Online Nonparametric Bayesian Activity Mining and Analysis From Surveillance Video

Vahid Bastani, *Graduate Student, IEEE,* Lucio Marcenaro, *Member, IEEE,* and Carlo S. Regazzoni, *Senior Member, IEEE*

***Abstract*—A method for online incremental mining of activity patterns from surveillance video stream is presented in this paper. The framework consists of a learning block in which Dirichlet process mixture model is employed for incremental clustering of trajectories. Stochastic trajectory pattern models are formed using Gaussian process regression of corresponding flow functions. Moreover, a sequential Monte-Carlo method based on Rao-Blackwellized particle filter is proposed for tracking and online classification as well as detection of abnormality during observation of an object. Experimental results on real surveillance video data are provided to show the performance of the proposed algorithm in different tasks of trajectory clustering, classification and abnormality detection.**

***Index Terms*—Incremental trajectory clustering, online activity analysis, abnormality detection, state dynamics learning, nonparametric Bayesian, Gaussian process, Dirichlet process mixture.**

I. Introduction

In recent years, the widespread usage of video cameras for control and safety has made surveillance video a consistent part of the global big data generated every day [1]. It is infeasible to employ human operators for monitoring this huge volume of data. Therefore, automated methods for continuous analysis of surveillance video are required to achieve maximum efficiency of the deployed cameras. An automated video analytic method must be able to learn the activity classes in an environment and reply to queries about the activity of a moving entity in real time. For a given environment, the main queries are determining if the activity of an object is a normal activity class for that environment or it is an abnormal unobserved behavior.

An object's activity can be defined as the sequence of actions that makes the object move. These actions can be observed through the object's motion pattern. One way to extract motion pattern information from video is to use optical flow estimation as shown in [2] and [3]. However, the most popular approach is to use tracking for forming trajectory of objects then considering trajectories as motion pattern descriptors. Consequently, trajectory learning and classification become central tasks for any video analytics method.

Activity recognition can be seen as supervised or unsupervised problem, on the basis of availability of labeled datasets. In supervised approaches such as [4]–[10], labeled data are used for learning models of each trajectory pattern that are then used for classification or abnormality detection purposes. On the other hand, in unsupervised problems [11]–[15], the dataset is unlabeled and the aim is to cluster similar trajectories and then use the clustered data to train models for classification. A particular class of unsupervised learning is incremental clustering [15], in which training data are not available at once and they have to be processed sequentially as new data is received. This approach is particularly useful in surveillance applications since it may not be feasible to have training datasets for every camera. In this case, one would leave an incremental clustering algorithm to process data streams from cameras and make the system gradually learn and organize activities it observes from the camera.

Whatever method is used for training, the learned models of trajectory patterns can be employed for different purposes. Some methods address only abnormality detection [16]–[18] while others perform classification and abnormality detection together [4], [8]. Trajectory retrieval [15] is another possible application. An important property of an activity analyzer is online classification and abnormality detection with partially observed trajectories, which is addressed in [4], [8], [9]. This is of great importance when timely actions are required in response to a particular observed activity or abnormality.

The purpose of this paper is to propose a unified framework for tracking, learning activities as trajectory patterns, online classification and abnormality detection. The main contributions of this paper can be summarized as follows:

- Online incremental trajectory clustering algorithm based on novel Bayesian nonparametric techniques.
- Online sequential trajectory classifier and abnormality detector based on sequential Monte-Carlo techniques.
- Defining a framework in which learned trajectories is used to improve visual object tracking.

The remainder of the paper is organized as follows: in Section II an overview of technically related works is presented and differences of the proposed method with respect to the state-of-the-art is highlighted. The background of the proposed algorithm is presented in Section III. In Sections IV, V and VI the proposed framework and related algorithms are presented in detail. The experimental evaluation of the proposed method is provided in Section VII and Section VIII concludes the paper.

V. Bastani. L. Marcenaro and C. S. Regazzoni are with the Department of Electrical, Electronics and Telecommunication Engineering and Naval Architecture (DITEN), University of Genova, 16145, Via All'Opera Pia 11A, Geona, Italy (email: vahid.bastani@ginevra.dibe.unige.it, lucio.marcenaro@unige.it, carlo.regazzoni@unige.it)

Manuscript received Month dd,yyyy; revised Month dd,yyyy.

II. Related Works

In this paper, a trajectory model based on flow functions is introduced. This was first proposed in [5] using Switching Dynamical Model (SDM) and extended in [6] and [9] to state dependent SDM system. Alternatively, in [4] Gaussian Process (GP) regression was used for modeling flow functions from a stochastic viewpoint. In this paper, GP is used for trajectory modeling because it allows high flexibility for learning time-variant non-linear flow functions. However, the proposed method of this paper is different form [4] in two ways. Firstly, the proposed method is incremental and unsupervised, which means no labeling is required from training data and the model is always updated by receiving new trajectory samples. Secondly, unlike [4], the trajectory classification and the abnormality detection are now integrated into the object tracker. Moreover, The integrated tracker uses the learned flow function in the tracking process which improves the tracking performance over time.

In completely unsupervised clustering problems, the number of data clusters is unknown. The Mean-shift clustering algorithm [19] and Dirichlet Process mixture (DPM) model [20] can be applied in such problems. In this paper, DPM is used since it fits perfectly in a probabilistic framework and it enables incremental clustering. The Mean-shift clustering algorithm was used in [21] for trajectory clustering in multiple feature spaces. DPM has also been deployed for the same purpose in [12] as a mixture of hidden Markov models and in [2] as mixture of temporal motifs. In both methods, DPM was used in batch learning framework using a Gibbs sampling inference technique, that is very slow in convergence and cannot be applied for online usage. An incremental trajectory learning method based on DPM is proposed in [15], where Discrete Fourier transform (DFT) features were used to represent the trajectory, then DPM was used as a mixture of DFT features distribution. The method of [15] employs Gibbs sampling for learning but it allows for sequential assignment of new batch data. That approach is applied for trajectories' retrieval, but it cannot be used as an online trajectory classifier, since it requires the whole trajectory to calculate DFT features and identify trajectory classes. In contrast, a recently proposed DPM variational inference technique [22] is used in our work for learning purposes. The variational inference is faster and is more efficient than Gibbs sampling. More importantly, the DPM in our method defines a mixture of stochastic flow functions, which make online sequential classification possible.

The Rao-Blackwellized Particle Filter (RBPF) proposed in this paper is a modified version of the technique proposed in [23] for online parameter estimation. A preliminary version of this method was proposed in [9] for supervised trajectory classification, where trajectories were modeled as potential functions instead of flow functions. Here, the method is improved by extending it for unsupervised approaches, and by modifying the algorithm to a flexible adaptive particle sampling from mixture of flow functions.

The framework proposed in this paper is able to provide unsupervised incremental clustering together with online classification and abnormality detection embedded inside particle filter tracker. These features have not been simultaneously addressed in the literature despite they are interrelated and necessary for truly intelligent video analytic system. Table I outlines the comparison of the proposed method with related references that partially addressed these features.

TABLE I: Comparison with state-of-the-art

Method	Unsupervised	Incremental Clustering	Online Classification	Abnormality Detection	Embedded Tracking
Proposed	✓	✓	✓	✓	✓
DPM [15]	✓	✓*	✗	✗	✗
GP [4]	✗	✗	✓	✓	✗
SDM [9]	✗	✗	✓	✗	✓
SDM [5]	✗	✗	✓	✗	✗
DPM [2]	✓	✗	✗	✗	✗
DPM [12]	✓	✗	✗	✗	✗
Mean-Shift [21]	✓	✗	✗	✗	✗

* With batch initialization and batch increments

III. Background

The purpose of this section is to introduce Gaussian process regression and Dirichlet process mixture model which are used in the proposed algorithm.

A. Gaussian Process Regression

A Gaussian Process (GP) defines a probability distribution over functions $f : \mathcal{X} \to \mathbb{R}$ which maps a domain vector space $\mathcal{X}$ to the real numbers $\mathbb{R}$ such that the marginal distribution of vectorized function values $\Gamma = [f(\mathbf{x}_1), \cdots, f(\mathbf{x}_N)]^T$ over any finite subset $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\} \subset \mathcal{X}$ has a multivariate Gaussian distribution [24]. A GP, denoted $f(\mathbf{x}) \sim GP(\bar{f}(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, is characterized by a mean function $\bar{f}(\mathbf{x})$, which is usually set to zero for notational simplicity, and a covariance function $k(\mathbf{x}, \mathbf{x}')$ that encodes covariance of two values, $f(\mathbf{x})$ and $f(\mathbf{x}')$.

The GP is widely used as function prior in nonlinear, nonparametric regression [25], and classification [26] problems. Given a noisy training data set $\mathcal{D} = \{X, \tilde{\Gamma}\}$ consists of noisy function values $\tilde{\Gamma} = [\tilde{f}(\mathbf{x}_1), \cdots, \tilde{f}(\mathbf{x}_N)]^T$ at the set of points $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, the problem is to estimate the values of function $\Gamma^* = [f(\mathbf{x}_1^*), \cdots, f(\mathbf{x}_N^*)]^T$ at a set of new points $X^* = \{\mathbf{x}_1^*, \cdots, \mathbf{x}_M^*\}$. The observation process is $\tilde{f}(\mathbf{x}) = f(\mathbf{x}) + e(\mathbf{x})$ where $f(\mathbf{x})$ has a GP prior and $e(\mathbf{x}) \sim \mathcal{N}(0, \sigma)$ is a Gaussian zero-mean white noise process. Since observation likelihood and the prior of function are Gaussian, the predictive posterior distribution of function values at new points remains a multivariate Gaussian with mean vector

$$\bar{\Gamma}^* := \mathbb{E}[\Gamma^* | \mathcal{D}] = \mathbf{K}_{.|*}[\mathbf{K}_{.|.} + \sigma^2 \mathbf{I}]^{-1} \tilde{\Gamma} \tag{1}$$

and covariance matrix

$$\begin{aligned} \mathbb{C}[\Gamma^* | \mathcal{D}] :=& \mathbb{E}[(\Gamma^* - \bar{\Gamma}^*)(\Gamma^* - \bar{\Gamma}^*)^T | \mathcal{D}] \\ =& \mathbf{K}_{*|*} - \mathbf{K}_{*|.}[\mathbf{K}_{.|.} + \sigma^2 \mathbf{I}]^{-1} \mathbf{K}_{.|*} \end{aligned} \tag{2}$$

where $\mathbf{K}_{*|.}$ denotes a matrix whose element in ith row and jth column is $k(\mathbf{x}_i^*, \mathbf{x}_j)$ and likewise for $\mathbf{K}_{.|*}$, $\mathbf{K}_{*|*}$ and $\mathbf{K}_{.|.}$ [24].

A valid covariance function in GP should be positive semidefinite kernel. There are many families of covariance functions and the choice in general depends on the properties of the process, which generated the data. Each kernel family has a set of hyper-parameters θ controlling the behavior of the functions generated by the GP. Training of GP involves choosing θ and σ such that the resulting GP maximizes marginal likelihood of the data or posterior probability of hyperparameters.

