Learning Pedestrian Trajectories with Kernels

Elisa Ricci, Francesco Tobia, Gloria Zen
Fondazione Bruno Kessler, FBK-irst, via Sommarive 18, Povo, 38100 Trento, Italy
{*eliricci,tobia,gzen*}*@fbk.eu*

Abstract

We present a novel method for learning pedestrian trajectories which is able to describe complex motion patterns such as multiple crossing paths. This approach adopts Kernel Canonical Correlation Analysis (KCCA) to build a mapping between the physical location space and the trajectory patterns space. To model crossing paths we rely on a clustering algorithm based on Kernel K-means with a Dynamic Time Warping (DTW) kernel. We demonstrate the effectiveness of our method incorporating the learned motion model into a multi-person tracking algorithm and testing it on several video surveillance sequences.

1. Introduction

Being able to automatically extract recurrent patterns is of crucial importance for many video surveillance applications such as visual object tracking or anomaly detection. An example of recurrent patterns is represented by pedestrian trajectories. Usually pedestrians tend to follow only few common trajectories while other paths are infrequent or never observed. Previous works have shown how motion patterns can be described with parametric models learned from a typical set of trajectories collected offline. For example in [3, 9] splines are used to represent human paths: the resulting algorithms are simple and computationally efficient but fail to model complex situations of multiple intersecting paths. Other approaches [1] consider a single crowd motion model. However these methods are more suited to describe crowded scenes and are not appropriate in a scenario as the one considered in this work where only few people are moving around following multiple paths.

In this paper we propose a novel approach to learn typical pedestrian paths. We first cluster trajectories using Kernel K-means equipped a DTW kernel [4] to effectively compare paths of different length. Then we learn a function which allows to estimate the instantaneous velocity of a target given its current position and the cluster membership. To this aim we propose a novel algorithm based on Kernel Canonical Correlation Analysis (KCCA) [6]: the idea is that in this way we impose points that are close together in the input space to have highly correlated outputs. This is meant to model the fact that usually neighboring points along similar trajectories correspond to movements in similar directions. The proposed approach has several advantages. First of all, the instantaneous velocity can be treated as a continuous valued vector allowing more accurate estimates w.r.t. previous methods which adopt discrete representations [2]. Second, by the kernels, both K-means and KCCA effectively capture the nonlinear structure of the data. Finally by first clustering trajectories and then incorporating the cluster membership in the algorithm which predicts the trajectory patterns we are able to model complex situations such as crossing paths.

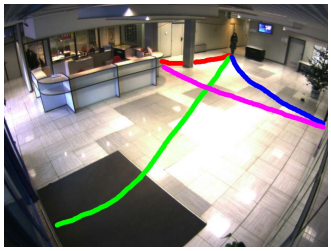


Figure 1. Typical paths in our scenario.

The contribution of this paper are threefold. (i) We show that the problem of learning motion patterns can be modeled as a multivariate regression task (*i.e.* we predict multiple outputs - the components of the instantaneous velocity vector- together) and that (ii) the learned motion model can be successfully incorporated into a multi-person tracking algorithm. (iii) We demonstrate that the DTW kernel previously proposed in [4] in the contest of speech recognition can be used effectively for pedestrian trajectory analysis.

2. Kernel Canonical Correlation Analysis

KCCA [6] has been proposed as the nonlinear extension of Canonical Correlation Analysis (CCA) by

the use of kernels. CCA [7] is a powerful statistical tool well suited to address tasks where we need to establish a relation between two sets of measurements. More specifically, given two matrices $\mathbf{S} \in R^{N\times M}$ and $\mathbf{T} \in R^{N\times Q}$ representing two sets of measurements, CCA finds directions (the canonical vectors) $\mathbf{w}_s$ and $\mathbf{w}_t$ such that the projections $\mathbf{w}_s^T\mathbf{S}$ and $\mathbf{w}_t^T\mathbf{T}$ are maximally correlated. While in CCA only linear projections are considered, KCCA account for nonlinear mappings. In analogy with other kernel methods ($e.g.$ Support Vector Machines), KCCA implicitly maps vectors $\mathbf{s}$ and $\mathbf{t}$ into an high-dimensional feature space through the transformations $\phi(\mathbf{s})$ and $\psi(\mathbf{t})$, and then performs traditional CCA in the two feature spaces. Considering the centered feature matrices $\mathbf{\Phi}_s$ and $\mathbf{\Phi}_t$ and the dual representations for the projection directions $\mathbf{w}_s = \mathbf{\Phi}_s^T\boldsymbol{\sigma}$ and $\mathbf{w}_t = \mathbf{\Phi}_t^T\boldsymbol{\tau}$ the problem of KCCA reduces to find $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$. It can be shown [6] that $\boldsymbol{\sigma}$ can be computed solving the generalized eigenproblem:

