Finding Prototypes to Estimate Trajectory Development in Outdoor Scenarios

Pau Baiget*, Eric Sommerlade+, Ian Reid+, Jordi Gonzàlez*
* Computer Vision Center, Edifici O. Campus UAB, 08193, Bellaterra, Spain
+ Dept. of Engineering Science, University of Oxford, U.K.
email: pbaiget@cvc.uab.es

Abstract

The incorporation of qualitative information into tracking systems is a challenging topic of research in the computer vision area. Within the context of an outdoor scenario, it is usual to find some patterns of human motion and, in general there is a reduced set of typical entrance and exit points. In this work we present a method to analyze an existing set of trajectories obtained in a selected outdoor environment. Trajectories are clustered based on their entrance and exit points in order to model prototypes that are used to on–line estimate the development of a new trajectory. Those prototypes are modeled using spline curves to avoid the inaccuracy pulled from the vision system. Interesting applications comprise abnormal behavior detection, spatio–temporal event analysis and nd semantic interpretation of human behavior based on the context.

1 Introduction

The contemporary ubiqu ity of surveillance cameras requires automated methods of video analysis to exploit the resulting data. With traffic analysis, sports commentary, site security and care for the elderly as a few examples, the number of application domains and their respective setting is legion, and hence demands adaptive strategies to classify the activity in the scene. These can be found for varying resolution levels, e.g. facial expression, a given human action, or – in this work – the trajectories of the agents in the scene, i.e. the location and velocity of targets such as vehicles or pedestrians while under surveillance.

Scene activity analysis is an active field of research, resulting in many properties extracted from the agent's movement in the scene [11, 6, 7]. The use of this extracted data is twofold. Primarily, the data is fed back to aid the lower level tracking system. For example the entry and exit points in the supervised area, such as doors, tunnels, support initialisation and termination of tracks and facilitate association of agents across disparate fields of view of multiple cameras. Typical trajectories in the scene are recoverd and then aid in reestablishment of correspondence after loss of track/occlusion. The second use is the interpretation of the observed behaviour, where entry/exit zones, junctions, paths and stop zones are labeled accordingly. This information is then fed to higher levels of the system, for example into natural language generators[8].

However, little work has been done on exploiting the predictive capabilities of these identified attributes. With regard to the complexity of higher level reasoning, where decisions demand longer processing time [8], a good prediction of target movement is vital.

The focus of this work lies not on classification of the observed behaviour, i.e. events that have passed already, but on facilitating a pro-active reasoning, to possibly prevent or predict future collisions. For this, we present a novel method of obtaining entry and exit points, which are then used to find protoype trajectories between these points, along with their uncertainty. These trajectories come in the formulation of piecewise continuous B-splines. Their properties allow an elegant yet simple acquisition of a target's future motion and its accompanying uncertainty. We demonstrate this method on a variety of real life scenarios.

The reminder of this paper is structured as follows. In the next section, we detail the related work, highlighting key differences to our approach. In section 3, we explain the algorithm in detail, and show the performance in certain scenarios in section 5. The paper finishes with conclusion and discussion of the results obtained and an outlook on the future work.

2 Related Work

Several approaches faced this problem for the traffic domain. Johnson and Hogg [5] trained a neural network to generate a probabilistic map of a scenario, assigning a probable direction to each pixel in the image plane. Fernyhough et al. [3] proposed a method to learn and classify semantic regions from a scenario. This approach recognizes common paths by extending trajectories with the spatial extent occupied by the agents in camera coordinates. Although the method does not need any a–priori information, it requires full trajectories and cannot handle on–line learning. In addition, this method does not use orientation to compute paths and thus does not distinguish between objects following the same route but different directions. The lack of conceptual labelling of the scenario is addressed by Makris and Ellis in [7], learning entry/exit zones and routes from trajectory samples. However, our contribution is focused not only in acquiring qualitative information from the environment, but we use this information to on–line estimate the development of a currently tracked trajectory.

3 Learning Trajectory Prototypes

The procedure described in this section processes a set $T = \{t_1, \ldots, t_n\}$ of human trajectories, defined as the sequence of ground–plane positions occupied by a human agent over in each time step, hence $t_i = \{(x_1, y_1), \ldots, (x_m, y_m)\}$. These trajectories have been obtained using a computer vision framework, consisting of a calibrated camera and a tracking system. The latter is in charge of obtaining the trajectory coordinates in the image–plane, and the former ensures the translation of the trajectories into ground–plane coordinates.