B. Data Assignment in GP Regression

The Bayesian Committee Machine (BCM) technique [27] was originally proposed as an approximation technique for reducing computation and speeding up GP regression. It relies on splitting the training data set into Q sets $\mathcal{D} = \{\mathcal{D}_1, \cdots, \mathcal{D}_Q\}$, where $\mathcal{D}_q = (X_q, \tilde{\Gamma}_q)$, and training Q separate estimator from each set. Under the approximation that

$$p(\tilde{\Gamma}_1, \cdots, \tilde{\Gamma}_Q | \Gamma^*, X) \approx \prod_{q=1}^{Q} p(\tilde{\Gamma}_q | \Gamma^*, X_q), \tag{3}$$

the predictive density becomes

$$p(\Gamma^* | \mathcal{D}) = C \times \frac{\prod_{q=1}^{Q} p(\Gamma^* | \mathcal{D}_q)}{p(\Gamma^*)^{Q-1}} \tag{4}$$

where C is a constant. The above density is generally intractable due to the proportionality constant. However, in the case of GP estimators, where all the distributions are Gaussian, the predictive density also becomes Gaussian with covariance matrix and mean vector

$$\mathbb{C}[\Gamma^* | \mathcal{D}]^{-1} \approx (1 - Q)\mathbf{K}_{*|*}^{-1} + \sum_{i=1}^{P} \mathbb{C}[\Gamma^* | \mathcal{D}^i]^{-1} \tag{5}$$

$$\mathbb{E}[\Gamma^* | \mathcal{D}] \approx \mathbb{C}[\Gamma^* | \mathcal{D}] \sum_{i=1}^{Q} \mathbb{C}[\Gamma^* | \mathcal{D}^i]^{-1} \mathbb{E}[\Gamma^* | \mathcal{D}^i] \tag{6}$$

BCM technique is used in our algorithm for selection of trajectory samples that should contribute to the model of a trajectory cluster. In the proposed model, each estimator corresponds to a GP trained by data from the trajectory of one object. In this way, retraining trajectory cluster model is avoided when new trajectories are assigned to the cluster. Instead, an independent GP is built using a new observed trajectory. Whenever a prediction of flow function values of a particular cluster is required, GPs correspond to trajectories in that cluster are combined using BCM technique. Equations (3 - 6) are good approximations when either the data sets are unconditionally independent or the number of prediction query points is large [27]. In our work, both conditions are exploited where the data partitions correspond to the data collected from trajectories of independent objects and the number of evaluation points is equal to the number of particles in a sequential Monte-Carlo algorithm (see Section VI for more detail).

C. Online Clustering by Variational Dirichlet Process Mixture

Dirichlet Process (DP), denoted by $DP(\alpha, p_H)$, is a distribution over discrete probability distributions, which is characterized by a concentration parameter $\alpha > 0$ and a base distribution p_H. The discreteness of distributions drawn from DP motivates its usage as prior for mixture models with random and possibly infinite number of mixture components [20], [28]. Such a model is called Dirichlet Process Mixture (DPM) model and it has been widely used for Bayesian data clustering where the number of clusters is not known. Recently DPM is shown to be effective for learning mixture of latent functions which are modeled by GP [29], [30].

With a slight abuse of notation let $X = \{x^i\}_{i=1}^n$ denote a random variable generated from DPM, then $x^i \sim p_F(x^i | \theta_i)$, where p_F is a parametric PDF representing distribution of mixture components, $\theta_i \sim p_G$ is the parameter of p_F and $p_G \sim DP(\alpha, p_H)$ is a random measure drawn from DP. Alternatively, using new variables $Z = \{z^i\}_{i=1}^n$ that indicate respective mixture component of each sample, the generative process can be expressed as

$$x^i | z^i, \{\phi_k\}_{k=1}^{\infty} \sim p_F(x^i | \phi_{z^i}) \tag{7}$$

where ϕ_k is the parameter of kth mixture components which is refered to as atom. Atoms are randomly drawn from DP base distribution

$$\phi_k \sim p_H, \ \ k = 1, 2, \cdots. \tag{8}$$

The indicator variable is drawn from a categorical distribution of mixture weights

$$z^i | \{\pi_k\}_{k=1}^{\infty} \sim Cat(\pi_1, \pi_2, \cdots) \tag{9}$$

where the mixture weights are given by stick-braking process with parameter α [20].

The main inference task in DPM is to learn number of mixture components, their parameters, and mixture weights given a set of observed samples. It involves the computation of the posterior function

$$p(p_G | X) = \sum_Z p(Z | X) p(p_G | X, Z), \tag{10}$$

which is intractable because the summation over all combination of partitions Z exponentially grows with n. Conventionally, Markov Chain Monte-Carlo (MCMC) methods such as Gibbs sampling are used for learning in DPM. However, practical usage of MCMC methods is limited due to their slow convergence rate. Thus, techniques based on variational approximations, which turns the learning problem into optimization problem, have become popular alternative learning approaches [31], [32].

In this paper the online variational technique proposed in [22] and [33] is used for incremental learning purposes. It is assumed that the posterior in (10) can be approximated by the variational density

$$p(p_G | \rho, \nu) = \sum_Z \prod_{i=1}^{n} \rho_i(z^i) \bar{p}(p_G | Z) \tag{11}$$

where $\prod_{i=1}^{n} \rho_i(z^i)$ replaces $p(Z|X)$ and $\bar{p}(p_G|Z)$ is a stochastic process on θ that is equivalent to generative process

$$\beta_0 p_H(\theta) + \sum_{k=1}^{K} \beta_k \delta(\theta - \phi_k) \tag{12}$$

where $(\beta_0, \cdots, \beta_k)$ have $K+1$-dimensional Dirichlet distribution $Dir(\alpha, \eta_1, \cdots, \eta_K)$ and $\phi_k \sim \tilde{p}_k$. Here $\tilde{p}_k$ is an independent distribution that approximate the posterior of kth mixture component. One important implication of this approximation is its resulting predictive law

$$\mathbb{E}_q[p(\theta|p_H, \alpha))] = \frac{\alpha}{\alpha+n} p_H(\theta) + \sum_{k=1}^{K} \frac{\eta_k^n}{\alpha+n} \tilde{p}_k(\theta), \tag{13}$$

which says that new samples from DPM are generated either from one of K seen components, whose parameter prior distribution is $\tilde{p}_k(\theta)$ with probability proportional to $\eta_k^n = \sum_{j=1}^{n} \rho_j(k)$, or from a new component of DP with probability proportional to α. The approximation enables a recursive estimation of optimal set of parameters, which minimizes the Kullback-Leiber divergence between (10) and (11). Let $(\rho_1, \cdots, \rho_i)$ and $(\tilde{p}_1, \cdots, \tilde{p}_K)$ be a set of parameters learned after processing the $(i-1)$th sample, then the optimal new parameters are

$$\begin{aligned}
\rho_i(k) &\propto \begin{cases} \omega_k^{i-1} \int_\theta \tilde{p}_k(\theta) p_F(x^i|\theta), & k \le K, \\ \alpha \int_\theta p_H(\theta) p_F(x^i|\theta), & k = K+1, \end{cases} \\
\tilde{p}_k(\theta) &\propto \begin{cases} p_H(\theta) \prod_{j=1}^{i} p_F(x^j|\theta)^{\rho_j(k)}, & k \le K. \\ p_H(\theta) p_F(x^i|\theta)^{\rho_i(k)}, & k = K+1. \end{cases}
\end{aligned} \tag{14}$$

Note that this sequential updating increases the number of components every time a new sample is processed. However, this is unnecessary, thus a thresholding mechanism is employed to introduce a new component only when the probability of that component is sufficiently large, i.e. $\rho_i(K+1) > e$. The optimal update equation is only applicable when p_H and $\tilde{p}$ are conjugate priors of p_F. In the non-conjugate cases, instead of $\tilde{p}$ it is suggested to maintain a MAP point estimates $\{\hat{\theta}_k\}_{k=1}^{K}$, which are computed using gradient descent algorithm.

IV. Proposed Algorithm Overview

In this section the framework of the proposed algorithm for online learning and classification of trajectory patterns is introduced. Let $\mathbf{x}_t = [x(t), y(t)]^T$ be the state vector at time t that indicates the position of an object in 2-D space. Given small enough time difference δ, the dynamics of the object can be generally represented as

$$\mathbf{x}_{t+\delta} = \mathbf{x}_t + \delta \times \tilde{\mathbf{f}}_r(\mathbf{x}_t, t), \tag{15}$$

where $\tilde{\mathbf{f}}_r = [\tilde{f}_r^x, \tilde{f}_r^y]^T$ is the flow vector for a particular trajectory pattern indexed by time-invariant variable r. $\tilde{\mathbf{f}}_r(\mathbf{x}, t)$ is the sum of noiseless flow function $\mathbf{f}_r(\mathbf{x}, t)$ and an independent stationary zero-mean Gaussian noise. The state vector sequence forms a Markov process governed by the flow function

$$\begin{aligned}
\mathbf{x}_{t+1}|\mathbf{x}_t, r, t &\sim p_f(\mathbf{x}_{t+1}|\mathbf{x}_t, r, t) \\
&= \mathcal{N}(\mathbf{x}_t + \mathbf{f}_r(\mathbf{x}_t, t), \mathrm{Cov}[\tilde{\mathbf{f}}_r(\mathbf{x}_t, t)]).
\end{aligned} \tag{16}$$

where it is assumed $\delta = 1$ for simplicity. The state vector is usually unobservable and its information is available through a dependent measurement vector $\mathbf{y}_t$ where the dependency is expressed with a conditional density function

$$\mathbf{y}_t|\mathbf{x}_t, t \sim p_g(\mathbf{y}_t|\mathbf{x}_t, t) \tag{17}$$

Using the state space model a trajectory pattern is characterized by a corresponding flow function and an initial state distribution $p_0(\mathbf{x}, r) = p(\mathbf{x}_0|r)$. In practice, there are a number of different trajectory patterns $r = 1, \cdots, K$, where given the measurement sequence $\mathbf{y}_{1:t}$ and assuming K known flow functions $\{\tilde{\mathbf{f}}_r\}_{r=1}^{K}$, it is possible to calculate an approximation of the joint posterior $p(\mathbf{x}_{1:t}, r|\mathbf{y}_{1:t})$ of state trajectory and trajectory pattern index. As described in Section VI this can be done efficiently using recursive Bayesian filtering.

