set of position samples of trajectories and a label \( L \in M \) specifying the class of the trajectory pattern \( D = \{(x_p^{(d)}(0 : t_d), L^{(d)})\}_{d=1}^D \). We start with choosing an arbitrary partition \( \{\mathcal{P}_n\}_{n=1}^N \), then the samples of slope vector of Eq.4 are calculated using Eq.1 and finite difference approximation of derivatives of all trajectories that fall in each partition. The optimality of the chosen partition is not considered here, but it is reasonable to guess that the method can be extended to learn it.

3.2. Transition Matrices and Initial State Probabilities

Instead of the sequence \( r(t) \) of actual PFS we use sequence \( k_n(t) \) of LMMs for learning and classification purpose. This makes the parameter learning phase more tractable by bypassing the need for reconstructing actual PFS from LMMs. However, unlike \( r(t) \), \( k_n(t) \) is not independent of the state \( x(t) \) since only a limited number of LMMs are allowed to be chosen while the object is in a particular zone.

\[
k_n(t) \text{ belongs to set of paired indexes } K_N = \bigcup_{n=1}^N \{(1, n), \cdots, (K_n, n)\}. \text{ where the pair } (i, \eta) \text{ represent the } i \text{th LMM in partition } \eta. \text{ The cardinality of } K_N \text{ is equal to total number of LMMs and each member specifies one of LMMs. Then, the dynamic of } k_n(t) \text{ for trajectory pattern } m \text{ is given by transition probabilities between LMMs}
\]

\[
P(k_n(t) = (j, \eta')|k_n(t-1) = (i, \eta), x_p(t) \in \mathcal{P}_{\eta'}, m) = \pi_{(i,\eta),(j,\eta')}^m,
\]

where \( \pi_{(i,\eta),(j,\eta')}^m \) is the element in column \( j \) and row \( i \) of inter-partition transition matrix \( \Pi_{\eta,\eta'}^m \) from partition \( \eta \) to \( \eta' \). Therefore, there are \( N^2 \) inter-partition transition matrices for each trajectory pattern \( m \), \( \{\Pi_{\eta,\eta'}^m\}_{\eta,\eta'=1}^N \), which are learned simply using normalized histogram of occurrences in the subset of \( D \) with members that are labeled as \( m \).

Likewise, it is straightforward to learn the prior probability of trajectory pattern \( P(m) \) using the frequency of labels in the training set. On the other hand, The learning of initial state probability density function \( p(x(0)|m) \) requires more consideration since the state vector is continues. Kernel density estimation method or histogram method after quantization can be used for this task [14]. The later is performed in the tests of this paper where the same partitioning method described in Sec.3.1 applied for quantization of position part of the state vector and a uniform quantizer in polar coordinates applied for the velocity part of state vector.
mean Gaussian noise as observation model. We use Tsai camera calibration model [15] with additive zero-mean Gaussian noise as observation model. The bottom layer variable $z(t)$ is the sequence of observations provided by the sensors, which in our case is the video data. The conditional probability density functions (CDF) of $x(t)$ and $k_n(t)$ are given by Eq.1 and Eq.5 respectively (note that the Gaussian distribution on $a_{k_n}$ in Eq.4 induces conditional Gaussian distribution for the CDF of $x(t)$).

4. PARTICLE FILTER BASED SEQUENTIAL TRAJECTORY CLASSIFIER

Fig.1 shows the proposed Dynamic Bayesian Network (DBN) of the state space model that describes the motion objects. The bottom layer variable $z(t)$ is the sequence of observations provided by the sensors, which in our case is the video data. The conditional probability density functions (CDF) of $x(t)$ and $k_n(t)$ are given by Eq.1 and Eq.5 respectively (note that the Gaussian distribution on $a_{k_n}$ in Eq.4 induces conditional Gaussian distribution for the CDF of $x(t)$).

4.1. Observation Model

We use Tsai camera calibration model [15] with additive zero-mean Gaussian noise as observation model

$$z(t) = h(x(t)) + \nu_z(t)$$

where $h(.)$ is a nonlinear function relates the ground position of the object to pixel position in the video frame. We assume that the object detection and data association is done and we are given the set of pixels at each frame $t$ as the pixels of the object. Then, we calculate the mean of these pixel positions as the observation $z(t)$ and covariance matrix of pixel positions as covariance matrix of observation noise $\nu_z(t)$.

4.2. Multiple Model Rao-Blackwellized Particle Filter

The type of system depicted in Fig.1 is known as Jump Markov System (JMS) for which particle filters have proven well performing state estimation techniques [16]. However in conventional JMS the dynamic of the discrete variable $k_n(t)$ is independent of continues state $x(t)$. The Rao-Blackwellized Particle Filter (RBPF) introduced in [17] is used here for state estimation and resampling algorithm from [18] is adopted to account for multiple switching model.