3.1 Human Trajectory Acquisition

The architecture of the tracking algorithm used to obtain trajectories [9] is based on background subtraction with group identification and structured into a modular and hierarchically-organized system, see [9] for further details. Using a camera calibration process, the ground–plane representation of the tracked trajectories is obtained, as shown in Fig. 1.(b).

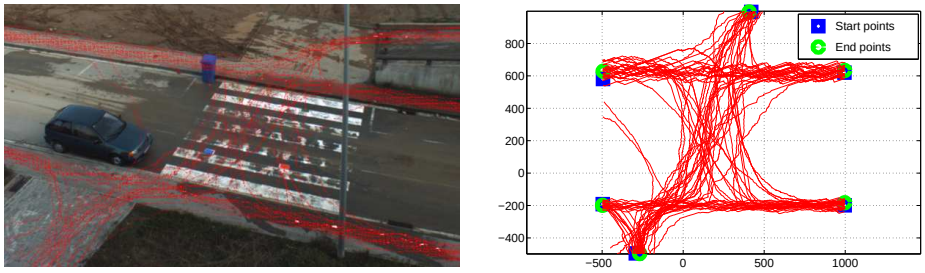


Figure 1: Trajectories depicted over the image plane. (b) Ground plane representation of the trajectories and the clustered start points (blue squares) and end points (green circles).

In the remaining of the paper, all references to a distance measure will refer to the 2D euclidean distance.

3.2 Clustering start and end points

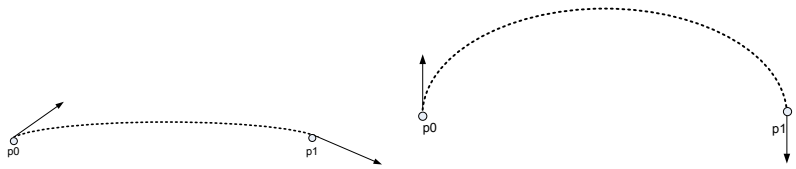
A region r of the scenario is considered to be an *start* of *end* point if the observation of the training trajectory set concludes that a non–trivial number of trajectories begin or end in that points. To this end, the first positions of the trajectory set are clustered to find the start points. For clarity purposes, we only explain in this section how to find the start points, and the same procedure is followed to find the end points, but using the last positions of the trajectory set.

The objective is to find a set E which contains the start points to the scenario. The fact that the exact number of start points is unknown beforehand discourages the use of parameter–specific clustering algorithms like k–means, although research done in that subject [1]. In order to avoid specifying the number of clusters, the *Quality Threshold* (QT) clustering algorithm [4] has been used. The clustering is based on a spatial measure, *diameter*, to establish the maximum distance between points that belong to the same cluster. In our case , since the scenario has been calibrated and trajectories are described in ground–plane coordinates, the distances between start points are expressed in real–world units, i.e. meters. Using this information, we establish a distance constraint D, expressed in meters, which describes the maximum distance allowed between two start points to belong to the same cluster.

$$\begin{aligned} S &= \{s_1,\ldots,s_k\} \\ s_i &= \{t_1,\ldots,t_m\}, \forall_{t_{p_1},t_{q_1}\in s_i} dist(t_{p_1},t_{q_1}) \leq D \end{aligned} \tag{1}$$

Following Eq. 1, the QT-clustering algorithm outputs the set S, which separates the training trajectory set T into k disjoint subsets. In each subset s_i, the initial trajectory positions ($t_{1_1},\ldots,t_{1_m}$ in Eq. 1) must be separated by a maximum distance D (called *complete linkage* clustering). Afterwards, a centroid is computed for each subset s_i by simply computing the average point from the point list $\{t_1,\ldots,t_m\}$.

$$\begin{aligned} C_S &= \{c_{s_1},\ldots,c_{s_k}\} \\ c_{s_i} &= \textstyle\sum t_j \in s_i/k\} \end{aligned} \tag{2}$$



(a) (b)

Figure 2: The speed vectors completely determine the curve of the spline between the points p_0 and p_1. In these two samples it can be seen how to obtain different splines by just varying the derivative vectors.

In spite of being computationally more expensive than other clustering algorithms like *k*-means, the benefit obtained avoiding setting predefined parameters allows the system to be adaptable to changes in the environment. Since the diameter is expressed as the maximum distance between points of the same cluster, the detected start / end points could be larger or smaller, depending on the selected scenario.

After applying the QT-clustering of the start and end points in the training trajectory set, two sets $C_S = \{c_{s_1}, \ldots, c_{s_k}\}$ and $C_E = \{c_{e_1}, \ldots, c_{e_l}\}$. Finally, the training trajectory set T is splitted into $k \times l$ subsets, each of them being identified by a start and an end point.