The block diagram of the proposed algorithm is shown in Fig. 1. This is a snapshot of the algorithm when q number of objects have been observed completely, from which K flow functions are identified and learned. Assume that object $q+1$ has been observed up to time $t-1$ and now the measurement $\mathbf{y}_t^{q+1}$ is being processed. The proposed RBPF in Section VI at each time instant provides the joint filtered posterior of the state $\mathbf{x}_t$ and class r of the object. Then a memory mechanism stores all the filtered states till the end of trajectory of object $q+1$ to form the joint posterior of complete trajectory and class of the object $p(\mathbf{x}_{1:T_{q+1}}, r|\mathbf{y}_{1:T_{q+1}}^{q+1})$, where T_q represents duration of trajectory q. The complete data are then passed to the clustering and learning algorithm introduced in Section V, which either assign this new data to one of already existing K classes or make a new class $K+1$ using DPM rules, then updates the corresponding flow function using the data observed from object $q+1$ according to GP and BCM equations. In the next two sections the details of algorithms used in two blocks of tracking/classification (Section VI) and clustering/learning (Section V) are presented.

V. Online Incremental Clustering of Trajectories

The proposed method for incremental clustering is described in this section. Firstly, training independent estimators on individual trajectories and combining them for forming a single probabilistic model will be described. Then, the procedure for incremental updating of this model as new sample trajectory is observed will be shown.

A. Flow Function Learning

Each observed trajectory $\mathbf{y}_{1:t}^q$ of object q bears some information about the underlying flow function and can contribute to learning. The set $\mathcal{D}_q = \{\hat{\mathbf{x}}_{1:T_q}^q, \dot{\hat{\mathbf{x}}}_{1:T_q}^q, \sigma_q, \theta_q, \rho_q(.)\}$ is the collection of all data gathered from tracking of object q, $\rho_q(k) \in \{0, 1\}$ is the cluster indicator function that is one

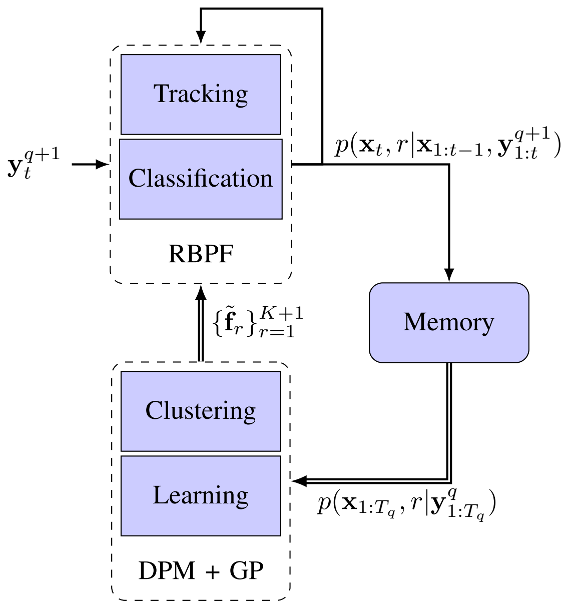

Tracking
Classification
RBPF
$\mathbf{y}_t^{q+1}$
$p(\mathbf{x}_t, r|\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t}^{q+1})$
Memory
$\{\tilde{\mathbf{f}}_r\}_{r=1}^{K+1}$
Clustering
Learning
$p(\mathbf{x}_{1:T_q}, r|\mathbf{y}_{1:T_q}^q)$
DPM + GP

Fig. 1: Block diagram of proposed algorithm. Information on single arrows is updated with each observation while information on double arrow is updated with each trajectory.

if the trajectory q belongs to pattern k and zero otherwise. $\hat{\mathbf{x}}_{1:T_q}^q$ is the estimated state trajectory

$$\hat{\mathbf{x}}_{1:T_q}^q = \mathbb{E}[\mathbf{x}_{1:T_q}^q|\mathbf{y}_{1:T_q}^q], \tag{18}$$

and $\hat{\dot{\mathbf{x}}}_{1:T_q}^q$ is the estimated flow (speed) at each trajectory point

$$\hat{\dot{\mathbf{x}}}_{1:T_q}^q = \mathbb{E}[\dot{\mathbf{x}}_{1:T_q}^q|\mathbf{y}_{1:T_q}^q], \tag{19}$$

which is calculated point-wise by using filtered state density

$$p(\dot{x}(t)|\mathbf{y}_{1:t+1}^q) = \iint \delta(\dot{x}(t) - (x_{t+1} - x_t))p(\mathbf{x}_{t:t+1}^q|\mathbf{y}_{1:t+1}^q)\, d\mathbf{x}_{t:t+1}^q, \tag{20}$$

for x direction and likewise for y direction. $\sigma_n = \{\sigma_q^x, \sigma_q^y\}$ is the mean standard deviation of estimated flow which serves as noise variance in GP, and $\theta_q = \{\theta_q^x, \theta_q^y\}$ is the MAP parameter of GP covariance function estimated using data of object q,

$$\theta_q^x = \underset{\theta}{\operatorname{argmin}} \text{ -}\log(p(\theta|\dot{x}(1:T_q), \hat{\mathbf{x}}_{1:T_q}^n, \phi)) \tag{21}$$

for x direction and likewise for y direction. The optimization is done by using probabilities calculated by GP and ϕ is the parameter of prior density of θ.

Each $\mathcal{D}_q$ corresponds to a flow estimator learned beased on the observation of the object q. However, clustering of trajectories implies that each cluster has a single flow estimator optimized over all data in the cluster. One can hold a single GP model for each cluster and recalculates its parameters every time a trajectory is assigned to that cluster. The complexity of parameter optimization and GP estimation of this approach increases with number of trajectories in the cluster. Instead, we use the approximation of BCM to estimate flow of a cluster using individual GP estimators built on each trajectory data $\mathcal{D}_q$.

Let $\tilde{\Gamma}_{r,x}^* = [\tilde{f}_r^x(\mathbf{x}_*^{(1)}, t_*^{(1)}), \cdots, \tilde{f}_r^x(\mathbf{x}_*^{(M)}, t_*^{(M)})]^T$ be vectors of x component flow function values that are going to be predicted (for y component the derivation is the same). Using BCM technique the predictive probability of $\tilde{\Gamma}_{r,x}^*$ when $r = k$ is expressed as

$$p(\tilde{\Gamma}_{r,x}^*|r = k) \propto \frac{\prod_{q=1}^{Q} [\rho_q(r)p(\tilde{\Gamma}_{r,x}^*|\mathcal{D}_q) + (1 - \rho_q(r))p(\tilde{\Gamma}_{r,x}^*)]}{p(\tilde{\Gamma}_{r,x}^*)^{Q-1}} \tag{22}$$

where $p(\tilde{\Gamma}_{r,x}^*)$ is the predictive distribution using GP prior without training data. Since $\rho_q(.)$ is either zero or one and all distributions are Gaussian, (22) will be a Gaussian with covariance and mean calculated as

$$\begin{aligned}\mathbb{C}[\tilde{\Gamma}_{r,x}^*|r = k]^{-1} = \\ &(1 - Q)\mathbb{C}[\tilde{\Gamma}_{r,x}^*]^{-1} + \\ &\sum_{q=1}^{Q} \rho_q(k)\mathbb{C}[\tilde{\Gamma}_{r,x}^*|\mathcal{D}_q]^{-1} - (1 - \rho_q(k))\mathbb{C}[\tilde{\Gamma}_{r,x}^*]^{-1}\end{aligned} \tag{23}$$

$$\begin{aligned}\mathbb{E}[\tilde{\Gamma}_{r,x}^*|r = k] = \\ &\mathbb{C}[\tilde{\Gamma}_{r,x}^*|r = k] \sum_{q=1}^{Q} \rho_q(k)\mathbb{C}[\tilde{\Gamma}_{r,x}^*|\mathcal{D}_q]^{-1}\mathbb{E}[\tilde{\Gamma}_{r,x}^*|\mathcal{D}_q]^{-1}\end{aligned} \tag{24}$$

where $\mathbb{C}[\tilde{\Gamma}_{r,x}^*]$ is the GP prior covariance matrix which is calculated with maximum prior probability parameters of the covariance function

$$\theta_0^x = \underset{\theta}{\operatorname{argmax}}\, p(\theta|\phi). \tag{25}$$

In practice it is of interest to have marginal mean $\mu_{r,x}^{(i)}$ and variance $\sigma_{r,x}^{(i)}$ of function value $\tilde{f}_r^x(\mathbf{x}_*^{(i)}, t_*^{(i)})$. These quantities can be found respectively by taking the ith element of vector $\mathbb{E}[\tilde{\Gamma}_{r,x}^*|r = k]$ and the ith diagonal element of matrix $\mathbb{C}[\tilde{\Gamma}_{r,x}^*|r = k]$.

B. *DPM incremental Update Rule*

In the last subsection, a method was described to probabilistically model a group of trajectories characterized by the same flow function. In this subsection, the aim is to describe how incremental learning of flow functions $\{\tilde{\mathbf{f}}_r\}_{r=1}^K$ can be done by observing moving objects over time. The algorithm is initialized with no cluster, i.e. $K = 0$, each time new trajectories are observed, they are assigned to the closest available clusters if present or new clusters are created when trajectories are detected as novelties. To this end, the idea of variational DPM approximation of Section III-C is used and cluster assignment update equations are defined, which are adapted to comply with GP model.

Let us assume q trajectories have been observed, which are grouped into K clusters. While a new trajectory $q + 1$ is being observed, the posterior cluster probabilities at each time instance t is computed as

$$\tilde{\rho}_{q+1}^t(r) \propto \begin{cases} \eta_r^q \gamma_{q+1}^t(r)c(\tau, m_r(t)), & r \leq K \\ \alpha\gamma_{q+1}^t(r), & r = K + 1 \end{cases} \tag{26}$$

where $\eta_r^q = \sum_{j=1}^q \rho_j(r)$ is the number of trajectories assigned to cluster r, $\gamma_q^t(r) = p(\mathbf{y}_{1:t}^q|r)$ is the observed trajectory likelihood of the object q up to time t given flow model of cluster r and $c(\tau, m_k(t))$ is a correction term, which will be introduced in the following paragraphs. In Section VI, a method is presented for online estimation of the quantity $\tilde{\rho}_{q+1}^t(r)$ at each time instant t, jointly with tracking using particle filtering. The clustering algorithm of this section uses values at the end of each trajectory, based on the final value of (26) at the end of trajectory $\tilde{\rho}_{q+1}^{T_{q+1}}(.)$; such a value is chosen as it represents the best estimate of the class of the object for clustering purposes.

Comparing (26) with (14) of on-line variational DPM, here we have two modifications. Firstly, instead of cluster parameter posterior, complete data likelihood is used, because the former is not feasible due to non conjugate structure for GP parameters. Thus, as suggested in [22], MAP point estimates of DP covariance functions parameters is used to compute the complete data likelihood and to have approximately the same effect. The second modification is the use of correction term. Note that, unlike conventional DPM, where the clusters are modeled by conventional probability distributions, here cluster models are GPs, which are distributions on functions. It is not possible here to compute the likelihood of mixture samples, because they are infinite dimensional functions. Instead, $\gamma_q(r)$ is evaluated on a finite subset of the function domain occupied by sample trajectory. The correction term in (26) is deployed in order to overcome the implications introduced by using GP as cluster models.