$$(\mathbf{K}_s + \gamma\mathbf{I})^{-1}\mathbf{K}_t(\mathbf{K}_s + \gamma\mathbf{I})^{-1}\mathbf{K}_t\boldsymbol{\sigma} = \lambda^2\boldsymbol{\sigma} \qquad (1)$$

where $\mathbf{K}_s$ and $\mathbf{K}_t$ denote the two centered kernel matrices with $[\mathbf{K}_s]_{i,j} = k_s(\mathbf{s}_i, \mathbf{s}_j)$ and $[\mathbf{K}_t]_{i,j} = k_t(\mathbf{t}_i, \mathbf{t}_j)$, I is the identity matrix and γ is a user-defined regularization parameter. Regularization is introduced to avoid overfitting penalizing the norms of the weight vectors. A typical choice for $k(\cdot)$ is the gaussian kernel $k(\mathbf{x}, \mathbf{y}) = e^{\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}}$. Then $\boldsymbol{\tau}$ can be obtained as:

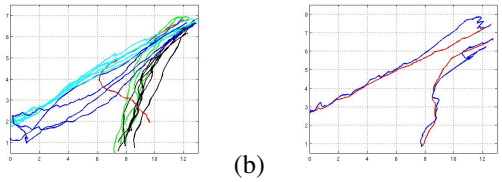
$$\boldsymbol{\tau} = \frac{1}{\lambda}(\mathbf{K}_s + \gamma\mathbf{I})^{-1}\mathbf{K}_t\boldsymbol{\sigma} \qquad (2)$$

Given any new datapoint $\tilde{\mathbf{s}}$, its projection $\boldsymbol{\pi}(\tilde{\mathbf{s}})$ on $\mathbf{w}_s$ is given by $\boldsymbol{\pi}(\tilde{\mathbf{s}}) = \mathbf{k}_{\tilde{s}}^T\boldsymbol{\sigma}$ with $[\mathbf{k}_{\tilde{s}}]_i = k(\mathbf{s}_i, \tilde{\mathbf{s}})$. Similarly, the projection of any $\tilde{\mathbf{t}}$ onto $\mathbf{w}_t$ is $\mathbf{k}_{\tilde{t}}^T\boldsymbol{\tau}$.

3. Learning Typical Pedestrian Paths

Our approach consists in three main phases. In the fist step ($preprocessing$) we automatically collect a set of trajectories and filter out noisy data. In the second phase ($clustering$) the paths are grouped according to their similarity. Finally during $modeling$ a compact representation of all observed trajectory patterns is learned. In the following we describe the three phases.

Preprocessing. We consider the scenario depicted in Fig.1 and we use a color-based particle filter [8] to track several people and automatically extract their paths. Since this phase is fully automatic the data tend to be noisy due to situations of tracking instability or failure. Therefore we manually discard outliers and we further remove noise from each trajectory by smoothing it using a moving average filter. We end up with a dataset of 80 different paths.



(a) (b)

Figure 2. (a) Clustered trajectories. (b) Paths extracted tracking with (red) and without (blue) the learned motion model

Clustering. We represent a trajectory as a sequence of flow vectors $F = \{\mathbf{f}_1, \ldots \mathbf{f}_M\}$ where $\mathbf{f}_t = [x_t,\ y_t,\ \delta x_t,\ \delta y_t]$ contains the position and the velocity of the target at time t. Then we group the flow vectors F_i according to their similarity. To be able to detect nonlinear clusters we resort on kernel methods and specifically on Kernel K-means (KK-means) [11]. In KK-means points are mapped to a higher-dimensional feature space using a nonlinear function, and then classical K-means partitions them by linear separators in the feature space. Due to lack of space we remind the reader to [11] for details about KK-means.