The set of variables of $N_p$ particles is $\{\alpha_{k_n}^{(i)}(t), x^{(i)}(0 : t), l^{(i)}, \omega^{(i)}\}_{i=1}^{N_p}$, where for each particle $i$ associated a weight $\omega^{(i)}$, the state history $x^{(i)}(0 : t)$, a label indicating the trajectory pattern $l^{(i)}$ the particle follows and the probability of LMM at current time,

$$\alpha_{k_n}^{(i)}(t) := P(k_n(t) = (l, \eta)|x^{(i)}(0 : t), z(1 : t)),$$

for all $l, \eta \in K_N$.

The resampling algorithm is activated whenever the number of effective particles falls below a threshold. However, using conventional resampling algorithms such as one introduced in [18] causes the filter to lose the diversity of trajectory pattern types among particles at very early stage of tracking. For this reason, we modified the resampling algorithm such that the proportion of particles that follow different trajectory patterns remain unchanged during the course of tracking.

The filtering algorithm starts with the reception of first observation $z(1)$. Then a distribution $P(m|z(1))$ is inferred based on $P(m)$ and $p(x(0)|m)$ using an arbitrary Bayesian inference method, here a one-step lag particle smoother is used [19]. Then $l^{(i)}$ is sampled from $P(m|x(1))$ and $x^{(i)}(0)$ is sampled from $p(x(0)|l^{(i)})$ for $i = 1, \cdots, N_p$. This procedure avoids initializing particles that are following improbable trajectory patterns given the initial observation and guarantees that the number of particles for each trajectory pattern is proportional to $P(m|x(1))$. Therefore, At each step of RBPF an empirical estimate of the posterior probability of trajectory pattern $m$ can be calculated as

$$\hat{P}(m|z(1 : t)) = \sum_{i|l^{(i)}=m} \omega^{(i)},$$

since the weights of the particles are proportional to the likelihood of observations given corresponding trajectory pattern and the initial number of particles is proportional to the prior probability of trajectory patterns.

5. EVALUATION

5.1. Dataset and Evaluation Procedure

The PDTV dataset [20] has been used for tests, which is the video of a road intersection. The available ground-truth of trajectories are used for learning process and the available foreground detections are used as measurements in the particle filter.

Five trajectory types, shown in Fig.2, are chosen for classification tests. The definition and the number of available samples of each type are:

1. A car that goes from right to left road, 14 samples.
2. A car that goes from left to right road, 2 samples.
3. A car that goes from right to upper road, 6 samples.
4. A car that goes from left to upper road, 2 samples.
5. A car that comes from left road, stops behind traffic light then goes to right road, 6 samples.
In order to avoid over-fitting effect, half of the samples of each trajectory pattern is used for learning and another half for classification.

Since the training data is highly unbalanced and the number of samples are very limited, we expand the training set by injecting a independent Gaussian random bias to each trajectory sample to achieve new trajectory samples. This process does not alter the shape of the trajectory but translate its position by a small random vector. In addition, throughout all processes the friction constant is chosen $\kappa = 0.2$ and the time difference set to $\delta = 1$. For learning LMMs the ground plane is divided into 36 non-overlapping rectangular partitions.

5.2. Results

Fig. 3 shows the posterior probability of trajectory pattern classes calculated at each time instance using Eq.7 for five sequences in the test set. The curves are the mean value of 100 independent Monte-Carlo run of classification algorithm. The solid curves correspond to the results with 100 particles and the dashed curves represent the result with 50 particles.

It can be seen that except for the test sequence (c) with 100 particles the MAP class is different from the true class at the beginning (Note that the difference of initial class probabilities in different figures is due to initialization procedure of particles). However, as the trajectory of the object evolves during its observation, the true class is identified as the MAP class. In all cases of Fig.3 (also in all sequences in test and training sets) eventually the maximum a posterior probability class is converged to true class of the test sequence. Specifically, in tests (a) and (c) and almost in (e) the MAP class probability is almost equal to one at the end.

The time instance at which the true trajectory pattern is recognized is different in tests because the discriminative feature of each test sequence appears at different time instance. However, this initial confusion is always between patterns with high physical initial similarity namely patterns 1 and 3.

The effect of the number of particles also depicted in Fig.3. Increasing the number particles not only increase the confidence about the true class but also advances the discrimination time. This is expected because more particles enables the algorithm to test more simultaneous hypotheses.

Fig. 3. Calculated posterior probability distribution of trajectory pattern classes for five test sequences. the horizontal axes is time (frame number).
6. REFERENCES