Consider the one-dimensional demonstrative example of Fig. 2. Suppose that the red trajectory in the lower plot has been observed first. From the noisy speed points of this trajectory, which are shown as red dots in upper plot of Fig. 2, a GP has been trained. The mean of the GP predictive posterior is shown as yellow line and its standard deviation is shown as gray region. The strip between the two dashed lines in the upper plot is the prior standard deviation region of the GP (the prior mean is zero everywhere). Now, suppose another trajectory depicted with blue dots has been observed. This trajectory behaves like the first one for $t < 40$ ($x < 1$), but it continues with constant speed to position $x = 1.5$ where it starts decelerating.

In the example of Fig. 2, the likelihood of blue trajectory given the GP trained on the red one is much larger than the likelihood of that given the prior GP $\gamma_2(1) >> \gamma_2(2)$ (here $K + 1 = 2$). The reason is that, up to $x = 1$, the second trajectory flow almost lies around the mean of first GP. However, after $x = 1.5$ the likelihood given the first GP and the prior GP is almost the same because no data was available from red trajectory in this domain and the posterior GP simply converges back to prior GP there. In this case, the DPM process allocates these two trajectories to the same cluster because of the high likelihood of first GP. However, generally it is difficult to say that these two trajectories should belong to the same cluster. Thus, a correction term was incorporated in (26) to compensate the situation where there is a lack of data. In this case, it should be preferred to decrease the assignment score of evaluated trajectory to a cluster when the trajectory

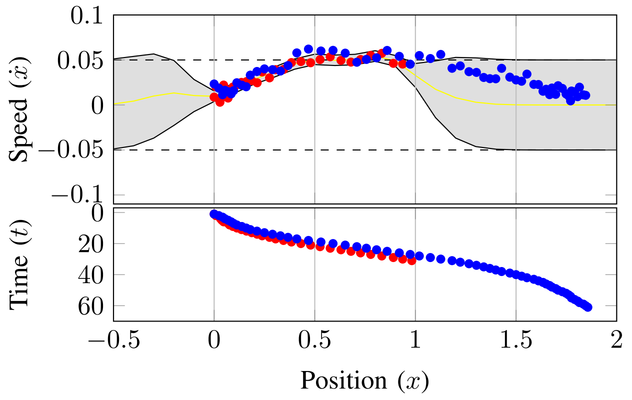


Fig. 2: A one dimensional example of trajectories in space-time and corresponding flow functions (used to motivate correction term). The red and blue dots are two different trajectories. Yellow line shows the learned GP flow from red trajectory while gray area shows uncertainty region.

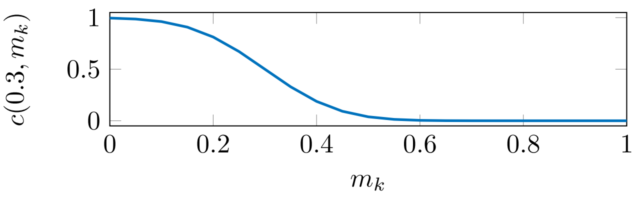


Fig. 3: An example of correction function with $\tau = 0.3$

deviates from the neighborhood of the training data of the cluster. To this end, the proportion of trajectory where the local flow given the cluster GP is similar to prior GP is computed, denoting it as $m_k \in [0, 1]$. Then a soft thresholding is used to gradually decrease the corresponding score as m_k increases. The function $c(\tau, m_k)$ is equal to one when $m_k \to 0$ and is equal to zero when $m_k \to 1$. It is a sigmoid-like function whose inflection point is τ:

$$c(\tau, m_k) = (1 + erf(-5\sqrt{\pi/2}(m_k - \tau)))/2, \tag{27}$$

where $erf(.)$ is standard error function. the shape of function is shown in Fig. 3 for $\tau = 0.3$.

Finally, the cluster indicator function $\rho_{q+1}(.)$ of $\mathcal{D}_{q+1}$ which is used in (23) and (24) is calculated as

$$\rho_{q+1}(r) = \begin{cases} 1 & r = \text{argmax}_k\, \tilde{\rho}_{q+1}(k) \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

Although this may causes information loss due to changing the soft clustering of $\tilde{\rho}_{q+1}(.)$ to hard clustering represented by $\rho_{q+1}(.)$, it simplifies the clustering problem and makes the use of BCM feasible in our problem.

The technique for incremental clustering and learning the flow functions is summarized in Algorithm 1. The complexity of the step 1 of the algorithm will be discussed in Section VI. In step 2 of the algorithm each evaluation of GP prediction equations is done in cubic order of number of points due to matrix inversion of (1) and (2). Thus, the step 2 has the complexity of $\mathcal{O}(ST_{q+1}^3)$ where S is the number of times the

GP prediction equations are evaluated in optimization problem. In practice, an upper bound is set for S and the gradient descent algorithm is terminated when the bound is reached. However, the computational load does not limit online usage of the algorithm since it is processed at the rate of the appearance of object (q), which is much lower than t in practice.

Algorithm 1 Incremental trajectory clustering

```
Input: $\{\mathcal{D}_i\}_{i=1}^{q}$, $\mathbf{y}_{1:T_{q+1}}^{q+1}$, $K$, $\phi$
Output: $\{\mathcal{D}_i\}_{i=1}^{q+1}$, $K$
  1) apply Algorithm 2 to find $\tilde{\rho}_{q+1}^{t}(r)$ for $t = T_{q+1}$ and
     $r = 1, \cdots, K+1$ , $\hat{\mathbf{x}}_{1:T_{q+1}}^{q+1}$, $\hat{\dot{\mathbf{x}}}_{1:T_{q+1}}^{q+1}$ and $\sigma_{q+1}$
  2) find $\theta_q$ of (21) by gradient descent algorithm
  3) find $\rho_{q+1}(r)$ from (28) for $r = 1, \cdots, K+1$
  4) if $\rho_{q+1}(K+1) = 1$ then $K = K+1$ and $\rho_i(K+1) = 0$
     for $i = 1, \cdots, q$
  5) let $\mathcal{D}_{q+1} = \{\hat{\mathbf{x}}_{1:T_{q+1}}^{q+1}, \hat{\dot{\mathbf{x}}}_{1:T_{q+1}}^{q+1}, \sigma_{q+1}, \theta_{q+1}, \rho_{q+1}(.)\}$
```

VI. Sequential Classification using Rao-Blackwellized Particle Filter

In the last section, the proposed method for incremental trajectory clustering was described. In this section, a sequential Monte-Carlo method based on Rao-Blackwellized Particle Filter (RBPF) is introduced for joint tracking and classification purposes. The output of filtering provides necessary information for calculation of required quantities used in the clustering algorithm.

A. Online state and cluster label probability estimation

The RBPF filter is proposed here for tracking and online estimation of trajectory classes. The technique is inspired by [23], where it has been used for tracking and parameter estimation for switching dynamical systems. The main idea is to marginalize the cluster index variable r during the particle propagation. The joint filtered density of state and cluster indicator variable can be factorized as (in this section we use object index q only whenever is necessary):

$$p(\mathbf{x}_{1:t}, r|\mathbf{y}_{1:t}) = p(r|\mathbf{x}_{1:t}, \mathbf{y}_{1:t})p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) \tag{29}$$

the second term of above density is approximated using a particle filter represented by the set of state trajectory particles together with their weights $\{\mathbf{x}_{1:t}^{(i)}, \omega_t^{(i)}\}_{i=1}^{N}$. The density of r is a multinomial distribution of the size $K+1$ conditioned on the state trajectory $\mathbf{x}_{1:t}$. In order to be able to marginalize r, for every particle $i = 1, \cdots, N$ the conditional density

$$\zeta_t^{(i)}(k) := p(\mathbf{x}_{1:t}^{(i)}, \mathbf{y}_{1:t}|r = k),\ k = 1, \cdots, K+1 \tag{30}$$

is assigned. In other words, a conditional probability of clusters is estimated using the information of the trajectory of each particle.

By including the conditional cluster probability to the particle system, the complete particle set $\{\mathbf{x}_{1:t}^{(i)}, \zeta_t^{(i)}(.), \omega_t^{(i)}\}_{i=1}^{N}$ for RBPF is formed. This variable set is updated at every iteration of the filter and is used to calculate required quantities such as complete data likelihood $\gamma_q(r)$ of (26), estimated trajectory $\hat{\mathbf{x}}_{1:T}$, estimated flow $\hat{\dot{\mathbf{x}}}_{1:T}$ and its error variance.

The RBPF algorithm starts by initializing $\mathbf{x}_0^{(i)} \sim p_0(\mathbf{x})$, $\omega_t^{(i)} = 1/N$ and $\zeta_t^{(i)}(k) = 1/(K+1)$ for $i = 1, \cdots, N$ and $k = 1, \cdots, K+1$. Each iteration begins with particles resampling to avoid their degeneration [34]. The resampling process results in a new set of the particles denoted by $\{\tilde{\mathbf{x}}_{1:t}^{(i)}, \tilde{\zeta}_t^{(i)}(.), \tilde{\omega}_t^{(i)}\}_{i=1}^{N}$. However, it unnecessary to perform resampling at each iteration. It can be done only when the number of effective particles becomes lower than a pre-defined threshold.

To propagate the state vector to the next time instant $t+1$ for every particle $i = 1, \cdots, N$ the new state vector is sampled from a proposal distribution

$$\mathbf{x}_{t+1}^{(i)} \sim p_s(\mathbf{x}_{t+1}|\tilde{\mathbf{x}}_{1:t}^{(i)}, \mathbf{y}_{1:t}), \tag{31}$$

Then the updated trajectory of particle i is $\mathbf{x}_{1:t+1}^{(i)} = \{\tilde{\mathbf{x}}_{1:t}^{(i)}, \mathbf{x}_{t+1}^{(i)}\}$. The cluster posterior probabilities have to be estimated using all the object history up to the current time. Consider the factorization of complete conditional data likelihood:

$$\begin{aligned} p(\mathbf{x}_{1:t+1},&\mathbf{y}_{1:t+1}|r) = \\ &p(\mathbf{x}_{1:t}, \mathbf{y}_{1:t}|r)p(\mathbf{x}_{t+1}|\mathbf{x}_t, r)p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}). \end{aligned} \tag{32}$$

The above factorization allows one to update $\zeta_t^{(i)}(.)$ recursively. Let us denote the conditional particle dynamic likelihood by

$$\xi_{t+1}^{(i)}(k) := p_f(\mathbf{x}_{t+1}^{(i)}|\mathbf{x}_t^{(i)}, k, t). \tag{33}$$

then we have

$$\zeta_{t+1}^{(i)}(k) = \zeta_t^{(i)}(k)\xi_{t+1}^{(i)}(k)g(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}, t+1). \tag{34}$$

for $i = 1, \cdots, N$ and $k = 1, \cdots, K+1$.