A main difficulty when clustering trajectories is that different paths have unequal length due to their time varying nature. To address this issue many approaches rely on a procedure of length normalization or dimensionality reduction, while other methods use size independent distance measures [10]. However normalization techniques result suboptimal since they often imply some loss of information, while size independent distances introduce the problem that in many cases they cannot be trivially transformed into a positive semidefinite kernel function as required by KK-means. To circumvent these problems in this paper we adopt the DTW kernel originally proposed in [4]:

$$K_{DTW}(F_a, F_b) = \sum_{\rho\in\mathcal{A}(F_a,F_b)} \prod_{i=1}^{|\rho|} \kappa(F_a(\rho(i)), F_b(\rho(i)))$$

which is a positive definite kernel under weak assumptions on $\kappa(\cdot)$. The idea behind this kernel is to compare two trajectories F_a and F_b considering the set $\mathcal{A}$ of all possible alignments ρ between them and use dynamic programming to compute the sum of all alignments scores. Note that this concept is different from the idea of standard DTW distance where only the optimal alignment is computed. Up to our knowledge this kernel, originally proposed for speech applications, has never been employed in the context of trajectories analysis. Our experiments demonstrate that it can be used to accurately cluster pedestrians paths. An example of clustering results is shown in Fig.2.a. Due to visualization purposes only a subset of the original set of 80 trajectories is depicted. In our experiments we set $\kappa(\mathbf{x}, \mathbf{y}) = e^{\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}}$.

Table 1. Average cross-validation error

	(a)		(b)		(c)	
	δx	δy	δx	δy	δx	δy
Synthetic	3.6	4.1	4.6	4.7	6.3	5.1
Real	5.9	6.1	7.6	8.9	6.7	8.6

Modeling. We use all the clustered sequences F_i of flow vectors to construct a training set $\mathcal{D} = \{(\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_N, \mathbf{t}_N)\}$. Here $\mathbf{s}_i = (x_i, y_i, c_i)$ indicates the position of the target and the associated cluster while $\mathbf{t}_i = (\delta x_i, \delta y_i)$. The goal is to use training data $\mathcal{D}$ to learn a mapping f between the location space $\mathcal{S} = \{\mathbf{s}_i \in R^3\}$ and the trajectory patterns space $\mathcal{T} = \{\mathbf{t}_i \in R^2\}$ which allows to predict the instantaneous velocity vector $\tilde{\mathbf{t}}$ for a new input datapoint $\tilde{\mathbf{s}}$. To this aim we propose the following algorithm:

Training phase
Input: *Training set $\mathcal{D}$, parameters σ_s, σ_t, γ*
1. *Set* $k_s(\mathbf{x}, \mathbf{y}) = e^{\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{\sigma_s^2}}$ *and* $k_t(\mathbf{x}, \mathbf{y}) = e^{\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{\sigma_t^2}}$.
2. *Compute λ, $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ solving (1) and (2).*
3. *For each training pair $(\mathbf{s}_i, \mathbf{t}_i)$ compute $\boldsymbol{\pi}(\mathbf{s}_i)$.*
Output: $\boldsymbol{\Pi} = \{\boldsymbol{\pi}(\mathbf{s}_1), \ldots, \boldsymbol{\pi}(\mathbf{s}_N)\}$, λ, $\boldsymbol{\sigma}$, $\boldsymbol{\tau}$

Test phase
Input: *A new input point $\tilde{\mathbf{s}}$, λ, $\boldsymbol{\sigma}$, $\boldsymbol{\tau}$, the parameter P.*
1. *Compute* $\boldsymbol{\pi}(\tilde{\mathbf{s}})$.
2. $\forall \boldsymbol{\pi}(\mathbf{s}_i) \in \boldsymbol{\Pi}$, *compute* $w_i = \|\boldsymbol{\pi}(\tilde{\mathbf{s}}) - \boldsymbol{\pi}(\mathbf{s}_i)\|^2$.
3. *Create the set $\boldsymbol{\Upsilon}_P = \{(\mathbf{t}_p, w_p) : \mathbf{t}_p \in \mathcal{D}\}$ of the P output vectors corresponding to the $\mathbf{s}_i$ associated to the P neighbors $\boldsymbol{\pi}(\mathbf{s}_i)$ closest to $\boldsymbol{\pi}(\tilde{\mathbf{s}})$ according to w_i.*
4. $\forall \mathbf{t}_p \in \boldsymbol{\Upsilon}_P$ *compute* $\tilde{w}_p = 1 - \frac{w_p}{\sum_{\boldsymbol{\Upsilon}_P} w_p}$.
5. *Compute* $\tilde{\mathbf{t}} = \frac{1}{P} \sum_{\boldsymbol{\Upsilon}_P} \tilde{w}_p \mathbf{t}_p$.
Output: $\tilde{\mathbf{t}}$