3.3 Trajectory Prototypes

Each of the subsets mentioned above corresponds to the pair $(s_i, e_j) \in C_S \times C_E$. Thus, the trajectories contained in the subset represent the *observed* paths that the tracked agents have taken to reach the end point e_j having entered the scenario from start point s_i. A combination of these trajectories will allow to create a *trajectory prototype*, i.e., a model of the trajectory that is usually performed to go from the start point s_i to the end point s_j. Nonetheless, the trajectories contained in the subset could be significantly different, even when the tracked agent has walked over the same area within the scenario. This could be provoked either by computer vision issues like measuring errors or by the velocity and body motion performed by the agent during his existence in the scenario. In order to solve the aforementioned problems, the trajectories are converted into a spline representation[2]. The main advantage over the initial structure of the trajectory is that the spline acts as a continuous function which takes values from $[0 \ldots 1]$, allowing to sample points with any required precision, getting rid of the inaccuracy pulled from the tracking system.

The spline representation of a trajectory $t = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ has to deal with the features of human motion, i.e.,complex curve shapes that might be modeled. Hence, a simple spline is not a representation robust enough to model all types of trajectories. Instead, we propose to use a concatenation of simple splines (also called *B-splines*) as the best representation. Thus, the spline representation of t, called $sp(t)$, will be $sp(t) = \{bsp_1, \ldots, bsp_d\}$.

In order to ease the notation, let us consider that one b-spline models a segment of a trajectory, e.g $\{x_p, \ldots, x_q\}$. The construction of the b-spline needs of two control points p_0 and p_1, which coincide with the starting and ending point of the spline curve. However, the curvature of the spline is determined by the *speed* in the entrance and exit of the spline, see Fig. 2. This speed is represented by two normalized vectors v_0 and v_1, i.e.

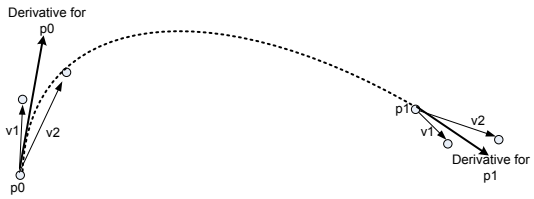


Figure 3: Computing the derivatives for a simple b-spline. The average of the vectors formed by p_0 and the subsequent trajectory points are normalized to obtain the derivative of the spline in p_0. The same is done for p_1, and that derivative is kept equal for the next spline.

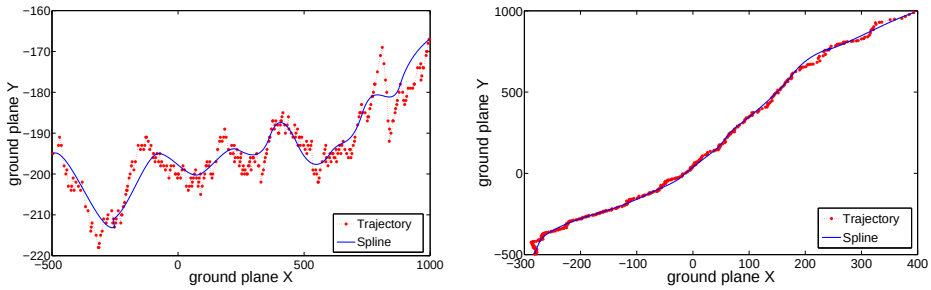


Figure 4: Example of trajectories fitted to splines. Trajectory points are more separated in the left graphic because it represents a smaller region.

the derivatives of the spline in the points p_0 and p_1. These two vectors are obtained considering the vectors between the points p_0 and p_1 and their successor points in the trajectory and computing the average, see Fig. 3(b). In the figure, the vectors v_1 and v_2 are obtained by joining p_0 with the next two positions in the trajectory. The more vectors are used to compute the derivatives, the better the b-spline will represent the sequence of positions $\{x_p,\ldots,x_q\}$. In the case of p_1, the points required to construct the vectors v_3 and v_4 are taken from the next part of the trajectory to be modeled. Thus, the derivative obtained for p_1 at the present b-spline will be the same as that for p_0 in the b-spline for the next trajectory segment. This fact allows to obtain a continuous and smooth concatenation of b-splines.