Finally, particle importance weights are updated according to

$$\omega_{t+1}^{(i)} = \frac{p(\mathbf{x}_{t+1}, \mathbf{y}_{t+1}|\tilde{\mathbf{x}}_{1:t}^{(i)}, \mathbf{y}_{1:t})}{p_s(\mathbf{x}_{t+1}|\tilde{\mathbf{x}}_{1:t}^{(i)}, \mathbf{y}_{1:t})}\tilde{\omega}_t^{(i)} \tag{35}$$

The values of $\omega_t^{(i)}$ are normalized over all particles after each iteration as usual in particle filtering.

The state transition probability in (33) as suggested by (16) is a Gaussian distribution

$$p_f(\mathbf{x}_{t+1}|\mathbf{x}_t^{(i)}, r) = \mathcal{N}(\mathbf{x}_t^{(i)} + [\mu_{r,x}^{(i)}, \mu_{r,y}^{(i)}]^T, \mathrm{diag}[\sigma_{r,x}^{(i)}, \sigma_{r,y}^{(i)}]) \tag{36}$$

where $\mu_{r,x}^{(i)}$, $\mu_{r,y}^{(i)}$, $\sigma_{r,x}^{(i)}$ and $\sigma_{r,y}^{(i)}$ are marginal mean and variance of the estimated flow components of cluster r at $\mathbf{x}_t^{(i)}$, which are extracted from all particles mean and covariance matrix computed according to (23) and (24) for each cluster. For cluster $K+1$, it is necessary to set $\rho_q(K+1) = 0$ for all trajectories observed before to enforce zero contribution of training data in the computations of the prior GP. Here BCM approximation is computed over all particles at the same time; in this way, the BCM performs better approximations since the number of evaluation points is large.

The mixture distribution of conditional transition probability densities is chosen here as the proposal distribution

$$p_s(\mathbf{x}_{t+1}|\tilde{\mathbf{x}}_{1:t}^{(i)}, \mathbf{y}_{1:t}) = \sum_{k=1}^{K+1} \tilde{\rho}_q^t(k) p_f(\mathbf{x}_{t+1}|\mathbf{x}_t, k) \tag{37}$$

where $\tilde{\rho}_q^t(k)$ is the corrected cluster probability of (26) computed up to time t.

B. Calculating required quantities

The variable $m_k(t)$ in (26) for $k = 1, \cdots, K$ shows the proportion of the trajectory where the likelihood of its dynamics given the kth flow function is approximately equal to the prior dynamic model $(K+1)$. It is calculated recursively according to

$$m_k(t+1) = \frac{tm_k(t) + w_\epsilon(|\hat{p}(\mathbf{x}_{t+1}|\mathbf{x}_t, k) - \hat{p}(\mathbf{x}_{t+1}|\mathbf{x}_t, K+1)|)}{t+1} \tag{38}$$

where $w_\epsilon(.)$ is a window function that is one if its argument is less than ϵ and zero otherwise. $\hat{p}(\mathbf{x}_{t+1}|\mathbf{x}_t, k)$ is the estimated conditional state transition likelihood

$$\hat{p}(\mathbf{x}_{t+1}|\mathbf{x}_t, k) = \sum_{i=1}^{N} \omega^{(i)} \xi_{t+1}^{(i)}(k) \tag{39}$$

The observed trajectory likelihood of (37) is computed by marginalizing the state trajectory from the complete data likelihood,

$$\gamma^{t+1}(k) = \sum_{i=1}^{N} \omega^{(i)} \zeta_{t+1}^{(i)}(k), \tag{40}$$

which together with $m_k(t+1)$ allows us to compute $\tilde{\rho}_q^{t+1}(k)$ for using in (37) at the next iteration.

We now turn to the computation of $\hat{\mathbf{x}}_{1:T}$ and $\hat{\dot{\mathbf{x}}}_{1:T}$, which are used in the training of GP. The expected state trajectory is simply the weighted average of the particle trajectory,

$$\hat{\mathbf{x}}_{1:T} = \sum_{i=1}^{N} \omega^{(i)} \mathbf{x}_{1:T}^{(i)}, \tag{41}$$

which can be calculated at each iteration of the filter or at the end of trajectory. However, the estimated flow however is computed at the each filter iteration according to the particle approximation of (20):

$$\hat{\dot{\mathbf{x}}}_t = \sum_{i=1}^{N}\sum_{j=1}^{N} \omega^{(i)}\omega^{(j)}(\mathbf{x}_{t+1}^{(i)} - \mathbf{x}_t^{(j)}). \tag{42}$$

C. RBPF Algorithm

Algorithm 2 summarizes an iteration of the proposed RBPF algorithm. In practice, each GP estimator uses a regularly down-sampled subset of its training points of length T for prediction. In this way, the computational order of the step 2 of the algorithm is $\mathcal{O}(qT^2N^2 + qN^3)$, where the cubic term is from the matrix inversions in (23) and (24) and quadratic term is from the matrix multiplication in (1) and (2). However, since only marginal values are needed in (36) all off-diagonal elements of the covariance matrices can be ignored before applying BCM equations. This makes computational complexity of matrix inversions linear with N. Furthermore, to prevent the scaling of the computation with the number of observed trajectories q, when it gets too large, a random subset of estimators with a fixed size $q' < q$ can be used, such that, at least one estimator is available for each class. However, this is not applied for the results of this paper and all observed trajectories have been used. Steps 3 and 6 of Algorithm 2 have computational complexity of $\mathcal{O}(KN)$ and step 9 is done in $\mathcal{O}(N^2)$.

Algorithm 2 An iteration of RB Particle filter

```
Input: ${\mathbf{x}_{1:t}^{(i)}, \zeta_t^{(i)}(.), \omega_t^{(i)}}_{i=1}^N$, ${\mathcal{D}_i}_{i=1}^q$, $\mathbf{y}_{t+1}^{q+1}$, $K$, $\phi$
Output:
1) Resampling
2) Calculate the mean and covariance of flow on ${\mathbf{x}_t^{(i)}}_{i=1}^N$ for every trajectory model $r = 1, \cdots, K+1$ from (23) and (24)
3) Compute parameters of $p_f(\mathbf{x}_{t+1}|\mathbf{x}_t^{(i)}, r)$ for $r = 1, \cdots, K+1$ and $i = 1, \cdots, N$ from (36)
4) Sample $\mathbf{x}_{t+1}^{(i)}$ for $i = 1, \cdots, N$ using (37)
5) Calculate $\omega_{t+1}^{(i)}$ for $i = 1, \cdots, N$ using (35)
6) Calculate $\zeta_{t+1}^{(i)}(k)$ for $k = 1, \cdots, K+1$ and $i = 1, \cdots, N$ from (34)
7) Calculate $m_k(t+1)$ and $\gamma^{t+1}(k)$ for $k = 1, \cdots, K+1$ from (38) and (40)
8) Calculate $\tilde{\rho}_{q+1}^t(r)$ $r = 1, \cdots, K+1$ from (26)
9) Calculate $\hat{\dot{\mathbf{x}}}_t$ using (42)
```

VII. Evaluation

In this section, the proposed algorithm is evaluated in different conditions and for a number of different tasks. Performances of the proposed algorithm in clustering trajectory patterns is compared with a baseline spectral clustering algorithm and more recent mean-shift based Multi-Feature Clustering [21] and DPM based incremental trajectory clustering of [15]. The behavior of the algorithm is considered for different parameter sets. The performance of proposed algorithm in online abnormality detection and activity classification is shown and compared with GP based supervised algorithm of [4]. Finally, it is demonstrated that the proposed algorithm improves the performances of the tracker by learning realistic dynamical models.

A. Experimental Setup and Data Sets

The proposed algorithm has been tested with different choices of parameters to evalute its sensitivity to different configuration. The concentration parameter of DP α is chosen from $\{0.2, 0.5, 2, 5\}$ and the dissimilarity threshold τ from $\{0.15, 0.25, 0.5\}$. These two parameters mainly control the sensitivity of the algorithm for making a new clusters when a deviation occurs. The range of variables is selected such that resulting number of clusters will be around the actual expected number (5 to 20 clusters). The number of particles N is equal to 500 in all experiments. The ϵ parameter in (38) is set to

10^{-3} to be small enough. The Squared Exponential covariance function with Automatic Relevance Determination (SE-ARD) [24] was chosen as the covariance kernel of the GPs:

$$k(\mathbf{x},\mathbf{x}',t,t') = \theta_s^2 \exp\left(-\frac{(x-x')^2}{2e^{\theta_{lx}}} - \frac{(y-y')^2}{2e^{\theta_{ly}}} - \frac{(t-t')^2}{2e^{\theta_{lt}}}\right), \quad (43)$$

where $e^{\theta_{lx}}$, $e^{\theta_{ly}}$ and $e^{\theta_{lt}}$ are length-scale parameters for each coordinate respectively. θ_s^2 is the variance parameter of the kernel, which is fixed in the experiments for every dataset. This parameter mainly affects the uncertainty about flow values in GP and will therefore affect span of the sampling in RBPF. As general rule, higher values should be preferred when the speed of objects in video are higher and more diverse. Normal distributions with the unit variance and mean chosen from $\{1, 2\}$ is used as the prior of θ_{lx}, θ_{ly} and θ_{lt} in the MAP estimation of kernel hyper-parameters. In the implementation, the GPML library[1] is used for training and evaluation of the GP. Using this kernel, the prior flow function becomes a zero-mean white noise process with variance θ_s^2 at every position and time.

Three datasets are used for the evaluation of the proposed algorithm (Fig. 4). The *Highway* (Fig. 4b) dataset provided by authors of [21][2] consists of 134 trajectories in 10 classes and the *Traffic* (Fig. 4c) dataset provided by authors of [15] consists of 1500 trajectories in 15 classes. These two datasets are used in the clustering performance evaluation. The PDTV dataset [35] consists of 51 trajectories in 16 classes (Fig. 4a). The PDTV dataset is used for demonstrating clustering performance as well as classification and abnormality detection. The value of kernel variance parameter θ_s^2 in PDTV and *Traffic* is chosen from $\{15, 30\}$ while in *Highway* larger values are chosen from $\{90, 120\}$ since there is higher variability in the speed of the objects on image plane.

The state and observation vector are the position of objects in the image plane. A Gaussian measurement model is assumed in (17) with the covariance matrix equal to the covariance of the object's foreground pixel positions. The observation vector $\mathbf{y}_t$ is extracted from each frame by computing the mean of the foreground pixel positions associated to the object. The segmented foreground pixels of each object is available in PDTV dataset and they are used as inputs. In the *Traffic* and *Highway* datasets only the trajectory points are available. Thus, trajectory points are regarded as $\mathbf{y}_t$ and a synthetic Gaussian noise is applied to the model (17).