Among previous works in the context of trajectory patterns learning the most similar to our algorithm is perhaps the one proposed in [5]. However in [5] δx and δy are considered separately and two independent regression functions are learned. Thus the underlying structure of the output space is completely ignored. On the contrary with KCCA we consider the output space $\mathcal{T}$ globally and we emphasize the correlations between the locations and the motion patterns space. The advantage of considering a unique model rather than two separate ones is confirmed by our experiments (Table 1). Note that incorporating the cluster membership c in the input vector is crucial since we circumvent the problem of predicting multiple outputs from a single input point: a situation that occur in case of multiple crossing paths where the same location can be associated with different velocity vectors. On the other hand, with KK-means and specifying a sufficient number of clusters we avoid this difficulty.

4. Motion Aware Multi-Person Tracking

To demonstrate the effectiveness of the learned motion model we use it to improve the performance of a people tracker. To this purpose, we resort on the multi-person tracking algorithm proposed by [8]. This approach is based on particle filters.

Let $\mathbf{s}_t = \{s_t^1 \ldots s_t^J\}$ denotes the hidden state which represents the spatial configuration of J different targets and $\mathbf{o}_t$ be the associated observations extracted from the image at time t. In Bayesian tracking the sequence of hidden states $\mathbf{s}_{1:t}$ is estimated based on the observed data $\mathbf{o}_{1:t}$. All Bayesian estimates of $\mathbf{s}_t$ follow from the posterior distribution $p(\mathbf{s}_t|\mathbf{o}_{1:t})$. In particle filters $p(\mathbf{s}_t|\mathbf{o}_{1:t})$ is approximated by a set of samples (the particles) $\{\mathbf{s}_t^i, \mathbf{w}_t^i\}$ each one associated with a weight $\mathbf{w}_t^i$ which indicates its "quality". In [8] the joint posterior distribution is approximated by the outer product of its marginals $p(\mathbf{s}_t|\mathbf{o}_{1:t}) \approx p(s_t^j|\mathbf{o}_{1:t})$. According to this approximation the dynamical model $p(\mathbf{s}_t|\mathbf{s}_{t-1}) = \prod_j p(s_t^j|s_{t-1}^j)$ accounts for the motion of the targets separately while the observation model $p(\mathbf{o}_t|\mathbf{s}_t)$ consider the joint objects configuration and involves the computation of a joint likelihood function which takes explicitly into account occlusions.

We modify the original tracking algorithm by incorporating the learned motion model into $p(\mathbf{s}_t|\mathbf{s}_{t-1})$. More specifically for each target we replace the original uniform isotropic motion model with: $p(s_t|s_{t-1}) = p(\mathbf{x}_t, c_t|\mathbf{x}_{t-1}, c_{t-1}) = p(c_t|\mathbf{x}_{t-1}, c_{t-1})p(\mathbf{x}_t|\mathbf{x}_{t-1}, c_{t-1})$ $i.e.$ we assume the current location and cluster indicator being conditionally independent from the location and the cluster indicator at time $t-1$. We further define $p(c_t|\mathbf{x}_{t-1}, c_{t-1}) \approx p(c_t|\mathbf{l}_{t-1}, c_{t-1})$, where $p(c_t|\mathbf{l}_{t-1}, c_{t-1})$ is a probability table learned from training data and obtained discretizing the space of all possible locations $\mathbf{x}$ into a finite set of points $\mathbf{l}_i$, $\mathbf{l}_i \in \mathcal{L} = \{1...L\}$. In practice we construct a regular grid on the floor and for each cell in the grid $\mathbf{l}_{t-1}$ we compute a table of co-occurrencies $p(c_t|c_{t-1})$. The assumption behind this approximation is that the transition between clusters is smooth w.r.t. location changes. We also define $p(\mathbf{x}_t|\mathbf{x}_{t-1}, c_{t-1}) = \mathcal{N}(\mathbf{x}_t; \mathbf{x}_{t-1} + \Delta\mathbf{x}_{t-1}, \Sigma_\mathbf{x})$. Here $\Delta\mathbf{x}_{t-1} = (\delta x_{t-1}, \delta y_{t-1}) = \tilde{\mathbf{t}}$ is computed from $\tilde{\mathbf{s}} = (x_{t-1}, y_{t-1}, c_{t-1})$ as indicated in the test phase of the algorithm in Section 3 while $\Sigma_\mathbf{x} = diag(\sigma_x, \sigma_y)$ with σ_x, σ_y user defined parameters.