Let $T_{ij} = \{\tau_1,\ldots,\tau_q\}$ be the trajectories from T that begin in the start point s_i and finish at the end point e_j. In order to generate their spline representation, the control points, i.e. those which separate the b–splines, are chosen to maintain a spatial and temporal consistency between each b–spline. To this end, the splines are created using K control points and thus $K-1$ b–splines. This parameter K is fixed for all the trajectories in the same T_{ij}, so their splines can be sampled obtaining the same proportion of points. Fig. 4 shows example trajectories and their conversion to the spline representation.

The resulting splines $\{sp(\tau_1),\ldots,sp(\tau_q)\}$ are then sampled to obtain D points of each b–spline:

$$SAMPLE(sp(\tau_1)) \quad = \quad [p_{1_1},...,p_{1_{D*(K-1)}}]$$

$$\vdots$$

$$SAMPLE(sp(\tau_q)) \quad = \quad [p_{q_1},...,p_{q_{D*(K-1)}}] \tag{3}$$

Finally, the prototype trajectory is generated by computing the average of each tuple $[p_{1\alpha},\ldots,p_{q\alpha}], \alpha \in [1\ldots D*(K-1)]$ from the previous sequences. A new spline P_{ij} is then fit to the resulting sequence, using K control points as done with the previous splines. This result constitutes the *common* path between the start point s_i and the end point e_j and will in the next section to measure how likely it is that an agent exits the scenario by e_j given the fact that comes from s_i.

4 Uncertainty Measuring

The objective is to measure the uncertainty issued by the entrance of a new target in the scene, and how its development within the ground–plane can give a guess about where the agent will be in the future time steps.

Let $t_* = \{(x_1,y_1),\ldots,(x_\mu,y_\mu)\}$ be the trajectory of a new agent that has been tracked within the ground–plane until time step μ. Considering the initial position of t_*, we estimate its start point s_* by choosing the closest start point from the start points C_S. Let $\{P_{*,1},\ldots,P_{*,||E||}\}$ be the prototypes that begin at the start point s_*. Additionally, let $\{c^1_{*,\theta},\ldots,c^{K-1}_{*,\theta}\}$ be the control points that were used to generate $P_{*,\theta}$, being $\theta \in [1\ldots||E||]$.

The uncertainty at time step μ is measured in terms of the distance between the position (x_μ,y_μ) and each of the above defined prototypes. Although there exist methods to compute the closest point on a spline curve, e.g. [10], they are still computationally expensive and, however, the control points of the spline serve as a simplified distance measure that does not dramatically affect the final result. Therefore, the distance is measured to one control point of each prototype. The control points are chosen in order, beginning at the nearest point to s_*. The current control point, $c^\gamma_{*,\theta}$ is replaced by the next if:

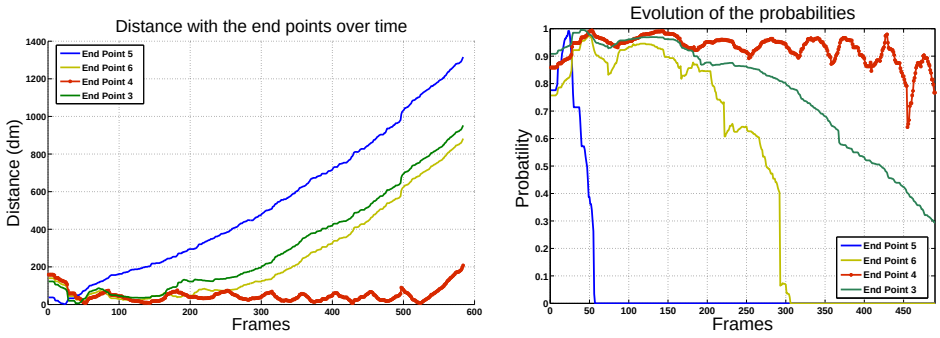
$$\begin{array}{lcl} dist(\{c^\gamma_{*,\theta},(x_\mu,y_\mu)\}) & > & dist(\{c^\gamma_{*,\theta},(x_{\mu-1},y_{\mu-1})\}) \\ & AND & \\ dist(\{c^\gamma_{*,\theta},(x_\mu,y_\mu)\}) & \geq & dist(\{c^{\gamma+1}_{*,\theta},(x_{\mu+1},y_{\mu+1})\}) \end{array} \tag{4}$$

Finally, $\{d^\mu_{*,1},\ldots,d^\mu_{*,||E||}\}$ are the distances from (x_μ,y_μ) to the prototypes at time step μ. Also, we consider the distances $\{D_1,\ldots,D_{||E||}\}$ from the current control point to the end point, passing through all the intermediate control points:

$$D_\theta = \sum_\gamma dist(c^\gamma_{*,\theta},c^{\gamma+1}_{*,\theta}) \tag{5}$$