B. Baseline Algorithms and Evaluation Criterion

The spectral clustering algorithm [36] is proposed in [14] for unsupervised trajectory clustering. The Longest Common Subsequence (LCSS) [37] and Dynamic Time Wrapping (DTW) [38] are used as distance measures for spectral clustering. Both distance measures are able to eliminate the effect of misalignment and different length of trajectories. Unlike our method, the spectral clustering algorithm is neither incremental nor it is able to estimate the number of clusters. In addition, the proposed algorithm is compared with the Multi-Feature Trajectory Clustering (MFTC) algorithm [21] and DPM trajectory clustering algorithm [15], which are able to estimate the number of clusters. The former method is a batch clustering algorithm while the latter can be employed in batch and incremental modes.

[1]http://www.gaussianprocess.org/gpml/code/matlab/doc/

[2]available from ftp://motinas.elec.qmul.ac.uk/pub/code/nadeem/QMUL_MFTC_software.zip

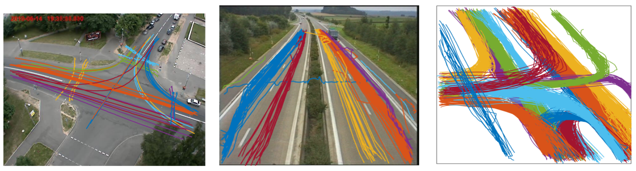

(a) PDTV (b) Highway (c) Traffic

Fig. 4: Datasets used in evaluation. Colors show ground-truth classes of trajectories.

The *Accuracy* of clustering is defined as the ratio of the maximum number of trajectories with the same ground-truth label in the cluster to the total number of trajectories in that cluster. The overall *Accuracy* is then the mean of the accuracy of all clusters [15]. The clustering *Accuracy* is the measure that shows how well the algorithm is able to build clusters of similar trajectories. It does not take into account the number of actual clusters. When the number of clusters is equal to the samples, the value of *Accuracy* is maximum. A more accurate criteria is the Adjusted Mutual Information (AMI) [39] that compares the result of clustering with the ground-truth, while eliminating the effect of random matching and the different permutation of class labels. The metric has values in range [0 1]: values close to zero mean two assignments are highly independent and values close to one indicate agreement in them. The AMI, unlike accuracy, is sensitive to the number of clustering labels.

In online classification and abnormality detection tasks, the proposed method is compared with the Gaussian Process Regression of Flow (GPRF) algorithm [4]. However, since GPRF is a supervised algorithm, the class labels of trajectories is used for its training purpose. For fairness of comparison, the GPRF is trained with q trajectories in the dataset and then applies the classification on $q + 1$ because the proposed algorithm uses only information from first q trajectories for processing $(q + 1)$th trajectory. Average per-class precision and recall [40] is used for online classification comparison and Receiver Operation Curve (ROC) is used for comparing abnormality detection performances.

C. Clustering Results

The effect of parameter's choice on the clustering algorithm is shown in the Fig. 5 for the PDTV Dataset. The figure depicts the number of learned clusters versus the DP concentration parameter α for different choices of dissimilarity threshold τ. The clustering behavior highly adapts with the choice of τ, while its sensitivity to α is less significant. The optimum choice of these parameters, however, depends on the recording environment. This is more evident when the cluster growing

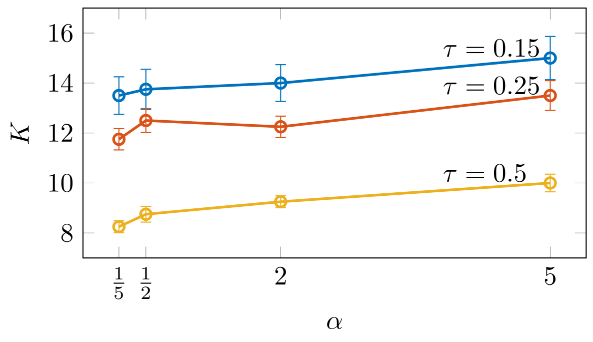
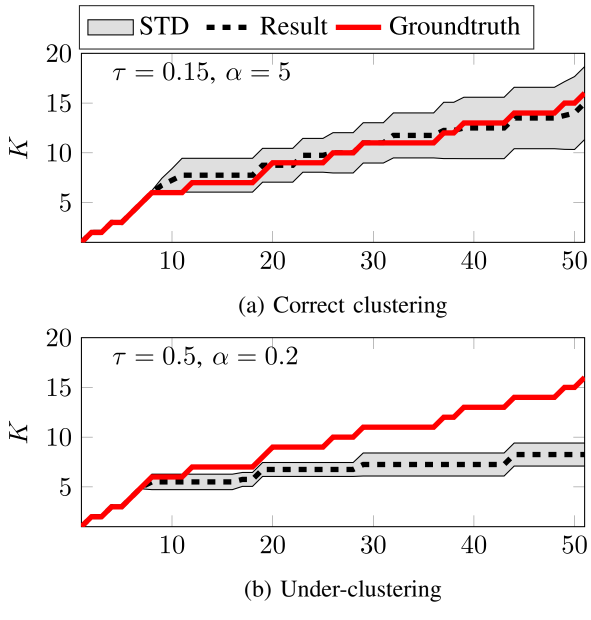


Fig. 5: Number of learned clusters K versus the DP concentration parameter α and dissimilarity threshold τ.



(a) Correct clustering

(b) Under-clustering

Fig. 6: Cluster growing rate versus the number of objects observed. Red line is the ground-truth. Horizontal axis is the number of trajectories observed (with fixed same ordering in all tests and ground-truth). Dashed line is the mean of results and gray region is the standard deviation of results.

rate is compared with the number of observed objects in the video shown in Fig. 6. An appropriate choice of parameters as in Fig. 6a results in the same growing rate as ground-truth. On the other hand, choosing larger τ and smaller α reduces the sensitivity of the algorithm to outliers and its tendency to merge more similar trajectories in the same group.

The performances of the clustering algorithm are shown and compared with other algorithms in Fig. 7 for PDTV and Fig. 8 for *Highway* dataset. The horizontal lines in these figures are the number of resulting clusters that is learned in the proposed algorithm and MFTC, while it should be set a priori in the spectral clustering algorithm. For MFTC algorithm the code provided by the authors of [21] is used. As depicted in Fig. 7 the proposed algorithm is more accurate than spectral clustering algorithm and it is comparable to MFTC algorithm in PDTV dataset. In the *Highway* dataset experiment (Fig. 8), our algorithm outperforms others in higher number of clusters.

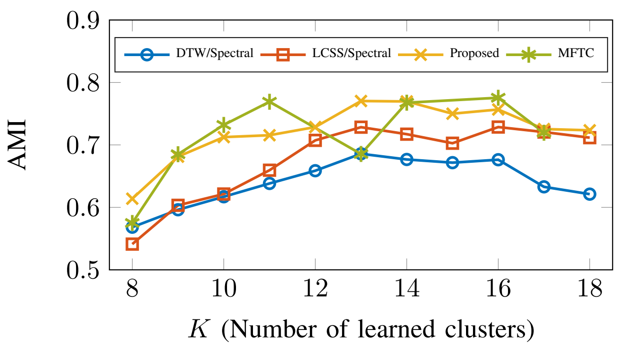


Fig. 7: Comparison of proposed algorithm clustering performance on PDTV dataset.

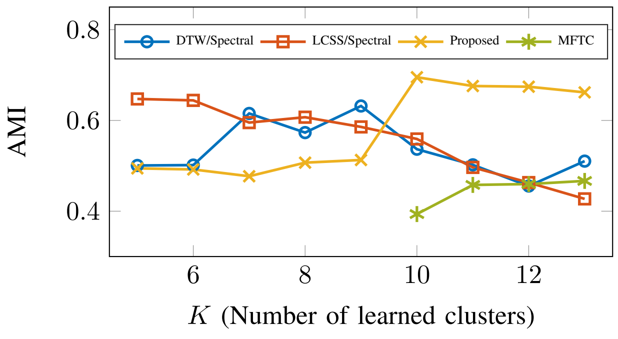


Fig. 8: Comparison of proposed algorithm clustering performance on *Highway* dataset.

In this dataset, MTFC does not result in less than 10 clusters with all parameter tunings. However, in the *Highway* dataset, with a lower number of clusters, the parametric spectral algorithm provides better performance than the proposed.

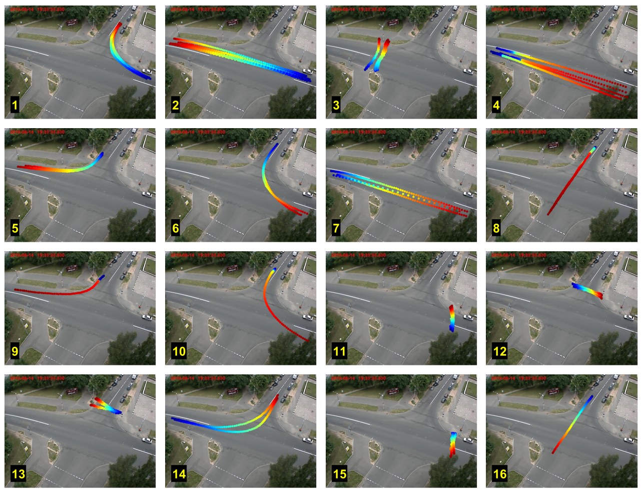
For comparison with DPM [15], the proposed algorithm is applied on the *Traffic* dataset and the result is compared with the results from [15] for the same dataset. As shown in Table II, the proposed algorithm outperforms the DPM by more than 10% *Accuracy* in its incremental mode and more than 7% *Accuracy* in its batch mode. The AMI for the DPM algorithm is not presented in Table II, because the AMI measure is not used in [15] for performance validation. The number of resulting clusters for the proposed algorithm was 19 for the values in the Table II, while for DPM the number of resulting clusters is unknown.

A typical qualitative result for PDTV dataset is shown in Fig. 9. The 16 ground-truth classes are shown in Fig. 9a and the resulting 19 clusters are shown in Fig. 9b. First type of clustering error occurs when the algorithm puts trajectories from different classes in one cluster as in cluster 10 in Fig.

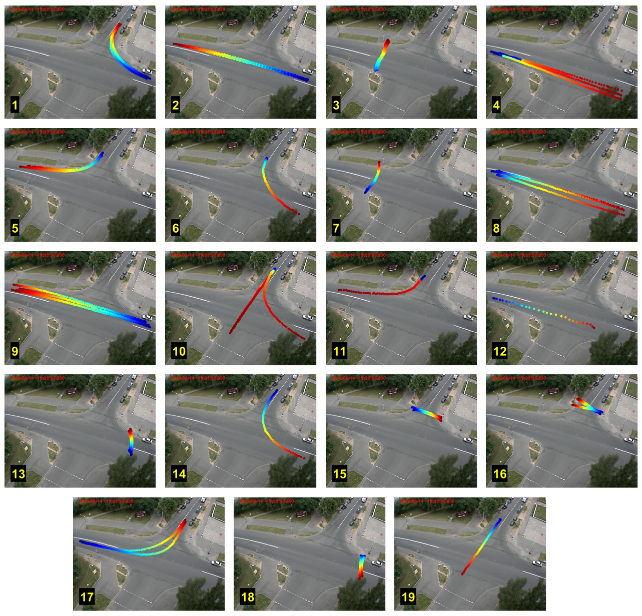
TABLE II: Comparison of Incremental Clustering Accuracy on *Traffic* Dataset

Method	DPM Batch [15]	DPM Incerementel [15]	Proposed
Accuracy	90.0%*	85.1%*	97.6%
AMI	-	-	0.72

* values are taken from [15]



(a) Ground-truth Classes



(b) Resulting Clusters

Fig. 9: Trajectory clusters in the PDTV dataset. (Color hue represents the time from blue to red.)