5. Experimental Results

We first evaluate the performance of the modeling algorithm on static data. The goal is to learn the mapping function f from a training set and use it to compute the instantaneous velocity on test data. We consider both

Table 2. Tracking results (F-measure)

	s1	s2	s3	s4	s5
n^o targets	2	2	3	3	3
Isotropic motion	0.63	0.94	0.60	0.86	0.58
Our approach	0.83	0.95	0.65	0.86	0.84

synthetic and real data. Synthetic data are represented by colored balls moving on a black background and following user defined trajectories. The trajectories were created generating δx and δy according to a Gaussian distribution. The real data consist of the 80 collected paths. We run a 5-fold crossvalidation experiment. Results are reported on Table 1 (we measure the error as $100 \times |\delta q^{GT} - \delta q^{OUT}|/\delta q^{GT}$, $q = x, y$). We compare: (a) our approach, (b) linear KCCA (KCCA with linear kernels in the input and output space), (c) Separate KCCA *i.e.* KCCA with correlations between the input space vector and δx modeled separately from those between the input space and δy. Table 1 clearly shows how modeling nonlinearity and jointly considering δx and δy produces more accurate estimates.

A second series of experiments aims to quantify the effectiveness of our approach in the contest of tracking. We focus our attention on video surveillance single-camera data. We first run the tracking algorithm with the isotropic motion model on several sequences. Then we consider the motion aware particle filter on the same sequences. A ground truth used to evaluate the results was derived by manually marking the body area on the image plane and annotating the identity of each person present every 5 frames. From the ground truth GS and the tracking estimate TS windows we measure the performance of both algorithms in terms of average F-measure $F = (2PR)/(P + R)$ where $P = (TS \cap GS)/TS$ is the precision and $R = (TS \cap GS)/GS$ the recall. This measure not only detects tracking failures situations such as target lost or change of identity but also indicates tracking quality. Table 2 shows the results on 5 sequences: it is clear that the learned dynamical model greatly improves tracking accuracy. The advantage of the proposed method can be observed also in Fig.2.b where the trajectories extracted during tracking are depicted. It is evident that paths corresponding to the learned motion model (red) are more stable than those obtained with the isotropic motion model (blue). Similar conclusions can be drawn from Fig.3. Without a learned motion model the target on the left is lost after the occlusion due to the presence of the column. On the contrary the motion aware particle filter successfully tracks both targets for the entire sequence. It is worth noting that the results presented in this Section also indirectly confirm the effectiveness of the proposed clustering approach. We encourage the reader to look at the supplementary material for further results[1].

[1] tev.fbk.eu/people/zen/trajectories_hall.html

Figure 3. Tracking results without (top) and with (bottom) learned motion model

6. Conclusions

We presented a new approach for learning pedestrian trajectories from real data. The proposed method which mainly relies on kernels has been successfully incorporated into a multi-person tracker. Its effectiveness has been shown through extensive tests on many video surveillance sequences. Due to its flexibility in the future we plan to extend the modeling algorithm augmenting the size of the input space trying to capture other patterns such as people interactions.

References

[1] S. Ali and M. Shah. Floor fields for tracking in high density crowd scenes. *In ECCV*, pages 1–14, 2008.

[2] G. Antonini, S. Martinez, M. Bierlaire, and J. Thiran. Behavioral priors for detection and tracking of pedestrians in video sequences. *IJCV*, 2006.

[3] P. Baiget, E. Sommerlade, I. Reid, and J. Gonzalez. Finding prototypes to estimate trajectory development in outdoor scenarios. *In THEMIS*, 2008.

[4] M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. *ICASSP*, 2007.

[5] D. Ellis, E. Sommerlade, and I. Reid. Modelling pedestrian trajectories with gaussian processes. *In Ninth International Workshop on Visual Surveillance*, 2009.

[6] D. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis; an overview with application to learning methods. *Neural Computation*, 16:2639–2664, 2004.

[7] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28:312–377, 1936.

[8] O. Lanz. Approximate bayesian multibody tracking. *IEEE Trans. PAMI*, 28(9):1436–1449, 2006.

[9] D. Makris and T. Ellis. Path detection in video surveillance. *Image and Vision Computing*, 20:895–903, 2002.

[10] B. Morris and M. M. Trivedi. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. *CVPR*, 2009.

[11] B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 1998.