With these measures we compute the probability $P(e|(x_\mu,y_\mu))$ that the agent finishes its trajectory in the end point $e \in \{1,\ldots,||E||\}$ given the current position:

$$P(e|(x_\mu,y_\mu)) = 1 - \frac{|D_e| - d^\mu_{*,e}}{|D_e|} \tag{6}$$



(a) (b)

Figure 5: (a) Evolution of the distance between the experimental trajectory t_1 w.r.t. the trajectory prototypes and the possible ending points. (b) Evolution of the probabilities for the experimental trajectory to end in each of the detected end points. The highlighted graphic represents the final end point of the trajectory.

5 Results

Experiments have been run in the urban outdoor scenario introduced in previous sections. This scenario is specially interesting due to the fact that several start and end points can be found, some of them very close. Also, these points are not very restricted and there are not phisical obstacles that impede to reach one end point from one of the a–priori expected start point.

We have processed a set of 200 complete trajectories obtained using one day of continuous recording. The training process takes about 5 minutes to cluster the start and end points and the probability estimation has been achieved in real–time, running 25 frames per second.

After clustering the training set, the proposed distance and uncertainty measure is tested with new trajectories. Results can be observed in Fig. 5.(a), where the distance with the splines is shown over time, and in Fig. 5.(b), showing the estimated probabilities of reaching one of the detected end points. In the initial frames, all the prototypes are very close to the current position, but it can be seen that when the trajectory develops within the scenario, it keeps following one of the prototypes.

6 Conclusions

In this work we have presented a method to analyze an existing set of trajectories, obtained by a tracking system in a selected outdoor environment. From these we have obtained the most frequent entrance and exit points and then trajectories have been clustered in order to model trajectory prototypes that have been applied to on–line estimate the development of a new trajectory. The prototypes have been modeled using splines curves in order to avoid the inaccuracy pulled from the vision system. We have computed the probabilities for a new trajectory to leave the scenario through a detected exit point, given the entrance point and the current trajectory position. The results shown in the previous section are promising, because they allow to improve two areas that are still on research. On the one hand,

the probability estimation can be a useful feedback to the tracking system when dealing with long occlusions or cluttered environments. On the other hand, these predictions are helpful towards high-level reasoning about human behavior within the environment.

Acknowledgements

This work is supported by EC grants IST-027110 for the HERMES project and IST-045547 for the VIDI-video project, and by the Spanish MEC under projects TIN2006-14606 and Consolider-Ingenio 2010 (CSD2007-00018). Jordi Gonzàlez also acknowledges the support of a Juan de la Cierva Postdoctoral fellowship from the Spanish MEC.

References

[1] K. Daniels and C. Giraud-Carrier. Learning the threshold in hierarchical agglomerative clustering. *5th International Conference on Machine Learning and Applications 2006*, pages 270–278, Dec. 2006.

[2] C. de Boor. *A practical guide to splines*. Springer-Verlag, New York, USA, 1978.

[3] J. H. Fernyhough, A. G. Cohn, and D. Hogg. Generation of semantic regions from image sequences. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume II*, pages 475–484, London, UK, 1996. Springer-Verlag.

[4] L.J. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: identification and analysis of coexpressed genes. *Genome Research*, 11:1106–1115, 1999.

[5] N. Johnson and David Hogg. Learning the distribution of object trajectories for event recognition. In *BMVC '95: Proceedings of the 6th British conference on Machine vision (Vol. 2)*, pages 583–592, Surrey, UK, UK, 1995. BMVA Press.

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

[7] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *IEEE Transactions on Systems Man and Cybernetics–Part B*, 35(3):397–408, June 2005.

[8] H.-H. Nagel. Steps toward a cognitive vision system. *AI Magazine*, 25(2):31–50, 2004.

[9] D. Rowe, I. Reid, J. Gonzàlez, and J. Villanueva. Unconstrained multiple-people tracking. In *28th DAGM, Berlin, Germany*, volume LNCS 4174, pages 505–514. Springer, 2006.

[10] M. Wang, J. Kearney, and K. Atkinson. Robust and efficient computation of the closest point on a spline curve. In *Proceedings of the 5th International Conference on Curves and Surfaces*, pages 397–406, San Malo, France, 2002.

[11] X.Wang, K. Tieu, and E. Grimson. Learning semantic scene models by trajectory analysis. In *Proceedings of the ECCV*, Graz, Austria, 2006.