9b. In this particular case, the two trajectories correspond to cars that stop for long time behind light and then move away. In this case, since trajectories are similar for long time and then starts diverging for short duration, the deviation was not enough to stimulate creation of new cluster. The second type of error is when the algorithm splits trajectories of the same class into separate clusters. This happened for classes 2, 3 and 7 of ground-truth Fig. 9a. Class 2 is split into clusters 2 and 9 in Fig. 9b. Class 3 is split into clusters 3 and 7 in Fig. 9b. Class 7 is split into clusters 8 and 12 in Fig. 9b. However, careful examination shows that split trajectories are slightly different with each other either in terms of speed (as in cluster 8 and 12 in Fig. 9b) or shape and position of trajectory (as in 3 and 7 in Fig. 9b).

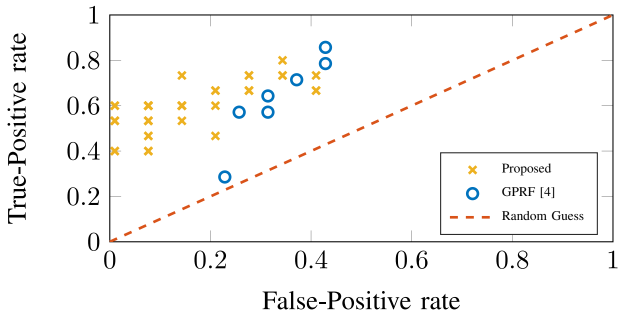


Fig. 10: Abnormality detection performance of the proposed algorithm in ROC space.

D. Abnormality Detection Results

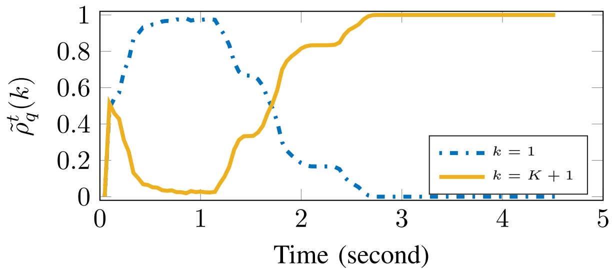
Abnormality in this paper is defined as a behavior (trajectory) pattern that has not been observed and learned before. This definition complies with [4] where the model is trained in a supervised way using normal trajectories, then it is used to detect abnormalities, However, abnormality may also be defined as detecting outliers in a dataset like in [17] where sparse trajectories are detected as abnormal. The former definition is closely related to the incremental clustering ability of the proposed algorithm, where generation of a new cluster corresponds to identify that the received trajectory does not match to any pattern learned up to that processing moment. This can be done online by considering the posterior class probability from equation (26) since $\tilde{\rho}^t_{q+1}(K+1)$ equals the probability that the trajectory belongs to an unseen class.

However, in practice, not all trajectories that are detected as new classes belong to semantically abnormal behavior (e.g. abnormal traffic action). Thus, a post processing is required to filter semantically abnormal trajectories and prevent the system from learning them (learning new clusters is default in proposed method). This can be done by the intervention of a user or an automated system to either act when an event of making a new cluster (abnormality detection) is generated or by revising the learned clusters later. In this section, the abnormality detection performance of the proposed method is presented as detecting the event that a trajectory belongs to a new cluster. This is equivalent to suppose that all learned trajectory patterns up to a moment are normal then testing abnormality for a new incoming trajectories.

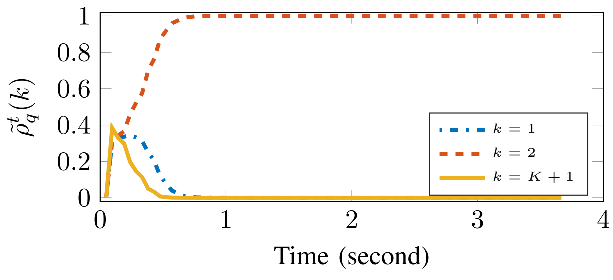
The performance of the proposed algorithm in detecting anomalies in the PDTV dataset is shown in ROC (Receiver Operating Curve) of Fig. 10 and compared [4]. The points in Fig. 10 are achieved with different parameter sets in our algorithm and different abnormality detection thresholds in [4]. The results show that the proposed algorithm is able to achieve higher true-positive rate with lower false-positive rate than the algorithm of [4].

E. Sequential Classification Results

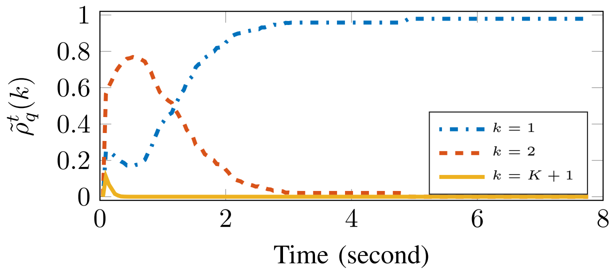
Calculating the maximum of $\tilde{\rho}^t_q(k)$ from (26) at each time instant provides a classification result for the trajectory being processed. The classification accuracy improves as more observations are made from the object. Fig. 11 provides three



(a)



(b)



(c)

Fig. 11: Online trajectory classification of trajectories number (a) 2, (b) 3 and (c) 45.

examples of online classification for the trajectories of objects numbers 2, 3 and 45 from the PDTV dataset respectively, which corresponds to class 1, 2 and 1 of Fig. 9a . The average $\tilde{\rho}_q^t(k)$ at each time instant over all experiments is shown for three particular objects. Object number 2 corresponds to the second trajectory received in the algorithm ($q = 2$). From Fig. 11a it is evident that it was first classified as class 1 due to the initial similarity between class 1 and class 2 trajectories (Fig. 9a). As time passes and it starts to deviate from class 1 pattern, it is identified as a new class $K + 1$, where $K = 1$ in this case. In Fig. 11b, apart from a short initial ambiguity, object number 3 is identified correctly as class 1. Fig. 11c is an example of an initial mis-classification where object number 45, which is from class 2, is initially classified to class 1. This is again due to the high similarity between the initial shapes of class 1 and 2. However, as soon as the discriminative feature of the trajectory is observed, the probability of the true class increases until it is very close to one.

The online classification performance of the proposed algorithm is compared with the supervised GPRF algorithm [4] in Fig. 12. The average per-class precision and recall [40] for the classification results is shown when different portions of trajectories are observed from the beginning. Considering the

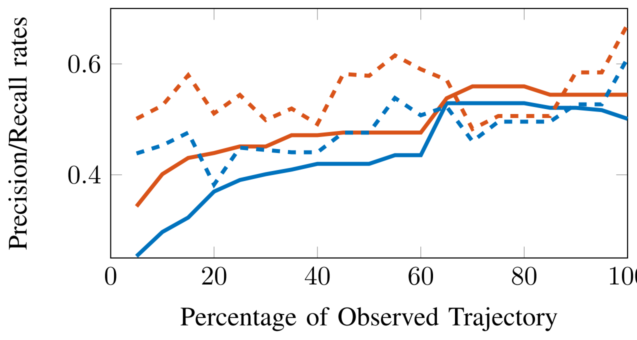


Fig. 12: Comparison of online classification performances. Precision and recall of the proposed algorithm is shown with the solid blue and red lines respectively. Precision and recall of the GPRF [4] is shown with the dashed blue and red lines respectively.

fact that the classification error in the proposed unsupervised algorithm may also be caused by the error in trajectory clustering, the proposed algorithm still gives comparable results to supervised GPRF algorithm where more than 60% of a trajectory is observed.

F. Tracking Improvement by Learning Flows

Object tracking is an integrated part of the proposed algorithm, which is realized using the RBPF. In traditional particle filter tracking algorithms, the dynamical model (16) has to be specified beforehand and they are usually linear and stationary models. In contrast, in the proposed model, dynamics of objects are learned and updated as flow function by each trajectory the system receives. For a given environment, the incremental clustering algorithm gradually builds a library of realistic flow models that are used in the particle filter. This improves the system's tracking ability with each experience. This effect is shown using two examples in Fig. 13, the average variance of the filtered state over a whole trajectory is provided versus the number trajectories from the same class observed by the system beforehand. Results shown in Fig. 13 correspond to the first and second trajectory classes in Fig. 9a. In both cases of Fig. 13, the tracking variance is higher when the trajectory class is observed for the first time. However, the performance improves (tracking variance decrease) as more instances of same trajectory pattern is seen, which allows the system to learn better flow function models for the corresponding trajectory pattern.

G. Computation Time

The algorithm is implemented in MATLAB® R2014b with unoptimized code[3]. Computation time of the algorithm on an Intel® Core™ i7 3.40GHz CPU and 12.0GB memory is presented in Fig. 14 as the frame-per-second (inverse of computation time of each frame) versus number of clusters.

[3]code is available from http://www.isip40.it/codes/dpgpf_activity_mining.zip

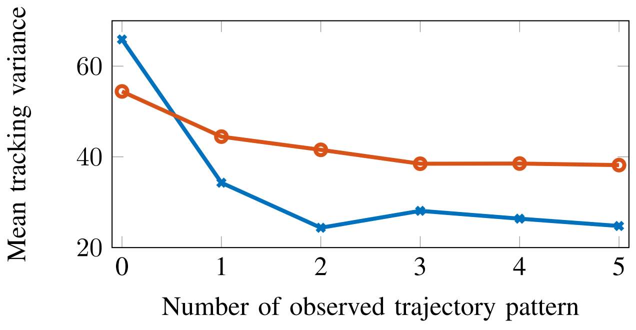


Fig. 13: Performance of tracking an object performing trajectory class 1 (blue) and 2 (red) versus the number of instances observed from the same trajectory class before.

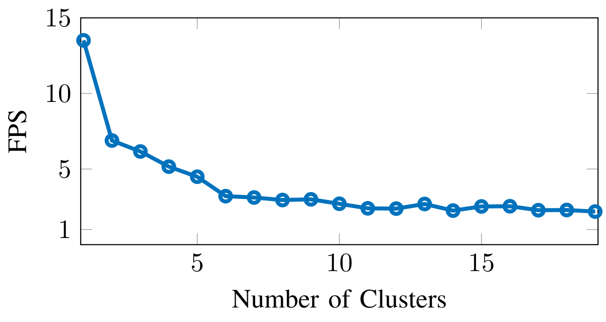


Fig. 14: Computation Time

VIII. Conclusion

A method for mining activity patterns from stream of surveillance video is presented in this paper. Advanced Bayesian nonparametric techniques of Dirichlet process mixture and Gaussian process are employed for incremental clustering and learning of trajectory patterns characterized by non-linear time-variant flow functions. A Rao-blakwellized particle filter is proposed for tracking and online classification of trajectory pattern and abnormality detection. Experimental results demonstrated the superior performance of the proposed algorithm as compared to state-of-the-art trajectory clustering algorithms. Moreover, the online classification and abnormality detection performances are presented for real video data. Furthermore, it is suggested that the proposed framework can be employed also to improve tracking performances as it is able to identify real dynamics of objects by learning them from experience.

In this paper, objects are assumed as points where their state is defined as their position in the image plane. A future direction of this work is to extend it for understanding more complicated trajectories of extended and group objects where higher dimensional state vectors would be used. Another future work is to use the idea of the proposed framework to understand interactions between objects in the scene. In this case, the dynamics of the objects are influenced by the environmental situation and the presence of other objects around.

References

[1] T. Huang, "Surveillance video: The biggest big data," *Computing Now*, vol. 7, no. 2, Feb. 2014. [Online]. Available: http://www.computer.org/web/computingnow/archive/february2014

[2] R. Emonet, J. Varadarajan, and J. M. Odobez, "Temporal analysis of motif mixtures using dirichlet processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 1, pp. 140–156, Jan 2014.

[3] I. Saleemi, L. Hartung, and M. Shah, "Scene understanding by statistical modeling of motion patterns," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, pp. 2069–2076.

[4] K. Kim, D. Lee, and I. Essa, "Gaussian process regression flow for analysis of motion trajectories," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, Nov 2011, pp. 1164–1171.

[5] J. C. Nascimento, M. A. T. Figueiredo, and J. S. Marques, "Trajectory classification using switched dynamical hidden markov models," *IEEE Transactions on Image Processing*, vol. 19, no. 5, pp. 1338–1348, May 2010.

[6] J. Nascimento, M. Figueiredo, and J. Marques, "Activity recognition using a mixture of vector fields," *Image Processing, IEEE Transactions on*, vol. 22, no. 5, pp. 1712–1725, May 2013.

[7] A. Dore and C. Regazzoni, "Interaction analysis with a bayesian trajectory model," *IEEE Intelligent Systems*, vol. 25, no. 3, pp. 32–40, 2010.

[8] F. Castaldo, F. A. N. Palmieri, V. Bastani, L. Marcenaro, and C. Regazzoni, "Abnormal vessel behavior detection in port areas based on dynamic bayesian networks," in *Information Fusion (FUSION), 2014 17th International Conference on*, July 2014, pp. 1–7.

[9] V. Bastani, L. Marcenaro, and C. Regazzoni, "A particle filter based sequential trajectory classifier for behavior analysis in video surveillance," in *Image Processing (ICIP), 2015 IEEE International Conference on*, Sept 2015, pp. 3690–3694.

[10] L. Marcenaro, L. Marchesotti, and C. Regazzoni, "Self-organizing shape description for tracking and classifying multiple interacting objects," *Image and Vision Computing*, vol. 24, no. 11, pp. 1179–1191, 2006.

[11] B. T. Morris and M. M. Trivedi, "Understanding vehicular traffic behavior from video: a survey of unsupervised approaches," *Journal of Electronic Imaging*, vol. 22, no. 4, p. 041113, Sep. 2013.

[12] V. Bastani, L. Marcenaro, and C. Regazzoni, "Unsupervised trajectory pattern classification using hierarchical dirichlet process mixture hidden markov model," in *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, Sept 2014, pp. 1–6.

[13] T. Nawaz, A. Cavallaro, and B. Rinner, "Trajectory clustering for motion pattern extraction in aerial videos," in *Image Processing (ICIP), 2014 IEEE International Conference on*, Oct 2014, pp. 1016–1020.

[14] B. T. Morris and M. M. Trivedi, "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2287–2301, Nov 2011.

[15] W. Hu, X. Li, G. Tian, S. Maybank, and Z. Zhang, "An incremental dpmm-based method for trajectory clustering, modeling, and retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 5, pp. 1051–1065, May 2013.

[16] C. Piciarelli, C. Micheloni, and G. L. Foresti, "Trajectory-based anomalous event detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1544–1554, Nov 2008.

[17] R. Laxhammar and G. Falkman, "Online learning and sequential anomaly detection in trajectories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 6, pp. 1158–1173, June 2014.

[18] F. Jiang, Y. Wu, and A. K. Katsaggelos, "A dynamic hierarchical clustering method for trajectory-based unusual video event detection," *IEEE Transactions on Image Processing*, vol. 18, no. 4, pp. 907–913, April 2009.

[19] D. Comaniciu and P. Meer, "Distribution free decomposition of multivariate data," *Pattern Analysis and Applications*, vol. 2, pp. 22–30, 1998.

[20] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical Dirichlet Processes," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1566–1581, Dec. 2006.

[21] N. Anjum and A. Cavallaro, "Multifeature object trajectory clustering for video analysis," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 11, pp. 1555–1564, Nov 2008.

[22] D. Lin, "Online learning of nonparametric mixture models via sequential variational approximation," in *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., 2013, pp. 395–403.

[23] E. zkan, F. Lindsten, C. Fritsche, and F. Gustafsson, "Recursive maximum likelihood identification of jump markov nonlinear systems," Feb 2015.

[24] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptative computation and machine learning series. University Press Group Limited, 2006.

[25] D. Nguyen-tuong, J. R. Peters, and M. Seeger, "Local gaussian process regression for real time online model learning," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 1193–1200.

[26] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Active learning with gaussian processes for object categorization," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, Oct 2007, pp. 1–8.

[27] V. Tresp, "A Bayesian Committee Machine," *Neural Computation*, vol. 12, no. 11, pp. 2719–2741, Nov. 2000.

[28] D. Blei, L. Carin, and D. Dunson, "Probabilistic topic models," *IEEE Signal Processing Magazine*, vol. 27, no. 6, pp. 55–65, Nov 2010.

[29] J. Ross and J. Dy, "Nonparametric mixture of gaussian processes with constraints," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, vol. 28, no. 3. JMLR Workshop and Conference Proceedings, May 2013, pp. 1346–1354.

[30] J. Hensman, M. Rattray, and N. Lawrence, "Fast nonparametric clustering of structured time-series," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 37, no. 2, pp. 383–393, Feb 2015.

[31] D. M. Blei and M. I. Jordan, "Variational inference for dirichlet process mixtures," *Bayesian Analysis*, vol. 1, pp. 121–144, 2005.

[32] M. Bryant and E. B. Sudderth, "Truly nonparametric online variational inference for hierarchical dirichlet processes," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 2708–2716.

[33] L. Wang and D. B. Dunson, "Fast Bayesian Inference in Dirichlet Process Mixture Models." *Journal of computational and graphical statistics*, vol. 20, no. 1, Jan. 2011.

[34] 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.

[35] N. Saunier, H. k. Ardö, J.-P. Jodoin, A. Laureshyn, M. Nilsson, A. s. Svensson, L. Miranda-Moreno, G.-A. Bilodeau, and K. Å strö m, "A public video dataset for road transportation applications," in *2014 Transportation Research Board 93rd Annual Meeting*, Jan. 2014.

[36] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug 2000.

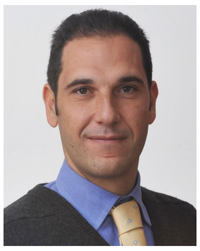
[37] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Data Engineering, 2002. Proceedings. 18th International Conference on*, 2002, pp. 673–684.

[38] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, Feb 1978.

[39] N. X. Vinh, J. Epps, and J. Bailey, "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance," *The Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, Mar. 2010.

[40] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427 – 437, 2009.

Vahid Bastani (S'14) is currently working toward the Ph.D. degree with the Information and Signal Processing for Telecommunication (ISIP40) Lab, University of Genova (Italy). He received Master of Telecommunication Systems Engineering from Shiraz University (Iran) in 2012 and Bachelor of Electrical and Electronics Engineering from Shiraz University of Technology (Iran) in 2009. Vahid's research interests include machine learning and signal processing techniques applied to video, time-series and dynamical system analysis. He serves as reviewer for IEEE Transactions on Circuits and Systems for Video Technology, Journal of Visual Communication and Image Representation and EURASIP Journal on Image and Video Processing.



Lucio Marcenaro (M'11) enjoys over 15 years experience in image and video sequence analysis, and authored about 100 technical papers related to signal and video processing for computer vision. An Electronic Engineering graduate from Genova University in 1999, he received his PhD in Computer Science and Electronic Engineering from the same University in 2003. From 2003 to 2010 he was CEO and development manager at TechnoAware srl. From March 2011, he became Assistant Professor in Telecommunications for the Faculty of Engineering at the Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture (DITEN) at the University of Genova where he teaches the courses of Pervasive Electronics and Computer Programming and Telematics Lab. He is the principal scientific and technical coordinator of the Ambient Awareness Lab (A2Lab), with TechnoAware srl. His main current research interests are: video processing for event recognition, detection and localization of objects in complex scenes, distributed heterogeneous sensors ambient awareness systems, ambient intelligence and bio-inspired cognitive systems. Lucio Marcenaro was General Chair of the Symposium on Signal Processing for Understanding Crowd Dynamics organized within the 2016 IEEE GlobalSIP. He was Industrial Chair of the 6th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS2009). He is in the technical committee of many surveillance related international conferences (IEEE AVSS, ICDP, ICIP) and international journals (IEEE TCSVT, Pattern Recognition Letters, Transactions on Sensor Networks). Since 2015 Lucio Marcenaro is an Expert on video analytics for ERNCIP Video Analytics & Surveillance for Security of Critical Infrastructures Thematic Group of the European Union.

Carlo S. Regazzoni (SM'00) is full professor of Cognitive Telecommunications Systems at DITEN, University of Genova, Italy. His main research interests include (see at www.isip40.it) cognitive dynamic systems, adaptive and self-aware video processing, tracking and recognition, generative models and inference schemes based on hierarchical dynamic Bayesian networks, software and cognitive radio. He has been responsible of several national and EU funded research projects. He is currently coordinator of international PhD courses on Interactive and Cognitive Environments involving several European universities. He is author of peer-reviewed papers on international journals (80) and international conferences/books (350). He served as general chair (IEEE AVSS2009), technical program chair (IEEE ICIP2005, NSIP2002), associate editor (IEEE Trans on Image Processing, IEEE Trans on Mobile Computing, et al.), guest editor (Proceedings of the IEEE, IEEE Signal Processing Magazine et al.) in international conferences and Journals. He has served in many roles in governance bodies of IEEE SPS. He is serving as Vice President Conferences IEEE Signal Processing Society in 2015-2017.